Opinion Mining in a Code-Mixed Environment: A Case Study with Government Portals

Deepak Gupta, Ankit Lamba, Asif Ekbal and Pushpak Bhattacharyya
Department of Computer Science & Engineering
IIT Patna, Patna
Bihar, India
{deepak.pcs16,ankit.cs12,asif,pb}@iitp.ac.in

Abstract

With the phenomenal growth in social media, citizens are coming forward to participate more in discussions on socially relevant topics including government policies, public health etc. India is not an exception to this, and the website *mygov.in* launched by the Government of India acts as a platform for discussion on such topics. People raise their viewpoints as comments and blogs on various topics. In India, being a diverse country, citizens write their opinions in different languages, which are often in mixed-languages. Code-Mixing refers to the mixing of two or more languages in speech or in a text, and this poses several challenges. In this paper, we propose a deep learning based system for opinion mining in an environment of code-mixed languages. The insights obtained by analyzing the techniques lay the foundation for better lives of citizens, by improving the efficacy and efficiency of public services, and satisfying complex information needs arising within this context. Moreover, understanding the deep feelings can help government to anticipate deep social changes and adapt to population expectations, which will help building Smart city.

1 Introduction

The report as published by Statista shows that there has been a phenomenal growth in the use of social media and messaging applications. It has grown 203 percent year-on-year in 2013, with overall application use rising 115 percent over the same period. This implies that 1.61 billion people are now active in social media around the world and this is expected to rise to 2 billion users in 2016, led by India. The research also reveals that users daily spend approximately 8 hours on digital media including social medias and and mobile internet usages.

At the heart of this interest is the ability for users to create and share contents via a variety of platforms such as blogs, microblogs, collaborative wikis, multimedia sharing sites, social networking sites etc. The unprecedented volume and variety of user-generated contents, as well as the user interaction networks constitute new opportunities for understanding social behavior and building socially intelligent systems. Therefore, it is important to investigate tools and methods for knowledge extraction from social media data.

In social media, contents are often written in mixed-languages, and this phenomenon is known as code-mixing. Code-Mixing or code-switching is defined as the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language. This phenomenon is prevalent among bi-lingual and multilingual individuals. This is a well-known trait in speech patterns of the average bilingual in any human society all over in the world. With the phenomenal growth in social media, people from different dialects participate on web portals to showcase their opinions. This diversity of users contributes to the non-uniformity in texts and as a result the data generated lead to code-mixed. There is also a tendency among the users to write in their own languages, but in transliterated forms. Transliteration is the process of converting a text from one form to the other. Transliteration is not merely a task of representing sounds of the original characters, ideally it should be done accurately and unambiguously. Hence, we must have a way to convert transliterated text into its own original script for effective analysis. One of the crucial issues in code-mixed languages is to identify the origin of a text for further processing. Hence, we must have a way to discriminate texts written in different scripts. Given a text, the task is to identify the origin of a text, i.e. the language in which it belongs to. In this paper we propose an approach based on deep learning for sentiment analysis[1] of the user comments written in a well-known public portal, namely mygov.in, where citizens express their opinions on different topics or government schemes. This will facilitate urban informatics (for building Smart Cities), where the goal is to analyze the opinions that lay the foundation for improving the lives of citizens, by improving the efficacy and efficiency of public services, and satisfying complex information needs arising within this context. Moreover, understanding the deep feelings can help government to anticipate deep social changes and adapt to population expectations, which will help building smart city. The first task that we address is a classification problem, where each word has to be labeled either with one of the two classes, either Hindi or English [2]. We propose a technique for language identification, which is supervised. We also do back-transliteration whenever necessary. The final step is to find the opinion expressed in each comment. We propose a deep convolutional neural network (CNN) based approach to solve this particular problem.

Most existing work on sentiment analysis makes use of handcrafted features for training of the supervised classifier. This process is expensive and requires significant effort to extract features. Moreover, handcrafted features which are generally specific to any particular domain requires to be altered once we focus on a different domain. Our proposed model does not use any handcrafted features for sentiment classification, and hence can be easily adapted to a new domain and language. The contributions of the present research can be summarized as follows: (i). we propose a deep learning based approach for opinion mining from a code-mixed data (ii). we develop a system that could be beneficial for building a smart city by providing useful feedback to the govt. bodies (iii). we create resources for sentiment analysis involving Indian language code-mixed data.

The rest of the paper is structured as follows:

Comments	Language Identification	Back-Transliteration	Opinion
Swachh bharat is a good initiative to unite every one. i wish our PM for the same	E E E E E E E E E E. E E E E E E E	Not-Required	Positive
jab tak puri tarh se polithin ke prayog band nhi hoga swatch bharat abhiyan kabhi pura nhin hoga	H H H H H H H H H H H H H H H H H H	जब तक पुरी तरह से पालीथिन के प्रयोग बन्द नही होगा स्वच्छ भारत अभियान कभी पुरा नहीं होगा	Negative

Table 1: Examples of comments along with the output at various steps of the proposed approach. Langage identification **E**: English, **H**: Hindi.

In Section 2 we present a brief literature overview. Pre-processing, annotation and statistics of the resources that we created are described in Section-3. Section-7 gives the details of our proposed convolutional network model to identify opinion from comments. The experimental setup along with the details of external data are described in Section-8. The obtained results, key observations and error analysis are discussed in Section-9. Finally, we conclude in Section-10.

2 Related Works

Nowadays deep learning models are being used to solve various natural language processing (NLP) problems. Usually, the input to any deep learning based model is the word representation. Some of the commonly used word representation techniques are word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), Neural language model (Mikolov et al., 2010), etc. Distributed representation of a word is one of the popularly used models (Hinton, 1984; Rumelhart et al., 1988). Similarly recurrent neural network (Liu et al., 2015) has been used for modeling sentence and documents. The numeric vectors, used to represent words are called word embedding. Word embedding has shown promising results in variety of the NLP applications, such as named entity recognition (NER) (dos Santos et al., 2015), sentiment analysis (Socher et al., 2013b) and parsing (Socher et al., 2013a; Turian et al., 2010). The convolutional neural network(CNN) (LeCun et al., 1998) was originally proposed for computer vision. The success of CNN has been seen in few of the NLP applications such as sentence modeling (Kalchbrenner et al., 2014), semantic parsing for question answering (Yih et al., 2014), query retrieval in web search (Shen et al., 2014), sentence classification (Kim, 2014; Socher et al., 2013b) etc. Collobert (Collobert et al., 2011) has also claimed the effectiveness of CNN in traditional NLP task such as PoS tagging, NER etc. Deep learning based architectures have shown success for sen-

[1]Here we use the terms opinion mining and sentiment analysis interchangeably

[2]Here, we consider that our contents are mixed in nature that contains either English or Hindi or both.

timent classification of tweets, such as (Tang et al., 2014; dos Santos and Gatti, 2014). The domain adaption for large scale sentiment classification has been handled through deep learning model (Glorot et al., 2011). In social media contents, code-mixing where more than one language is mixed is very common that demands special attention. Significant characteristics of code mixing have been pointed out in some of the works such as (Milroy and Muysken, 1995; Alex, 2007; Auer, 2013). In a multi-lingual country like India, code-mixing poses a big challenge to handle the contents in social media. Chinese-English code mixing in Macao (San, 2009) and Hong Kong (Li, 2000) indicated that linguistic constructions predominantly trigger code mixing. The work reported in (Hidayat, 2012) showed that Facebook users tend to mainly use inter-sentential switching over intra-sentential. A code-mixed speech corpus of English-Hindi on student interviews is presented in (Dey and Fung, 2014). It shows analysis and motivations of code mixing, and discusses in what grammatical contexts code mixing occurs (Dey and Fung, 2014) . To the best of our knowledge we do not see the use of any deep learning that addresses the problem of sentiment analysis in a code-mixed environment.
In our current work we discuss the scope for text analysis based on deep learning architecture on government data / citizen views which can very well frame a new concept of better e-governance.

3 Resource Creation

We design a web-crawler to crawl user comments from *mygov.in* portal. We consider the comments written for the section of 'cleanliness in school curriculum' under *Swachh Bharat Abhiyaan*. In total 17,155 cleaned[3] comments were crawled from the web [4]. The contents are mixed in nature containing both *English* and *Hindi*. Hence, it poses several challenges to extract meaningful information.

3.1 Pre-processing

We pre-process the crawled data in order to effectively extract opinion from the comments. Since we extract the comments from an open platform where anyone has the freedom to give their opinions the way they want, there was a necessity to pre-process the data before its use. We perform the following steps:

- First we manually remove comments which are neither in English script nor in Devanagari (Hindi). For e.g., अधार्मिकाः यदा धर्मं वदन्ति स्वार्थसाधने । ते धर्मं दूषयन्त्येव
- While analyzing the comments, we noticed that some of the comments having shorter length do not contain any vital opinion. Therefore, we removed all the comments which have less than 5 words. For e.g. *Clean India, Swachh Bharat Abhiyan* etc.
- We define regular patterns to discard the strings containing html symbols, e.g. *&#039, &quot, &#8364, &trade*.
- We remove all webpage and HTML references from the data by using proper regular expressions. For e.g. `https://www.youtube.com/watch?v=wP1bmk`.
- We also observe that there are quite a few cases where the comments are duplicated. We remove all the duplicates and keep only one copy for each comment.

3.2 Data Annotation

In order to test the effectiveness of our method we manually annotate a portion of the data. It is to be noted that we perform language identification to identify the origin of written text. Thereafter, we distribute the data into two groups, one containing comments in English and the other containing comments in Hindi. We manually annotate 492 Hindi and 519 English comments using two sentiment classes, positive or negative. Sample examples are given in Table-1. The data were annotated by two experts. In order to assess the quality of a annotations by both annotators we calculate inter-rater agreement. We compute Cohen's Kappa coefficient (Kohen, 1960) agreement ratio that showed the agreements of 95.32% and 96.82% for Hindi and English dataset, respectively. We present a brief statistics of our crawled-data in Table-2.

4 Language Identification(LI)

The problem of language identification concerns with determining the origin of a given word. The task can be modelled as a classification problem, where each word has to be labeled with one of the two classes, *Hindi* or *English*. Our proposed

[3]this number is after the Pre-processing of comments
[4]https://mygov.in/group-issue/cleanliness-school-curriculum/

# of comments	17155
# of sentences	54568
Average No. of sentence/comments	3
No. of identified English Coments	14096
No. of identified Hindi Coments	3059
Total no. of tokens	$10,26,612$

Table 2: Statistics of crawled data from *mygov.in*

method for language identification is supervised. In particular we develop the systems based on four different classifiers, *random forest*, *random tree*, *support vector machine* and *decision tree*. For faster computation, we use Sequential Minimal Optimization (Platt, 1998) implementation of SVM. Random tree (Breiman, 2001) is basically a decision tree, and in general used as a weak learner to be included in some ensemble learning method. Random forest (Breiman, 2001) is a kind of ensemble learner. We use the Weka implementations[5] of these classifiers. In order to further improve the performance we construct an ensemble by combining the decisions of all the classifiers using majority voting. We use the Character n-gram, word normalization and gazetteer based features as used in (Gupta et al., 2014) to build our model. The accuracy of this model on a gold standard test set was 87.52%. A public comment is a sequence of sentences, which are made up of several word-level tokens. Each token of a sentence is labeled with one of the classes (denoting English or Hindi) that correspond to its original script. Based on the classes assigned at the token-level we classify the sentence based on the majority voting. The sentence is classified to belong to that particular class which appears most in the sentence. Mathematically, it can be defined as follows:

$$S = \{x | x \in lang(t), \forall t \in T\}$$

$$Lang(comments) = \underset{s \in S}{\operatorname{argmax}}(f(s)) \quad (1)$$

where $f(s)$ is cardinality function, $x \in \{$Hindi, English$\}$ $lang(t)$ is the language of a token t; and T denotes all the tokens in a comment.

5 Transliteration

Most of Hindi comments are in their transliterated forms. In order to train an effective word embedding model we need to have enough data. We have abundant of data sources from Hindi Wikipedia. We back-transliterate the roman script into Devanagari script. A transliteration system takes as input a character string in the source language and generates a character string in the target language as output. The transliteration algorithm (Ekbal et al., 2006) that we used here can be conceptualized as two-levels of decoding: (a) segmenting source and target language strings into transliteration units (TUs); and (b). defining appropriate mapping between the source and target TUs by resolving different combinations of alignments and unit mappings. The TU is defined based on a regular expression. For a given token belonging to 'non-native' script X[6] written in English Y as the observed channel output, we have to find out the most likely English transliteration Y that maximizes $P(Y|X)$. The Indic word(Hindi) is divided into TUs that have the pattern C^+M, where C represents a vowel or a consonant or conjunct and M represents the vowel modifier or matra. An English word is divided into TUs that have the pattern C^*V^*, where C represents a consonant and V represents a vowel (Ekbal et al., 2006). The most appropriate mappings between the source and target TUs are learned automatically from the bilingual training corpus. The transliteration of the input word is obtained using direct orthographic mapping by identifying the equivalent target TU for each source TU in the input and then placing the target TUs in order. We have used three types of statistical model to obtained transliterated output.

Model-I: This is a kind of monogram model where no context is considered, i.e.

$$P(X,T) = \Pi_{i=1}^{k} P(<x,t>_i) \quad (2)$$

Model-II: This model is built by considering next source TU as context.

$$P(X,T) = \Pi_{i=1}^{k} P(<x,t>_i | x_{i+1}) \quad (3)$$

Model-III: This model incorporates the previous and the next TUs in the source and the previous target TU as the context.

$$P(X,T) = \Pi_{i=1}^{k} P(<x,t>_i \mid <x,t>_{i-1}, x_{i+1}) \quad (4)$$

The overall transliteration process attempts to produce the best output for the given input word

[5]http://www.cs.waikato.ac.nz/ml/weka/

[6]Denotes Indian languages written in roman script and mixed with English language

using Model-III. If the transliteration is not obtained then we consult Model-II and then Model-I in sequence. If none of these models produces the output then we consider a literal transliteration model developed using a dictionary. The accuracy of this model on a gold standard test set was 83.82%.

6 Baseline Models for Sentiment Analysis

In this section we describe the baseline model that we build for sentiment analysis.

6.1 Representation of comments

An effective representation of comment is important to uncover the opinion associated with a comment. Since we deal with a code-mixed environment, it is not very straightforward to represent the tokens. Here, we describe the representation techniques used only for our baseline input. The input to the CNN based sentiment analysis model is discussed in Section-7.

Representation of English comments: We use the well-known *word2vec*[7] tool to generate the word vectors of each token. We use freely available Google news word embedding model trained on news data. A comment is finally represented by a vector composed of the word vectors of the individual tokens. The vector is generated as follows:

$$Reps(comment) = \frac{\sum_{t_i \in Comment(T)} Reps(t_i)}{\textit{number Of Lookups}} \tag{5}$$

Here, $Reps(t)$ is the token representation obtained by Google news word embedding and *number of lookups* is equal to the number of tokens from the comments present in the word embedding model.

Representation of Hindi comments: For Hindi we build our own word embedding model. For training we use the data sets obtained from the Hindi Wikipedia and some other sources (Joshi. et al., 2010; Al-Rfou et al., 2013) including all the comments that we crawled. We use *skip-gram* representation (Mikolov et al., 2013) for the training of *word2vec* tool. Further, we use Eq-5 to obtain the representations of Hindi comments. We set dimension to 200 and window size as 5. After we represent the comments in terms of vectors, we develop two baselines to compare with our proposed approach.

[7]https://code.google.com/archive/p/word2vec/

6.2 Baseline-1

The hypothesis behind this baseline being the fact that, if two comments have the same sentiments (positive or negative) then the similarity between them would be higher than any other comments having different sentiments. We use the IMDB movie reviews data sets (Maas et al., 2011) containing 2K positive and 2K negative reviews for English and Hindi movie reviews, and tourism data (Joshi. et al., 2010) to compare the opinions represented in our Hindi comments. This algorithm takes as input a comment and the source documents (i.e. datasets that we collected). Each comment and all the documents belonging to a particular set (positive set: containing all the positive comments and negative set: containing all the negative comments) are represented as vectors following the representation techniques that we discussed in Subsection-6.1. We compute the cosine similarity of a given comment with respect to all the documents present in a 'positive set' or 'negative set'. Based on the dominance of average cosine similarity, we assign the opinion. We sketch the steps in Algorithm-1.

6.3 Baseline-2: SVM based Model

We construct our second baseline using SVM that classifies the comments into positive and negative. The model is trained with the word-embedding representations as discussed in Subsection-6.1. For English, we use the IMDB movie review dataset for training. For Hindi, we use Hindi movie reviews and tourism datasets which were used in building the first baseline. In order to perform the experiment we use SVM implementation *libsvm* (Chang and Lin, 2011) with linear kernel.

7 Sentiment Analysis using CNN

We propose a method for sentiment analysis using convolutional neural network (CNN). Our model is inspired by the network architectures used in (Kim, 2014; Kalchbrenner et al., 2014) for performing various sentence classification tasks. Typically, a CNN is composed of *sentence representation matrix, convolution layer, pooling layer* and *fully connected layer*. Our proposed system architecture is shown in Fig-1. Now We describe each component of CNN in the following:

```
Input: A comment $C$, Positive reviews data,
       Negative reviews data
Output: Sentiment of a comments $Sent(C)$
$n_P$= no. of Positive reviews data
$n_N$=no. of Negative reviews data
begin
    Calculate the cosine similarity of
      comment with every positive review
    for $i = 1$ to $n_P$ do
        $sim(C, P_i) = \frac{\vec{Reps}(C) \cdot \vec{Reps}(P_i)}{\|Reps(C)\| \|Reps(P_i)\|}$
    end
    Calculate the average positive similarity
      $AvgPoS(C)$ of comment $C$ as follows:
        $AvgPoS(C) = \frac{\sum_{i=1}^{n_P} sim(C, P_i)}{n_P}$
    Calculate the cosine similarity of
    comment with every negative review.
    for $i = 1$ to $n_N$ do
        $sim(C, N_i) = \frac{\vec{Reps}(C) \cdot \vec{Reps}(N_i)}{\|Reps(C)\| \|Reps(N_i)\|}$
    end
    Calculate the average positive similarity
      $AvgNeg(C)$ of comment $C$ as follows:
        $AvgNeg(C) = \frac{\sum_{i=1}^{n_N} sim(C, N_i)}{n_N}$
    if AvgPoS(C) >AvgNeg(C) then
        $Sent(C) = Positive$ ;
    else
        $Sent(C) = Negative$ ;
    end
    return $Sent(C)$
end
```

Algorithm 1: Sentiment of comments using benchmark data sets

7.1 Sentence Representation Matrix

The input to our model is a comment *C* having 'n' words. Each token $t_i \in C$ is represented by its distributed representation $\mathbf{x} \in \mathbb{R}^k$. The distributed representation **x** is looked up into the word embedding matrix **W**. We build a sentence representation matrix **C** by concatenating the distributed representation x_i for every i^{th} token in the comment *C*. The sentence representation matrix $x_{1:n}$ can be represented as:

$$x_{1:n} = x_1 \otimes x_2 \ldots \otimes x_n \tag{6}$$

where $\otimes$ is concatenation operator. After this step, network learns to capture low-level features of words from word embedding, and then project to the higher levels. In the next step network performs the series of operations on the matrix that we obtained.

7.2 Convolution

In order to extract the common patterns throughout the training set, we use convolution operation on the sentence representation matrix. A convolution operator is applied on sentence representation that involves a filter $\mathbf{F} \in \mathbb{R}^{m \times k}$, which is applied to a window of m words and produces a new feature c_i. A feature c_i is generated from window of word $x_{i:i+m-1}$ as follows.

$$c_i = f(\mathbf{F}.x_{i:i+m-1} + b) \tag{7}$$

where f is non-linear function and b is a bias term. The feature c_i is the result of element-wise product between a filter matrix **F** and column of $x_{i:i+m-1}$, which is then summed to a single value in addition of bias term b. This filter **F** can be applied to each possible window of a word in comment C. This generates a set of features which are also called as *feature map*. More formally possible window of m words in a comment C having size n would be $\{ x_{1:h}, x_{2:h+1}, \ldots, x_{n-m+1:n} \}$. A feature map **c** can be generated by applying each possible window of word.

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \tag{8}$$

The process described above is able to extract on feature map with one filter matrix. In order to form a deeper representation of data, deep learning models apply a set of filters that work in parallel generating multiple feature maps.

7.3 Pooling

The aim of pooling layer is to aggregate information and reduce representation. The output of convolution layer is fed into the pooling layer. There are several ways to apply pooling operations on the output of convolution layer. The well-known pooling operations are: *max pooling, min pooling, average pooling, and dynamic pooling*. We apply max pooling operation (Collobert et al., 2011) over the feature map and take the maximum value as the feature corresponding to this particular filter **F**.

7.4 Fully Connected Layer

Finally, the output of pooling layers p is subjected to a fully connected softmax layer. It computes the probability distribution over the given labels (positive or negative):

$$P(y = j|c, p, a) = softmax_j(p^T\mathbf{w} + a) = \frac{e^{p^T w_j + a_j}}{\sum_{k=1}^{K} e^{p^T w_k + a_k}} \quad (9)$$

where b_k and w_k are the bias and weight vector of the k^{th} labels.

8 Datasets and Experimental setup

8.1 Datasets

Our language identification system is trained on FIRE-2014 (Choudhury et al., 2014) Hindi-English query word labeling data sets. We back-transliterate FIRE-2013 (Roy et al., 2013) data sets. Detailed statistics of training and test data used in our experiment are shown in Table-4.

8.2 Regularization

In order to overcome the effect of overfit of network, we apply dropout on the penultimate layer of the network with a constraint on l_2-norms of the weight vectors (Hinton et al., 2012). Dropout prevents feature co-adaptation by randomly setting to zero a portion of hidden units during the forward propagation when passing it to the softmax output layer.

8.3 Network Training and Hyper-parameters

We use the rectified linear units (Relu) throughout training as a non-linear activation function. However, we also experiment with *sigmoid* and *tanh* but it could not perform better than *Relu*. For English, we use 15% of training data of English movie reviews as the development data to fine-tune the hyper-parameters of CNN. Similarly, we use 10% of training data of Hindi reviews as the development data to tune the hyper-parameter of CNN. The stochastic gradient descent (SGD) over mini-batch is used to train the network and backpropagation algorithm (Hecht-Nielsen, 1989) is used to compute the gradients. The Adadelta (Zeiler, 2012) update rule is used to tune the learning rate. A brief details of hyper-parameters are listed in Table-5.

Parameter	Parameter Name	English CNN	Hindi CNN
d^x	Word embedding dimension	300	200
n	Maximum length of comments	50	50
m	Filter window sizes	3,4,5,6	3,4,5,6
$\mathbf{c}$	Maximum feature maps	100	100
r	Dropout rate	0.5	0.5
e	Maximum epochs	50	60
mb	Mini-batch size	50	50
λ	Learning rate	0.2	0.2

Table 5: Neural network hyper-parameters

9 Results and Discussion

Results of our proposed model based on deep CNN are shown in Table-3. The first baseline model which is based on cosine similarity measure does not perform well. We observe that this is biased towards a specific class. However, the second baseline performs well to identify the opinions for both Hindi and English. Performance of our proposed CNN based model is encouraging for both the languages. We experiment with different filter sizes for both the languages. The effect of varying window size is shown in Fig-2. From further analysis of the system outputs, we come up with the following observations:

- Performance of the proposed model for Hindi is not at par with English. One possible reason could be the less amount of data that we use to train the network. The amount of data on which the word embedding model is trained is also less for Hindi.
- Results show that, for Hindi, the system performs better for the negative opinion. This may be due to the un-equal distribution (1035 vs.559 of negative vs. positive) of training instances.
- Model performance depends upon the size of filter window being used to train the network. It is clearly shown in Fig-2. The system performs well with window size combination {3,4,5} for Hindi. The best suited window size(s) for English is {4,5,6}.

9.1 Error Analysis

We analyze the system outputs to understand the shortcomings of the proposed system. We observe the following:

1. We observe that many errors were due to incorrect classification of positive comments

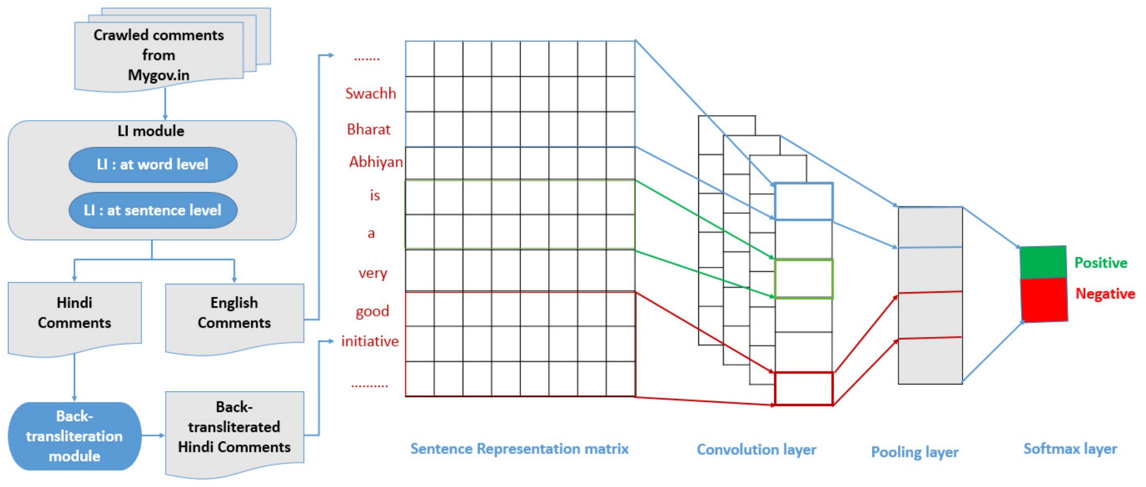

Figure 1: Proposed system architecture

Model	English Data				Hindi Data				Overall			
	R	P	F	Acc	R	P	F	Acc	R	P	F	Acc
Baseline-1	58.75	70.89	50.01	56.06	51.05	54.73	36.77	44.71	54.90	62.81	43.38	50.38
Baseline-2	67.21	67.73	67.25	67.82	57.70	70.5	54.30	57.72	62.45	69.11	60.77	62.77
Code-mixed-CNN	71.90	71.80	71.66	71.68	65.21	68.71	60.68	61.58	68.56	70.25	66.17	66.63

Table 3: Results of baselines and the proposed model. The best results obtained from the window size combination $\{3,4,5\}$ with feature maps size 100 are reported here. Here, **R** : Recall, **P**: Precision, **F**: F-score and **Acc**: Accuracy

Language	Train			Test		
	Sources	# Positive instances	# Negative instances	Sources	# Positive instances	# Negative instances
English	IMDB movie Review(Maas et al., 2011)	2000	2000	Our crawled data	279	240
Hindi	Tourism Review(Joshi. et al., 2010)	98	100	Our crawled data	282	210
	Movie Review(Joshi. et al., 2010)	127	125			
	SAIL(Patra et al., 2015)	334	810			
	Total	*559*	*1035*			

Table 4: Training and test data statistics used in our experiments

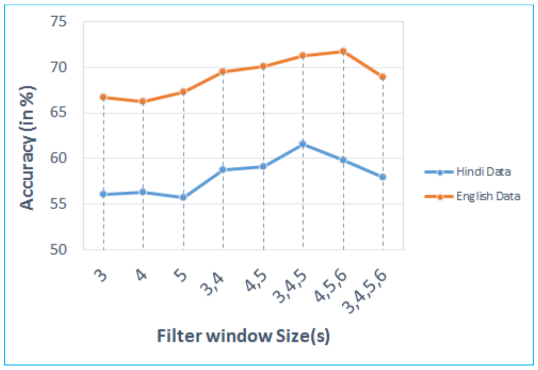

Figure 2: Effect of varying window size on accuracy in CNN model on both languages

into negative. The possible reason is that the comments seem to have conflicting opinions. For e.g., *"The scheme is fantastic but it would not be successful until polybag get banned...."*

2. Some positive comments were mis-classified as negative due to the presence of strong negative triggeres in context of suggestion, but not in the context of government scheme[8]. For e.g.
" इस अभियान से बहुत खुश हूँ पर आप ऐसा कानून लाईये जिससे जो गन्दगी करे उसे ***पैसे भरने*** *पड़े और* ***दंड*** *हो "*
(*very happy with the campaign but implement a law so that those who litter get* ***penalized*** *and* ***punished***.) In this example, opinion is positive about the scheme, but user has given some suggestions with negative opinion bearing words such as **पैसे भरने** and **दंड** .

3. A number of errors were also due to annotations of suggestive comments as positive. Our proposed approach often fails to classify the suggestive comments properly due to the presence of negative triggers such as *should not, should also, does no mean, very few,*

[8]Suggestion is given with respect to any government scheme.

but, at least, should be added etc.. "Swach bharat abhiyaan ***should also*** *include cleaning of polluted rivers"*
(iv). The model could not perform well to classify the comments which are conditional in nature. For e.g., *"unless we make judicial system fair, fast, economical and secure we won't see good days I bet my life on it."* System predicts this comment as positive, but actual opinion should be negative.

10 Conclusion

In this paper we propose a deep learning based opinion miner that could be beneficial for urban informatics, where the goal is to work for the betterment of human lives through social media networks. We build this model based on the user comments crawled from *M*ygov.in portal where users write comments on various topics related to govt schemes. We create our own resources in order to build the model. One of the major challenges was to handle the code-mixed cases where the language of more than one script is mixed. We have developed algorithms to solve three crucial problems, *viz.* language identification, machine transliteration and opinion mining. Experiments on Hindi and English show encouraging results for the tasks. Further we would like to enhance the size of the corpus, build deep CNN models utilizing hand-crafted features and study the effectiveness of the proposed model.

Acknowledgment

We acknowledge Salam Amitra Devi (lexicographer, IIT Patna) for contributing in annotation of dataset.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Beatrice Alex. 2007. *Automatic detection of English inclusions in mixed-lingual data with an application to parsing*. Ph.D. thesis, University of Edinburgh.

Peter Auer. 2013. *Code-switching in conversation: Language, interaction and identity*. Routledge.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

M Choudhury, G Chittaranjan, P Gupta, and A Das. 2014. Overview and datasets of fire 2014 track on transliterated search. In *Pre-proceedings 6th workshop FIRE-2014*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Anik Dey and Pascale Fung. 2014. A hindi-english code-switching corpus. In *LREC*, pages 2410–2413.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. *Proceedings of the COLING/ACL*, pages 191–198.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Deepak Kumar Gupta, Shubham Kumar, and Asif Ekbal. 2014. Machine learning approach for language identification & transliteration. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 60–64. ACM.

Robert Hecht-Nielsen. 1989. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE.

Taofik Hidayat. 2012. An analysis of code switching used by facebookers (a case study in a social network site). *Student essay for the study programme Pendidikan Bahasa Inggris (English Education) at STKIP Siliwangi Bandung, Indonesia*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Geoffrey E Hinton. 1984. Distributed representations.

Aditya Joshi., Balamurali A. R., and Pushpak Bhattacharyya. 2010. A fall-back strategy for sentiment analysis in a new language: a case study for hindi. In *International Conference on Natural Language Processing*, pages 1081–1091.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jacob Kohen. 1960. A coefficient of agreement for nominal scale. *Educ Psychol Meas*, 20:37–46.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

David Li. 2000. Cantonese-english code-switching research in hong kong: A y2k review. *World Englishes*, 19(3):305–322.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon*, pages 2326–2335. Citeseer.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lesley Milroy and Pieter Muysken. 1995. *One speaker, two languages: Cross-disciplinary perspectives on code-switching*. Cambridge University Press.

Braja Gopal Patra, Dipankar Das, Amitava Das, and Rajendra Prasath. 2015. Shared task on sentiment analysis in indian languages (sail) tweets-an overview. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 650–655. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

John Platt. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, April.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview and datasets of fire 2013 track on transliterated search. In *FIRE-13 Workshop*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3.

Hong Ka San. 2009. Chinese-english code-switching in blogs by macao young people.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.