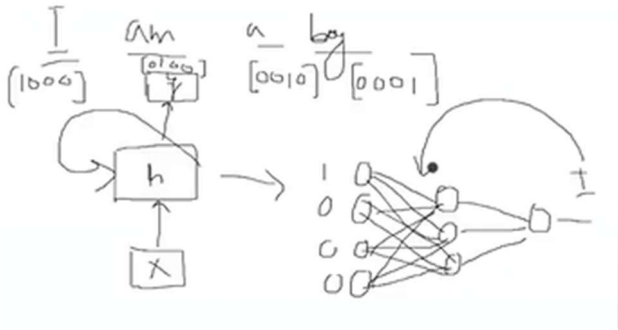
RNN – means sequential architecture

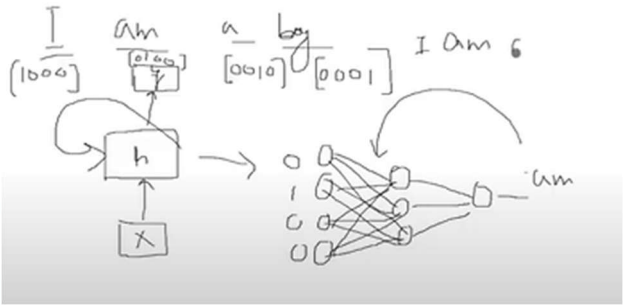
Example: I am a boy (it is token / sequence)

Sequence convert into embedding / vectors: I [1000] + am [0100] + a [0010] + boy [0001]

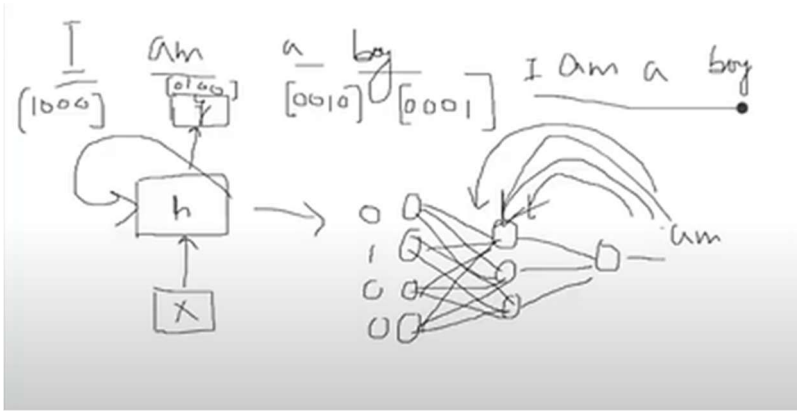
For unfold in terms of RNN – 4words



1st input store (that is I)



2nd input store (that is am)



Like this all inputs store (means it learn input in form of sequence)

RNN is good for small data only. Once bigger data/sequences is input it can't be able to give results. To resolve this use LSTM.

Two types of models: Homogenous model and Heterogeneous model

Reference: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

What Is a Neural Network?

A Neural Network consists of different layers connected to each other, working on the structure and function of a human brain. It learns from huge volumes of data and uses complex algorithms to train a neural net.

Several neural networks can help solve different business problems. Let's look at a few of them.

- Feed-Forward Neural Network: Used for general Regression and Classification problems.
- Convolutional Neural Network: Used for object detection and image classification.
- Deep Belief Network: Used in healthcare sectors for cancer detection.
- RNN: Used for speech recognition, voice recognition, time series prediction, and natural language processing.

What Are Recurrent Neural Networks (RNN)?

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to process sequences of data. They work especially well for jobs requiring sequences, such as time series data, voice, natural language, and other activities.

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

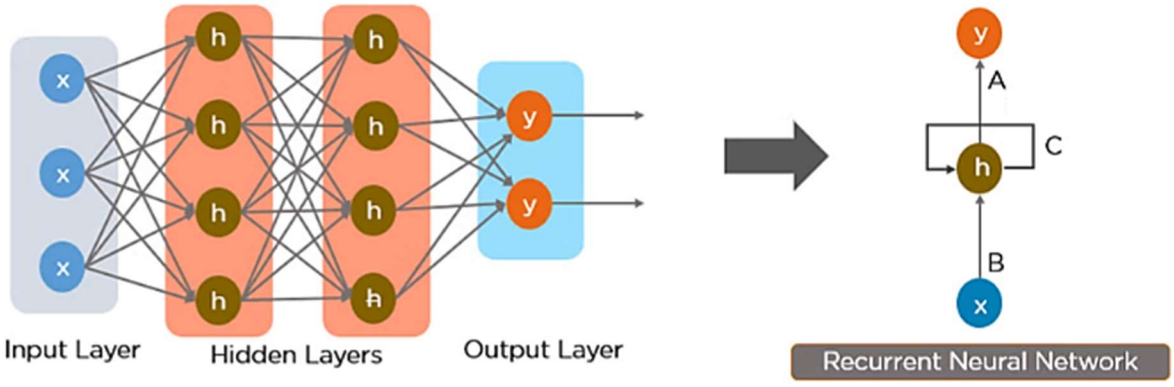
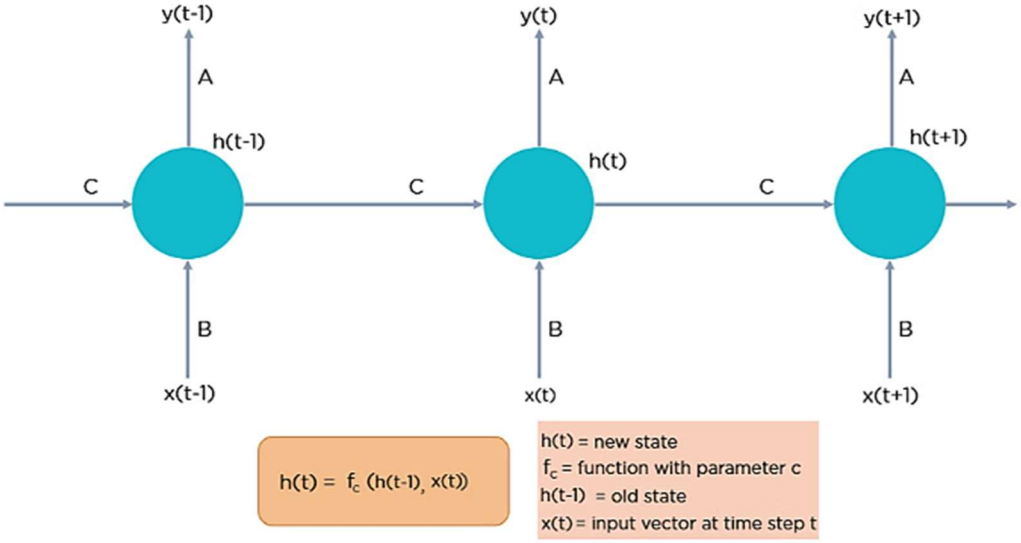


Fig: Simple Recurrent Neural Network

The output at any given time is fetched back to the network to improve on the output. Image shows unfolding of RNN



Why Recurrent Neural Networks?

RNN were created because there were a few issues in the feed-forward neural network:

- Cannot handle sequential data
- Considers only the current input
- Cannot memorize previous inputs

How Does Recurrent Neural Networks Work?

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.

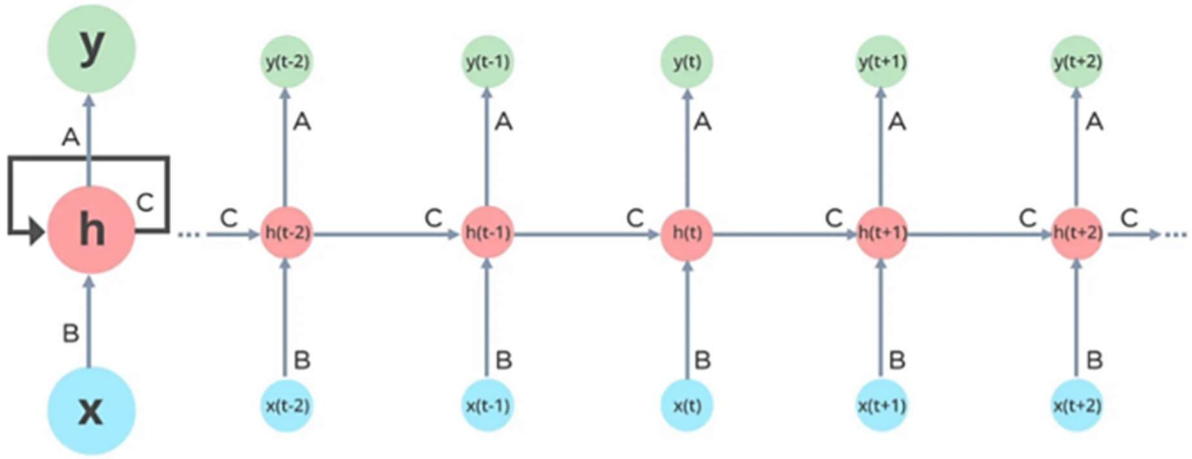


Fig: Working of Recurrent Neural Network

Feed-Forward Neural Networks vs Recurrent Neural Networks

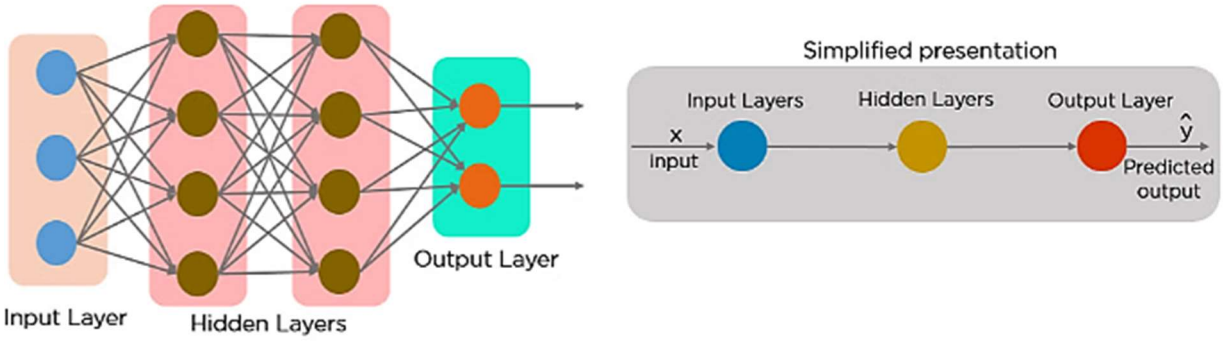


Fig: Feed-forward Neural Network

Applications of Recurrent Neural Networks

- Image captioning
- Time series prediction
- Natural language processing
- Machine Translation

Types of Recurrent Neural Networks

There are four types of Recurrent Neural Networks:

- One to One
- One to Many
- Many to One
- Many to Many

Two Issues of Standard RNNs

1. Vanishing Gradient Problem

Recurrent Neural Networks enable you to model time-dependent and sequential data problems, such as stock market prediction, machine translation, and text generation.

RNNs suffer from the problem of vanishing gradients. The gradients carry information used in the RNN, and when the gradient becomes too small, the parameter updates become insignificant. This makes the learning of long data sequences difficult.

2. Exploding Gradient Problem

While training a neural network, if the slope tends to grow exponentially instead of decaying, this is called an Exploding Gradient. This problem arises when large error gradients accumulate, resulting in very large updates to the neural network model weights during the training process.

Variant RNN Architectures

LSTM Networks: LSTMs handle long-term dependencies by using input, forget, and output gates to control information flow, preventing the vanishing gradient problem.

GRU Networks: GRUs simplify LSTMs with only reset and update gates, offering efficient long-term memory management.

Bidirectional RNNs: These process sequences in both forward and backward directions, capturing context from past and future inputs.

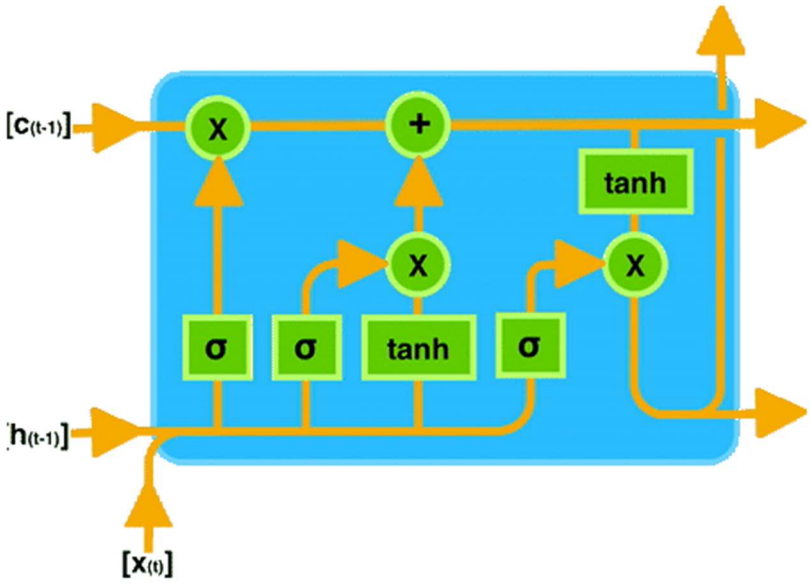
Encoder-Decoder RNNs: Used for sequence-to-sequence tasks, this architecture encodes input into a fixed-length vector, which a decoder then uses to generate output.

Attention Mechanisms: Attention selectively focuses on relevant parts of the input sequence, improving performance on tasks with long inputs.

Reference: https://medium.com/@sachinsoni600517/unlocking-the-power-of-long-short-term-memory-lstm-networks-7a02da292d7a

Long Short-Term Memory (LSTM)

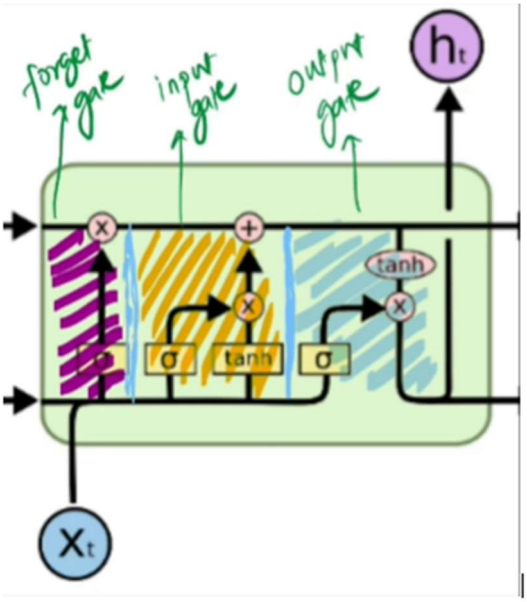
Long Short-Term Memory (LSTM) networks emerge as powerful tools. In this introductory journey, we'll unravel the mysteries of LSTM networks, exploring their architecture and practical applications.



LSTM Architecture:

The architecture of LSTM includes three key components: The **forget gate**, which decides what information to discard from the long-term memory, the **input gate**, which determines what new information to store in the long-term memory, and the **output** gate in LSTM determines what information from the long-term memory is used to produce the final output of the LSTM cell at a particular time step.

It regulates the flow of information from the long-term memory to the current cell output, ensuring that only relevant information is considered in generating the output.



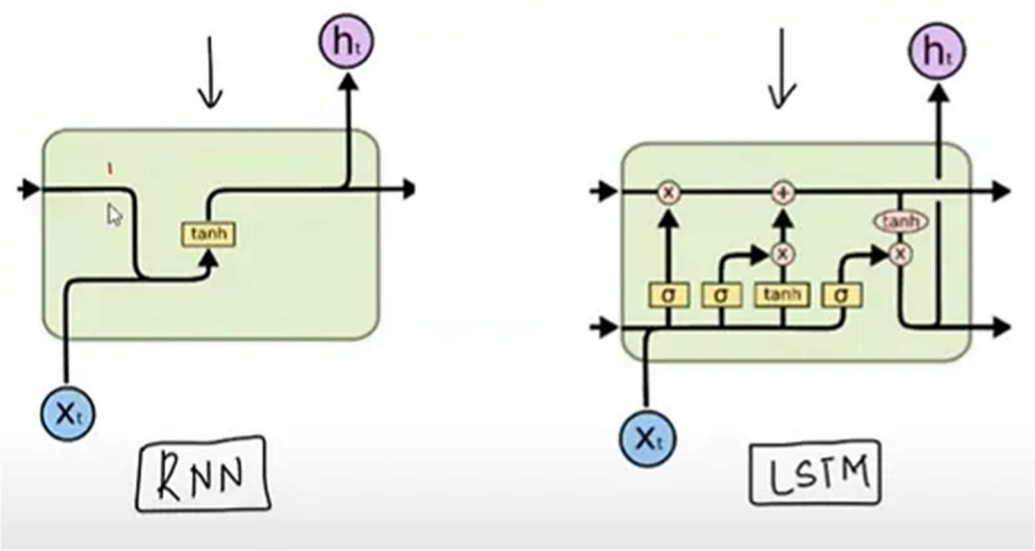
RNN VS LSTM

RNN (Recurrent Neural Network):

- Designed for sequential data, RNNs have loops that allow information to persist.
- They suffer from the "vanishing gradient" problem, making it difficult to learn long-term dependencies.
- Suitable for short sequences, but less effective for longer sequences due to their simpler structure.

LSTM (Long Short-Term Memory):

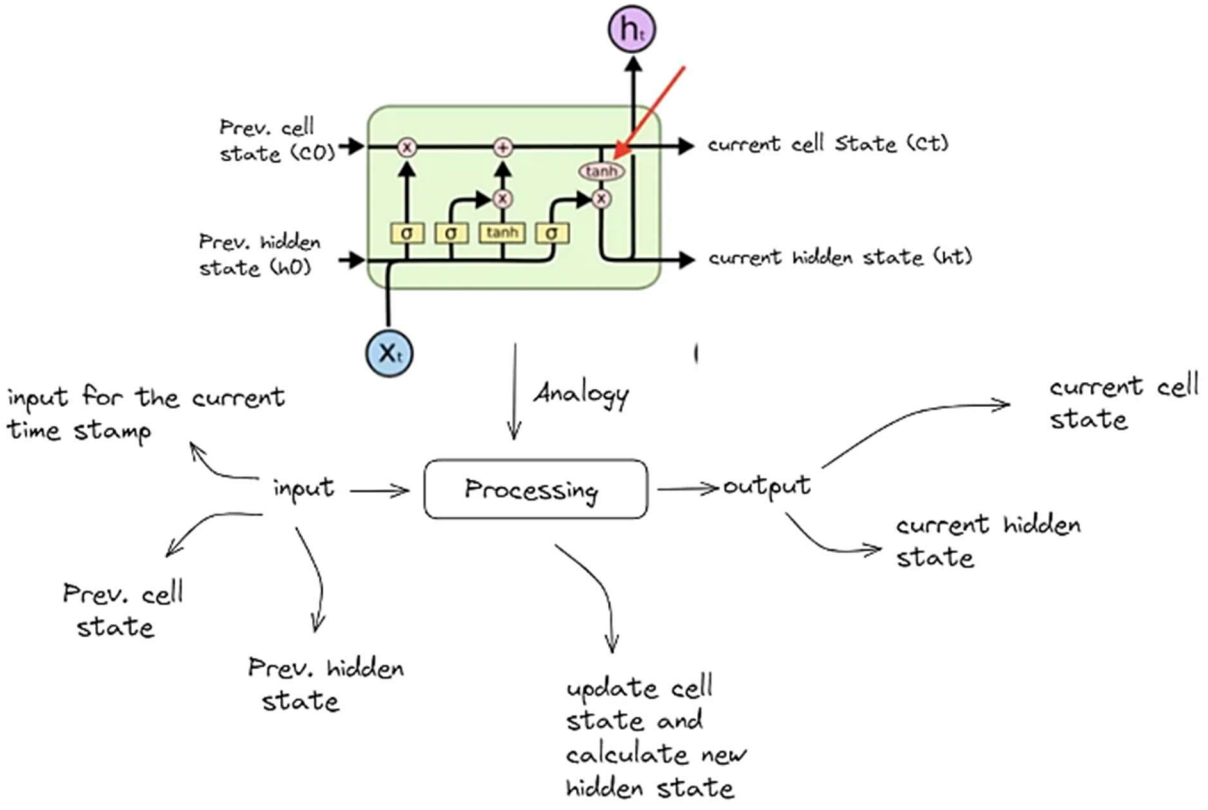
- A type of RNN specifically designed to overcome the limitations of traditional RNNs.
- Uses special units called memory cells with gates (input, forget, output) to control the flow of information.
- Effectively captures long-term dependencies, making it ideal for tasks involving longer sequences like time-series prediction and language modeling.



RNN vs LSTM

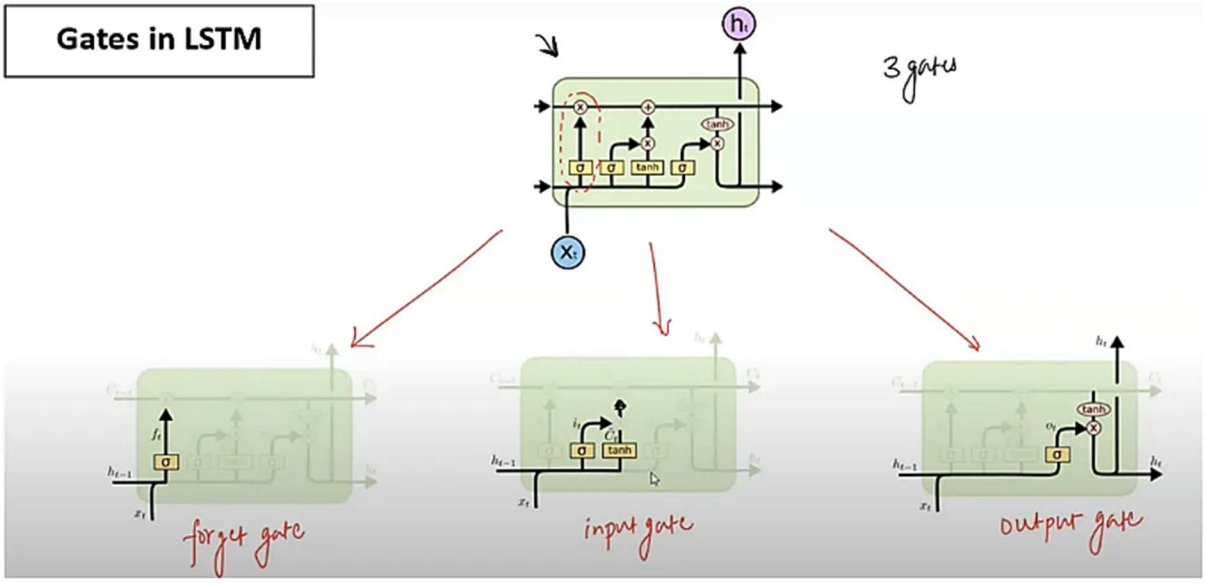
LSTM Working :

In the LSTM model, we can think of three main stages: input, processing, and output. At the input stage, we have three inputs: the input for the current state, previous cell state, and previous hidden state. During processing, the model updates the cell state(c0 -> ct) and calculates the new hidden state (h0 -> ht).



Finally, at the output stage, we have two outputs: the current cell state and the current hidden state. These outputs provide the information needed for further processing or decision-making in the model.

Understanding the gates in LSTM:



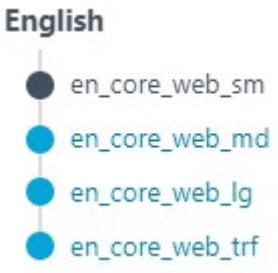
For Natural language or RNN two mostly used libraies are:

- NLTK – Natural language tool kit
- Spacy Website: https://spacy.io/models/en

SPACY

English pipeline optimized for CPU. Components: tok2vec, tagger, parser, senter, ner, attribute_ruler, lemmatizer.

LANGUAGE	EN English
TYPE	CORE Vocabulary, syntax, entities
GENRE	WEB written text (blogs, news, comments)
SIZE	SM 12 MB
COMPONENTS	tok2vec, tagger, parser, senter, attribute_ruler, lemmatizer, ner
PIPELINE	tok2vec, tagger, parser, attribute_ruler, lemmatizer, ner
VECTORS	0 keys, 0 unique vectors (0 dimensions)



Above image shows models

Steps for NLP Model

1. Text Collection

Data Gathering: Collect raw text data from various sources such as web scraping, APIs, databases, or manually entered text.

2. Text Preprocessing

Text Cleaning: Remove irrelevant characters, punctuation, special symbols, and HTML tags.

Lowercasing: Convert all text to lowercase to ensure uniformity.

Tokenization: Split the text into individual words or tokens.

Stopwords Removal: Remove common, non-informative words (e.g., "the," "and," "is") that do not add significant value.

Stemming: Reduce words to their root form by chopping off prefixes or suffixes (e.g., "running" to "run").

Lemmatization: Convert words to their dictionary form (e.g., "better" to "good").

Text Normalization: Handle misspellings, abbreviations, and other inconsistencies in the text.

Handling Special Words: Address specific cases like names, dates, numbers, etc.

3. Feature Extraction

Vectorization: Convert text data into numerical representations using methods like:

Bag of Words (BoW): Represents text as a set of word occurrences.

TF-IDF (Term Frequency-Inverse Document Frequency): Weighs words based on their frequency and importance.

Word Embeddings: Convert words into dense vectors capturing semantic meaning (e.g., Word2Vec, GloVe, FastText).

Sentence Embeddings: Create vector representations for entire sentences (e.g., BERT, GPT).

4. Model Selection

Traditional Models:

Naive Bayes: A probabilistic classifier based on Bayes' theorem.

Support Vector Machines (SVM): Finds the hyperplane that best separates classes.

Logistic Regression: Used for binary or multi-class classification.

Deep Learning Models:

RNNs (Recurrent Neural Networks): Handles sequential data, such as time series or text.

LSTMs/GRUs: Variants of RNNs that handle long-term dependencies better.

CNNs (Convolutional Neural Networks): Often used for text classification tasks.

Transformers: State-of-the-art models like BERT, GPT, and T5 that understand context and sequence in text.

5. Model Training

Data Splitting: Split the data into training, validation, and test sets.

Model Training: Train the model on the training data.

Hyperparameter Tuning: Adjust model parameters to optimize performance (e.g., learning rate, batch size).

Cross-Validation: Validate the model using different data subsets to ensure robustness.

6. Model Evaluation

Accuracy: The percentage of correctly predicted labels.

Precision, Recall, F1-Score: Metrics to evaluate classification performance, especially for imbalanced datasets.

Confusion Matrix: A table that shows the true positives, true negatives, false positives, and false negatives.

AUC-ROC Curve: Evaluates the model's ability to distinguish between classes.

Perplexity: A measure used in language models to evaluate the quality of predictions.

7. Model Deployment

Model Serialization: Save the trained model using formats like .h5, .pb, or .pkl.

Serving: Deploy the model in a production environment using REST APIs, microservices, or cloud platforms.

Monitoring: Track model performance over time and retrain as necessary.

8. Post-Processing

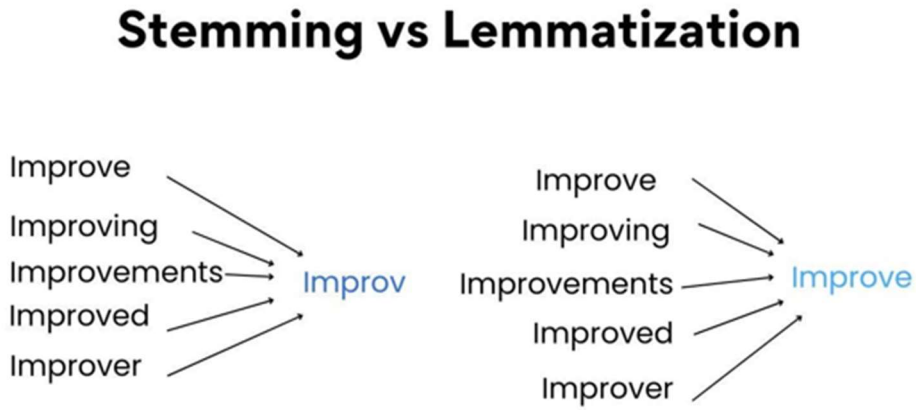
Result Interpretation: Convert the model's output back to human-readable text or labels.

Error Analysis: Analyze incorrect predictions to improve the model.

Feedback Loop: Incorporate user feedback for continuous model improvement.

This workflow can vary depending on the specific NLP task (e.g., text classification, sentiment analysis, machine translation, etc.), but these steps provide a comprehensive overview of the process.

Stemming Vs Lemmatization



Types of Stemming:

1. **Porter Stemmer:**

Developed by Martin Porter in 1980, the Porter Stemmer is one of the most popular stemming algorithms. It applies a set of rules to iteratively strip suffixes from words to reduce them to their root form. It's relatively simple and widely used in text processing.

Example: "caresses" becomes "caress," "flies" becomes "fli," "troubled" becomes "troubl."

Snowball Stemmer:

Also developed by Martin Porter, the Snowball Stemmer is an improvement over the original Porter Stemmer. It's often referred to as the "Porter2" stemmer.

The Snowball Stemmer uses a more refined and efficient set of rules and is designed to be more consistent and accurate across different languages. It supports multiple languages, making it more versatile than the original Porter Stemmer.

Example: "running" becomes "run," "happily" becomes "happi."

The Snowball Stemmer is generally considered more advanced and flexible, particularly for multilingual applications.

Note: Stemming and Lemmatization apply where words are nit important,