Highlights

Hinglish Language Sarcasm Detection System - Performance of Various Embedding and Models

Hari Thapliyal

- Created 109 Hinglish Language Sarcasm Detection Models using different embedding and classifiers.
- Performance Evaluation of 109 Hinglish Language Sarcasm Detection Models.

Hinglish Language Sarcasm Detection System - Performance of Various Embedding and Models

Hari Thapliyal[a,a] (Researcher)

[a] dasarpAI, Lord Krishna Green, Doon University Road, Dehradun, 248001, Uttrakhand, India

ARTICLE INFO

Keywords:
Hinglish Language Text,
Sarcasm Detection in Hinglish Language,
Embedding for Hinglish Language,
NLP for Hinglish Language

Abstract

Hindi is third[1] most spoken language on our planet. Like English which is written in Roman script, Hindi also does not have its own script but almost all the Hindi speaking people write Hindi in Devanagari script. Hinglish is a mix language and it is spoken by Hindi speaking, English educated people and they can add words from other Indian languages during their conversation. Unlike Hindi Hinglish has its own script and this script is called Hinglish script. Hinglish script has characters borrowed characters from Roman and Devanagari scripts. (WikipageA) states that 65% of Indian population is under 35 years age. Several disruptions like low cost mobile phone, extremely cheap data, digital India initiatives by government of India has caused huge surge in Hinglish language content. Hinglish language context is available in audio, video, images, and text format. We can find Hinglish content in comment box of online product, news articles, service feedback, WhatsApp messages, social media like YouTube, Facebook, twitter etc. With the increasing number of education and sophisticated people in Indian society it is obvious that people do not say negative things directly even when they want to say. Generally, an educated mind is more diplomatic than less educated. In this paper we are demonstrating a system which can help in automatic sarcasm detection in Hinglish language. In this work no word, either Indian language words written in Roman or English word written in Devanagari is translated or transliterated. We developed our dataset with the help of 3 Hinglish language speakers. In this work we used ten classification libraries for classification work and developed 109 classification models, including 4 classification models developed using neural network. We analysed the performance of those models against the embedding and classifier used. Our best model with fastTextWiki embedding and Naïve Bayesian classifier gives 76% accuracy, 78% recall, 75% precision, 76% F1 score and 80% AUC.

1. Architecture, Parameters of Classifier & Embedder

We created 109 models using 4 Neural Network Architectures, 4 Word Embedding without Transfer Learning, 4 Embedding with Transfer Learning, 2 Mix Embedding, 8 Classifiers, 5 Classifier Using Task Transfer. Architecture created, parameters used for creating embedding, parameters used for creating classifiers, features created will be discussed in this section.

1.1. Neural Network Architecture

1.1.1. CNN Architecture without Transfer Learning

```
embedding\_dim = 200
sent\_size =119
batch\_size=100
cnnmodel = Sequential()

#embedding layer
cnnmodel.add(layers.Embedding(vocab\_size,
    embedding\_dim, input\_length=sent\_size))
#CNN layer
cnnmodel.add(layers.Conv1D(128, 5,
activation='relu'))
cnnmodel.add(layers.GlobalMaxPooling1D())

#FC layer
cnnmodel.add(layers.Dense(10,
activation='relu'))
cnnmodel.add(layers.Dense(1,
activation='sigmoid'))

#Add loss function, metrics, optimizer
cnnmodel.compile(optimizer='adam',
              loss='binary\_crossentropy',
              metrics=['accuracy'])
```

1.1.2. CNN with Transfer Learning from fastText_Wiki

```
vocab\_size= 9156
Weights = Weight of above vocabulary from
fastText\_Wiki finetuned model
Dim=300
input\_length = 119
Remaining architecture same as above.
```

hari.prasad@vedavit-ps.com (H. Thapliyal)
www.dasarpai.com (H. Thapliyal)
orcid(s): 0000-0001-7907-865X (H. Thapliyal)

1.1.3. CNN with Transfer Learning from IndicFT

All parameters remained same as above. Weights are taken from IndicFT finetuned model.

1.1.4. RNN Architecture without Transfer Learning

```
embedding\_dim = 200
sent\_size =119
batch\_size=100
rnnmodel=Sequential()

#embedding layer
rnnmodel.add(layers.Embedding(
vocab\_size, embedding\_dim,
input\_length=sent\_size))

#lstm layer
rnnmodel.add(LSTM(128,return
\_sequences=True,dropout=0.2))

#Global Maxpooling
rnnmodel.add(GlobalMaxPooling1D())

#Dense Layer
rnnmodel.add(Dense(64,activation='relu'))
rnnmodel.add(Dense(1,activation='sigmoid'))

#Add loss function, metrics, optimizer
rnnmodel.compile ( optimizer='adam',
    loss='binary\_crossentropy',
    metrics=["acc"])

#Adding callbacks
es = EarlyStopping(monitor='val\_loss',
mode='min', verbose=1,patience=3)

mc=ModelCheckpoint( 'best\_model.h5',
    monitor='val\_acc',  mode='max',
    save\_best\_only=True, verbose=1)
```

1.2. Parameters – Word Embedding without Transfer Learning

1.2.1. TFIDF

```
Vectorizer: TfidfVectorizer
Parameters: max\_features=300,
    ngram \_range=(1,2)
    pca = PCA(n\_components=200)

Final embedded dataset has 200 features
Dimension: 200
```

1.2.2. BOW

```
Vectorizer : CountVectorizer
Parameters: max\_features=300,
     ngram\_range=(1,2)
     pca = PCA(n\_components=200)

Final embedded dataset has 200 features
Dimension: 200
```

1.2.3. Word2VEC

```
Vectorizer: Word2vec
Parameters: feature\_size=15,
     window\_context=20,
     min\_count = 1,
     sg=1,
     sample= 1e-3,
     iter=5000
     Dimension: 15
Final embedded dataset has 15 features
```

1.2.4. fastText

```
Vectorizer: gensim.models.fasttext
import FastText
Tokenizer Params:
    feature size=50, # Word vector dimensionality
    window\_context=20, # Context window size
    min\_word\_count = 1 # Minimum word count
    skip gram=1,  # skip-gram model
    sample=1e-3,
    #Downsample setting for frequent words
    iter=5000
Final embedded dataset has 50 features
```

1.3. Parameters - Word Embedding from Transfer Learning

1.3.1. IndicBERT

```
Tokenizer: ai4bharat/indic-bert
    Tokenizer Params: tokenizer.encode\_plus(text,
    add\_special\_tokens=True,
    max\_length=200) ["input\_ids"]
    Dimension: 768
```

1.3.2. mBERT

```
Tokenizer: bert-base-multilingual-uncased
    Tokenizer Params: tokenizer.encode\_plus(text,
    add\_special\_tokens=True,
    max\_length=200) ["input\_ids"]
    Dimension: 768
```

1.3.3. IndicFT

```
Pretrained vector: indicnlp.ft.hi.300.vec
Tokenizer Params :
    fasttext.train\_supervised (train\_file,
 lr=0.5, epoch=25, wordNgrams=2,
 bucket=200000, pretrainedVectors,
 dim=300)
```

1.3.4. fastText_wiki

```
Pretrained vector: wiki.hi.300.vec
Tokenizer Params:
    fasttext.train\_supervised (train\_file,
 lr=0.5, epoch=25, wordNgrams=2,
 bucket=200000, pretrainedVectors,
 dim=300)
```

1.4. Parameters - Features from Other Techniques

1.4.1. Lexical Features

Manually created 33 lexical features. Following POS are used to create lexical features ['NOUN', 'ADP', 'VERB', 'AUX', 'PRON', 'PROPN', 'PART', 'DET', 'PUNCT', 'ADJ', 'SCONJ', 'CCONJ', 'NUM', 'ADV', 'INTJ', 'X'] POS Features: 16 POS features, Number of these POS in a sentence. POS Presence Features: 16 Binary features of above POS (whether these POS are present in the sentence 0/1) number of words

1.4.2. Combined

The best performing embedding transfer combined with above created lexical features. This embedding contains features from IndicFT & Lexical. Dimension: 333

1.5. Parameters - Classifiers

1.5.1. Logistic Regression – LR

LogisticRegression (C=.01, max_iter=1000, random_state=100)

1.5.2. Light Gradient Boost Machine – LGBM

```
lgbm.LGBMClassifier(colsample_bytree=1.0,
   importance_type='split', learning_rate=0.1,
   max_depth=-1, min_child_samples=20,
   min_child_weight=0.001, min_split_gain=0.0,
   n_estimators=100, n_jobs=-1, num_leaves=31,
   objective=None,
   random_state=100, reg_alpha=0.0,
   reg_lambda=0.0, silent=True,
   subsample=1.0, subsample_for_bin=200000,
   subsample_freq=0)
```

1.5.3. Naïve Bayesian – NB

Default Parameters

1.5.4. Support Vector Classifier – SVC

Default Parameters

1.5.5. AdaBoost – ADB

Default Parameters

1.5.6. Gradient Boost Machine – GBM

```
GradientBoostingClassifier(ccp\_alpha=0.0,
criterion='friedman\_mse', init=None,
learning\_rate=0.1,
loss='deviance', max\_depth=3,
max\_features=None, max\_leaf\_nodes=None,
min\_impurity\_decrease=0.0,
min\_impurity\_split=None,
min\_samples\_leaf=1,
min\_samples\_split=2,
min\_weight\_fraction\_leaf=0.0,
n\_estimators=100,
n\_iter\_no\_change=None,
presort='deprecated',
random\_state=100, subsample=1.0,
tol=0.0001,
validation\_fraction=0.1, verbose=0,
warm\_start=False)
```

1.5.7. Random Forest Classifier – RFC

Default Parameters

1.5.8. Perceptron

Default Parameters

1.6. Parameters - Task Transfer

1.6.1. mBERT- Transformer

```
Tokenizer: bert-base-multilingual-uncased
Params of Token: pad\_sequences( list (map
(bert\_tokenizer.convert\_tokens\_to\_ids,
train\_tokens)),
maxlen=256, truncating="post",
padding="post", dtype="int")

Model: bert-base-multilingual-uncased
Parameter of Model :
BertTransformer(bert\_tokenizer\_multi,
bert\_model\_multi, max\_length=200)
```

1.6.2. mBERT – Pytorch

```
Tokenizer: bert-base-multilingual-uncased
Params of Tokenizer:
list(map(lambda t: ['[CLS]'] +
Tokenizer.tokenize(t)[:255]
+ ['[SEP]'], X))
tokens\_ids = pad\_sequences
(list(map(bert\_tokenizer.convert
\_tokens\_to\_ids, Tokens)),
maxlen=256, truncating="post",
padding="post", dtype="int")

Model: bert-base-multilingual-uncased
Params of Model: Batch\_Size: 2,
Ephoch=10, dropout=10\%,
add\_special\_tokens=True,
max\_length=self.max\_length
```

1.6.3. IndicBERT

```
Tokenizer: ai4bharat/indic-bert
Params of Tokenizer:
tokenizer.encode\_plus(text,
add\_special\_tokens=True,
max\_length=200) ["input\_ids"]

Model: ai4bharat/indic-bert
```

1.6.4. IndicFT

```
Pretrained vector: indicnlp.ft.hi.300.vec
Params of ftmodel\_indicnlp300\_vec  =
fasttext.train\_supervised(train\_file,
 dim=300, lr=0.5,
 epoch=25, wordNgrams=2,
 bucket=200000,pretrainedVectors)
```

1.6.5. fastText_wiki

```
  Pretrained vector: wiki.hi.300.vec
    Params of ftmodel\_wiki300\_vec =
    fasttext.train\_supervised(train\_file,
dim=300, lr=0.5,
epoch=25,wordNgrams=2,
bucket=200000, pretrainedVectors)
```

2. Results & Evaluation

2.1. Evaluation & Sampling Methods of Results

In all our experiments we calculated five metrics namely Accuracy, Recall, Precision, F1 Score and AUC score. But all the tables are shorted based on the accuracy score, highest to lowest. Although we displayed all the results but on accuracy score we considered top 2 performing model as the best models.

Across all the 109 models developed using Task Transfer, different kind of embedding and classifiers used the best Accuracy score is 76% with Naïve Bayesian Classifier when fastTextWiki embedding are used. AUC score is another good metrics to compares models. AUC score is free from thresholds, which is used to optimize the model performance. The best AUC score is 81% which we get when SVC (linear) is used with fastTextWiki transferred embedding. Lexical features could not demonstrate results in top 10. One interesting thing to note is when fastTextWiki embedding, the best embedding, is combined with Lexical features then overall accuracy of the model drops 2%. The fastText embedding, which is created using fastText library and without transfer learning, is one of the worse performer when used with NB. Perceptron classifier does not work good no matter what kind of embedding are used.

Classifier	Embedding Name	AUC	Acc	Recall	Prec.	F1
NB	fastTextWiki	0.80	0.76	0.78	0.75	0.76
TT	fastTextWiki	0.81	0.76	0.71	0.79	0.75
NB	IndicFT	0.77	0.74	0.70	0.76	0.73
LR	IndicFT	0.78	0.74	0.70	0.75	0.73
SVC	IndicFT	0.79	0.74	0.71	0.76	0.73
ADB	IndicFT	0.79	0.74	0.72	0.76	0.74
XGB	IndicFT	0.79	0.74	0.70	0.76	0.73
NB	Combined	0.79	0.74	0.76	0.74	0.75
PyrotchTT	mBERT	0.80	0.74	0.69	0.76	0.72
SVC	fastTextWiki	0.81	0.74	0.67	0.79	0.72

Table 1: Top 10 Best Models

Classifier	Embedding Name	AUC	Acc	Recall	Prec.	F1
ADB	mBERT	0.56	0.53	0.67	0.52	0.59
Perceptron	Word2Vec	0.52	0.52	0.47	0.52	0.49
NB	mBERT	0.55	0.52	0.30	0.53	0.38
DT	BOW	0.56	0.52	0.51	0.53	0.52
NB	BOW	0.58	0.52	0.39	0.52	0.45
Perceptron	mBERT	0.50	0.50	0.24	0.51	0.33
Perceptron	Lexical	0.50	0.50	0	0	0
Perceptron	Combined	0.50	0.50	0.01	0.50	0.02
NB	Word2Vec	0.64	0.50	0.95	0.50	0.66
NB	fastText	0.66	0.50	0.95	0.50	0.66

Table 2: Bottom 10 Worse Models

2.2. Evaluating Task Transferred Models

In task transfer learning experiments, we used four base models and fined tuned classification task using our train data. The fastTextWiki task transfer model gives the best accuracy 76%. We had an interesting observation that IndicBERT & mBERT are giving bad results when we use them for direct task transfer. This is more interesting because BERT is one of the best performing models for classification task in English language and IndicBERT is tuned for Hindi language. However, with different set of parameters when we try mBERT on GPU machine with Pytorch implementation then performance improves significantly from 58% accuracy to 74% accuracy.

Embedding Name	AUC	Acc	Recall	Prec	F1
fastTextWiki	0.81	0.76	0.71	0.79	0.75
mBERT (Pytorch)	0.80	0.74	0.69	0.76	0.72
IndicFT	0.81	0.74	0.71	0.76	0.74
mBERT (Transformer)	0.60	0.58	0.65	0.57	0.61
IndicBERT (Transformer)	0.61	0.58	0.63	0.57	0.6

Table 3: Metrics for Task Transfer Models

2.3. Evaluating Embedding & Classifier based on Average Metrics of Models

When we average out the performance of embedding for all the classifiers, we find those models which has IndicFT embedding are giving best Accuracy & F1 score. On the other hand if we average out performance of the classifiers for all the embedding we find RNN gives the best accuracy of 68%.

Embedding	Avg AUC	Avg Acc	Avg Recall	Avg Prec.	Avg F1
IndicFT	0.77	0.72	0.67	0.75	0.71
Keras_Tokenizer	0.74	0.68	0.68	0.69	0.68
fastTextWiki	0.76	0.69	0.64	0.73	0.67
Word2Vec	0.64	0.59	0.74	0.58	0.64
fastText	0.64	0.58	0.75	0.57	0.64
Combined	0.74	0.68	0.59	0.70	0.62
IndicBERT	0.64	0.61	0.59	0.62	0.59
TFIDF	0.63	0.59	0.58	0.60	0.59
BOW	0.63	0.59	0.55	0.60	0.57
Lexical	0.67	0.62	0.56	0.58	0.56
mBERT	0.60	0.57	0.58	0.57	0.56

Table 4: Best Embedding - Average (All Metrics & Classifiers)

Classifier	Avg AUC	Avg Acc	Avg Recall	Avg Prec.	Avg F1
RNN	0.74	0.68	0.7	0.67	0.68
CNN	0.73	0.66	0.68	0.66	0.67
RFC	0.72	0.65	0.71	0.65	0.67
SVC	0.7	0.65	0.68	0.65	0.66
GBC	0.69	0.65	0.65	0.66	0.65
XGB	0.7	0.64	0.63	0.65	0.64
LGBM	0.7	0.65	0.63	0.65	0.64
LR	0.69	0.64	0.64	0.64	0.64
ADB	0.67	0.63	0.63	0.63	0.63
DT	0.62	0.6	0.66	0.6	0.63
NB	0.67	0.6	0.64	0.61	0.6
Perceptron	0.56	0.56	0.36	0.55	0.41

Table 5: Best Classifier - Average (All Metrics & Classifiers)

2.4. Evaluating Transferred Embedding

As mentioned earlier, we have used 5 transfer embedding techniques. On fastTextWiki embedding NB gives the best accuracy 76%. On IndicFT embedding NB gives the best accuracy 74%. Nor classifier could give more than 62% accuracy on mBERT embedding. The best accuracy of IndicBERT embedding is with RFC classifier, 70% accuracy.

Classifier	AUC	Acc	Recall	Prec	F1
NB	0.80	0.76	0.78	0.75	0.76
TT	0.81	0.76	0.71	0.79	0.75
SVC	0.81	0.74	0.67	0.79	0.72
LR	0.81	0.72	0.66	0.75	0.70
XGB	0.78	0.71	0.64	0.74	0.69
RFC	0.79	0.71	0.65	0.74	0.69
LGBM	0.78	0.70	0.63	0.72	0.67
ADB	0.79	0.70	0.65	0.73	0.69
GBC	0.78	0.69	0.62	0.72	0.67
CNN	0.74	0.65	0.74	0.63	0.68
Perceptron	0.63	0.63	0.36	0.78	0.49
DT	0.64	0.63	0.63	0.63	0.63

Table 6: Embedding Transfer - fastText Wiki

Classifier	AUC	Acc.	Recall	Prec.	F1
NB	0.77	0.74	0.70	0.76	0.73
LR	0.78	0.74	0.70	0.75	0.73
SVC	0.79	0.74	0.71	0.76	0.73
ADB	0.79	0.74	0.72	0.76	0.74
XGB	0.79	0.74	0.70	0.76	0.73
TT	0.81	0.74	0.71	0.76	0.74
DT	0.71	0.72	0.61	0.77	0.68
Perceptron	0.72	0.72	0.56	0.81	0.66
LGBM	0.79	0.72	0.67	0.74	0.7
GBC	0.79	0.72	0.67	0.75	0.71
RFC	0.79	0.72	0.68	0.75	0.71
CNN	0.71	0.66	0.65	0.66	0.66

Table 7: Embedding Transfer- IndicFT

Classifier	AUC	Acc.	Recall	Prec.	F1
DT	0.63	0.62	0.65	0.61	0.63
SVC	0.63	0.60	0.66	0.59	0.63
GBC	0.64	0.60	0.65	0.60	0.62
RFC	0.64	0.60	0.65	0.59	0.62
LR	0.61	0.58	0.68	0.57	0.62
LGBM	0.63	0.58	0.64	0.57	0.60
XGB	0.61	0.57	0.62	0.56	0.59
ADB	0.56	0.53	0.67	0.52	0.59
NB	0.55	0.52	0.30	0.53	0.38
Perceptron	0.50	0.50	0.24	0.51	0.33

Table 8: Embedding Transfer - mBERT

Classifier	AUC	Acc.	Recall	Prec.	F1
RFC	0.71	0.70	0.70	0.70	0.70
XGB	0.71	0.66	0.65	0.67	0.66
LGBM	0.69	0.64	0.58	0.67	0.62
GBC	0.68	0.62	0.58	0.63	0.60
DT	0.62	0.60	0.62	0.60	0.61
ADB	0.64	0.60	0.59	0.60	0.59
NB	0.61	0.59	0.67	0.58	0.62
LR	0.59	0.57	0.61	0.57	0.59
Perceptron	0.56	0.56	0.22	0.67	0.33
SVC	0.60	0.56	0.65	0.55	0.59

Table 9: Embedding Transfer- IndicBERT

Classifier	AUC	Acc.	Recall	Prec.	F1
NB	0.79	0.74	0.76	0.74	0.75
GBC	0.78	0.72	0.66	0.74	0.7
XGB	0.80	0.71	0.65	0.74	0.69
LGBM	0.78	0.70	0.63	0.72	0.67
ADB	0.78	0.70	0.64	0.74	0.68
LR	0.80	0.70	0.61	0.74	0.67
RFC	0.80	0.70	0.63	0.74	0.68
SVC	0.75	0.68	0.70	0.67	0.68
DT	0.63	0.63	0.63	0.63	0.63
Perceptron	0.50	0.50	0.01	0.50	0.02

Table 10: Embedding Transfer- Combined Embedding

2.5. Evaluating Non-Transferred Embedding

None of the non-transfer embedding techniques could deliver good results. The best non-transfer embedding technique is Lexical, which gives the best accuracy 66%. The fastText non-transfer embedding gives the best accuracy 64%.

Classifier	AUC	Acc.	Recall	Prec.	F1
LGBM	0.68	0.64	0.72	0.62	0.66
ADB	0.64	0.62	0.71	0.61	0.65
GBC	0.64	0.62	0.70	0.61	0.65
SVC	0.67	0.62	0.75	0.60	0.66
XGB	0.69	0.62	0.63	0.62	0.63
LR	0.64	0.61	0.76	0.58	0.66
DT	0.60	0.58	0.79	0.56	0.65
RFC	0.69	0.57	0.89	0.54	0.67
Perceptron	0.52	0.52	0.47	0.52	0.49
NB	0.64	0.50	0.95	0.50	0.66

Table 11: Word2Vec Embedding

Classifier	AUC	Acc.	Recall	Prec.	F1
GBC	0.63	0.62	0.57	0.64	0.60
SVC	0.68	0.62	0.58	0.64	0.61
Perceptron	0.60	0.60	0.61	0.60	0.60
LR	0.64	0.69	0.52	0.61	0.56
LGBM	0.66	0.60	0.54	0.62	0.58
RFC	0.66	0.60	0.56	0.61	0.58
DT	0.61	0.58	0.73	0.57	0.64
NB	0.60	0.56	0.53	0.56	0.54
ADB	0.60	0.56	0.55	0.56	0.56
XGB	0.65	0.56	0.59	0.56	0.58

Table 12: TFIDF Embedding

Classifier	AUC	Acc.	Recall	Prec.	F1
RFC	0.68	0.64	0.68	0.62	0.65
ADB	0.61	0.62	0.61	0.62	0.62
LGBM	0.65	0.62	0.56	0.63	0.59
GBC	0.65	0.62	0.55	0.63	0.59
SVC	0.69	0.62	0.61	0.63	0.62
XGB	0.69	0.62	0.59	0.62	0.61
LR	0.63	0.60	0.50	0.62	0.56
Perceptron	0.55	0.55	0.51	0.55	0.53
DT	0.56	0.52	0.51	0.53	0.52
NB	0.58	0.52	0.39	0.52	0.45

Table 13: BOW Embedding

Classifier	AUC	Acc.	Recall	Prec.	F1
XGB	0.66	0.64	0.61	0.64	0.63
GBC	0.63	0.62	0.74	0.59	0.66
LGBM	0.66	0.62	0.65	0.61	0.63
SVC	0.67	0.61	0.75	0.59	0.66
LR	0.65	0.58	0.83	0.55	0.66
Perceptron	0.56	0.56	0.65	0.56	0.6
ADB	0.61	0.56	0.52	0.56	0.54
RFC	0.71	0.56	0.90	0.54	0.67
DT	0.54	0.54	0.87	0.52	0.65
NB	0.66	0.50	0.95	0.50	0.66

Table 14: fastText Embedding

Classifier	AUC	Acc.	Recall	Prec.	F1
LGBM	0.69	0.66	0.70	0.64	0.67
GBC	0.71	0.66	0.71	0.65	0.68
SVC	0.72	0.66	0.69	0.66	0.67
RFC	0.72	0.66	0.75	0.64	0.69
LR	0.74	0.66	0.57	0.70	0.63
ADB	0.68	0.62	0.63	0.62	0.63
DT	0.64	0.61	0.60	0.61	0.61
XGB	0.64	0.60	0.63	0.59	0.61
NB	0.69	0.58	0.32	0.67	0.43
Perceptron	0.50	0.50	0	0	0

Table 15: Lexical Feature Engineering

2.6. Evaluating Models: Transfer Learning vs No-Transfer

We can clearing see the transfer learning is giving the best result. If we do not use any transfer learning techniques then results are far behind. The best score when we do transfer embedding & transfer task, both, is 76% accuracy while without any kind of transfer the best result is 68% accuracy.

TL Type	Embed Name	Classifier	AUC	Acc	Recall	Prec.	F1
EMB	fastTextWiki	NB	0.80	0.76	0.78	0.75	0.76
Task	fastTextWiki	TT	0.81	0.76	0.71	0.79	0.75
EMB	IndicFT	NB	0.77	0.74	0.70	0.76	0.73
EMB	IndicFT	LR	0.78	0.74	0.70	0.75	0.73
EMB	IndicFT	SVC	0.79	0.74	0.71	0.76	0.73
EMB	IndicFT	ADB	0.79	0.74	0.72	0.76	0.74
EMB	IndicFT	XGB	0.79	0.74	0.70	0.76	0.73
EMB	Combined	NB	0.79	0.74	0.76	0.74	0.75
Task	mBERT (Pytorch)	PyrotchTT	0.80	0.74	0.69	0.76	0.72
EMB	fastTextWiki	SVC	0.81	0.74	0.67	0.79	0.72

Table 16: Transfer Learning (Task & Embedding) Models

Embedding Name	Classifier	AUC	Accu.	Recall	Prec.	F1
Keras_Tokenizer	CNN	0.74	0.68	0.65	0.70	0.67
Keras_Tokenizer	RNN	0.74	0.68	0.70	0.67	0.68
Lexical	LGBM	0.69	0.66	0.70	0.64	0.67
Lexical	GBC	0.71	0.66	0.71	0.65	0.68
Lexical	SVC	0.72	0.66	0.69	0.66	0.67
Lexical	RFC	0.72	0.66	0.75	0.64	0.69
Lexical	LR	0.74	0.66	0.57	0.70	0.63
fastText	XGB	0.66	0.64	0.61	0.64	0.63
Word2Vec	LGBM	0.68	0.64	0.72	0.62	0.66
BOW	RFC	0.68	0.64	0.68	0.62	0.65

Table 17: Top 10- Non-Transfer Learning Models) Models

2.7. Evaluating Models with Embedding Used

With Naïve Baysian classifier fastTextWiki gives the best accuracy.The results of NB with IndicFT is not far behind. On NB no other embedding is doing good work.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
fastTextWiki	0.80	0.76	0.78	0.75	0.76
IndicFT	0.77	0.74	0.70	0.76	0.73
Combined	0.79	0.74	0.76	0.74	0.75
IndicBERT	0.61	0.59	0.67	0.58	0.62
Lexical	0.69	0.58	0.32	0.67	0.43
TFIDF	0.60	0.56	0.53	0.56	0.54
mBERT	0.55	0.52	0.30	0.53	0.38
BOW	0.58	0.52	0.39	0.52	0.45
Word2Vec	0.64	0.50	0.95	0.50	0.66
fastText	0.66	0.50	0.95	0.50	0.66

Table 18: Naive Bayesian with All Embeddings

SVM classifier is giving same accuracy with IndicFT and fastTextWiki embedding. The results of the other embedding are 6% less accuracy.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.79	0.74	0.71	0.76	0.73
fastTextWiki	0.81	0.74	0.67	0.79	0.72
Combined	0.75	0.68	0.70	0.67	0.68
Lexical	0.72	0.66	0.69	0.66	0.67
Word2Vec	0.67	0.62	0.75	0.60	0.66
TFIDF	0.68	0.62	0.58	0.64	0.61
BOW	0.69	0.62	0.61	0.63	0.62
fastText	0.67	0.61	0.75	0.59	0.66
mBERT	0.63	0.60	0.66	0.59	0.63
IndicBERT	0.60	0.56	0.65	0.55	0.59

Table 19: Support Vector Machine with All Embeddings

Logistic Regression performs the best with IndicFT embedding but fastTextWiki with Logistic Regression is not far behind.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.78	0.74	0.70	0.75	0.73
fastTextWiki	0.81	0.72	0.66	0.75	0.7
Combined	0.80	0.70	0.61	0.74	0.67
Lexical	0.74	0.66	0.57	0.70	0.63
Word2Vec	0.64	0.61	0.76	0.58	0.66
BOW	0.63	0.60	0.50	0.62	0.56
TFIDF	0.64	0.60	0.52	0.61	0.56
mBERT	0.61	0.58	0.68	0.57	0.62
fastText	0.65	0.58	0.83	0.55	0.66
IndicBERT	0.59	0.57	0.61	0.57	0.59

Table 20: Logistic Regression with All Embeddings

XGBoost, AdaBoost classifer performs the best with IndicFT.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.79	0.74	0.70	0.76	0.73
fastTextWiki	0.78	0.71	0.64	0.74	0.69
Combined	0.80	0.71	0.65	0.74	0.69
IndicBERT	0.71	0.66	0.65	0.67	0.66
fastText	0.66	0.64	0.61	0.64	0.63
Word2Vec	0.69	0.62	0.63	0.62	0.63
BOW	0.69	0.62	0.59	0.62	0.61
Lexical	0.64	0.60	0.63	0.59	0.61
mBERT	0.61	0.57	0.62	0.56	0.59
TFIDF	0.65	0.56	0.59	0.56	0.58

Table 21: XG Boost with All Embeddings

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.79	0.74	0.72	0.76	0.74
Combined	0.78	0.70	0.64	0.74	0.68
fastTextWiki	0.79	0.70	0.65	0.73	0.69
BOW	0.61	0.62	0.61	0.62	0.62
Word2Vec	0.64	0.62	0.71	0.61	0.65
Lexical	0.68	0.62	0.63	0.62	0.63
IndicBERT	0.64	0.60	0.59	0.60	0.59
TFIDF	0.60	0.56	0.55	0.56	0.56
fastText	0.61	0.56	0.52	0.56	0.54
mBERT	0.56	0.53	0.67	0.52	0.59

Table 22: AdaBoost with All Embeddings

GBC classifer is performing equally well on Combined embedding and IndicFT.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
Combined	0.78	0.72	0.66	0.74	0.7
IndicFT	0.79	0.72	0.67	0.75	0.71
fastTextWiki	0.78	0.69	0.62	0.72	0.67
Lexical	0.71	0.66	0.71	0.65	0.68
TFIDF	0.63	0.62	0.57	0.64	0.6
fastText	0.63	0.62	0.74	0.59	0.66
Word2Vec	0.64	0.62	0.70	0.61	0.65
BOW	0.65	0.62	0.55	0.63	0.59
IndicBERT	0.68	0.62	0.58	0.63	0.6
mBERT	0.64	0.60	0.65	0.60	0.62

Table 23: Gradient Boost Classifier with All Embeddings

LGBM classifier is performing the best on IndicFT but fastTextWiki embedding is not far behind.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.79	0.72	0.67	0.74	0.7
fastTextWiki	0.78	0.70	0.63	0.72	0.67
Combined	0.78	0.70	0.63	0.72	0.67
Lexical	0.69	0.66	0.70	0.64	0.67
Word2Vec	0.68	0.64	0.72	0.62	0.66
IndicBERT	0.69	0.64	0.58	0.67	0.62
BOW	0.65	0.62	0.56	0.63	0.59
fastText	0.66	0.62	0.65	0.61	0.63
TFIDF	0.66	0.60	0.54	0.62	0.58
mBERT	0.63	0.58	0.64	0.57	0.6

Table 24: Light Gradient Boost Model with All Embeddings

RFC classifer work the best with IndicFT & fastTextWiki embeddings. But the results of IndicBERT embedding with RFC is not far behind.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.79	0.72	0.68	0.75	0.71
fastTextWiki	0.79	0.71	0.65	0.74	0.69
IndicBERT	0.71	0.70	0.70	0.70	0.70
Combined	0.80	0.70	0.63	0.74	0.68
Lexical	0.72	0.66	0.75	0.64	0.69
BOW	0.68	0.64	0.68	0.62	0.65
mBERT	0.64	0.60	0.65	0.59	0.62
TFIDF	0.66	0.60	0.56	0.61	0.58
Word2Vec	0.69	0.57	0.89	0.54	0.67
fastText	0.71	0.56	0.90	0.54	0.67

Table 25: Random Forest Classifier with All Embeddings

Perceptron works the best with IndicFT, other embedding has less accuracy than IndicFT with perceptron. With Perceptron IndicFT give the best F1 score but with AdaBoost none of the embedding is able to give more than 70% of F1 score.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.72	0.72	0.56	0.81	0.66
fastTextWiki	0.63	0.63	0.36	0.78	0.49
TFIDF	0.60	0.60	0.61	0.60	0.60
IndicBERT	0.56	0.56	0.22	0.67	0.33
fastText	0.56	0.56	0.65	0.56	0.6
BOW	0.55	0.55	0.51	0.55	0.53
Word2Vec	0.52	0.52	0.47	0.52	0.49
mBERT	0.50	0.50	0.24	0.51	0.33
Lexical	0.50	0.50	0	0	0
Combined	0.50	0.50	0.01	0.50	0.02

Table 26: Perceptron with All Embeddings

Decision tree on IndicFT give best accuracy 72%. But on fastTextWiki its performance is 9% less.

Embedding Name	AUC	Acc.	Recall	Prec.	F1
IndicFT	0.71	0.72	0.61	0.77	0.68
Combined	0.63	0.63	0.63	0.63	0.63
fastTextWiki	0.64	0.63	0.63	0.63	0.63
mBERT	0.63	0.62	0.65	0.61	0.63
Lexical	0.64	0.61	0.60	0.61	0.61
IndicBERT	0.62	0.60	0.62	0.60	0.61
Word2Vec	0.60	0.58	0.79	0.56	0.65
TFIDF	0.61	0.58	0.73	0.57	0.64
fastText	0.54	0.54	0.87	0.52	0.65
BOW	0.56	0.52	0.51	0.53	0.52

Table 27: Decision Tree with All Embeddings

2.8. Evaluating CNN & RNN Models

Results of CNN & RNN classification models is not comparable to other models discussed earlier. Even if we use the best embedding and transfer it to our CNN model, we are not getting more than 66% accuracy.

Classifier	Embedding Name	AUC	Acc.	Recall	Prec.	F1
CNN	Keras_Tokenizer	0.74	0.68	0.65	0.70	0.67
RNN	Keras_Tokenizer	0.74	0.68	0.70	0.67	0.68
CNN	IndicFT	0.71	0.66	0.65	0.66	0.66
CNN	fastTextWiki	0.74	0.65	0.74	0.63	0.68

Table 28: CNN & RNN Model Results

2.9. Comparing Results with Other Works

However, did not find any work which has been on Hinglish language and using twitter and normal blog text together for sarcasm work yet we are putting a table below to demonstrate other work and compare the progress made by our work

Table 29: Comparing Results with Other Works

#	Paper	Language	Text Type & Metrics
1	Irony Detection in Twitter: The Role of Affective Content. (Faŕias, Patti and Rosso, 2016)	English, Twitter	Acc: 73-96% depends upon datasets and classifier.
2	Natural Language Processing Based Features for Sarcasm Detection: An Investigation Using Bilingual Social Media Texts. (Suhaimin, Hijazi, Alfred and Coenen, 2017)	English, Twitter	Acc: 82.5%
3	Semantics-aware BERT for Language Understanding. (Zhang, Sun, Galley, Chen, Brockett, Gao, Gao, Liu and Dolan, 2020)	English, Normal Text	Acc: 94.6% on Large dataset of SST2
4	Multi-Rule Based Ensemble Feature Selection Model for Sarcasm Type Detection in Twitter. (Sundararajan and Palanisamy, 2020)	English, Twitter	Acc: 86.61% to 99.79% Depending upon the type of sarcasm. Final classifier is RF
5	Sarcasm Detection in Typo-graphic Memes (Kumar, Singh and Kaur, 2019)	English, Instagram Images	Acc: 73.25% to 87.95% depending upon the classifier used.
6	Sarcasm detection on twitter : A Behavioural Modeling Approach. (Rajadesingan, Zafarani and Liu, 2015)	English, Tweet	Acc: 83.46%
7	Lexicon-Based Sentiment Analysis in the Social Web. (Asghar, Kundi, Khan and Ahmad, 2014)	English, Tweet	Acc: 95.24%
8	Harnessing Context Incongruity for Sarcasm Detection. (Joshi, Sharma and Bhattacharyya, 2015)	English, Tweet	F1: 61%
9	Contextualized Sarcasm Detection on Twitter (Bamman and Smith, 2015)	English, Tweet	Acc: 85.1%
10	Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. (Turney, 2002)	English, Opinion Survey of Products	Acc: 74.39%
11	Towards Multimodal Sarcasm Detection: An Obviously Perfect Paper (Castro, Hazarika, Pérez-Rosas, Zimmermann, Mihalcea and Poria, 2020)	English, Clips of YouTube, TV Shows, Transcription	F1: 71.8%
12	A Transformer-based approach to Irony and Sarcasm detection (Potamias, Siolas and Stafylopatis, 2020)	English, Irony/SemVal-2018-Task, Reddit SARC2.0 politics, Riloff Sarcastic Dataset	Acc: 85% to 94% depending upon dataset
13	Detecting Sarcasm is Extremely Easy ;-) (Parde and Nielsen, 2018)	English, Tweet, Amazon product reviews	F1: 59% (Twitter) F1: 78% (Amazon)
14	CARER: Contextualized Affect Representations for Emotion Recognition (Saravia, Liu, Huang, Wu and Chen, 2018)	English, Tweets	Acc: 81% with CARER
15	The perfect solution for detecting sarcasm in tweets #not (Liebrecht, Kunneman and Van den Bosch, 2013)	Dutch, Tweets	AUC: 77%

#	Paper	Language	Text Type & Metrics
16	A2Text-net: A novel deep neural network for sarcasm detection (Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer and Stoyanov, 2019)	English, Tweet, News Headlines, Reddit	F1: 71% - 90% depending upon dataset with A2Text classifer
17	Sarcasm as contrast between a positive sentiment and negative situation (Riloff, Qadir, Surve, Silva, Gilbert and Huang, 2013)	English, Tweet	F1: 51%
18	Exploring the fine-grained analysis and automatic detection of irony on Twitter (Hee, Lefever and Hoste, 2018)	English, Tweet	Acc: 67.54% (SVM) Acc: 68.27% (LSTM)
19	Exploiting Emojis for Sarcasm Detection (Subramanian, Sridharan, Shu and Liu, 2019)	English, Twitter, Facebook	F1: 89.36% (Twitter) F1: 97.97% (facebook)
20	A novel automatic satire and irony detection using ensembled feature selection and data mining. (Ravi and Ravi, 2017)	English, Newswire, Satire news articles, Amazon	F1: 96.58% (L+T+D features) + GR feature selector + SVM RBF Classifier
21	Automatic Satire Detection: Are You Having a Laugh? (Burfoot and Baldwin, 2009)	English, Newswire and Satire news articles	F1: 79.8%
22	Semi-supervised recognition of sarcastic sentences in twitter and Amazon (Davidov, Tsur and Rappoport, 2010)	English, Twitter, Amazon	F1: 78% Amazon F1: 83% Twitter
23	Identifying Sarcasm in Twitter: A Closer Look. In (González-Ibáñez, Muresan and Wacholder, 2011)	English, Twitter	Acc: 55.59% to 75.78% depending upon tweet format.
	Sarcasm Detection Work in Hindi Language		
1	Sentiment Analysis of Hindi Review based on Negation and Discourse Relation. (Mittal and Agarwal, 2013)	Hindi, Movie Reviews	Acc: 80.21%
2	A Sentiment Analyzer for Hindi Using Hindi Senti Lexicon. (Sharma, Sangal, Pawar, Sharma and Bhattacharyya, 2014)	Hindi, Movie Reviews, Product Reviews	Acc: 85 to 89.5%
3	Sarcasm Detection in Hindi sentences using Support Vector (Desai and Dave, 2016)	Hindi, various online sources (using polarity levelled corpora)	Acc: 84%
4	Sentiment Analysis in a Resource Scarce Language: Hindi. (Jha, N, Shenoy and R, 2016)	Hindi, Movie Reviews	Acc: 92.2% to 100% depending upon unigram or bigram feature and classifer
5	Harnessing Online News for Sarcasm Detection in Hindi Tweets (Bharti, Babu and Jena, 2017)	Hindi, Tweets	Acc: 79.4%
6	Context-based Sarcasm Detection in Hindi Tweets. (Bharti, Babu and Raman, 2018)	Hindi, Tweets	Acc: 87%
7	A Corpus of English-Hindi Code-Mixed Tweets for Sarcasm Detection (Swami, Khandelwal, Singh, Akhtar and Shrivastava, 2018)	Hindi-English, Tweets	Acc: 78.4% with RF
8	BHAAV- A Text Corpus for Emotion Analysis from Hindi Stories (Kumar et al., 2019)	Hindi, Short stories	Acc: 62%
9	**Sarcasm Detection in Hinglish Language** by Hari Thapliyal	**Hinglish Language, Twitter + Blog Text**	**Accuracy: 76%, Recall: 78%, Precision: 75%, F1: 76%, AUC: 80%**

References

Asghar, M.Z., Kundi, F.M., Khan, A., Ahmad, S., 2014. Lexicon-based sentiment analysis in the social web. J. Basic. Appl. Sci. Res 4, 238–248.

Bamman, D., Smith, N.A., 2015. Contextualized sarcasm detection on twitter. Proceedings of the 9th International Conference on Web and Social Media, ICWSM 2015 , 574–577.

Bharti, S.K., Babu, K.S., Jena, S.K., 2017. Harnessing online news for sarcasm detection in hindi tweets. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10597 LNCS, 679–686. doi:10.1007/978-3-319-69900-4_86.

Bharti, S.K., Babu, K.S., Raman, R., 2018. Context-based sarcasm detection in hindi tweets. 2017 9th International Conference on Advances in Pattern Recognition, ICAPR 2017 , 410–415doi:10.1109/ICAPR.2017.8593198.

Burfoot, C., Baldwin, T., 2009. Automatic satire detection: Are you having a laugh? ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf. , 161–164URL: http://search.cpan.org/perldoc?

Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R., Poria, S., 2020. Towards multimodal sarcasm detection (an obviously perfect paper). ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference , 4619–4629.

Davidov, D., Tsur, O., Rappoport, A., 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. CoNLL 2010 - Fourteenth Conference on Computational Natural Language Learning, Proceedings of the Conference , 107–116.

Desai, N.P., Dave, A.D., 2016. Sarcasm detection in hindi sentences using support vector machine. International Journal of Advance Research in Computer Science and Management Studies 4, 8–15. URL: www.ijarcsms.com.

Fa ́rias, D.I.H., Patti, V., Rosso, P., 2016. Irony detection in twitter: The role of affective content. ACM Transactions on Internet Technology 16, 1–24. URL: http://dx.doi.org/10.1145/2930663https://dl.acm.org/doi/10.1145/2930663, doi:10.1145/2930663.

González-Ibáñez, R., Muresan, S., Wacholder, N., 2011. Identifying sarcasm in twitter: A closer look. ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies 2, 581–586.

Hee, C.V., Lefever, E., Hoste, V., 2018. Exploring the fine-grained analysis and automatic detection of irony on twitter. Language Resources and Evaluation 52, 707–731. doi:10.1007/s10579-018-9414-2.

Jha, V., N, M., Shenoy, P.D., R, V.K., 2016. Sentiment analysis in a resource scarce language:hindi. International Journal of Scientific & Engineering Research 7, 968–980. doi:10.14299/ijser.2016.09.005.

Joshi, A., Sharma, V., Bhattacharyya, P., 2015. Harnessing context incongruity for sarcasm detection. ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference 2, 757–762. doi:10.3115/v1/p15-2124.

Kumar, A., Singh, S., Kaur, G., 2019. Fake news detection of indian and united states election data using machine learning algorithm. International Journal of Innovative Technology and Exploring Engineering 8, 1559–1563. doi:10.35940/ijitee.K1829.0981119.

Liebrecht, C., Kunneman, F., Van den Bosch, A., 2013. The perfect solution for detecting sarcasm in tweets #not, pp. 29–37.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. URL: https://github.com/pytorch/fairseqhttp://arxiv.org/abs/1907.11692.

Mittal, N., Agarwal, B., 2013. Sentiment analysis of hindi review based on negation and discourse relation. Sixth International Joint Conference on Natural Language Processing , 57–62URL: http://www.aclweb.org/website/old_anthology/W/W13/W13-43.pdf#page=57.

Parde, N., Nielsen, R., 2018. Detecting sarcasm is extremely easy ;-), pp. 21–26. doi:10.18653/v1/W18-1303.

Potamias, R.A., Siolas, G., Stafylopatis, A.G., 2020. A transformer-based approach to irony and sarcasm detection. Neural Computing and Applications doi:10.1007/s00521-020-05102-3.

Rajadesingan, A., Zafarani, R., Liu, H., 2015. Sarcasm detection on twitter:a behavioral modeling approach. WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining , 97–106doi:10.1145/2684822.2685316.

Ravi, K., Ravi, V., 2017. A novel automatic satire and irony detection using ensembled feature selection and data mining. Knowledge-Based Systems 120, 15–33. URL: http://dx.doi.org/10.1016/j.knosys.2016.12.018, doi:10.1016/j.knosys.2016.12.018.

Riloff, E., Qadir, A., Surve, P., Silva, L.D., Gilbert, N., Huang, R., 2013. Sarcasm as contrast between a positive sentiment and negative situation. EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference , 704–714.

Saravia, E., Liu, H.C., Huang, Y.H., Wu, J., Chen, Y.S., 2018. Carer: Contextualized affect representations for emotion recognition, pp. 3687–3697. doi:10.18653/v1/D18-1404.

Sharma, D.S., Sangal, R., Pawar, J.D., Sharma, R., Bhattacharyya, P., 2014. A sentiment analyzer for hindi using hindi senti lexicon.

Subramanian, J., Sridharan, V., Shu, K., Liu, H., 2019. Exploiting emojis for sarcasm detection. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11549 LNCS, 70–80. doi:10.1007/978-3-030-21741-9_8.

Suhaimin, M.S.M., Hijazi, M.H.A., Alfred, R., Coenen, F., 2017. Natural language processing based features for sarcasm detection: An investigation using bilingual social media texts. ICIT 2017 - 8th International Conference on Information Technology, Proceedings , 703–709doi:10.1109/ICITECH.2017.8079931.

Sundararajan, K., Palanisamy, A., 2020. Multi-rule based ensemble feature selection model for sarcasm type detection in twitter. Computational Intelligence and Neuroscience 2020. doi:10.1155/2020/2860479.

Swami, S., Khandelwal, A., Singh, V., Akhtar, S.S., Shrivastava, M., 2018. A corpus of english-hindi code-mixed tweets for sarcasm detection.

Turney, P.D., 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews , 417–424URL: http://arxiv.org/abs/cs/0212032.

WikipageA, . Demographics of india - wikipedia. URL: https://en.wikipedia.org/wiki/Demographics{_}of{_}India.

Zhang, Y., Sun, S., Galley, M., Chen, Y.C., Brockett, C., Gao, X., Gao, J., Liu, J., Dolan, B., 2020. Dialogpt : Large-scale generative pre-training for conversational response generation, pp. 270–278. URL: https://github.com/mgalley/, doi:10.18653/v1/2020.acl-demos.30.

Hari Thapliyal is a Data Science and Project Management professional. He is a mentor, trainer, coach, consultant, meditator, philosopher and blogger. In his 28+ years professional career in software development, project management, training and consulting he has been deeply involved in all kind of roles from software design, development, quality assurance, training, mentoring, PMO head and many others. He has deep interests in diverse subjects like BFSI Sector, Artificial Intelligence, Data Mining, Data Analytics, Deep Learning, ML

Modelling, NLP, Economics, Physics, Sanskrit, Vedic Chanting, Vedanta, Healing, History, Culture, Project Management, Meditation and Spirituality. This helps him to understand that how and where to discover new equilibrium among many variables like religion, culture, ethics, morality, societies, business, process automation, and new age technologies like AI, NLP, Deep Learning, GAN, Robotics, Cryptocurrency etc. He is Xpert Coach and Mentor for various AI, ML courses of upGrad and he is founder of dasarpAI an AI Training, Consulting startup.