



Deep Learning based Language Identification in Code-mixed Social Media Corpora

- Group 9




Objective:

- Word level language identification in indian social media corpora taken from facebook, twitter and whatsapp that exhibit code mixing in english-hindi and english-telugu languages as well as blend of both the language pairs.
- The performance of deep learning on this task is compared to feature-based learning ie., recursive neural networks techniques Bi-LSTM is contrasted to CRF classifier.
- The results show the deep learners outscoring the CRF, with the bidirectional LSTM demonstrating the best language identification performance.



Corpus data collection and statistics

- Code mixed dataset for language identification is taken from 12th international conference on nlp ICON-2015.
- Data is collected from leading social-media websites like facebook, whatsapp, twitter.
- The dataset is comprised of facebook posts and comments, tweets and whatsapp chat conversations
- The datasets consists of 70,333 words. This dataset contains **5913** code-mixed sentences. These sentences are tokenized into words. And the tokenized words of each sentence are separated by new line.

- 
- EN-HI data corpus

| Language Label | Label Frequency |
|----------------|-----------------|
| English | 17308 |
| Hindi | 15171 |
| Universal | 7263 |
| Named entity | 1104 |
| Other labels | 294 |

- EN-TE data corpus

| Language Label | Label Frequency |
|----------------|-----------------|
| English | 8812 |
| Telugu | 8757 |
| Universal | 10805 |
| Named entity | 743 |
| Other labels | 5 |

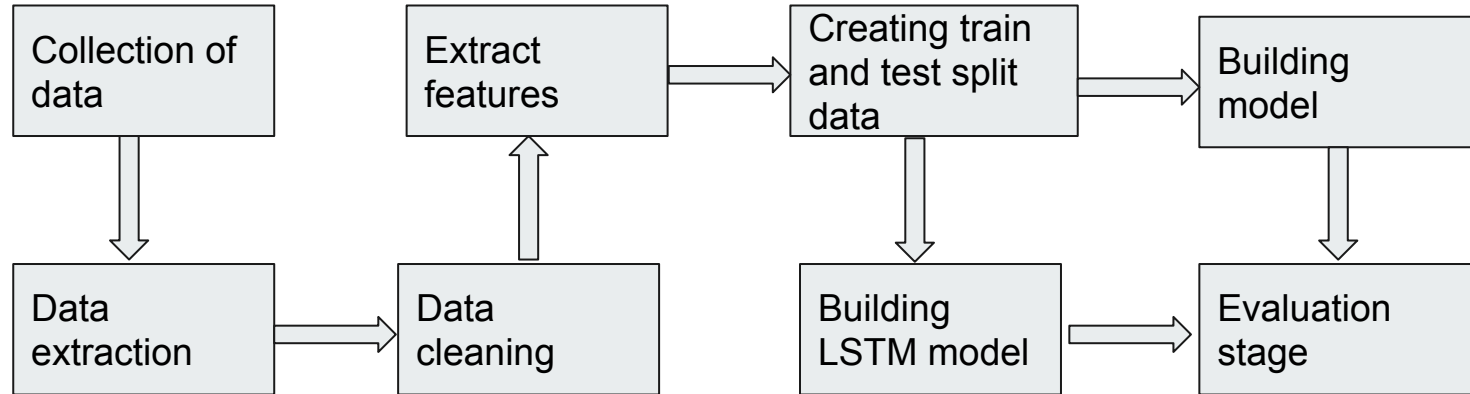
- Out of total data 30% data is kept for testing and remaining data is used for training the model.



Each word is annotated with a language identification tag which are :

| | |
|-------|--------------|
| en | english |
| hi | hindi |
| univ | universal |
| acro | acronym |
| mixed | Mixed data |
| ne | Named entity |
| te | telugu |
| undef | undefined |

Model Pipeline






Conditional Random Fields

- CRF is a simple, customizable and open source implementation.
- CRF model is implemented using sklearn-crfsuite on our data set.
- It is used for labeling of sequential data such as nlp.
- CRF applications are more noted in Name entity recognition, POS tagging etc.,
- We will use sklearn-crfsuite to train a CRF model on our training data
- Data is retrieved as messages containing details of word, label and postag
- For training a CRF model input should be a dictionary of features for every word



Feature Extraction

- Features are extracted like word parts, simplified POS tags, lower/title/upper flags, features of nearby words and converted them to sklearn-crfsuite format - each sentence is converted to a list of dicts.
- For this model, features used are:
 - Current word and its POS tag: We consider the current testing word and its POS tag to predict the language label.
 - BOS: if current word is beginning of sentence
 - EOS: if current word is ending of sentence

- 
- Next word and its POS tag: We define the next word and its POS tag to capture the relation between current and next word if current word is not EOS.
 - Previous word and its POS Tag: To extract the context of current word, we took the previous word and its POS tag if current word is not BOS.
 - Words with upper letters: this feature is TRUE if string contains only upper case letters otherwise it is FALSE.
 - Numeric digits: whether whole string is numeric or not.
 - Title case: this feature is TRUE if string starts with upper case otherwise FALSE.
 - For testing purpose dataset is divided into 7:3 ie., 30% data is for testing and 70% data is for training.



Training and Evaluation

- CRF model is built using sklearn-crfsuite and default algorithm L-bfgs is used.
- `c1`, `c2` are set to 0.1 value which are coefficients of L1,L2 regularizations.
- The model is set to do 100 iterations for optimization
- `all_possible_transitions` is set to `TRUE` so that CRFsuite generates transition features that associate all of possible label pairs
- Flat classification report from metrics is used to provide precision and F1-score for all labels respectively.



Bi-LSTM- deep learning approach

- A recurrent neural network (RNN) is a deep learning approach where the recurrent connections are loops in the network that allow it to maintain a memory based on history information, enabling the model to predict the current output conditioned on long-distance features.
- Bi-lstm is implemented using Tensorflow and Keras
- The proposed model for deep learning was trained using word embeddings.



Word Level Indexing

- Data retrieved as messages with words, labels and postag, in CRF classifier, is also used here.
- Words, a list of all words from dataframe is retrieved and a unique word “ENDPAD” is appended at last.
- Tags, a list of all tags from dataframe is retrieved.
- Indices are allotted to words and tags and saved in a dictionary
- Max length in all messages is retrieved and input and output messages are padded with indices of “ENDPAD” and “undef” respectively.
- Padding is necessary as model cannot take messages with different lengths.



Building Bi-Lstm model

- For testing purpose dataset is splitted into 7:3 ie., 70% data is used for training purpose and 30% data is used for testing purpose.
- Training and testing output data is converted into one-hot encoding.
- Layers are defined using keras with tensorflow background.
 - Input layer with shape=(max length,)
 - Embedding layer
 - SpatialDropout1D with dropout rate of 1%
 - Bidirectional(LSTM()) with 100 units and a recurrent dropout of 1%
 - Timedistributed(Dense()) layer with softmax activation
 - Model layer with defined Input layer as inputs and output of Dense layer as outputs.



Compiling and Fitting the model

- Model is compiled with categorical-cross-entropy loss and adam optimizer and accuracy as metrics.
- Model is fitted with
 - Input training data and one-hot encoding of output training data.
 - Batch size of 32 with 3 epochs
 - Plot Losses Keras from livelossplot is given as callbacks for model to visualize accuracy and loss of training data.
- Finally, the model is evaluated and accuracy is noted.
- Predicted data from model is converted to integers from one-hot encoding using argmax from numpy.
- Flat classification report from metrics is used to provide precision and F1-score for all labels respectively which is used to compare with CRF classifier.



Results

(A) Precision and F1 scores for **English- Hindi** dataset

| | Precision/Accuracy | F1 score |
|---------|--------------------|----------|
| CRF | 92% | 92% |
| Bi-LSTM | 89% | 89% |



Results

(B) Precision and F1 scores for **English- Telugu** dataset

| | Precision/Accuracy | F1 score |
|---------|--------------------|----------|
| CRF | 83% | 83% |
| Bi-LSTM | 93% | 93% |



Results

(C) Precision and F1 scores for **English-Hindi- Telugu** dataset

| | Precision/Accuracy | F1 score |
|---------|--------------------|----------|
| CRF | 88% | 88% |
| Bi-LSTM | 95% | 95% |



Observations

- CRF classifier outperform LSTM only on English- Hindi dataset
- The proposed LSTM approach improves on the CRF-based approach by incorporating pre-trained word embeddings instead of learning those from data, allowing the classifiers to learn in an unsupervised setting.



Conclusion

- Deep learning-based approach to the task of language identification is achieved at a reasonable accuracy levels compared to a supervised approach such as CRF.



Related work

- Deep Learning-Based Language Identification in English-Hindi-Bengali Code-Mixed Social Media Corpora <https://doi.org/10.1515/jisys-2017-0440>
- Part-of-Speech Tagging for Code-Mixed English-Hindi Twitter and Facebook Chat Messages
- Collecting and Annotating Indian Social Media Code-Mixed Corpora
[10.1007/978-3-319-75487-1_32](https://doi.org/10.1007/978-3-319-75487-1_32)
- Word level language identification for english-telugu code mixed data - *KCIS IIT Hyd*
- POS Tagging For Code-Mixed Indian Social Media Text : Systems from IIIT-H for ICON NLP Tools Contest