

Classification results for Orientation

Binary					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.7934	'C': 1, 'kernel': 'linear'	0.6691	0.6591	0.6632
Random Forest	0.7448	'n_estimators': 10	0.5462	0.5106	0.4584
Logistic Regression	0.8041	'C': 10	0.6813	0.6666	0.6722
Multinomial Naive Bayes	0.7448		0.8510	0.5333	0.475
ANN using Word Tokenizer	0.7391		*0.00	*0.00	0.410
ANN using Word Tokenizer (with removing stopwords)	0.7391		*0.00	*0.00	0.4138
ANN using GloVe & LSTM	0.7391		*0.00	*0.00	0.4138

Ternary					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.5574	'C': 1, 'kernel': 'linear'	0.4837	0.5266	0.4930
Random Forest	0.5016	'n_estimators': 10	0.5340	0.4925	0.4722
Logistic Regression	0.5574	'C': 1	0.5370	0.5714	0.5453
Multinomial Naive Bayes	0.5148	0.5148	0.6424	0.4055	0.3785
ANN using Word Tokenizer					
ANN using Word Tokenizer (with removing stopwords)					
ANN using GloVe & LSTM	0.3043		0.5652	1.0000	0.6190

5-Point Scale					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.4230	'C': 20, 'kernel': 'rbf'	0.2226	0.2687	0.2222
Random Forest	0.3508	'n_estimators': 10	0.4584	0.3272	0.3515
Logistic Regression	0.4328	'C': 5	0.3423	0.3107	0.3193
Multinomial Naive Bayes	0.3574		0.3595	0.2293	0.1890
ANN using Word Tokenizer					
ANN using Word Tokenizer (with removing stopwords)					
ANN using GloVe & LSTM					

Classification results for Evaluation

Binary					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.7097	'C': 20, 'kernel': 'rbf'	0.6962	0.6771	0.6690
Random Forest	0.6609	'n_estimators': 10	0.6962	0.6771	0.6690
Logistic Regression	0.6923	'C': 5	0.6678	0.6666	0.6660
Multinomial Naive Bayes	0.6087		0.6676	0.6145	0.5815
ANN using Word Tokenizer	0.5913		0.5913	1.0000	0.7432
ANN using Word Tokenizer (with removing stopwords)	0.5913		0.5913	1.0000	0.7432
ANN using GloVe & LSTM	0.6000		0.5965	1.0000	0.7432

Ternary					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.6098	'C': 20, 'kernel': 'rbf'	0.4467	0.4919	0.4638
Random Forest	0.5574	'n_estimators': 10	0.3570	0.4129	0.3824
Logistic Regression	0.6066	'C': 5	0.5175	0.5077	0.4989
Multinomial Naive Bayes	0.5475		0.4003	0.4555	0.3988
ANN using Word Tokenizer					
ANN using Word Tokenizer (with removing stopwords)					
ANN using GloVe & LSTM	0.1739		0.5913	1.0000	0.5887

5-Point Scale					
Method	Accuracy	Hyper Parameter Tuning	Precision	Recall	F1-Score
SVM	0.3443	'C': 10, 'kernel': 'rbf'	0.2763	0.2645	0.2168
Random Forest	0.3279	'n_estimators': 10	0.2834	0.2702	0.2641
Logistic Regression	0.3607	'C': 10	0.4096	0.4075	0.3817
Multinomial Naive Bayes	0.2820		0.4299	0.2907	0.2694
ANN using Word Tokenizer					
ANN using Word Tokenizer (with removing stopwords)					
ANN using GloVe & LSTM					

My Thoughts:

I have chosen those model which is marked by **GREEN** colors. Because I first looked for **F1-Score** than **accuracy** because it works well when our data is imbalanced. But I have a doubt on ternary classification for **evaluation** due to **little accuracy** which is using “ANN using GloVe & LSTM” model.

Asterisk (*) sign indicated I need to look it again what’s the exact reason for showing these values.

The main reason for the little accuracy of my deep learning models is that I have used to train my model only by using the **text** parameters & due to small data-set. If I can use **multiple parameters** as **inputs** then It will be a good model in terms of accuracy & F1-Score.

From those tables, Some of the Deep Learning Models have no result. After classifying by orientation I thought that there is no need to go with them due to little accuracy but further I realized **F1-Scores** can good impact over accuracy.

Future Improvements:

I have applied these three (Word2Vec, GloVe, Count Vectorization) word embedding method in this dataset so far. Due to lack of data, I couldn’t get good scores. And these methods ain’t context-sensitive embeddings. So, I think, I should apply ELMo & BERT as well or similar types of algorithms. Or I can use some kinds of **scoring algorithms** that is used in this (A Hybrid Approach for Sentiment Analysis Applied to Paper Reviews) paper.

Model Name	Context-sensitive embeddings	Learned representations
Count Vectorization	No	words
Word2Vec	No	words
GloVe	No	words
ELMo	Yes	words
BERT	Yes	sub-words

Fig-1: Word Embedding Technics

It would be better if I could build similar types of Model (Fig-2) by using Tensorflow. I tried but failed to mismatching tensors while executing the model for training purposes. I need to learn more in order to fix it up.

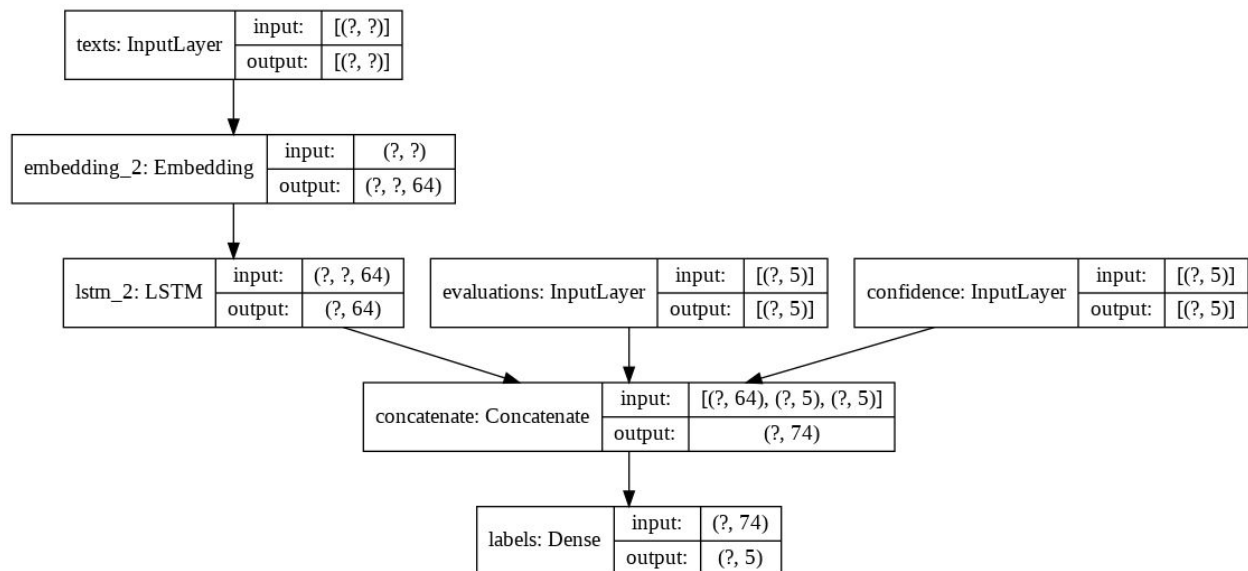


Fig-2: Multilayer Inputs Sample (for ANN)

Finally, I tried my best to prove myself for getting an opportunity to work with you however, I strongly believe that I can do better than that.