Towards Sub-Word Level Compositions for Sentiment Analysis of Hindi-English Code Mixed Text

Ameya Prabhu [*1], Aditya Joshi [*2], Manish Shrivastava [3] and Vasudeva Varma [2]

[1]Centre for Visual Information Technology [2]Search and Information Extraction Lab [3]Language Technologies Research Center

International Institute of Information Technology, Hyderabad (India)

Overview

Code Mixed Text

Code Mixing is a natural phenomenon of embedding linguistic units such as phrases, words or morphemes of one language into an utterance of another. e.g. Text in Hindi written in roman alphabets

आपका दिन शुभ हो aapka din shubh ho

Sentiment Analysis has been widely explored in the past decade. There are state of the art solutions for individual languages producing results at par with human cognition. However progress in the area of Code Mixed text is still in it's initial stages. In this work, we approach this problem by experimenting with comparable approaches for sentiment analysis in individual language, and present our system which advocates the use of sub-word representations using Long Short Term Memory (LSTM) in our deep neural architecture.

Problem Definition

Given a piece of Hindi-English code mixed text from social media, determine the sentiment polarity. The text may contain noisy characters, english words, or hindi words transliterated into roman script (English character set).

Challenges

Contractions, morpheme stretching, non standard spellings

Multiple transliterations owing to non-phonemic nature of script

Dataset

We collected user comments on the posts on following public Facebook pages:

Narendra Modi, *Prime Minister of India*	38M likes	www.facebook.com/narendramodi
Salman Khan, *Indian Actor*	33M likes	www.facebook.com/BeingSalmanKhan

Data Cleaning and Annotation

- Remove sentences which are
  - (i) not in roman script,
  - (ii) longer than 50 words, or
  - (iii) were complete English sentences
- Each sentence is annotated by two annotators on a three polarity scale – positive, negative or neutral.
- Only the sentences marked with same sentiment polarity by both the annotators are considered for the experiment.

Total Sentences: 4981;

Considered for experiment: 3879;

Kappa Agreement: 0.64; 77% sentences

Sample Sentences in the Dataset

Example *(Meaning)*	**Sentiment Polarity**
Aisa PM na hua hai aur naa hee hoga *Neither there has been a PM like him, nor there will be any*	Positive
abe kutte tere se kon baat karega *Who would talk to you, dog?*	Negative
Trailer dhannnsu hai bhai *Trailer is awesome, brother*	Positive

Issues in processing code mixed text

Word	Meaning	Appearing Variations
बहुत (bahut)	very	bahout bohut bhout bauhat bohot bahut bhaut bahot bhot
मुबारक (mubaarak)	wishes	mobarak mubarak mubark
प्यार (pyaar)	love	pyaar peyar pyara piyar pyr piyaar pyar

Hypothesis

Our aim is to perform sentiment analysis on the curated dataset that contains code mixed text from public Facebook groups. Most commonly used statistical approaches learn word-level feature representations. Lexicon based approaches for the SA task perform a dictionary look up to obtain an individual score for words in a given sentence and combine these scores to get the sentiment polarity of a sentence. We propose a method of generating sub-word level representations through 1-D convolutions on character inputs for a given sentence.

Character-level RNNs (Char-RNNs) have recently become popular. They [6] do not have the limitation of vocabulary, hence can freely learn to generate new words. This freedom, in fact, is an issue: Language is composed of lexical units made by combining letters in some specific combinations. The complexity arises because the mappings between meaning and its construction from characters is arbitrary. Character models may be apriori inappropriate models of language as characters individually do not usually provide semantic information. For example, while **"King – Man + Woman = Queen"**, is semantically interpretable,
"Cat – C + B = Bat" lacks any linguistic basis.
But, groups of characters may serve semantic functions. E.g. **"Un + Holy = Unholy"**.

Since sub-word level representations can generate meaningful lexical representations and individually carry semantic weight, we believe that sub-word level representations consisting composition of characters might allow generation of new lexical structures and serve as better linguistic units than characters.

We propose using intermediate sub-word feature representations learned by the filters during convolution operation. Unlike traditional approaches that add sentiment scores of individual words, we propagate relevant information with LSTM and compute final sentiment of the sentence. Incorporating sub-word level representations into the design of our models should result in better performance. This would also serve as a test scenario for the broader hypothesis proposed by Chris Dyer in ICLR 2016 keynote [3] - Incorporating linguistic priors in network architectures lead to better performance of models.

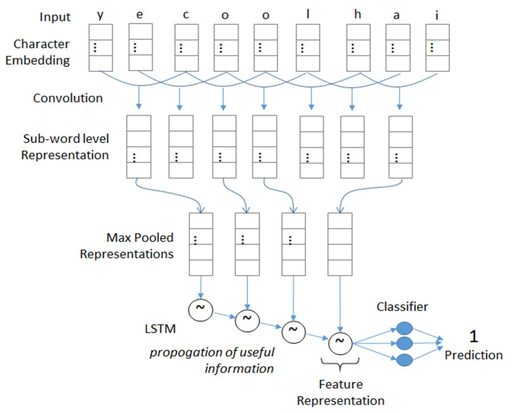

Figure 1. Illustration of the proposed methodology

Model and Discussion

For a sentence *S* made up of a sequence of characters *[c1, …, cL]*, the representation is given by the matrix $Q \in R^{d \times l}$ where *d* is the dimensionality of character embedding. We perform a convolution of Q with filter $H \in R^{d \times m}$, add a bias *b,* and apply a non linearity *g* to obtain a feature map given by: $f[i] = g((Q[:, i:i+m-1] * H) + b)$

Finally, we pool the maximal responses from p feature representations corresponding to selecting sub-word representations as: $y_i = max(f[p * (i : i + p - 1)])$

LSTM [1] is suited for learning to propagate and remember useful information, is applied, providing a sentiment vector representation for the input. We then compute values of activation of the forget gate, which can be used to compute the information stored in memory cell at time t. With It - the input gate, ~ Ct - the candidate value for the state of the memory cell at time t and ft - the the new state of memory cell Ct, we can compute the output feature representation by

$$O_t = \sigma(Wy_t + Uh_(t-1) + V(C_t + b)$$
$$h_t = O_t tanh(C_t)$$

where W,U and V are weight matrices and bi are biases. After l steps, hl represents the relevant information retained from the history. That is then passed to a fully connected layer which calculates the final sentiment polarity.

Visualizations in Figure 4 shows how the proposed model is learning to identify sentiment lexicons. We see that different filters generally tend to learn mappings from different parts, interestingly showing shifting trends to the right which maybe due to LSTM picking their feature representation in future time steps. The words sections that convey sentiment polarity information are captured despite misspelling in examples. In our dataset, the words which were sevely misspelt or stretched also show high activation.

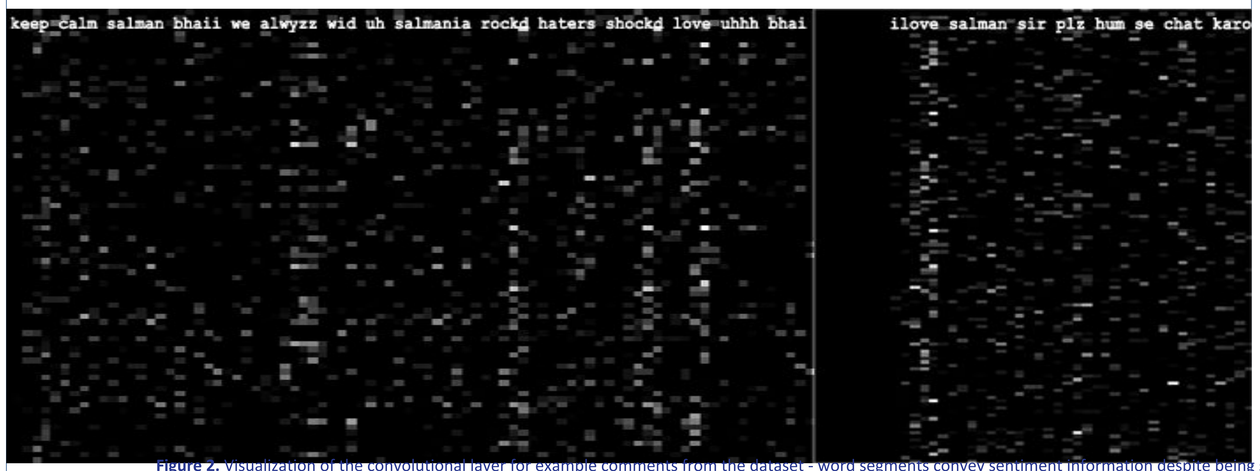

Figure 2. Visualization of the convolutional layer for example comments from the dataset - word segments convey sentiment information despite being severely misspelt

Experiments

Our dataset is divided into 3 splits- Training, validation and testing. We first divide the data into randomized 80-20 train test split, then further randomly divide the training data into 80-20 split to get the final training, validation and testing data.

The architecture of the proposed system (Subword-LSTM) is described in Figure 2. We compare it with a character-level LSTM (Char-LSTM) following the same architecture without the convolutional and maxpooling layers. We use Adamax (Kingma and Ba, 2014) (a variant of Adam based on infinity norm) optimizer to train this setup in an end-to-end fashion using batch size of 128. We use very simplistic architectures because of the constraint on the size of the dataset.

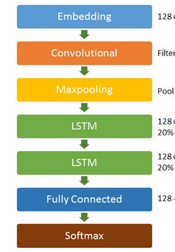

Figure 3. Architecture

We compared the result of standard sentiment analysis systems on our dataset. We observe that Multinomial Naive Bayes performs better than SVM [3] for snippets, providing additional validation to the hypothesis given by [4]. We also observe that unigrams perform better than bigrams and Bag of words performs better than tf-idf in contrast to trends in English, as the approaches inducing more sparsity would yield to poorer results because our dataset is inherently very sparse. The comparable previous work [5] utilizes on transliteration of the words to root language (here, Hindi) and uses a sentiment lexicon to compute the final polarity. However due to heavy misspelling and inconsistent code mix in social media text, this approach leads to incorrect transliterations, thus final results.

Method	**Reported In**	**Our Dataset**		**SemEval '13**	
		Accuracy	F1-Score	Accuracy	F1-Score
NBSVM (Unigram)	(Wang and Manning, 2012)	59.15%	0.5335	57.89%	0.5369
NBSVM (Uni+Bigram)	(Wang and Manning, 2012)	62.50%	0.5375	51.33%	0.5566
MNB (Unigram)	(Wang and Manning, 2012)	66.75%	0.6143	58.41%	0.4689
MNB (Uni+Bigram)	(Wang and Manning, 2012)	66.36%	0.6046	58.40%	0.469
MNB (Tf-Idf)	(Wang and Manning, 2012)	63.53%	0.4783	57.82%	0.4196
SVM (Unigram)	(Pang and Lee, 2008)	57.60%	0.5232	57.60%	0.5232
SVM (Uni+Bigram)	(Pang and Lee, 2008)	52.96%	0.3773	52.90%	0.3773
Lexicon Lookup	(Sharma et al., 2015)	51.15%	0.252	N/A	N/A
Char-LSTM	Proposed	59.80%	0.511	46.60%	0.332
Subword-LSTM	Proposed	69.70%	0.658	60.57%	0.537

Table 1. Results and comparision with other approaches

Conclusions

We introduce Sub-Word Long Short Term Memory model to learn sentiments in a noisy Hindi-English Code Mixed dataset. We discuss that due to the unavailability of NLP tools for Hi-En Code Mixed text and noisy nature of such data, several popular methods for Sentiment Analysis are not applicable. The solutions that involve unsupervised word representations would again fail due to sparsity in the dataset.

Sub-Word LSTM interprets sentiment based on morpheme-like structures and the results thus produced are significantly better than baselines.

Future Directions

Further work should explore the effect of scaling of RNN and working with larger datasets on the results. We would like to explore more deep neural network architectures that are able to capture sentiment in Code Mixed and other varieties of noisy data from the social web.

Meanwhile we also promote the need of curation of such datasets in mostly used code-mixing language combinations that would lead to further advancement in the area of language understanding.

Presenter Contact Information

Ameya Prabhu
IIIT-Hyderabad
Email: ameya.prabhu@research.iiit.ac.in
Website: researchweb.iiit.ac.in/~ameya.prabhu/
Phone: +91-8142060275

References

1. Alex Graves. 2013. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850.
2. http://videolectures.net/iclr2016_dyer_model_architecture/
3. Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. Found. Trends Inf. Retr., 2(1-2):1–135, January.
4. Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the ACL: Short Papers - Volume 2, ACL '12, Pages 90–94, Stroudsburg, PA, USA.
5. Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In 2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015
6. Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. CoRR, abs/1508.06615