Artificial Intelligence

N Ambalavanan

Project Name: **Fake News Detection Using NLP**

Phase 2 Project

Problem Statement:

In this phase, we can explore innovative techniques such as ensemble methods and deep learning architectures to improve the prediction system's accuracy and robustness.

Consider exploring advanced techniques like deep learning models (e.g., LSTM, BERT) for improved fake news detection accuracy.

Introduction:

In this phase, we have to explore and research some deep learning models and techniques to select a best model and technique which helps to solve our problem and improve the model's performance.

We will discuss certain deep learning architectures which proposed by large organizations to avoid the mislead of information in the social media.

Let's discuss a few deep learning models and their metrics and also see about suitable techniques for our problem.

There are a lot of Machine Learning and Deep Learning Models present outside, few of them are

- **Decision Tree**
- **Random Forest**
- **k-Nearest Neighbors classifiers**
- **SVM (Support Vector Machines)**
- **RNN(Recurrent neural Network)**
- **LSTM(Long Short Term Memory)**
- **CSI(CAPTURE,SCORE,INTEGRATE)**

which is a hybrid deep model especially built for fake news detection. Since our problem is to determine whether the news is true or fake basically binary classification problem.

Some other models are TI-CNN (Text and Image convolutional neural network) is trained with both images and text information simultaneously. The convolutional neural network makes the model to see the entire input at once, and it can be trained much faster than LSTM and many other RNN models.

Naive Bayes (NB) Classifier is a deterministic algorithm that uses the Bayes theorem to classify data. Naive Bayes classifiers work by correlating the use of tokens (typically words, or sometimes other constructions, syntactic or not), with spam and non-spam e-mails and then using Bayes theorem to calculate a probability that an email is or is not a spam message.

From the analyses, we can say that the machine learning models such as Random Forest, Decision tree, SVM, NB are performing good as neural networks for small datasets but they are less performing in large datasets. At the same time, CNNs and RNNs embedded deep learning models are performing well in both small and large sized datasets. Comparatively, the hybrid models perform better than ordinary neural networks. Now, we will see some of the hybrid models in detailed manner which are CNN-LSTM,CNN-BILSTM,BI-LSTM,LSTM.

LSTM(Long Term Short Memory) :

The architecture of LSTM

LSTMs deal with both Long Term Memory (LTM) and Short Term Memory (STM) and for making the calculations simple and effective it uses the concept of gates.

- **Forget Gate:** LTM goes to forget gate and it forgets information that is not useful.

- **Learn Gate:** Event (current input) and STM are combined together so that necessary information that we have recently learned from STM can be applied to the current input.

- **Remember Gate:** LTM information that we haven't forget and STM and Event are combined together in Remember gate which works as updated LTM.

- **Use Gate:** This gate also uses LTM, STM, and Event to predict the output of the current event which works as an updated STM.

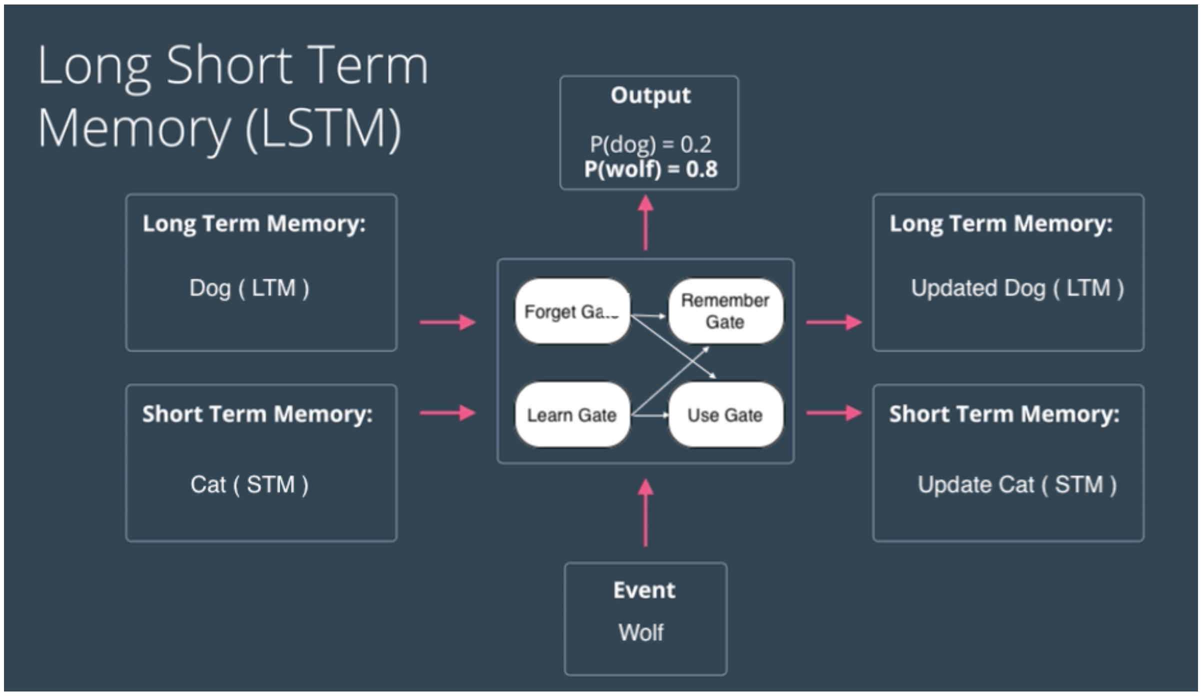
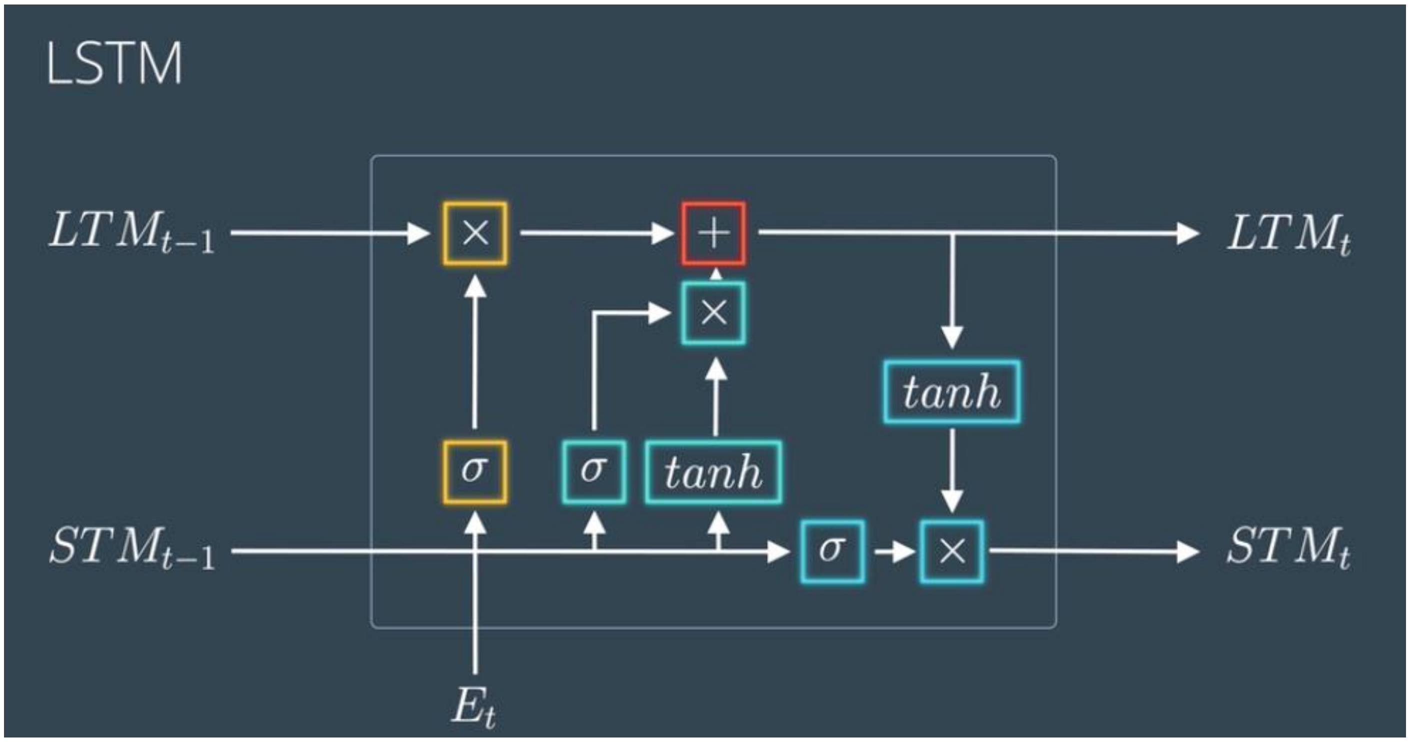


Figure : **Structure of the gates**

Mathematical Structure

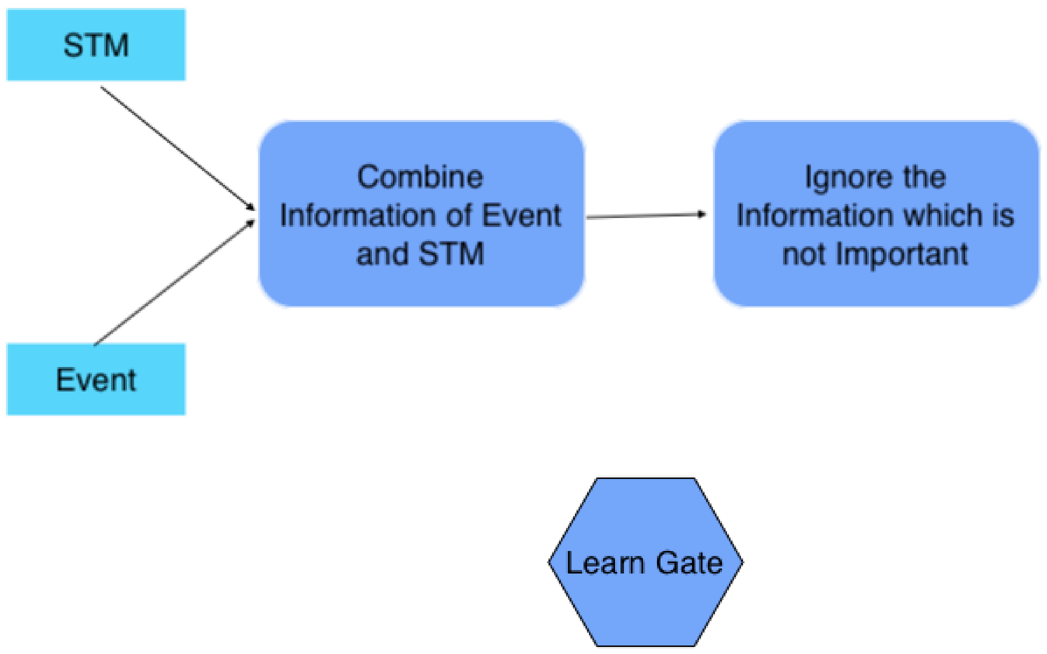


Breaking Down the Architecture

of LSTM:

- Learn Gate
- Forget Gate
- Remember Gate
- Use Gate

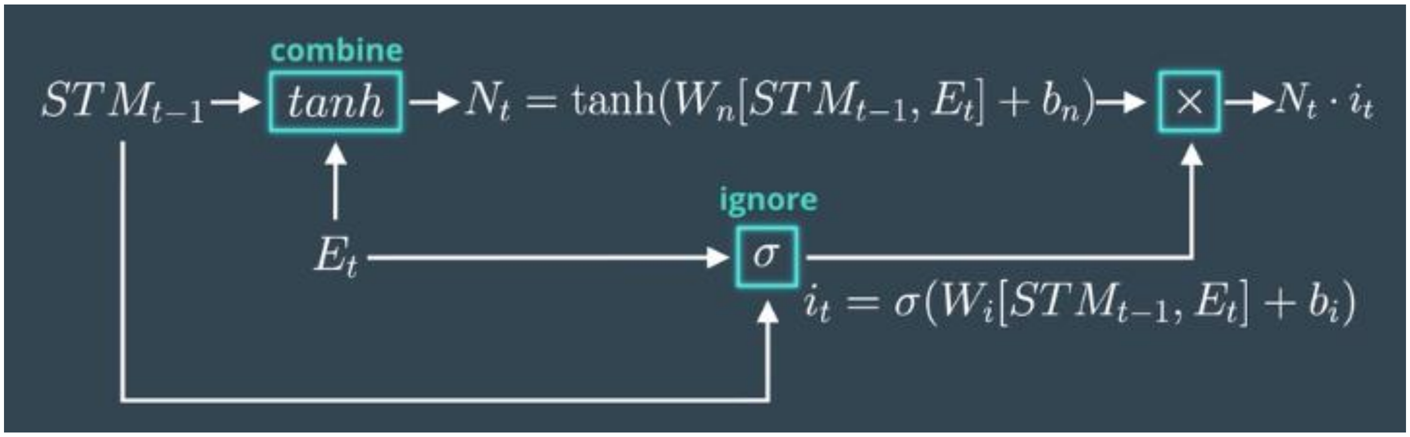
Learn Gate: Takes Event (E_t) and Previous Short Term Memory (STM_{t-1}) as input and keeps only relevant information for prediction.


STM
Event
Combine Information of Event and STM
Ignore the Information which is not Important
Learn Gate

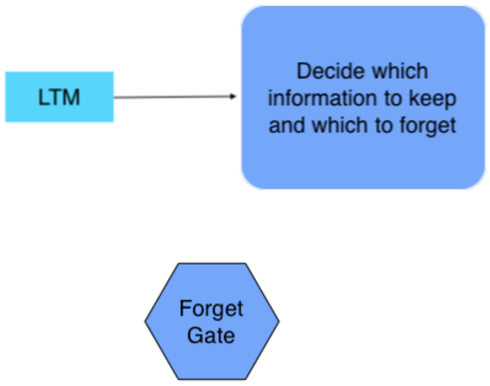
Calculation:

- Previous Short Term Memory STM_{t-1} and Current Event vector E_t are joined together $[STM_{t-1}, E_t]$ and multiplied with the weight matrix W_n having some bias which is then passed to tanh (hyperbolic Tangent) function to introduce non-linearity to it, and finally creates a matrix N_t.
- For ignoring insignificant information we calculate one Ignore Factor i_t, for which we join Short Term Memory STM_{t-1} and Current Event vector E_t and multiply with weight matrix W_i and passed through Sigmoid activation function with some bias.

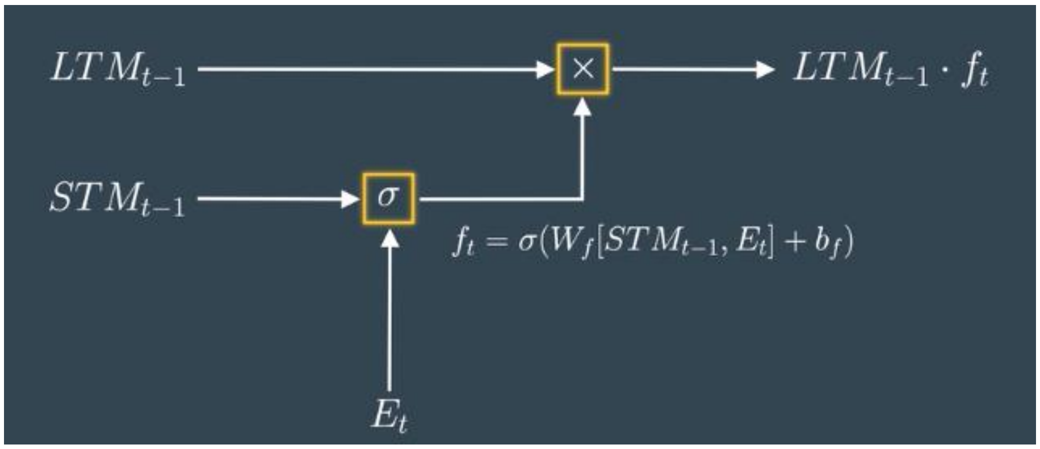
- Learn Matrix N_t and Ignore Factor i_t is multiplied together to produce learn gate result.


combine
STM_{t-1}
$tanh$
$N_t = \tanh(W_n[STM_{t-1}, E_t] + b_n)$
$\times$
$N_t \cdot i_t$
E_t
ignore
σ
$i_t = \sigma(W_i[STM_{t-1}, E_t] + b_i)$

The Forget Gate: Takes Previous Long Term Memory (LTM_{t-1}) as input and decides on which information should be kept and which to forget.

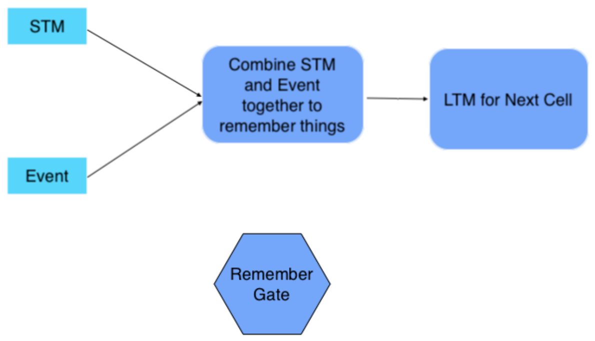


Calculation:

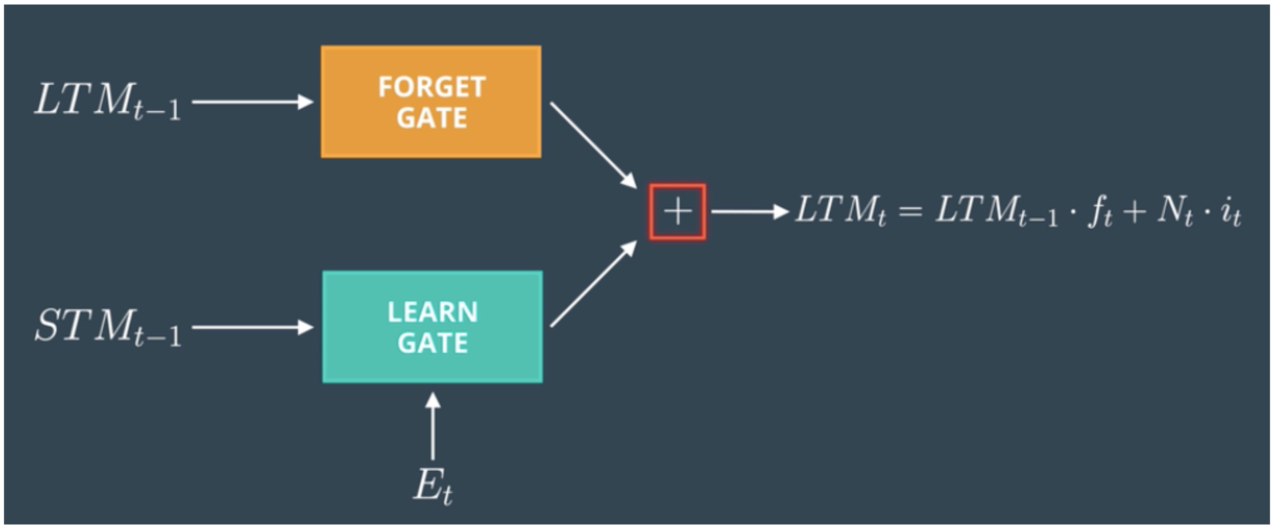


- Previous Short Term Memory STM_{t-1} and Current Event vector E_t are joined together $[STM_{t-1}, E_t]$ and multiplied with the weight matrix W_f and passed through the Sigmoid activation function with some bias to form Forget Factor f_t.
- Forget Factor f_t is then multiplied with the Previous Long Term Memory (LTM_{t-1}) to produce forget gate output.

The Remember Gate: Combine Previous Short Term Memory (STM_{t-1}) and Current Event (E_t) to produce output



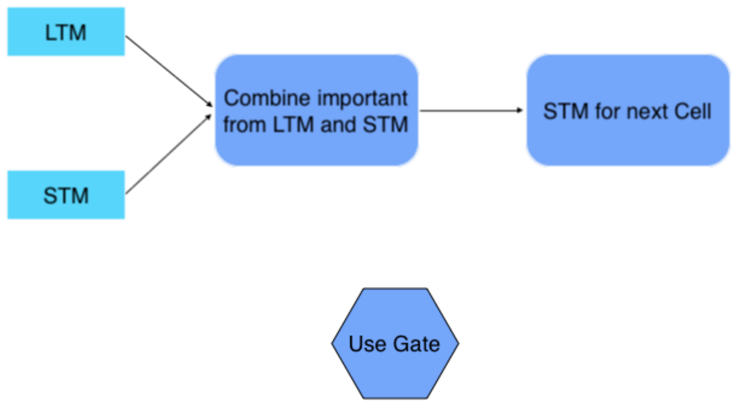
Calculation:



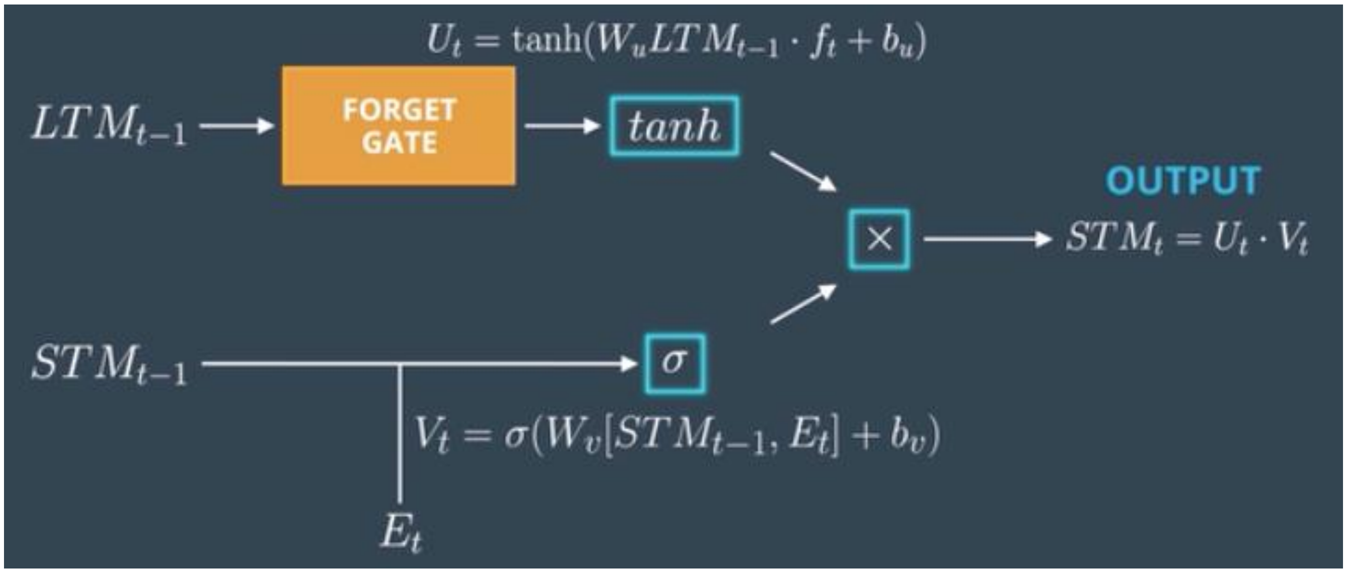
The output of Forget Gate and Learn Gate are added together to produce an output of Remember Gate which would be LTM for the next cell.

The Use Gate:

Combine important information from Previous Long Term Memory and Previous Short Term Memory to create STM for next and cell and produce output for the current event.

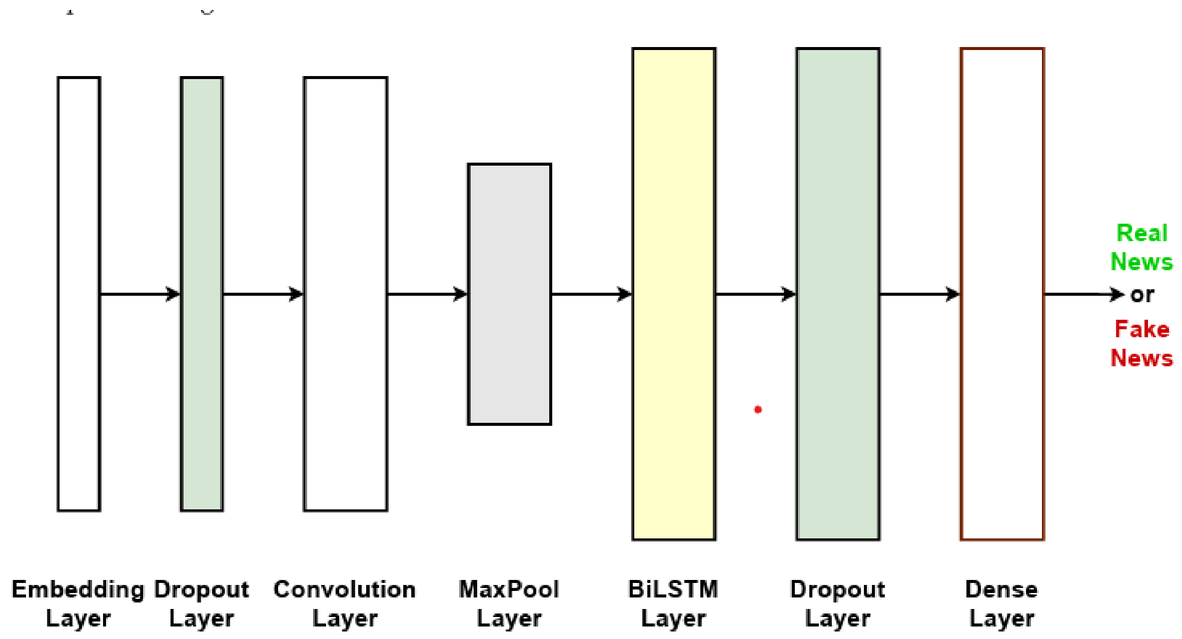


Calculation



- Previous Long Term Memory ($_{LTM-1}$) is passed through Tangent activation function with some bias to produce U_t.
- Previous Short Term Memory (STM_{t-1}) and Current Event (E_t)are joined together and passed through Sigmoid activation function with some bias to produce V_t.
- Output U_t and V_t are then multiplied together to produce the output of the use gate which also works as STM for the next cell.

Architecture of the proposed hybrid model:



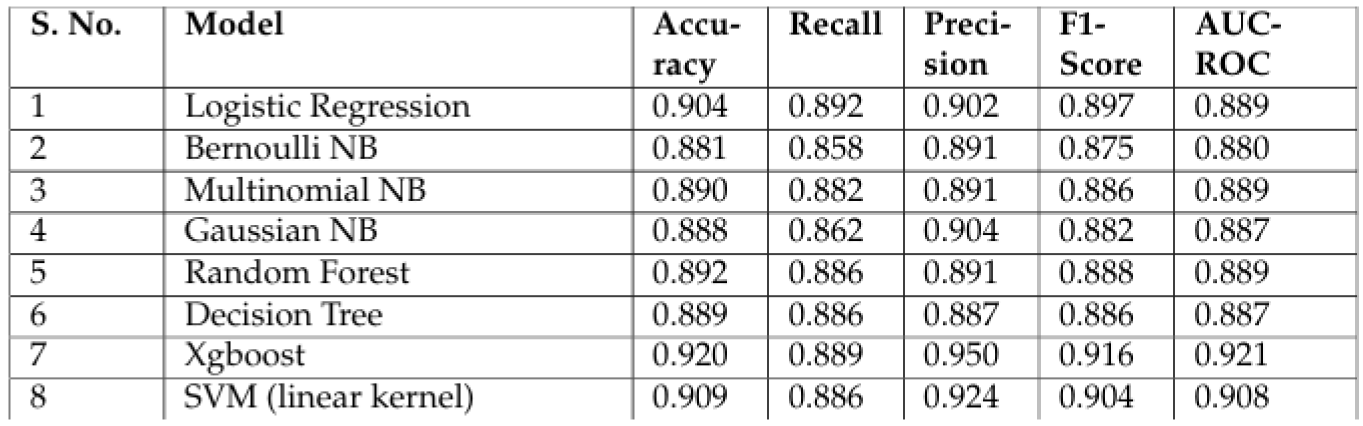
Techniques used in each layer:

The **embedding layer** is the first layer of the network. **This is the input layer where the model is fed the training data.** The **pre-trained word embeddings** are used by giving the prepared embedding matrix as **starting weights**. To reduce the **effect of overfitting**, the **next layer is a Dropout layer with a rate of 0.3**. The 3rd layer is a one-dimensional **CNN layer (Conv1D)** that has **64 filters of size 5x5** to extract local features, along with **ReLU** as the **activation function**. In the next layer, vectors are pooled (MaxPooling1D) with a window size of 4. The **BiLSTM layer** that follows receives the pooled feature maps. This information is utilised to train the **BiLSTM** that outputs **long-term dependent features of input while keeping memory**. The dimension of the output is set to 128.Next**, another Dropout layer is added with a rate of 0.3.** The **final layer of model is a dense layer.**

Here the vectors are classified as real or fake. **Sigmoid is used as a Activation function** in this layer. Binary cross-entropy is used as the loss function and **Adaptive Moment Estimation (Adam) is used as the optimizer**. Training of the model is performed with a batch size of 64.

Results and Analysis:

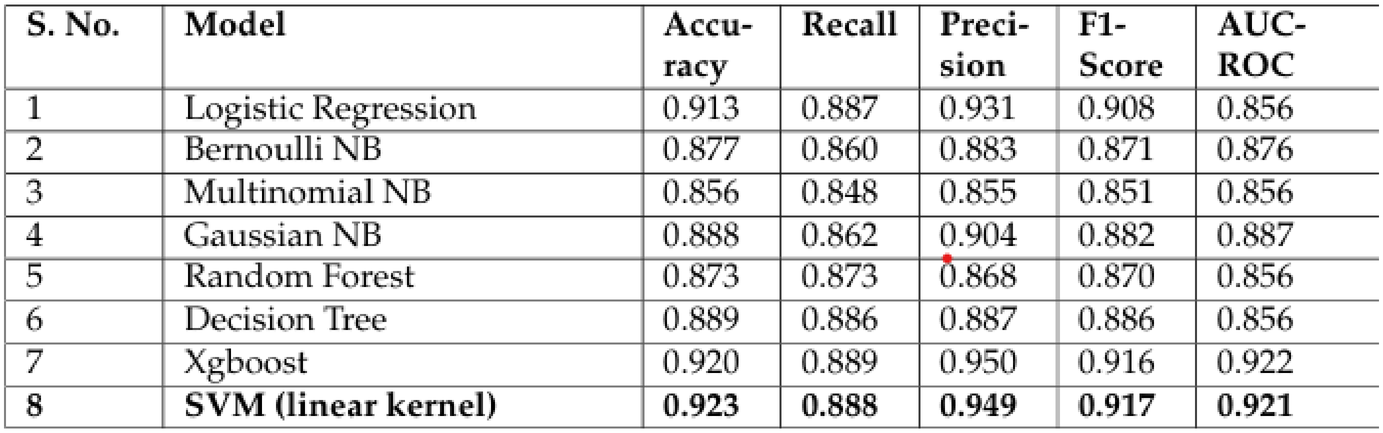
This is the last section of the Document. Now, we will see the performance of the various machine learning and deep learning models using different various feature methods.



S. No.	Model	Accu-racy	Recall	Preci-sion	F1-Score	AUC-ROC
1	Logistic Regression	0.904	0.892	0.902	0.897	0.889
2	Bernoulli NB	0.881	0.858	0.891	0.875	0.880
3	Multinomial NB	0.890	0.882	0.891	0.886	0.889
4	Gaussian NB	0.888	0.862	0.904	0.882	0.887
5	Random Forest	0.892	0.886	0.891	0.888	0.889
6	Decision Tree	0.889	0.886	0.887	0.886	0.887
7	Xgboost	0.920	0.889	0.950	0.916	0.921
8	SVM (linear kernel)	0.909	0.886	0.924	0.904	0.908

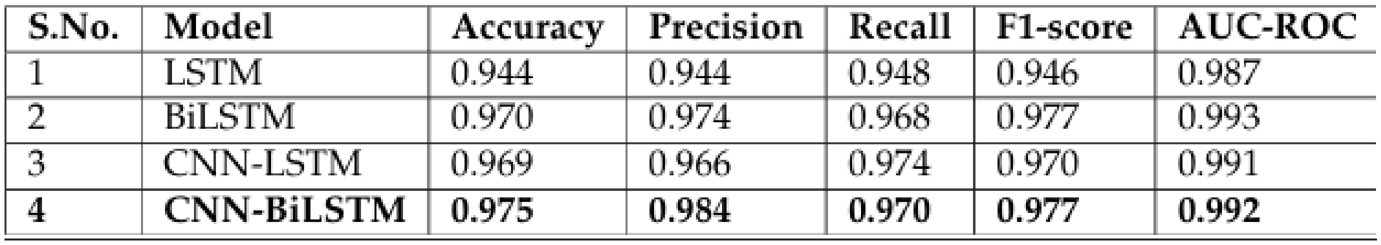
Performance of different Machine Learning methods on the consolidated dataset using **Bag of Words Count Vectorizer Feature.**

Performance of different Machine Learning methods on the consolidated



S. No.	Model	Accu-racy	Recall	Preci-sion	F1-Score	AUC-ROC
1	Logistic Regression	0.913	0.887	0.931	0.908	0.856
2	Bernoulli NB	0.877	0.860	0.883	0.871	0.876
3	Multinomial NB	0.856	0.848	0.855	0.851	0.856
4	Gaussian NB	0.888	0.862	0.904	0.882	0.887
5	Random Forest	0.873	0.873	0.868	0.870	0.856
6	Decision Tree	0.889	0.886	0.887	0.886	0.856
7	Xgboost	0.920	0.889	0.950	0.916	0.922
8	**SVM (linear kernel)**	**0.923**	**0.888**	**0.949**	**0.917**	**0.921**

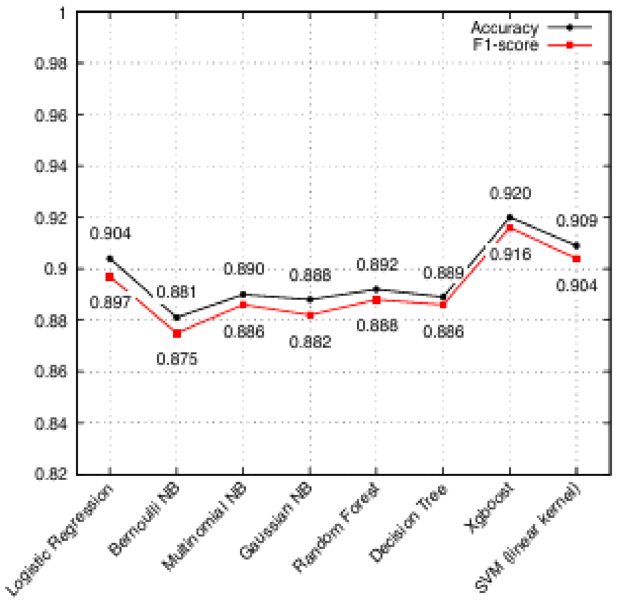
dataset **using TF-IDF Feature**



S.No.	Model	Accuracy	Precision	Recall	F1-score	AUC-ROC
1	LSTM	0.944	0.944	0.948	0.946	0.987
2	BiLSTM	0.970	0.974	0.968	0.977	0.993
3	CNN-LSTM	0.969	0.966	0.974	0.970	0.991
4	**CNN-BiLSTM**	**0.975**	**0.984**	**0.970**	**0.977**	**0.992**

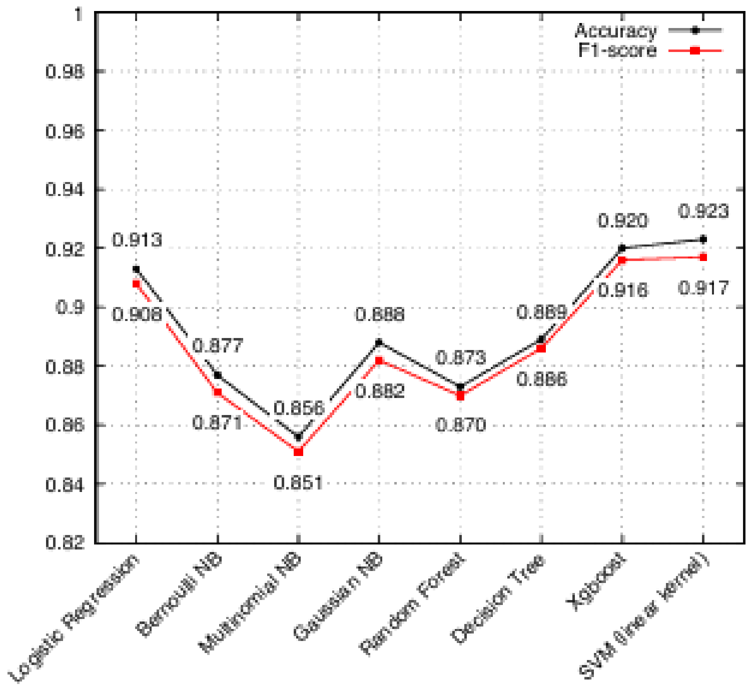
Performance of Deep learning models on the consolidated dataset with Word2Vec embedding

<u>Performance of ML models using bag of words feature:</u>

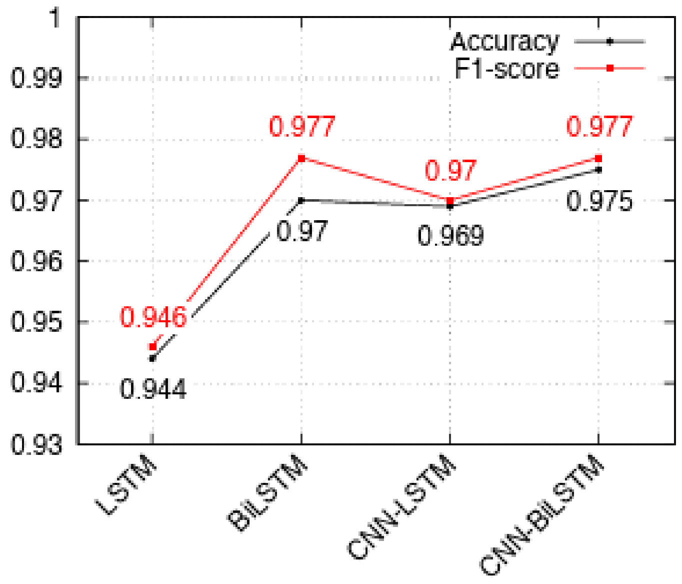

Accuracy
F1-score
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.904
0.897
0.881
0.875
0.890
0.886
0.888
0.882
0.892
0.888
0.889
0.886
0.920
0.916
0.909
0.904
Logistic Regression
Bernoulli NB
Multinomial NB
Gaussian NB
Random Forest
Decision Tree
Xgboost
SVM (linear kernel)

Performance of ML models using

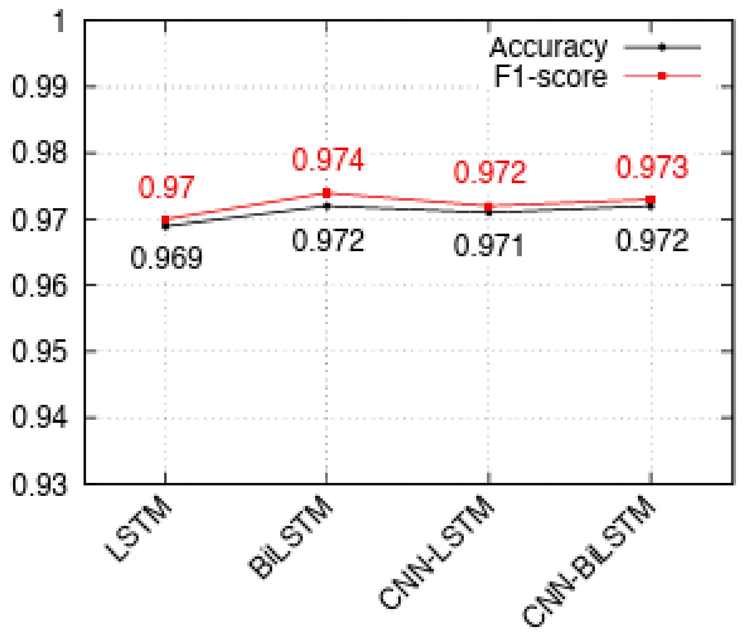
TF-TDF feature:



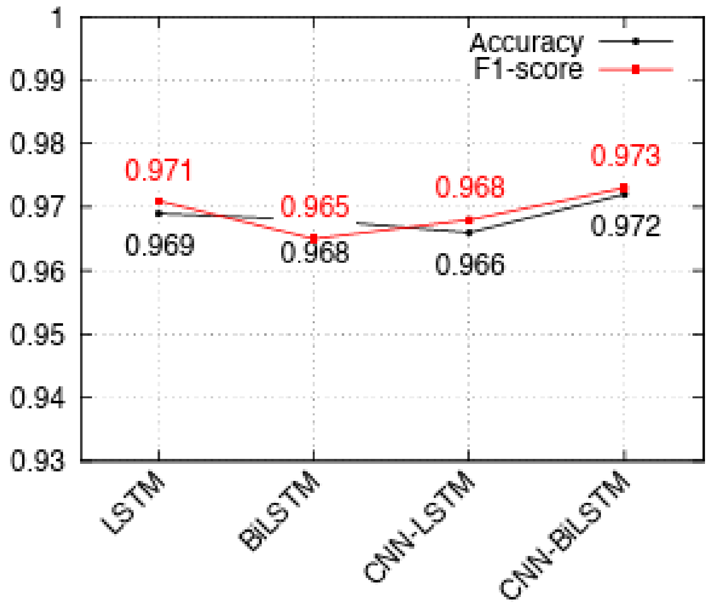
Performance of DL models using word2vec embedding:



Performance of GloVe Embedding:



Performance of Fasttext Embedding:


Accuracy
F1-score
LSTM
BiLSTM
CNN-LSTM
CNN-BiLSTM