



**Indian Institute of Technology (BHU), Varanasi**

भारतीय  
प्रौद्योगिकी  
संस्थान  
काशी हिन्दू विश्वविद्यालय

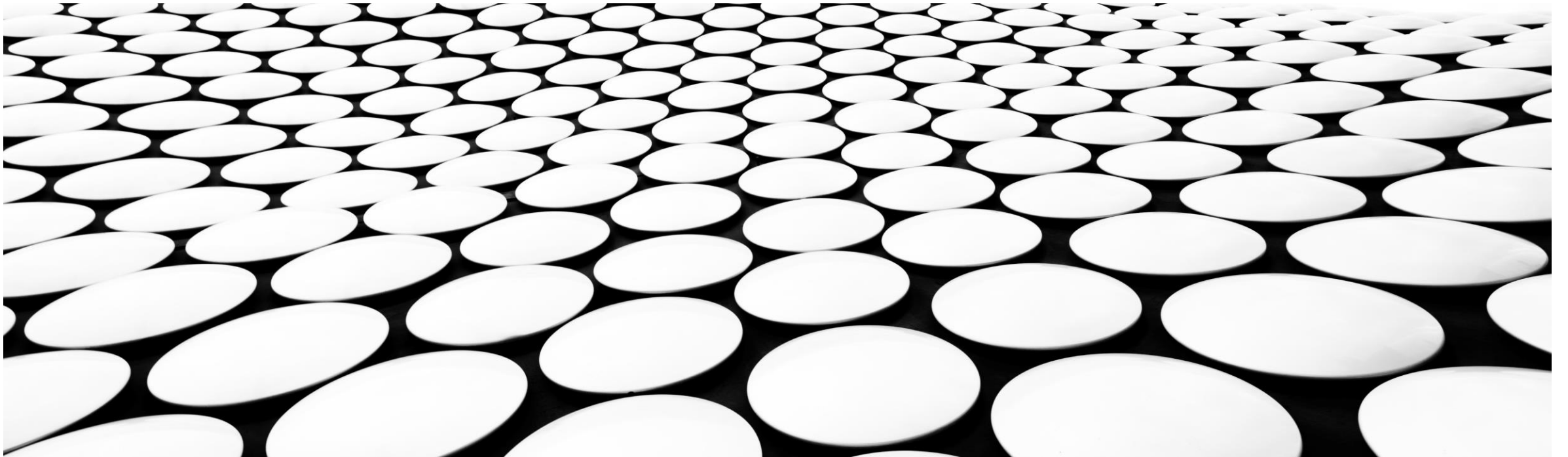


INDIAN  
INSTITUTE OF  
TECHNOLOGY  
BANARAS HINDU UNIVERSITY

---


# EXPLORATORY PROJECT

SENTIMENT ANALYSIS USING SOCIAL MEDIA POSTS



---

## TEAM MEMBERS



**BIRESH  
AGARWAL  
(20095030)**

**MAYANK  
SINGH  
(20095065)**

**PRASHASTI  
TRIPATHI  
(20095083)**

**SANIDHYA  
TAPARIA  
(20095100)**

---

## **ACKNOWLEDGEMENT**

We would like to thank our highly respected and esteemed teacher Dr. Shivam Verma, who gave us this opportunity to work on this project. His useful suggestions for this whole work are deeply acknowledged.

We got to learn a lot from this project about NLP and ML.

At last, we would like to extend my heartfelt thanks to our parents, friends and seniors, because without their help this project would not have been successful.

# OVERVIEW



*Sentiment Analysis* is the process of determining the emotional tone behind a series of words, used to gain an understanding of the the **attitudes**, **opinions** and **emotions** expressed within an **online** mention.

We have used the **twitter sentiment analysis dataset** available on **Kaggle** after processing it.

WHAT

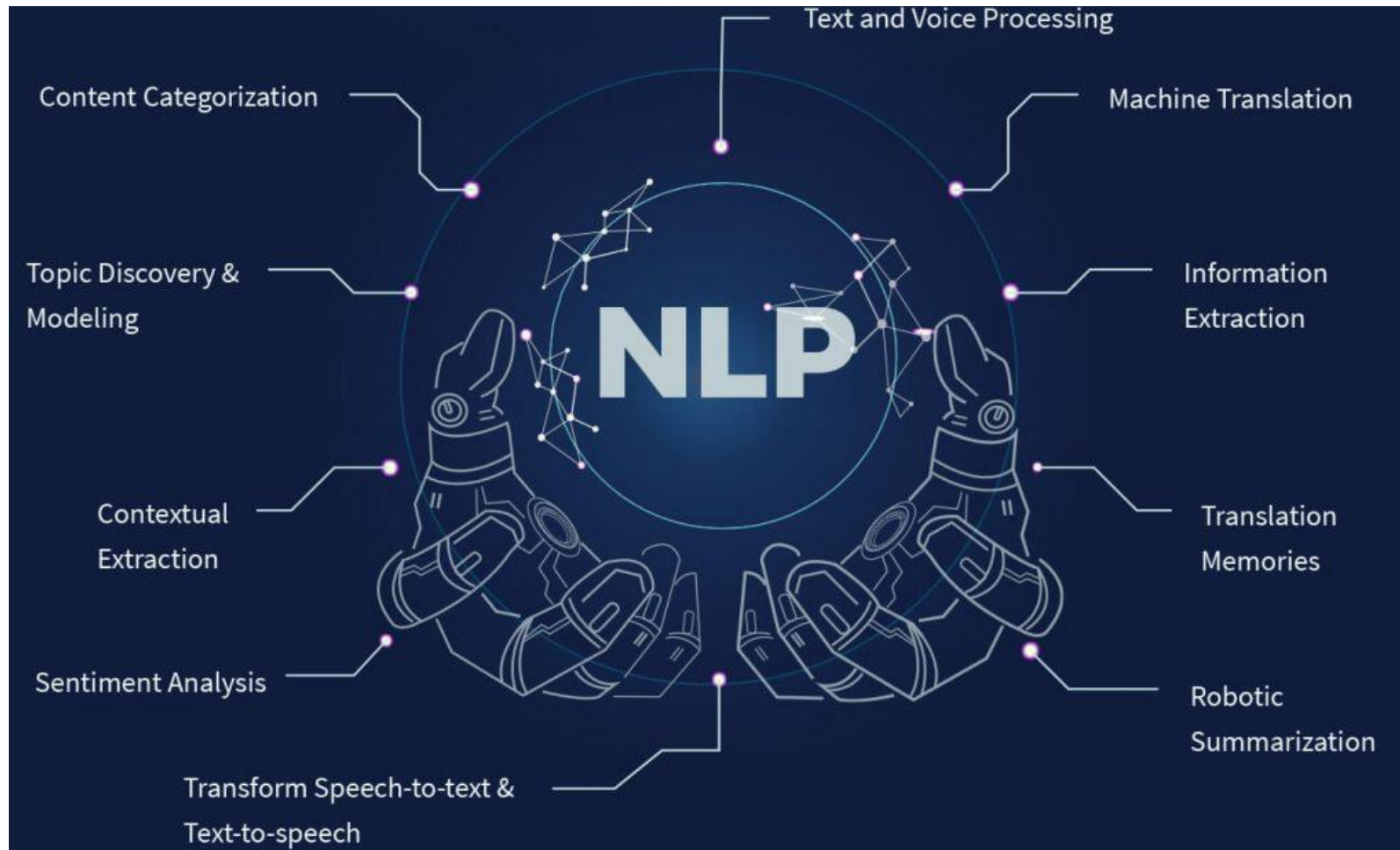
USES

DATASET

It is extremely useful in **social media monitoring** as it allows us to gain an overview of the **wider public opinion** behind certain topics. Sentiment analysis has become an integral part of **marketing** and can help government agencies to analyse **mental health** problems.

# CHALLENGES

- The main problems that exist in the current techniques are: inability to perform well in different domains, inadequate accuracy and performance in sentiment analysis based on insufficient labeled data, incapability to deal with complex sentences that require more than sentiment words and simple analyzing.
- For example-Polarity, words such as “love” and “hate” are high on positive (+1) and negative (-1) scores in polarity. These are easy to understand., but there are in-between conjugations of words such as “not so bad” that can mean “average” and hence lie in mid-polarity (-0.75). Sometimes phrases like these get left out, which dilutes the sentiment score.
- Tone can be difficult to interpret verbally, and even more difficult to figure out in the written word. Things get even more complicated when one tries to analyze a massive volume of data that can contain both subjective and objective responses. Brands can face difficulties in finding subjective sentiments and properly analyzing them for their intended tone.
- The problem with social media content that is text-based, like Twitter, is that they are inundated with emojis. NLP tasks are trained to be language specific. While they can extract text from even images, emojis are a language in itself. Most emotion analysis solutions treat emojis like special characters that are removed from the data during the process of text mining. But doing so means that companies will not receive holistic insights from the data.
- Idioms like actually Machine learning programs don't necessarily understand a figure of speech. For example, an idiom like “not my cup of tea” will boggle the algorithm because it understands things in the literal sense. Hence, when an idiom is used in a comment or a review, the sentence can be misconstrued by the algorithm or even ignored. To overcome this problem a sentiment analysis platform needs to be trained in understanding idioms. When it comes to multiple languages, this problem becomes manifold .Likewise there exists many more problems in this domain.



# NATURAL LANGUAGE PROCESSING



It refers to the use of algorithms to determine properties of natural or human language so that computers can understand what humans have written or said. NLP includes teaching computer systems how to extract data from bodies of written text, translate from one language to another, and recognize printed or handwritten words. Notably, NLP is the field that allows for our everyday use of virtual assistants.



The best-known library for NLP is the Natural Language Toolkit, or the NLTK.



## NLP & NEURAL NETWORKS

- The use of statistics in NLP started in the 1980s and heralded the birth of what we called **Statistical NLP** or **Computational Linguistics**. Since then, many machine learning techniques have been applied to NLP. These include naïve Bayes, k-nearest neighbours, hidden Markov models, conditional random fields, decision trees, random forests, and support vector machines.
- The use of neural networks for NLP did not start until the early 2000s. But by the end of 2010s, neural networks transformed NLP, enhancing or even replacing earlier techniques. This has been made possible because we now have more data to train neural network models and more powerful computing systems to do so.
- In traditional NLP, features were often hand-crafted, incomplete, and time consuming to create. Neural networks can learn multilevel features automatically. They also give better results.

# MODELS AND TECHNIQUES

NOW WE WOULD PRESENT TO YOU THE MACHINE  
LEARNING MODELS AND NEURAL NETWORKS  
WHICH WE HAVE USED IN OUR PROJECT.

# FINE TUNING BERT FOR SENTIMENT ANALYSIS

Created a baseline model, in which we have used a TF-IDF vectorizer and a Naive Bayes classifier and got accuracy of **78.18%**. The transformers library can help us fine-tune the state-of-the-art BERT model and yield an accuracy rate **10%** higher than the baseline model.

# NAIVE BAYES CLASSIFIER

THE PROBABILITY OF "B"  
BEING TRUE GIVEN THAT  
"A" IS TRUE



THE PROBABILITY  
OF "A" BEING  
TRUE



$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

↑  
THE PROBABILITY  
OF "A" BEING TRUE  
GIVEN THAT "B" IS  
TRUE

↖  
THE PROBABILITY  
OF "B" BEING  
TRUE



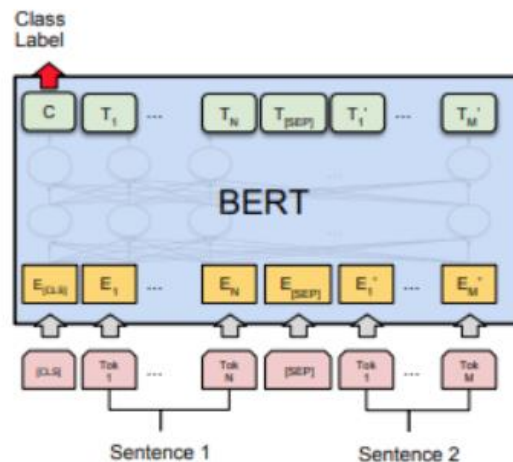
# Towards Naïve Bayesian Classifier

---

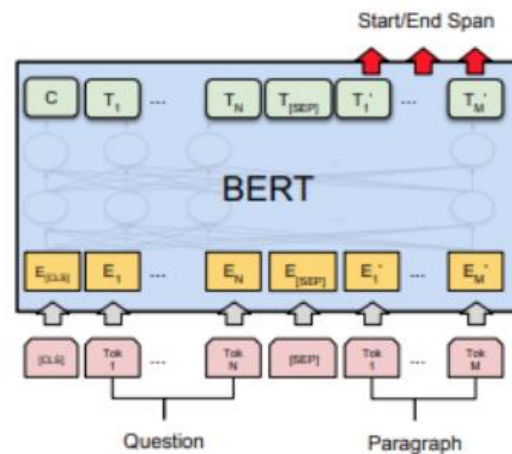
- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

- This greatly reduces the computation cost:  
Only counts the class distribution



(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



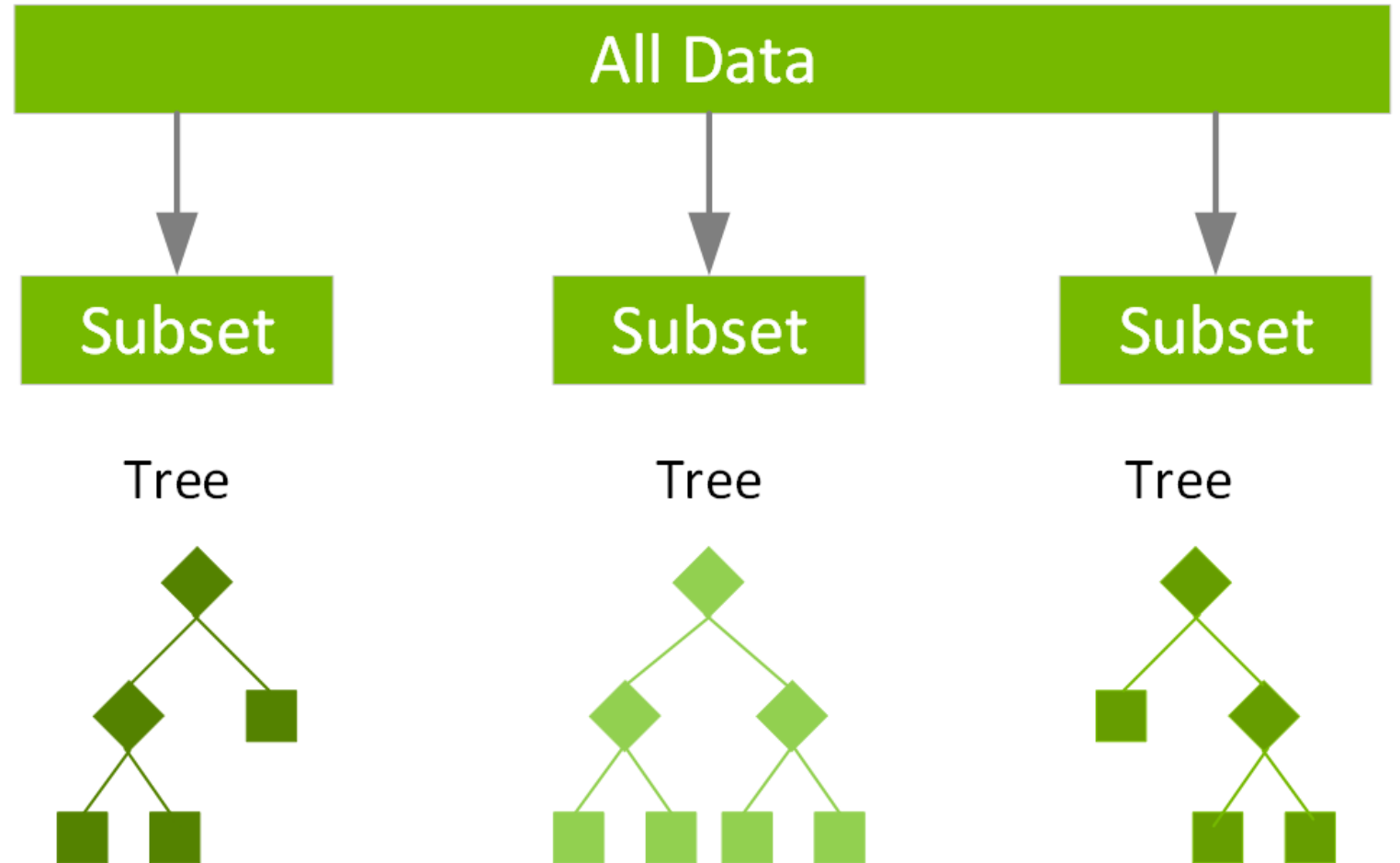
(c) Question Answering Tasks:  
SQuAD v1.1

- BERT makes use of **Transformer**, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an **encoder** that reads the text input and a **decoder** that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism of the Transformer is necessary.

# XGBoost

- **XGBoost**, which stands for **Extreme Gradient Boosting**, is an implementation of Gradient Boosted decision trees.
- XGBoost is especially widespread because it has been the winning algorithm in a number of recent Kaggle competitions (open data science competitions for prediction or any other kind of task).
- The training dataset of this algorithm contains users' review texts, summaries and their overall rating related to the mentioned product.
- This algorithm takes a text string as an input and loads an XGBoost model trained on the Twitter Tweets Dataset to predict its sentiment. Its output is in two basic categories: positive or negative; symbolized as 1 or 0. Its output is in two basic categories: positive or negative; symbolized as 1 or 0.

## LOGIC BEHIND XGBOOST ALGORITHM:





# WHY XGBOOST?



## STEPS INVOLVED IN ITS DATA PRE-PROCESSING:

- Filter out *symbols* (eg. question marks, full stops, etc)
- Remove *stop words* (ie. the non-polarising words like “wasn’t”, “shan’t”, etc)
- Text Normalization or *Tokenizing*.
- *Stemming* (For example, if we were to stem these words [‘connect’, ‘connected’, ‘connection’, ‘connects’], we’d get the word ‘connect’ for all of those instances.)
- *Word Embedding* or Vectorising (we have called the matrix as Bag of Words).

## IMPLEMENTING THE MODEL:

```
# Bag of words features
# Extracting train and test BoW features

train_bow = bow[:1600000,:]
test_bow = bow[1600000:,:]
# print(train_bow.shape[0])
train_bow.shape[0]==data['target'].shape[0]
# # splitting data into training and validation set
xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, data['target'], random_state=42, test_size=0.3)

# XGB Classifier on training data(15m)
from sklearn.metrics import f1_score, accuracy_score
from xgboost import XGBClassifier
xgb_model = XGBClassifier(max_depth=6, n_estimators=1000).fit(xtrain_bow, ytrain)
prediction = xgb_model.predict(xvalid_bow)
accuracy_score(yvalid, prediction)

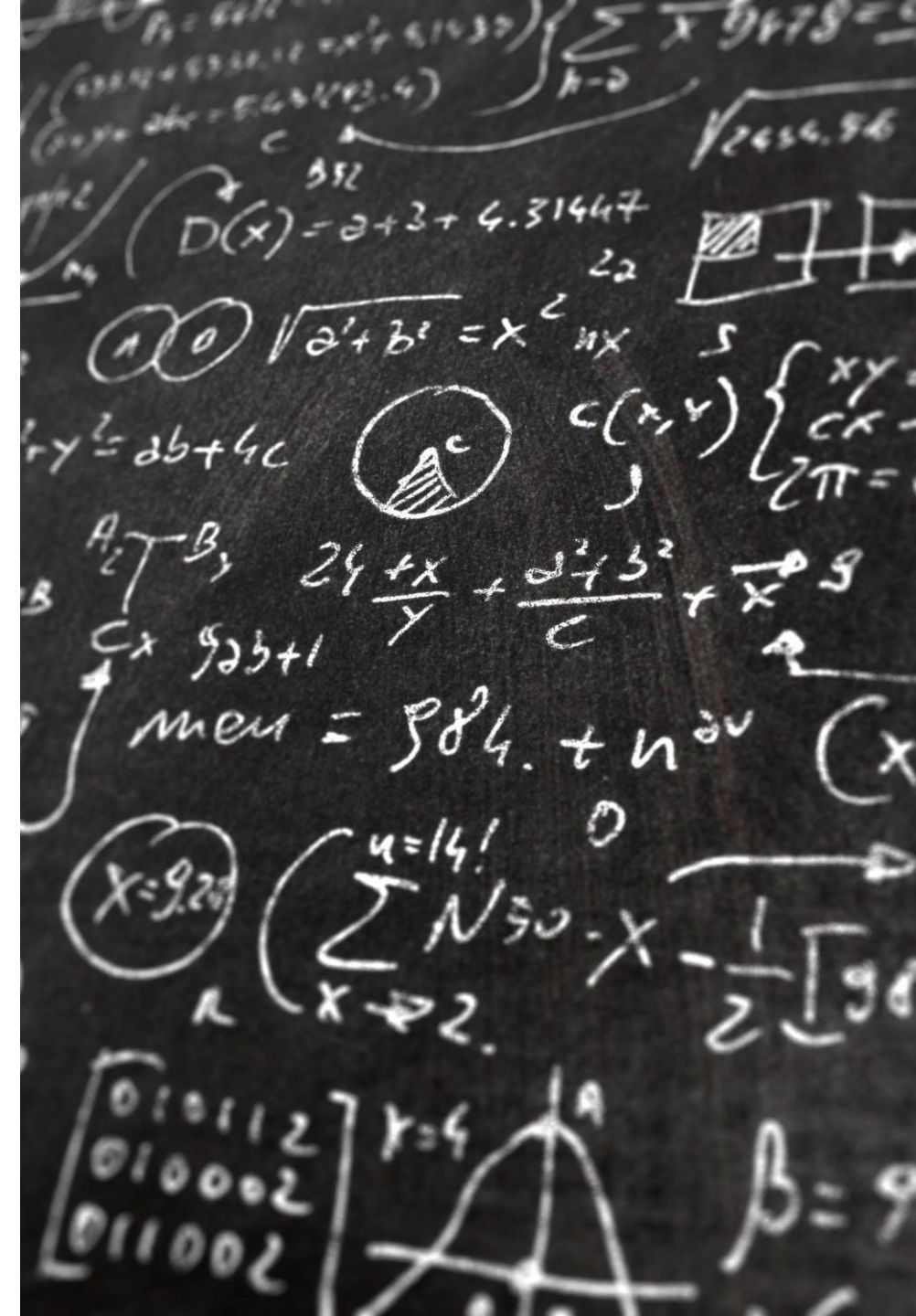
0.7348729166666667

# # XBG Classifier on test data
# xgb_model = XGBClassifier(max_depth=6, n_estimators=1000).fit(xtrain_bow, ytrain)
prediction = xgb_model.predict(xtrain_bow)
accuracy_score(ytrain, prediction)

0.7399723214285714
```

## CONCLUSION:

We get a score of 73.99% which is not bad for first attempt. We can improve the score by optimizing the algorithm. One way of achieving that is through hyperparameter tuning. The above score is solely based on the training data, we are interested on how the model works on unseen data.



# Embedding + Stacked LSTM

## Table of Contents

Embedding Explained

LSTM Explained

Tokenization

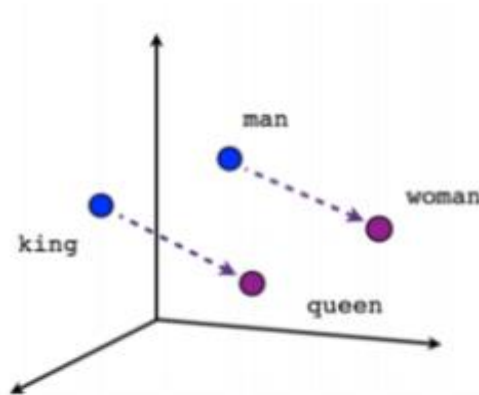
Word Embedding using GloVe

Model Training

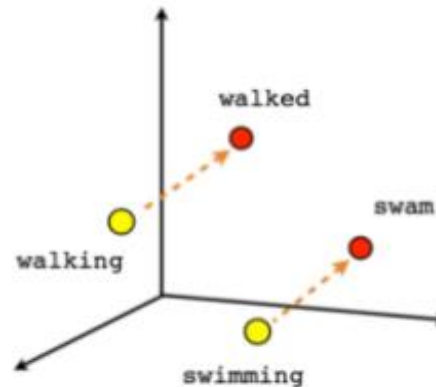
Conclusion

# What is Embedding??

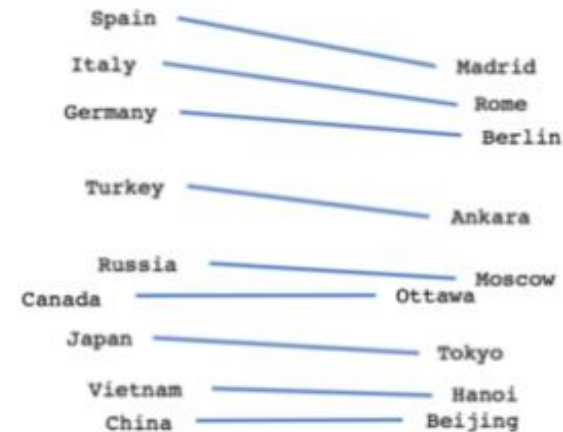
- A word embedding is a learned representation for text where words that have the same meaning have a similar representation.
- For example, words like “mom” and “dad” should be closer together than the words “mom” and “ketchup” or “dad” and “butter”.



Male-Female



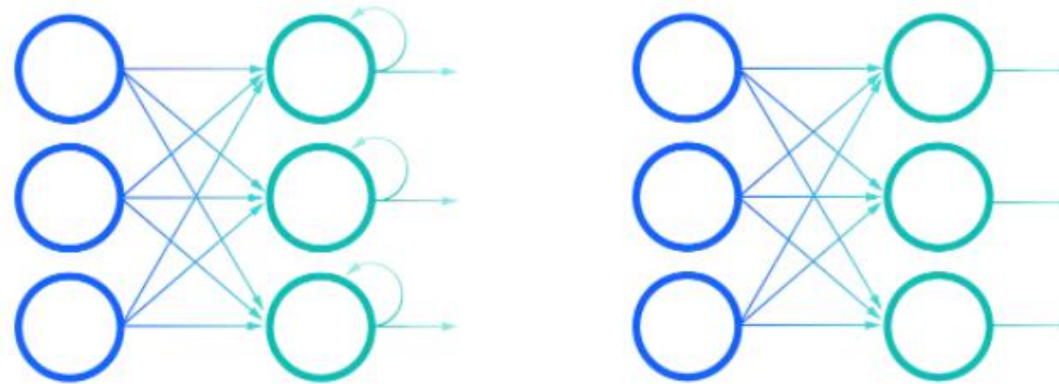
Verb tense



Country-Capital

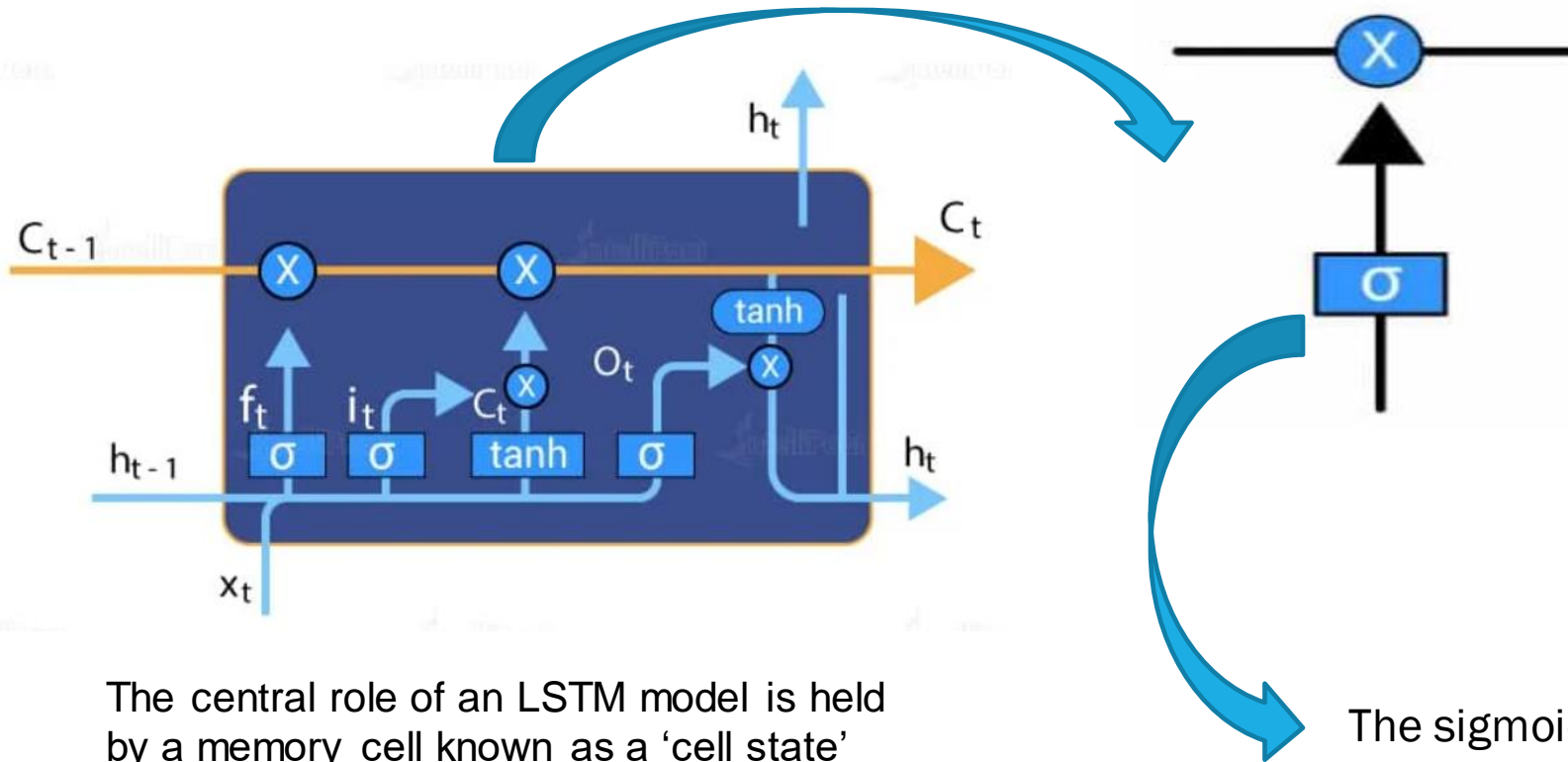
# What is LSTM??

- LSTM (Long Short Term Memory Network) is an advanced type of RNN (Recurrent neural network).
- RNN is a type of artificial neural network which uses sequential data or time series data. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence.



Comparison of Recurrent Neural Networks (on the left) and Feedforward Neural Networks (on the right)

# Logic Behind LSTM



The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time..

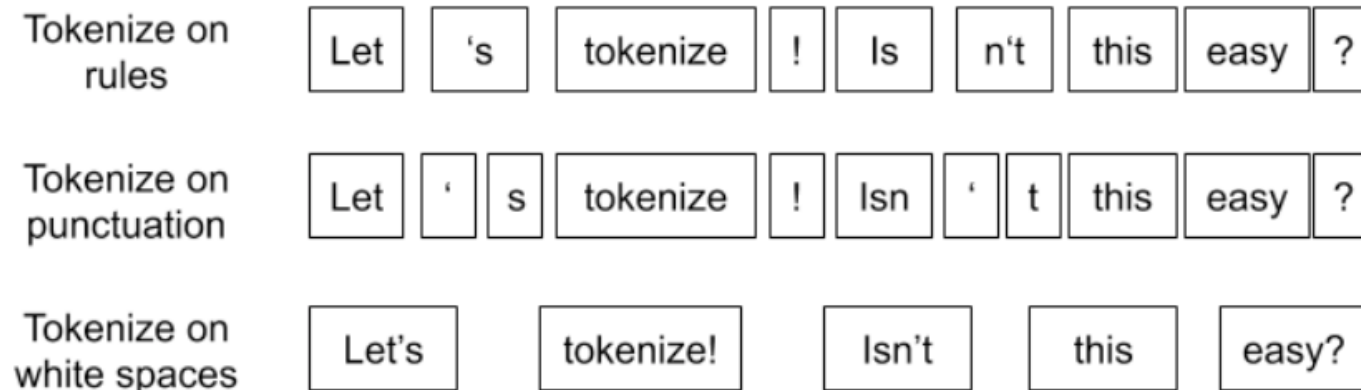
Information is regulated by gates containing a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.

The sigmoid layer gives out numbers between zero and one, where zero means 'nothing should be let through,' and one means 'everything should be let through.'



# Tokenization

- It is a particular kind of document segmentation. It breaks up text into smaller chunks or segments called tokens.
- A tokenizer breaks unstructured data, natural language text, into chunks of information that can be counted as discrete elements.

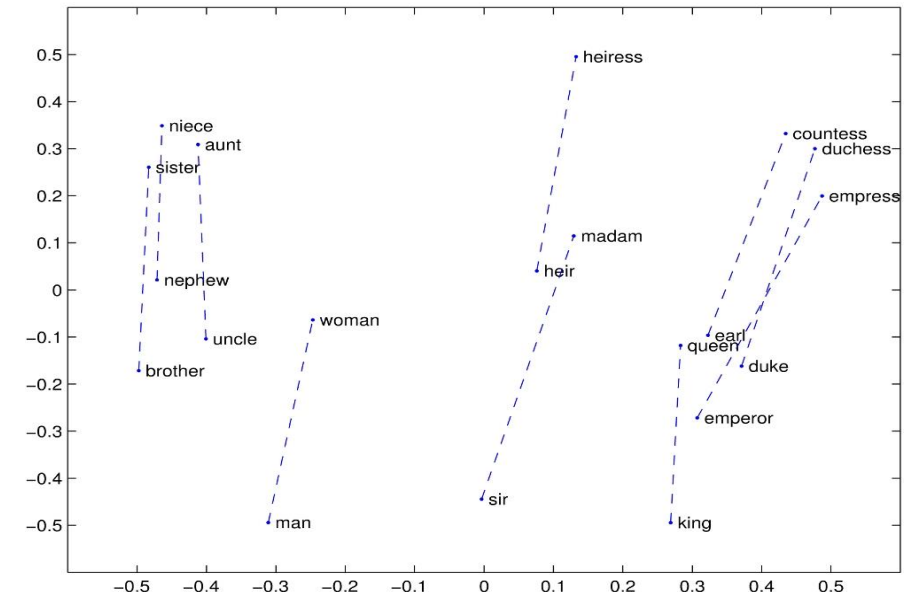
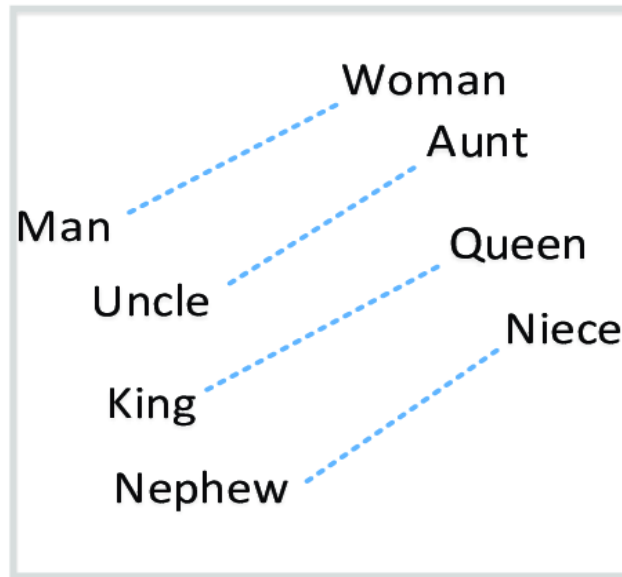
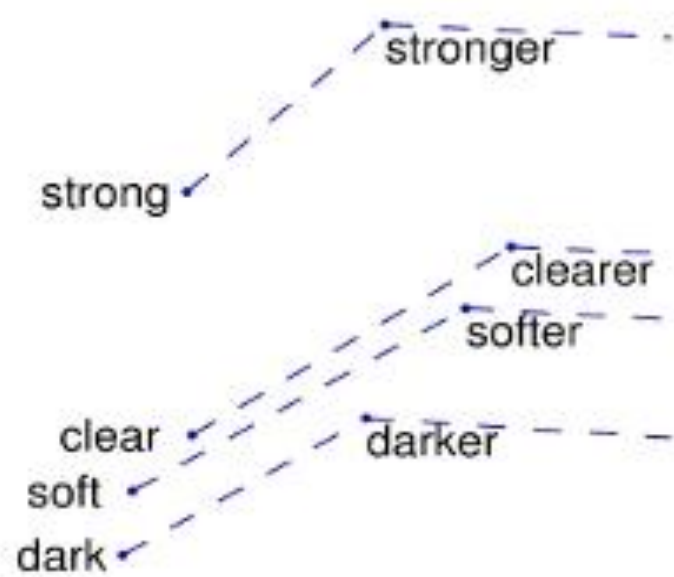


**Let's tokenize! Isn't this easy?**



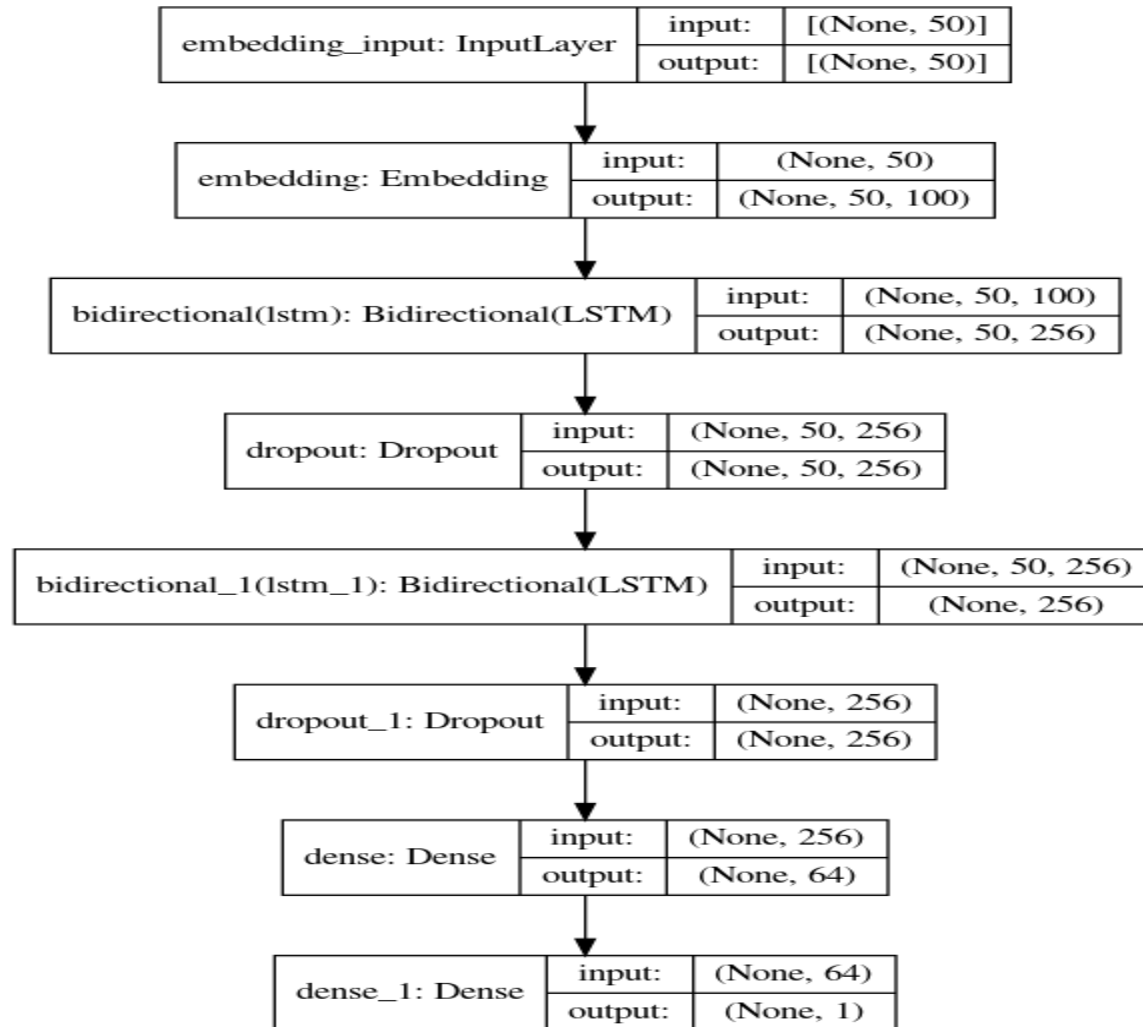
# Word Embedding using GloVe

- GloVe stand for Global Vectors for Word Representation and it is an unsupervised learning algorithm for obtaining vector representations for words.
- Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.



Examples of Word Embedding using GloVe

# Model Structure



# Conclusion

The model gave an overall validation accuracy of 77% with a macro-avg of 77% and also weighted average of 77% which can be seen in the below output printed using `classification_report` function of **skLearn**.

	precision	recall	f1-score	support
0	0.78	0.75	0.76	79800
1	0.76	0.79	0.77	80200
accuracy			0.77	160000
macro avg	0.77	0.77	0.77	160000
weighted avg	0.77	0.77	0.77	160000



**THANKYOU**