18CSE359T Natural Language Processing

- Offered to Final Year B.Tech. CSE
- By Dept. of C.Tech.

Unit 5 Topics

- Introduction to Probabilistic Approaches
- Statistical Approaches to NLP Tasks
- Sequence Labeling
- Problems - Similarity Measures
- Word Embeddings
- CBOW
- Skip-gram
- Sentence Embeddings
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)

Introduction to Probabilistic Approaches

Probabilistic Approaches

- Probabilistic modeling is a ***statistical approach (mathematical model)*** that incorporates the influence of ***random events or actions*** to ***predict the likelihood of future outcomes***.
- It is a ***quantitative modeling*** technique that anticipates various potential results, potentially extending beyond historical occurrences.
- In these models, the outcomes of certain events are not determined with absolute certainty, but they are associated with ***probabilities.***

Ex of Probabilistic Approaches in NLP

- A language model in NLP is a probabilistic statistical model that determines the probability of a given sequence of words occurring in a sentence based on the previous words.
- It helps to predict which word is more likely to appear next in the sentence. Hence it is widely used in predictive text input systems, speech recognition, machine translation, spelling correction etc.
- The input to a language model is usually a training set of example sentences. The output is a probability distribution over sequences of words. We can use the last one word (unigram), last two words (bigram), last three words (trigram) or last n words (n-gram) to predict the next word as per our requirements.

Some key characteristics and uses of probabilistic models Approaches

- ***Uncertainty:*** Probabilistic models are used to account for uncertainty or randomness in data. They allow us to express how likely different outcomes are and quantify uncertainty.
- ***Probabilities:*** These models use ***probabilities to describe the likelihood*** of different events or outcomes. This is often done through probability distributions, which describe how likely different values are for a particular variable.
- ***Applications:*** Probabilistic models are used in various applications, such as statistical inference, machine learning, artificial intelligence, risk assessment, finance, ***natural language processing***, and more. They can be used to make predictions, estimate parameters, and simulate complex systems.

Some key characteristics and uses of probabilistic models Approaches

- Common examples of probabilistic models include the normal distribution for modeling continuous data, the Bernoulli distribution for modeling ***binary outcomes (yes/no),*** and more complex models like ***Hidden Markov Models*** for sequence data, ***Bayesian networks*** for representing probabilistic relationships among variables (***Bayesian probability uses prior information to update beliefs***), and various machine learning algorithms that incorporate uncertainty in decision-making.
- ***Learning and Inference:*** In machine learning, probabilistic models are often used for ***learning from data and making predictions***. They can model the uncertainty in data and provide measures of confidence in predictions.
- Probabilistic models are a fundamental tool for dealing with uncertainty and making informed decisions in situations where complete determinism is not applicable or practical.

Importance of Probabilistic ML Model

- One of the most significant advantages of the probabilistic modeling technique is that it provides a ***comprehensive understanding of the uncertainty*** associated with predictions.
- An example of a probabilistic classifier that ***assigns a probability of 0.9 to the 'Dog'*** class suggests the classifier is ***quite confident that the animal in the image is a dog.*** It is heavily dependent on the opposing concepts of uncertainty and confidence.
- Probabilistic Approaches or Algorithms play a vital role in Artificial Intelligence and Cognitive Science for generating, reasoning, decisions.

Ex of Probabilistic ML Model

Given all the previous patients I've seen (below are their symptoms and diagnosis)...

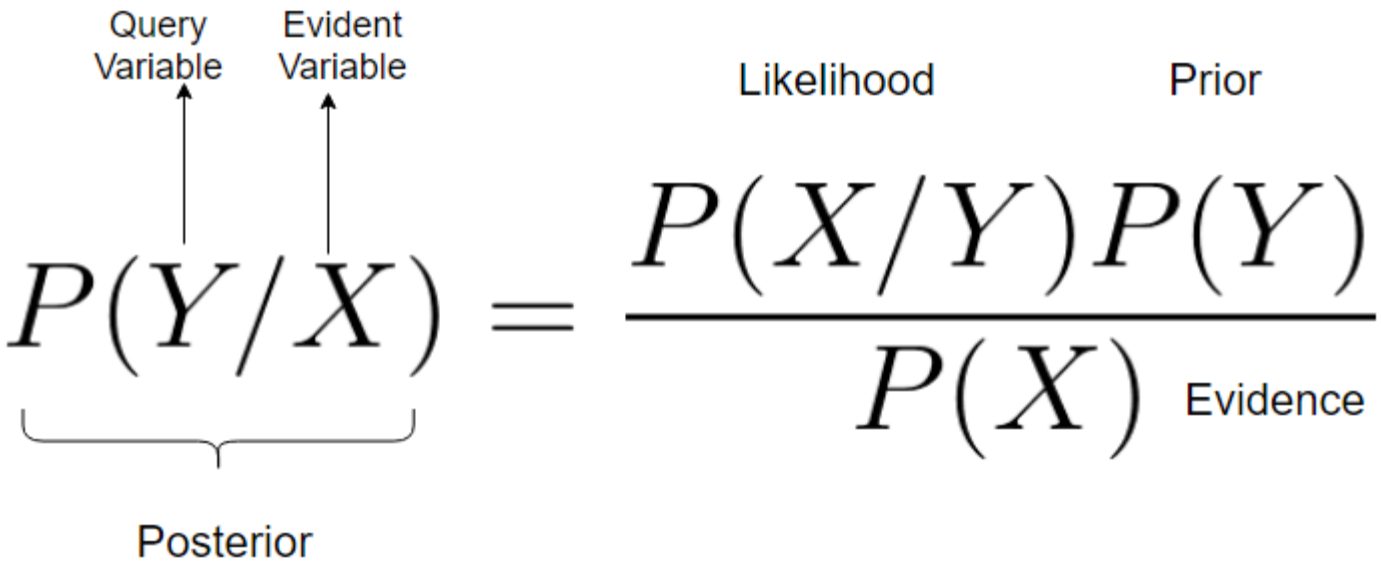
chills	runny nose	headache	fever	flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Do I believe that a patient with the following symptoms has the flu?

chills	runny nose	headache	fever	flu?
Y	N	Mild	Y	?

Ex of Probabilistic ML Model

- ***This model is based on bayes theorem.***



Probabilistic Inference: It is the ***computation of posterior probabilities*** for query variables ***(Y)*** given evidence variables ***(X)***.

Types of Probabilistic ML Model

- ***The main purpose of a probabilistic model is to classify / prediction the data.***
- In classification, we represent text as 'd' and assign it to a class 'c.' The goal is to learn the probability of text 'd' belonging to class 'c,' denoted as ***p(c/d).***

Typically, the classification process involves two stages: ***Representation and Classification.***

1.Representation Process: The first stage involves transforming text into a fixed representation, or in other words, learning ***'d.'***

2.Classification: In the second stage, we classify the document based on the learned representation, which is effectively estimating ***p(c/d).***

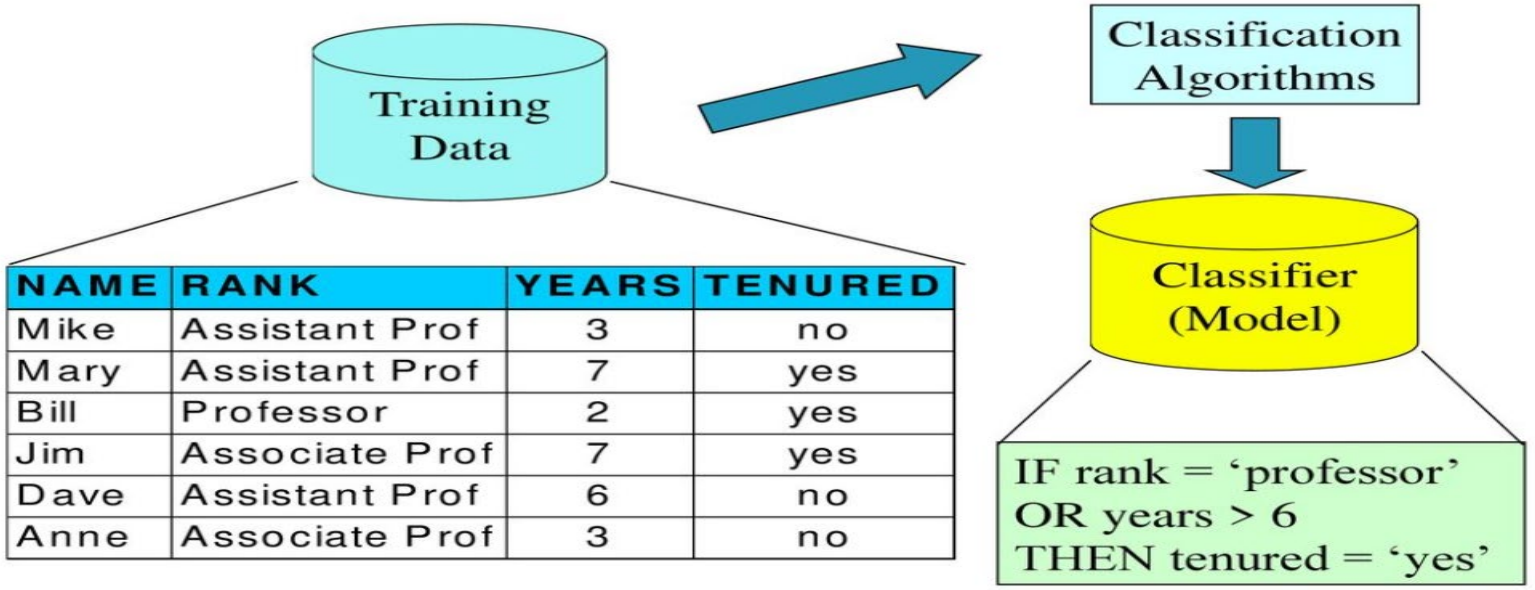
Types of Probabilistic ML Model

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

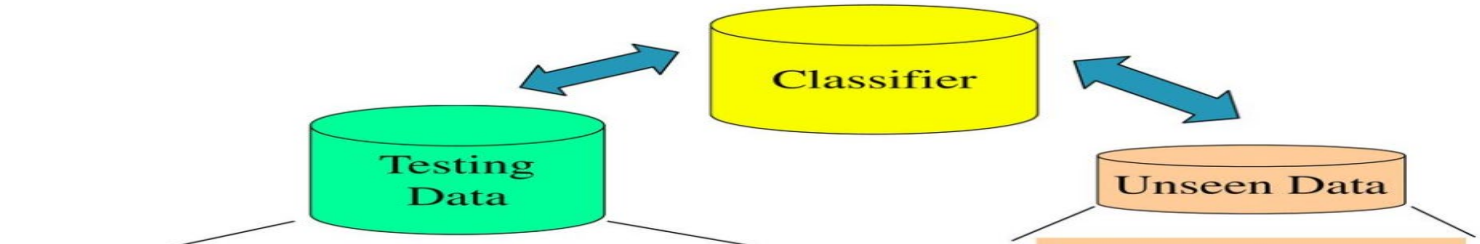
Types of Probabilistic ML Model

Learning Step: Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification step: Estimating accuracy of model



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Types of Probabilistic ML Model

- ***Generative Learning:*** In this approach, the goal is to learn the ***joint probability distribution p(c, d),*** capturing the relationship between class labels (c) and the data (d).
- ***Discriminative Learning:*** Conversely, discriminative learning focuses on estimating the conditional probability ***p(c/d),*** emphasizing the ***probability of class (c) given a specific data point (d).***

Statistical Approaches to NLP Tasks

Different approaches to natural language processing Task

There are three types of NLP approaches:

1.Rule-based Approach / Symbolic approach – The symbolic approach towards NLP is more about human-developed rules. A programmer writes a set of grammar rules to define how the system should behave.

2.Statistical approach– Based on statistical analysis - The statistical approach to natural language processing depends on finding patterns in large volumes of text. By recognising these trends, the system can develop its own understanding of human language.

3.Connectionist Approach or "Hybrid" Approach– The third approach is a combination of statistical and symbolic approaches.

What is statistical NLP?

- Statistical NLP aims to perform ***statistical inference*** for the field of NLP.
- Statistical inference consists of taking some data generated in accordance with some unknown probability distribution and making inferences.

Statistical NLP approach

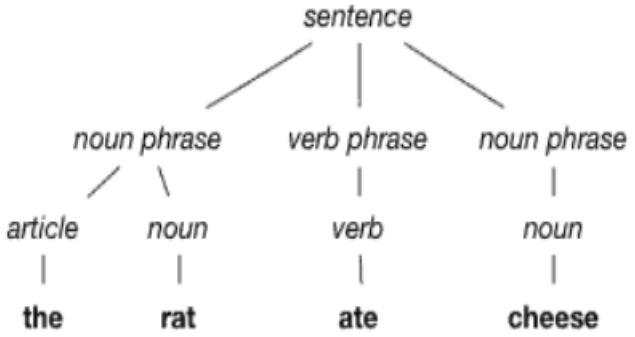
- A statistical NLP approach is a type of ***machine learning approach.***
- Statistical NLP involves using ***statistical models and algorithms*** to process and understand human language, making it a subset of machine learning techniques applied to language-related tasks.
- Traditional machine learning approaches include probabilistic modeling, likelihood maximization, and linear classifiers. Notably, ***these are not neural network models.***

Statistical NLP approach

- Machine learning (also called statistical) methods for NLP involve using ***AI algorithms to solve problems without being explicitly programmed.***
- Instead of working with human-written patterns, ML models find those patterns independently, just by analyzing texts.
- There are ***two main steps*** for preparing data for the machine to understand.

1. ***Text annotation and formatting:*** Any NLP task starts with data preparation.
    - In NLP tasks, this process is called ***building a corpus***.
    - Corpora (plural for corpus) are collections of texts used for ***ML training.*** You can't simply feed the system your whole dataset of emails and expect it to understand what you want from it. That's why texts must be ***annotated*** — enhanced by providing a larger meaning.
    - ***Tokenization, part-of-speech tagging, and parsing (describing the syntactic structure of a sentence) are some forms of annotations***.
    - Annotated data is then organized into standard formats, thus becoming structured.

Statistical NLP approach



One of the stages in text annotation — parsing

Feature engineering: Apart from a corpus, machines use ***features to perceive text***. Features are different characteristics like “language,” “word count,” “punctuation count,” or “word frequency” that can tell the system what matters in the text. Data scientists decide what features of the text will help the model solve the problem, usually applying their domain knowledge and creative skills. Say, the frequency feature for the words now, immediately, free, and call will indicate that the message is spam. And the punctuation count feature will direct to the exuberant use of exclamation marks.

Statistical NLP approach

- ***Model training and deployment.***
  - The prepared data is then fed to the algorithm for training.
  - Training done with labeled data is called supervised learning and it has a great fit for most common classification problems.
  - Some of the popular ***statistical / ML algorithms for NLP tasks*** are Decision Trees, Naive Bayes, Support-Vector Machine, Conditional Random Field, etc.
  - After training the model, data scientists test and validate it to make sure it gives the most accurate predictions and is ready for running in real life.
  - Though often, AI developers use pretrained language models created for specific problems.

Limitation of Statistical NLP approaches

- Just like the rule-based approach requires ***linguistic knowledge*** to create rules, machine learning methods are ***only as good as the quality of data and the accuracy of features created by data scientists.*** This means that while ML is better at ***classification than rules,*** it falls short in two directions:

  •The ***complexity of feature engineering***, which requires researchers to do massive amounts of preparation, thus not achieving full automation with ML; and

  •The ***curse of dimensionality***, when the volumes of data needed grow exponentially with the dimension of the model, thus creating data sparsity.

Below are the NLP tasks that use statistical approaches, highlighted in bold text

- *Language Modeling:* Statistical language models, such as ***n-grams and hidden Markov models***, estimate the ***likelihood of word sequences*** in a given language. These models are fundamental for tasks like speech recognition, text generation, and machine translation.
- *Part-of-Speech Tagging:* Statistical models assign grammatical categories (e.g., nouns, verbs, adjectives) to words in a text. ***Hidden Markov Models (HMMs***) and ***Conditional Random Fields*** (CRFs) are commonly used for this task.
- *Named Entity Recognition (NER):* Statistical methods identify and classify entities, such as names of people, places, and organizations in text. ***Conditional Random Fields and deep learning techniques*** are often applied for NER.

Below are the NLP tasks that use statistical approaches, highlighted in bold text

4. *Sentiment Analysis:* ***Statistical approaches*** assess the sentiment or emotion expressed in a piece of text. These methods are used in applications like social media sentiment analysis and customer feedback analysis.

5. *Machine Translation:* Statistical models, including phrase-based and statistical machine translation (SMT) models, are utilized to translate text from one language to another by analyzing bilingual corpora.

6. *Text Classification:* Statistical techniques like ***Naïve Bayes, Logistic Regression, and Support Vector Machines (SVMs)*** are used for tasks such as spam detection, document categorization, and sentiment analysis.

Below are the NLP tasks that use statistical approaches, highlighted in bold text

7. *Information Retrieval:* Statistical ranking models, like ***TF-IDF (Term Frequency-Inverse Document Frequency) and BM25***, assist in retrieving relevant documents from large collections based on user queries.
8. *Language Generation:* Statistical methods generate human-like text for chatbots and natural language generation systems. ***Markov models and neural language models are commonly employed.***
9. *Syntax and Parsing:* Statistical parsers analyze the grammatical structure of sentences**.** ***Probabilistic context-free grammars and statistical parsers are*** used for syntactic analysis.
10. *Text Summarization:* Text summarization models, including extractive and abstractive methods, uses ***statistical techniques*** to condense longer texts into concise summaries.

Below are the NLP tasks that use statistical approaches, highlighted in bold text

11. *Speech Recognition:* Statistical models like **HMMs** and deep learning techniques, including recurrent neural networks (RNNs), convert spoken language into text.

12. *Topic Modeling:* Statistical methods like **Latent Dirichlet Allocation (LDA)** identify topics within a collection of documents based on word co-occurrence patterns.

13. *Coreference Resolution:* **Statistical models** determine which words or expressions in a text refer to the same entities, essential for context understanding.

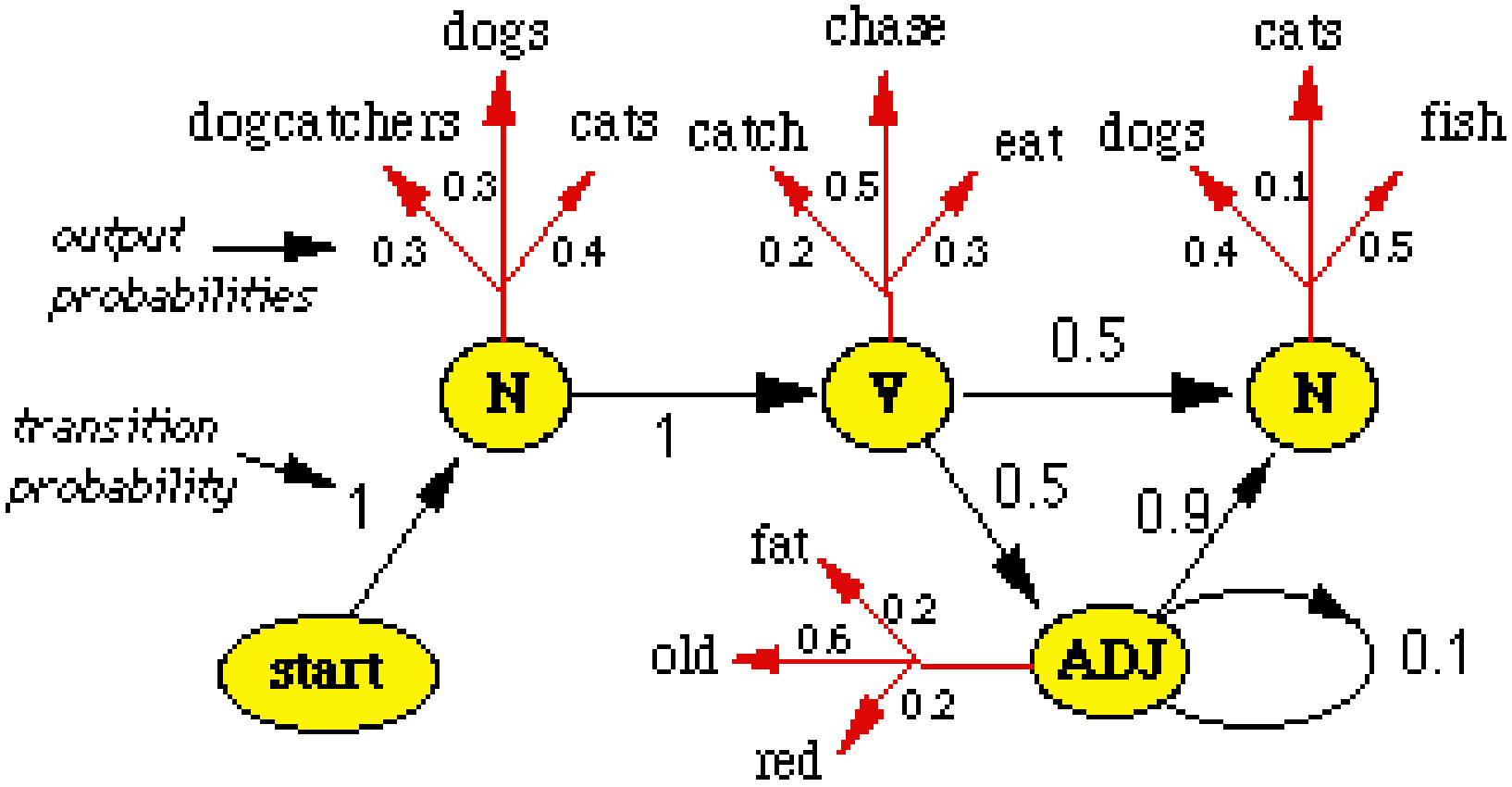
Statistical NLP approaches

- There are many ***different machine learning algorithms*** that can be used in the statistical approach to NLP.
- ***Some of the most common algorithms include:***
- ***Support vector machines (SVMs):*** SVMs are a type of supervised learning algorithm that can be used for a variety of NLP tasks, such as classification, regression, and ranking. SVMs work by finding a hyperplane that separates the data into two or more classes with the largest possible margin.
- ***Naive Bayes classifiers:*** Naive Bayes classifiers are a type of probabilistic classifier that can be used for a variety of NLP tasks, such as spam filtering, sentiment analysis, and text classification. Naive Bayes classifiers work by assuming that the features of the data are independent of each other, given the class label.

Statistical NLP approaches

- ***Hidden Markov models (HMMs):*** HMMs are a type of statistical model that can be used to model sequential data, such as speech and text. HMMs work by representing the data as a sequence of hidden states, where each state is associated with a probability distribution over the observed data.
- A ***HMM***, is a set of states (lexical categories in our case) with directed edges labeled with transition probabilities that indicate the probability of moving to the state at the end of the directed edge, given that one is now in the state at the start of the edge. The states are also labeled with a function which indicates the probabilities of outputting different symbols if in that state (while in a state, one outputs a single symbol before moving to the next state). In our case, the symbol output from a state/lexical category is a word belonging to that lexical category.

Statistical NLP approaches



Statistical NLP approaches

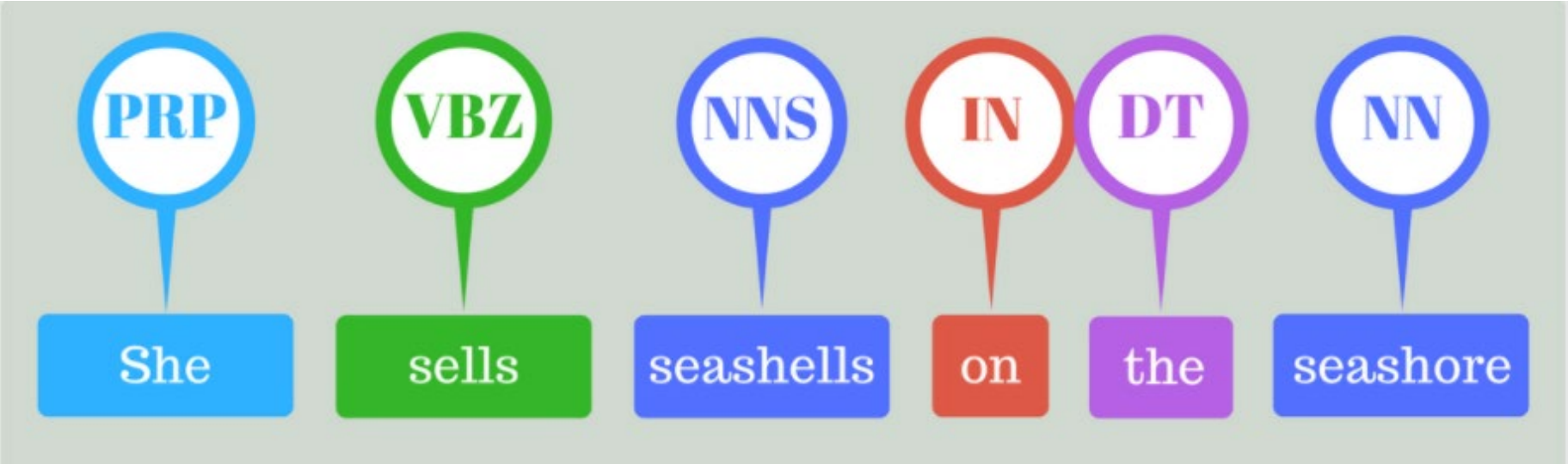
- ***Decision trees:*** Decision trees are a type of supervised learning algorithm that can be used for a variety of NLP tasks, such as classification and regression. Decision trees work by recursively partitioning the data into smaller and smaller subsets, until each subset contains only data points from the same class.Random forests: Random forests are an ensemble learning algorithm that combines the predictions of many decision trees to produce more accurate predictions.
- ***Random forests*** are often used for NLP tasks such as classification, regression, and ranking.Hidden Markov Models (HMMs) and Naive Bayes classifiers are both probabilistic approaches to NLP. They are both based on the assumption that the data can be modeled as a probability distribution.However, they are also commonly used in conjunction with statistical methods. For example, HMMs are often used to model the sequence of words in a sentence, while Naive Bayes classifiers are often used to classify text into different categories.

Sequence Labeling

Sequence Labeling

- Sequence labeling is a NLP and machine learning task that involves ***assigning a label or category*** to each element in a sequence of data.
- This ***sequence of data*** could be a sentence, a paragraph, a document, or any other ordered set of elements.
- Each ***element*** in the sequence is typically a ***word, a subword, a character, or a token,*** depending on the specific application.
- The primary goal of sequence labeling is to ***extract information*** or ***classify (determine) the elements/token/word within a sequence***.
- This classification approach can be ***independent*** (each word is treated independently) or ***dependent*** (each word is dependent on other words).

Sequence Labeling



POS Tagging of the sentence “She sells seashells on the seashore”

Sequence Labeling

- **Words and their roles**
- Determining the role of a word in a sentence, such as whether it is a noun or an adjective, is a fundamental task in natural language processing known as "***part-of-speech tagging***" (POS tagging).
- POS tagging is essential for understanding the grammatical and semantic structure of a sentence.

Sequence Labeling

- Sequence tagging (or sequence labeling) refers to a set of Natural Language Processing (NLP) tasks that ***assign labels or tags*** to tokens or other units of text.
- ***Common applications*** of sequence labeling include:
  - Named Entity Recognition (NER)
  - Part-of-Speech Tagging (POS)

Sequence Labeling

1. ***Named Entity Recognition (NER):***
   - ***Identifying and classifying entities***, such as names of persons, organizations, locations, dates, and more in text.
   - For example, in the sentence "***Apple Inc. is headquartered in Cupertino, California***," NER would label "Apple Inc." as an ***organization*** and "Cupertino, California" as a ***location***.
2. ***Part-of-Speech Tagging (POS):***
   - ***Assigning grammatical parts of speech to each word in a sentence***, such as nouns, verbs, adjectives, and adverbs.
   - For example, in the sentence "***She quickly runs***," POS tagging would label "She" as a ***pronoun***, "quickly" as an adverb, and "runs" as a ***verb.***

Sequence Labeling

- Sequence tagging (or sequence labeling) refers to a set of Natural Language Processing (NLP) tasks that ***assign labels or tags*** to tokens or other units of text.
- When the tags are named entities, we are then dealing with named entity recognition (NER). When the tags are parts of speech, this task is called part-of-speech (PoS) tagging.
- Sequence taggers are trained using ***supervised training techniques***, making them easier to evaluate and compare.

Main task of Sequence Labeling

- Sequence labeling is used for a wide range of ***NLP tasks***, such as:
  - Part-of-Speech Tagging
  - Named Entity Recognition
  - Chunking
  - Semantic Role Labeling.

Main task of Sequence Labeling - Part-of-Speech Tagging

- Part-of-Speech Tagging (POS Tagging) is a sequence labeling task.
- It is the ***process of marking up a word in a text (corpus)*** as corresponding to a particular ***part of speech (syntactic tag***), based on both its definition and its context.
- POS Tagging is a ***helper task*** for many tasks about NLP:
  - Word Sense Disambiguation, Dependency Parsing.
- POS tagging is the process of labeling the parts of speech (such as nouns, verbs, and adjectives) in a sentence.
- In information retrieval, it can help the search engine to ***provide more relevant*** and accurate results by distinguishing different parts of speech.

Main task of Sequence Labeling

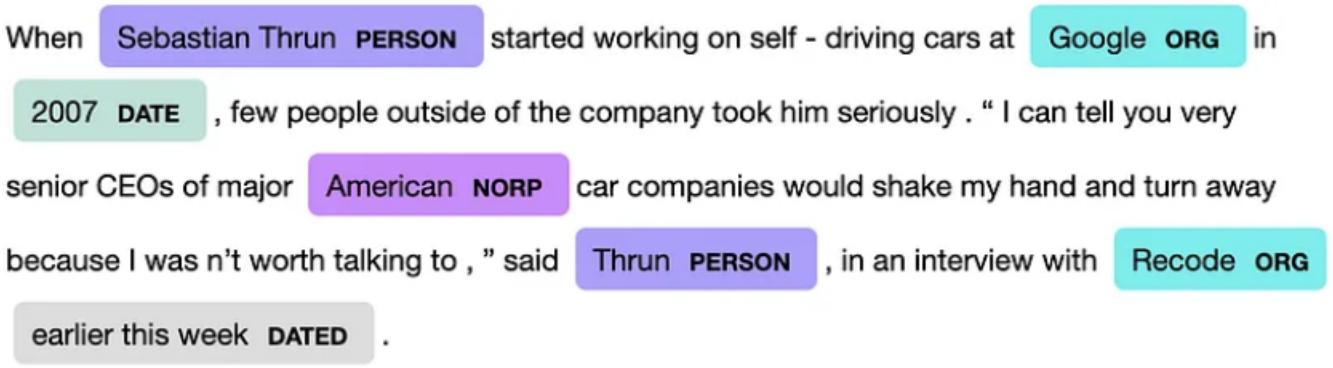
- Here is an example :

```
import nltk

def tag_sentence(sentence):
    tokens = nltk.word_tokenize(sentence)
    tagged_tokens = []
    for token in tokens:
        if token.endswith('ed'):
            tagged_tokens.append((token, 'VERB'))
        elif token.istitle():
            tagged_tokens.append((token, 'PROPER NOUN'))
        else:
            tagged_tokens.append((token, 'NOUN'))
    return tagged_tokens

sentence = "The cat chased the mouse."
tagged_tokens = tag_sentence(sentence)
print(tagged_tokens)

# Output: [('The', 'PROPER NOUN'), ('cat', 'NOUN'), ('chased', 'VERB'),
# ('the', 'PROPER NOUN'), ('mouse', 'NOUN'), ('.', 'NOUN')]
```

Main task of Sequence Labeling

Named Entity Recognition

- Named entity recognition (NER) is the task of ***identifying and classifying named entities*** (such as people, organizations, and locations) in text.
- As POS tagging, it can be used to ***extract information from a large corpus of texts*** and help identify more quickly what is wanted.



NER on a paragraph.

Main task of Sequence Labeling

- NER can be used for ***customer services, medical purposes, document categorization.***
- For example, ***extracting aspects from customer reviews*** with NER helps identification of semantics. Or extracting medical diseases and drugs from a text helps identification of treatment.

Main task of Sequence Labeling - Chunking

Chunking

- Chunking is a task of sequence labeling that involves ***dividing a sequence of words*** into ***chunks or non-overlapping sub-sequences***.
- These chunks are typically ***tagged with a label*** that indicates their type or role in the sequence.
- Chunking typically focuses on ***identifying noun phrases, verb phrases, and other meaningful constituents*** in a sentence, rather than ***individual words.***
- Suppose we have the following sentence: "The quick brown fox jumps over the lazy dog."

Using chunking, we might divide the sentence into the following ***chunks:*** "The quick brown fox" (noun phrase)

"jumps" (verb phrase) "over the lazy dog" (noun phrase)

Main task of Sequence Labeling

Chunking

- So, in a chunking task, the goal is to group words together into meaningful phrases or chunks, and each chunk is typically labeled with its grammatical role in the sentence, like NP or VP.
- This can be a helpful intermediate step in various NLP tasks, such as parsing and information extraction.

Sequence Labeling Models

- Sequence labeling can be done with various methods.
- While traditional models are based on corpus statistics (Hidden Markov Models, Maximum Entropy Markov Models, Conditional Random Field, etc.), recent models are based on neural networks (Recurrent Neural Networks, Long Short-Term Memory, BERT, etc.).

Problems - Similarity Measures

Problems - Similarity Measures

- Text similarity is a key concept in NLP, as it allows you to ***compare and analyze different texts based on their content, structure, and meaning.***
- ***Text similarity metrics*** can be broadly classified into ***two categories***: ***lexical and semantic.***
- ***Lexical similarity*** measures the degree of overlap or similarity between two texts based on their ***words, characters, or n-grams.***
- ***Semantic similarity*** measures the degree of similarity between two texts based on their ***meaning, context, or concepts.***
- Some ***examples*** of lexical similarity metrics are Euclidean distance, Jaccard index, and cosine similarity.
- Some ***examples*** of semantic similarity metrics are WordNet, word embeddings, and transformer models.

Problems - Similarity Measures

- Text similarity metrics can be used for various NLP tasks, such as text summarization, plagiarism detection, document clustering, sentiment analysis, and question answering.
- For each task, you need to select a text similarity metric that matches the level and type of similarity you want to capture.
- For example, for ***text summarization,*** use a ***semantic similarity metric*** like cosine Similarity, Word Embedding Similarity (e.g., Word2Vec or FastText), or even BERT-based embeddings to measure the semantic overlap between text units.
- For ***plagiarism detection,*** often involves identifying exact or near matches between texts. ***Lexical similarity metrics*** such as Jaccard Similarity, Longest Common Subsequence (LCS), and even simple methods like fingerprinting or n-gram overlap are used to detect similarities at the word or character level for this task.

Problems - Similarity Measures

- The similarity measure is usually expressed as a numerical value; and these values indicate the ***degree of similarity*** between data ***samples or objects.***

1. ***High Similarity:*** A ***similarity measure of 1*** typically means that the data objects are ***very similar or nearly identical***. In other words, a similarity score of 1 indicates a strong resemblance or a high level of agreement between the objects being compared.
2. ***Low Similarity:*** A ***similarity measure of 0*** typically means that the ***data objects are dissimilar*** or have no shared characteristics. A score of 0 indicates that there is no resemblance or agreement between the objects.

Types of Similarity measures / metrics

- Longest Common Substring (LCS)
- Euclidean Distance
- Manhattan Distance (City Block Distance)
- Minkowski distance
- Cosine similarity
- Jaccard similarity

Types of Similarity measures / metrics

- ***Longest Common Substring (LCS)***
  - The LCS is a common example of ***character-based similarity measure.***
  - Given two strings, s1 of length n1 and s2 of length n2, it simply considers the length of the longest string (or strings) which is substring of both s1 and s2.
  - ***Applications: data deduplication and plagiarism detection.***

Example: the LCS of strings 'Lydia' and 'Lydoko' simply returns 3.

```
LCS('Lydia', 'Lydoko') = 3
```

		L	Y	D	I	A
	0	0	0	0	0	0
L	0	1	0	0	0	0
Y	0	0	2	0	0	0
D	0	0	0	3	0	0
O	0	0	0	0	0	0
K	0	0	0	0	0	0
O	0	0	0	0	0	0

Types of Similarity measures / metrics

- ***Euclidean Distance***
  - Euclidean distance is a ***token-based similarity distance.***
  - Given two points in Rn, the Euclidean distance metric is simply the length of a line segment between these two points.
  - It is also often referred as the l2-norm, the l2-distance or the Pythagorean metric and can be expressed as:

$$\text{Euclidean distance:} \quad D_{euclidean}(x, y) = \|x - y\|_2 = \sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2}$$

  - ***Properties:***
    The Euclidean distance is symmetric, positive and obeys the triangle inequality.

- $\mathrm{D}_{euclidean}(x, y) = \mathrm{D}_{euclidean}(y, x)$
- $\mathrm{D}_{euclidean}(x, y) \geqslant 0$ and $\mathrm{D}_{euclidean}(x, y) = 0 \Leftrightarrow x = y$
- $\mathrm{D}_{euclidean}(x, y) \leqslant \mathrm{D}_{euclidean}(x, z) + \mathrm{D}_{euclidean}(y, z)$

Types of Similarity measures / metrics

- ***Manhattan Distance***
  - Manhattan distance measures the distance between two points by summing the absolute differences of their coordinates.

- **Formula (for 2D space):**

  $$\text{Manhattan Distance} = |x_2 - x_1| + |y_2 - y_1|$$

- **Formula (for n-dimensional space):**

  $$\text{Manhattan Distance} = \Sigma|x_i - y_i|$$

Types of Similarity measures / metrics

- ***Minkowski Distance***
  - Minkowski distance is a general distance metric that encompasses both ***Euclidean and Manhattan distances*** as special cases.
  - It allows you to adjust the distance measurement based on a parameter "p," which can be used to fine-tune the metric for different applications.

$$\text{Minkowski distance:} \quad D_{minkowski}(x, y) = \|x - y\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

  - It is equal to the Manhattan distance if p=1, equal to the Euclidean distance if p=2.

Types of Similarity measures / metrics

- ***Cosine Similarity Calculation:*** To calculate the cosine similarity between two vectors A and B, use the following formula:

  Cosine Similarity (A, B) = (A • B) / (||A|| * ||B||)

  - (A • B) is the dot product of vectors A and B, which is the sum of the pairwise products of their components.
  - ||A|| and ||B|| represent the magnitudes (or lengths) of vectors A and B, respectively. These magnitudes are the square roots of the sum of the squares of the individual vector components.
  - ***Interpretation:*** The resulting cosine similarity value ranges from -1 (perfectly dissimilar) to 1 (perfectly similar). ***A value of 0 indicates no similarity.***

Types of Similarity measures / metrics

- ***Jaccard Similarity***
  - The Jaccard distance measures how dissimilar two multisets are: the lower the distance, the more similar the two multisets.
  - It is computed using the Jaccard index (or Jaccard similarity coefficient) which is the ratio of the cardinal of the intersection of the multisets to the cardinal of their union. The distance is then obtained by subtracting the index from 1.

$$\text{Jaccard index:} \quad J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad \text{and} \quad 0 \leqslant J(A,B) \leqslant 1$$

$$\text{Jaccard distance:} \quad d_J(A,B) = 1 - J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \quad \text{and} \quad 0 \leqslant d_J(A,B) \leqslant 1$$

  - These expressions are undefined if both A and B are empty sets, in which case we define the index and distance to be 1 and 0 respectively.

Types of Similarity measures / metrics

- ***Ex of Jaccard Similarity***
  - The Jaccard distance works quite well in practice, especially for sparse data. For example, a streaming service like Netflix could represent customers as multisets of movies watched, and then use Jaccard distance to measure the similarity between two customers, i.e. how close their tastes are. Then, based on the preferences of two users and their similarity we could potentially make recommendations to one or the other.

Word Embeddings

- It is an approach for representing words and documents.
- Word Embedding or Word Vector is a ***numeric vector input*** that represents a ***word in a lower-dimensional space.***
- It allows words with similar meanings to have a similar representation.
- They can also approximate meaning. A word vector with 50 values can represent 50 unique features.

Features: Anything that relates words to one another. E.g.: Age, Sports, Fitness, Employed, etc. Each word vector has values corresponding to these features.

Word Embeddings

Goal of Word Embeddings

- To reduce dimensionality
- To use a word to predict the words around it
- Interword semantics must be captured

Word Embeddings

How are Word Embeddings used?

- They are used as input to machine learning models. Take the words —-> Give their numeric representation —-> Use in training or inference
- To represent or visualize any underlying patterns of usage in the corpus that was used to train them.

Word Embeddings

Word2Vec:

In Word2Vec every word is assigned a vector. We start with either a ***random vector or one-hot vector.***

One-Hot vector: A representation where only one bit in a vector is 1.If there are 500 words in the corpus then the vector length will be 500. After assigning vectors to each word we take a window size and iterate through the entire corpus. While we do this there are two neural embedding methods which are used:

Word Embeddings

Bag of words (BOW)

A bag of words is one of the popular word embedding techniques of text where each value in the vector would represent the count of words in a document/sentence. In other words, it extracts features from the text. We also refer to it as vectorization.

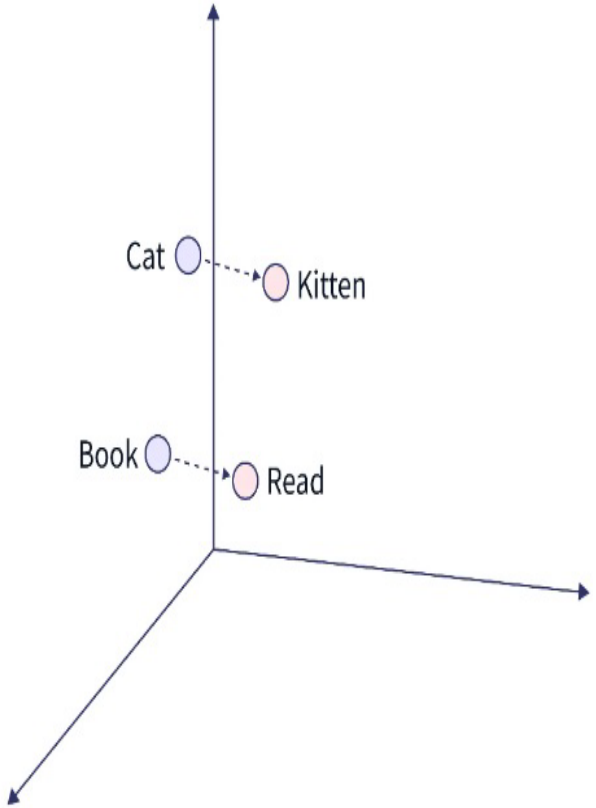
To get you started, here's how you can proceed to create BOW.

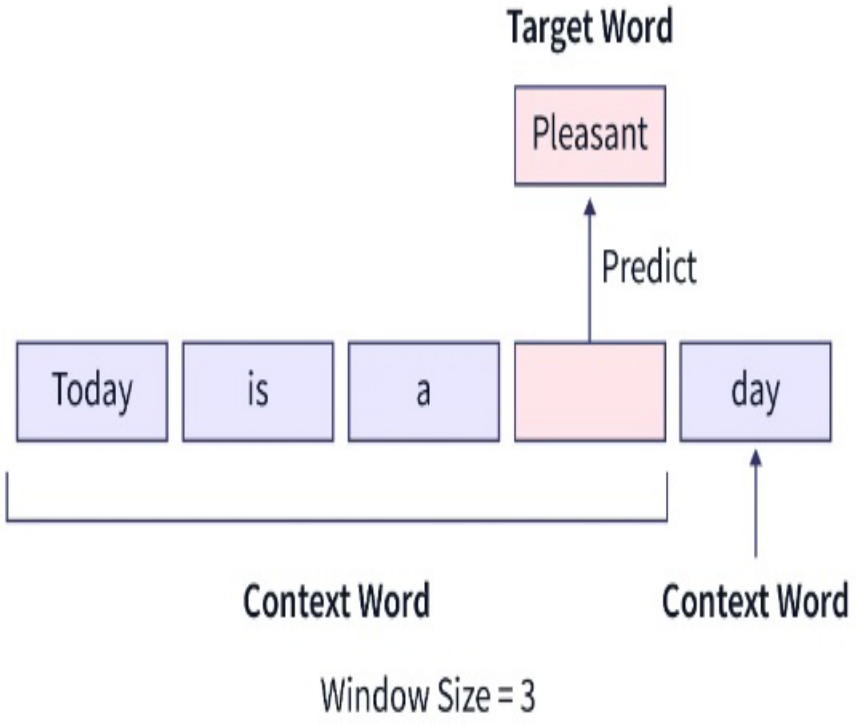
In the first step, you have to tokenize the text into sentences.
Next, the sentences tokenized in the first step have further tokenized words.
Eliminate any stop words or punctuation.
Then, convert all the words to lowercase.
Finally, move to create a frequency distribution chart of the words.

Continuous Bag of words (BOW)

- The continuous bag-of-words (CBOW) model is a neural network for natural language processing tasks such as language translation and text classification. It is based on predicting a target word given the context of the surrounding words. The CBOW model takes a window of surrounding words as input and tries to predict the target word in the center of the window.
- The model is trained on a large text dataset and learns to predict the target word based on the patterns it observes in the input data.
- The CBOW model is often combined with other natural language processing techniques and models, such as the skip-gram model, to improve the performance of natural language processing tasks.

Continuous Bag of words (BOW)

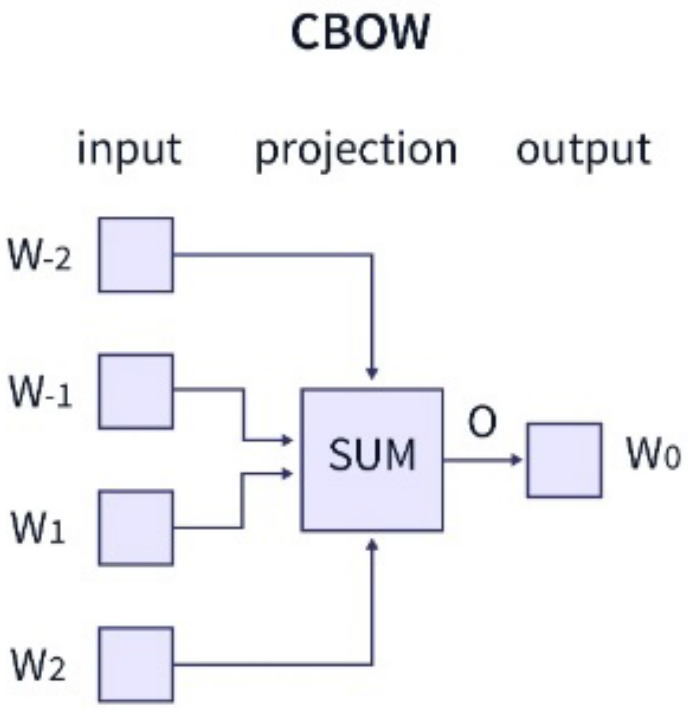




CBOW

CBOW - Architecture

The CBOW model attempts to comprehend the context of the words around the target word to predict it. Consider the previous phrase, "It is a pleasant day." The model transforms this sentence into word pairs (context word and target word). The user must configure the window size. The word pairings would appear like this if the context word's window were 2: ([it, a], is), ([is, nice], a) ([a, day], pleasant). The model tries to predict the target term using these word pairings while considering the context words.



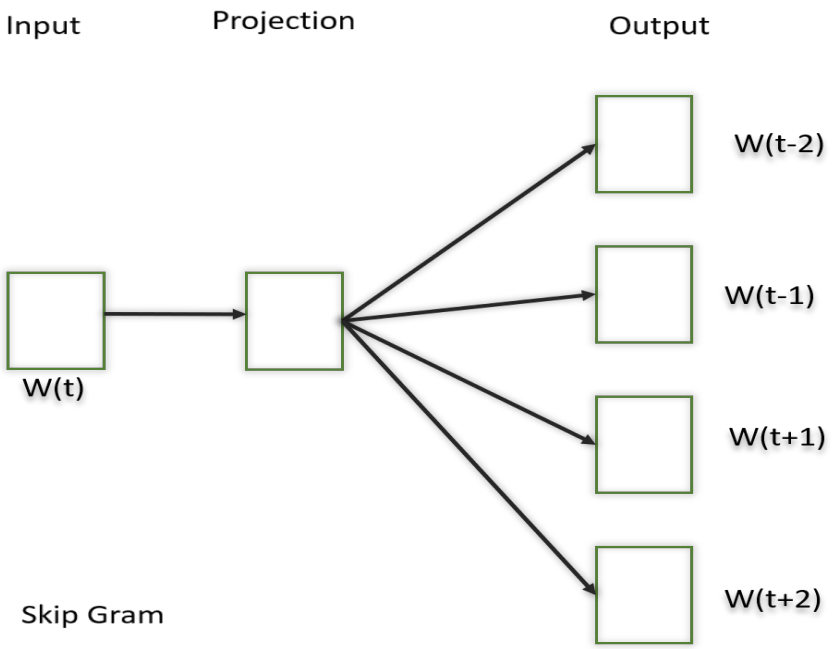
CBOW - Architecture

https://www.scaler.com/topics/nlp/cbow/

Skip-gram

Skip Gram

In this model, we try to make the central word closer to the neighboring words. It is the complete opposite of the CBOW model. It is shown that this method produces more meaningful embeddings.



After applying the above neural embedding methods we get trained vectors of each word after many iterations through the corpus. These trained vectors preserve syntactical or semantic information and are converted to lower dimensions. The vectors with similar meaning or semantic information are placed close to each other in space.

Skip Gram

https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c

Sentence Embeddings

Sentence Embeddings

- In NLP, sentence embedding refers to **a numeric representation of a sentence in the form of a vector of real numbers, which encodes meaningful semantic information**.
- It enables comparisons of sentence similarity by measuring the distance or similarity between these vectors.
- Techniques like Universal Sentence Encoder (USE) use deep learning models trained on large corpora to generate these embeddings, which find applications in tasks like text classification, clustering, and similarity matching.

Sentence Embedding Models

- Sentence embedding models are designed to encapsulate the semantic essence of a sentence within a fixed-length vector.
- Unlike traditional Bag-of-Words (BoW) representations or one-hot encoding, sentence embeddings capture context, meaning, and relationships between words.
- This transformation is crucial for enabling machines to grasp the subtleties of human language.

Methods of Sentence Embedding

Several methods are employed to generate sentence embeddings:

1. **Averaging Word Embeddings:** This approach involves taking the average of word embeddings within a sentence. While simple, it may not capture complex contextual nuances.
2. **Pre-trained Models like BERT:** Models like BERT (Bidirectional Encoder Representations from Transformers) have revolutionized sentence embeddings. BERT-based models consider the context of each word in a sentence, resulting in rich and contextually aware embeddings.
3. **Neural Network-Based Approaches:** Skip-Thought vectors and InferSent are examples of neural network-based sentence embedding models. They are trained to predict the surrounding sentences, which encourages them to understand sentence semantics.

Noteworthy Sentence Embedding Models

1. **BERT (Bidirectional Encoder Representations from Transformers):** BERT has set a benchmark in sentence embeddings, offering pre-trained models for various NLP tasks. Its bidirectional attention and contextual understanding make it a prominent choice.
2. **RoBERTa:** An evolution of BERT, RoBERTa fine-tunes its training methodology, achieving state-of-the-art performance in multiple NLP tasks.
3. **USE (Universal Sentence Encoder):** Developed by Google, USE generates embeddings for text that can be used for various applications, including cross-lingual tasks.

RNN

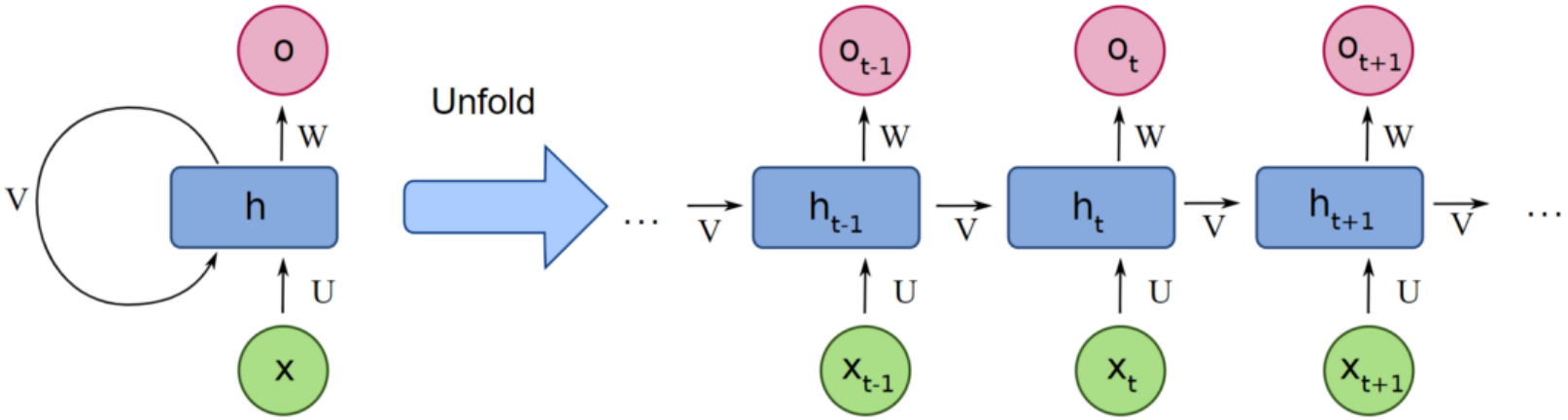
RNN

- A Deep Learning approach for modelling sequential data is Recurrent Neural Networks (RNN).
- RNNs were the standard suggestion for working with sequential data before the advent of attention models.
- Specific parameters for each element of the sequence may be required by a deep feedforward model.

What is RNN?

- RNNs Recurrent Neural Networks are a type of neural network that are designed to process sequential data.
- They can analyze data with a temporal dimension, such as time series, speech, and text.
- RNNs can do this by using a hidden state passed from one timestep to the next.
- The hidden state is updated at each timestep based on the input and the previous hidden state.
- RNNs are able to capture short-term dependencies in sequential data, but they struggle with capturing long-term dependencies.

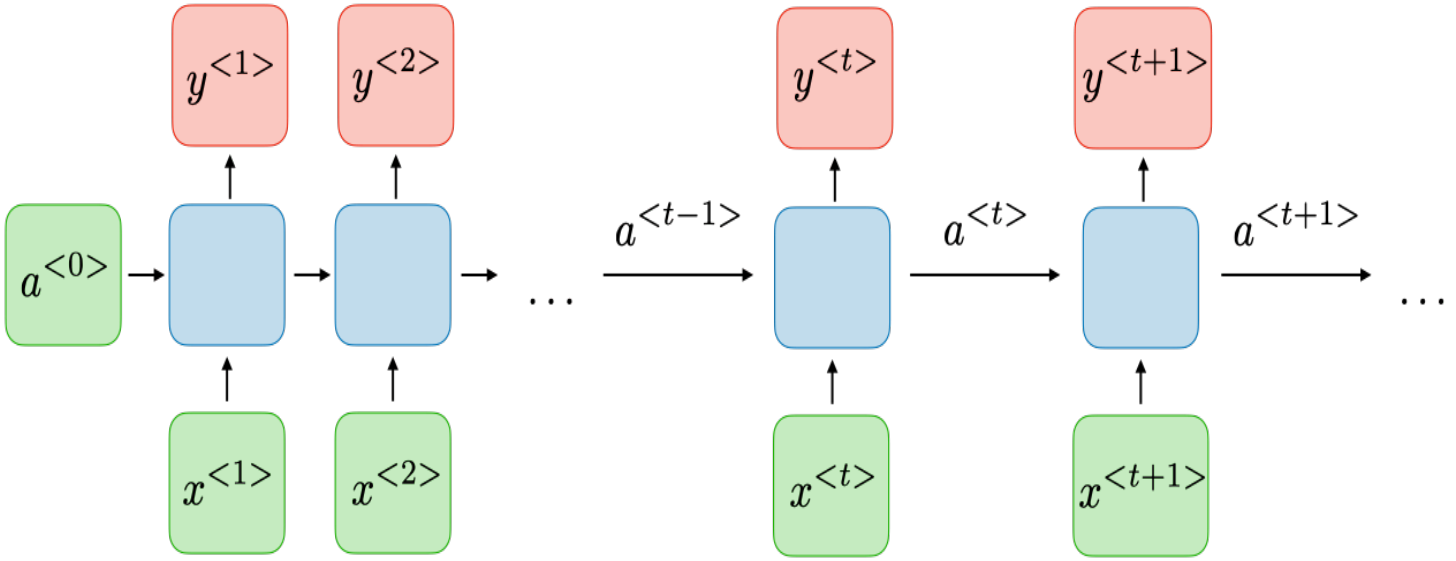
RNN MODEL



- Recurrent Neural Networks use the same weights for each element of the sequence, decreasing the number of parameters and allowing the model to generalize to sequences of varying lengths.
- RNNs are a type of neural network that can be used to model sequence data. RNNs, which are formed from feedforward networks, are similar to human brains in their behaviour.

- All of the inputs and outputs in standard neural networks are independent of one another, however in some circumstances, such as when predicting the next word of a phrase, the prior words are necessary, and so the previous words must be remembered.
- As a result, RNN was created, which used a Hidden Layer to overcome the problem. The most important component of RNN is the Hidden state, which remembers specific information about a sequence.

The Architecture of a Traditional RNN

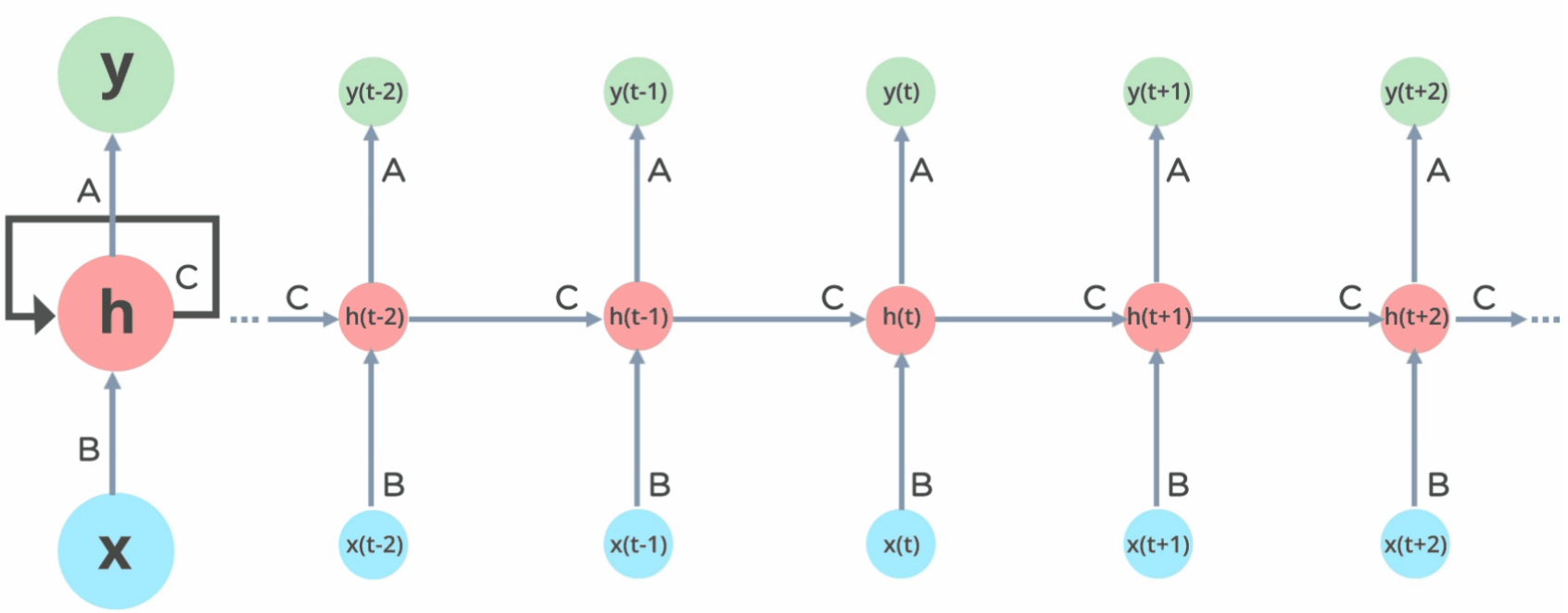


For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.

How does Recurrent Neural Networks work?



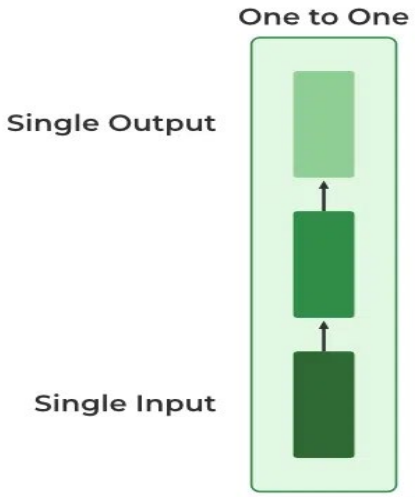
Types of RNN

There are four types of RNNs based on the number of inputs and outputs in the network.

1. One to One
2. One to Many
3. Many to One
4. Many to Many

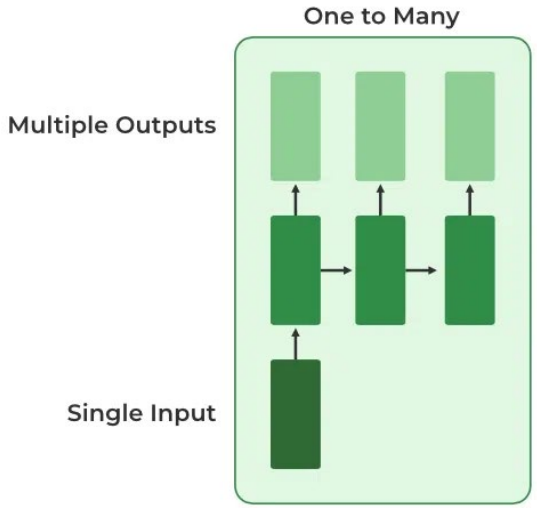
ONE to ONE

- This type of RNN behaves the same as any simple Neural network it is also known as Vanilla Neural Network.
- In this Neural network, there is only one input and one output.



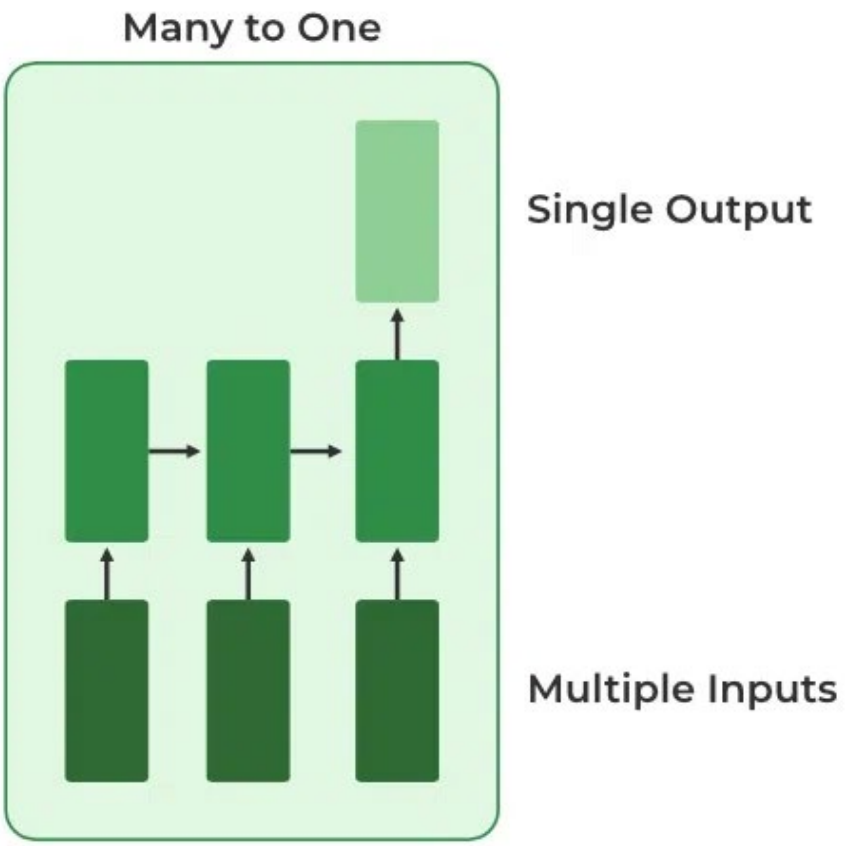
ONE to MANY

- In this type of RNN, there is one input and many outputs associated with it.
- One of the most used examples of this network is Image captioning where given an image we predict a sentence having Multiple words.



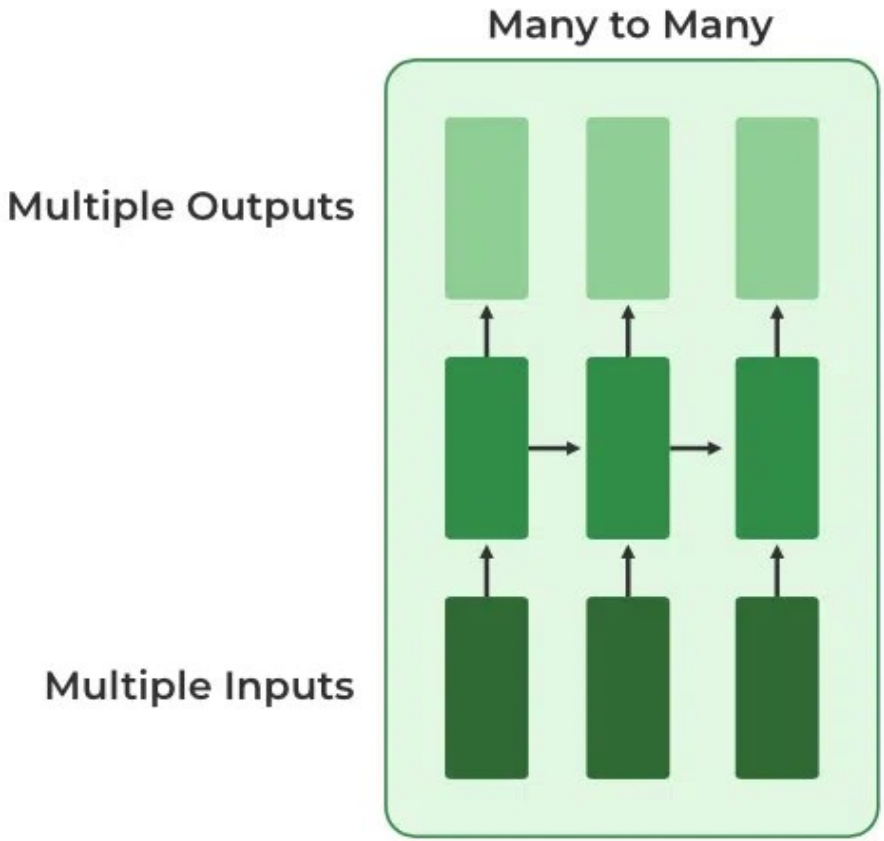
Many to One

- In this type of network, Many inputs are fed to the network at several states of the network generating only one output.
- This type of network is used in the problems like sentimental analysis.
- Where we give multiple words as input and predict only the sentiment of the sentence as output.


Many to One
Single Output
Multiple Inputs

Many to Many

- In this type of neural network, there are multiple inputs and multiple outputs corresponding to a problem.
- One Example of this Problem will be language translation.
- In language translation, we provide multiple words from one language as input and predict multiple words from the second language as output.


Many to Many
Multiple Outputs
Multiple Inputs

LSTM

Learning Objectives

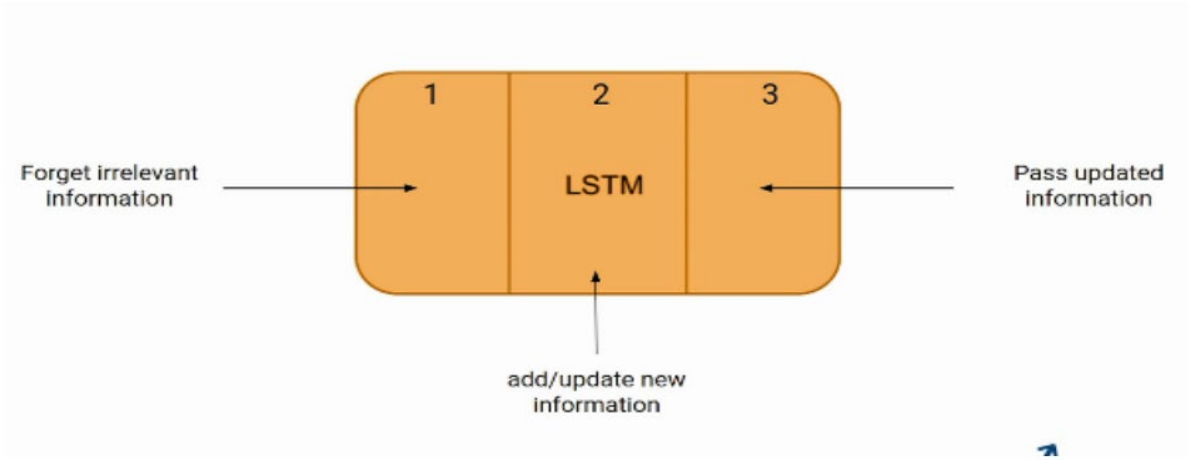
- Understand what LSTM is.
- Understand the architecture and working of an LSTM network.
- Learn about the different parts/gates in an LSTM unit.

What is LSTM?

- LSTMs Long Short-Term Memory is a type of RNNs Recurrent Neural Network that can detain long-term dependencies in sequential data.
- LSTMs are able to process and analyze sequential data, such as time series, text, and speech.
- They use a memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs.
- LSTMs are widely used in various applications such as natural language processing, speech recognition, and time series forecasting.

LSTM Architecture

- Here is the internal functioning of the LSTM network.
- The LSTM network architecture consists of three parts, as shown in the image below,
- Each part performs an individual function.



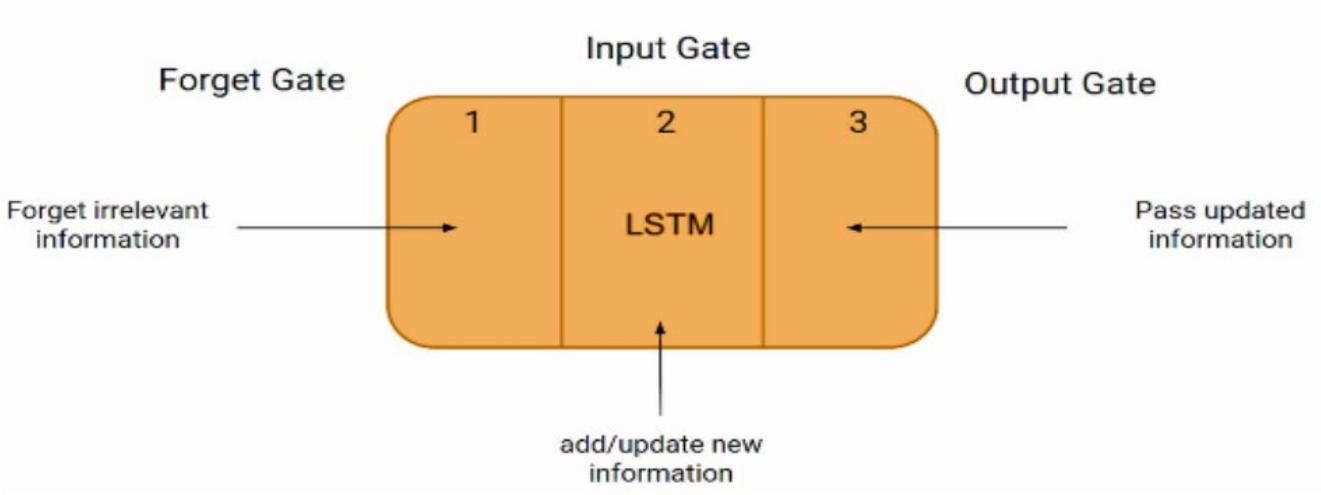
The Logic Behind LSTM

- The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten.
- In the second part, the cell tries to learn new information from the input to this cell.
- At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.
- This one cycle of LSTM is considered a single-time step.

The Logic Behind LSTM

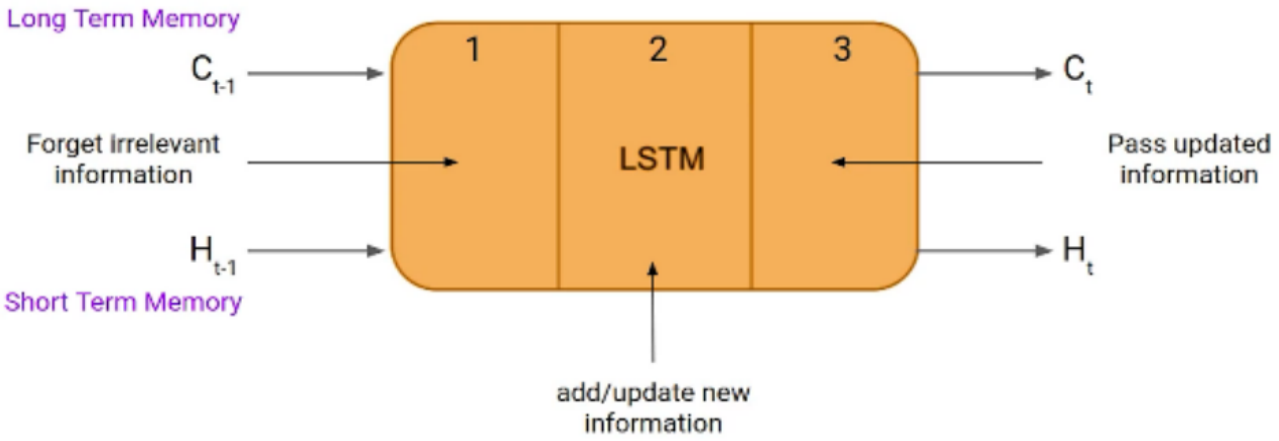
- These three parts of an LSTM unit are known as gates.
- They control the flow of information in and out of the memory cell or lstm cell.
- The first gate is called **Forget gate, the** second gate is known as **the Input gate,** and the last one is **the Output gate**.
- An LSTM unit that consists of these three gates and a memory cell or lstm cell can be considered as a layer of neurons in traditional feedforward neural network, with each neuron having a hidden layer and a current state.

LSTM with Gates

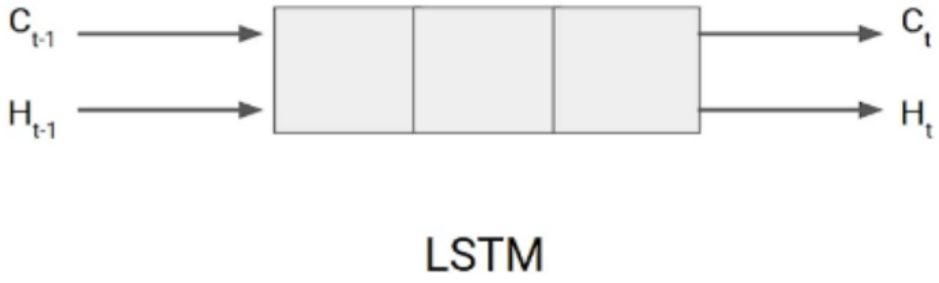


- Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp.
- In addition to that, LSTM also has a cell state represented by C(t-1) and C(t) for the previous and current timestamps, respectively.
- Here the hidden state is known as Short term memory, and the cell state is known as Long term memory.

LSTM with Gates



It is interesting to note that the cell state carries the information along with all the timestamps.



Example of LTSM Working

Bob is a nice person. Dan on the other hand is evil.

- Here we have two sentences separated by a full stop.
- The first sentence is “Bob is a nice person,” and the second sentence is “Dan, on the Other hand, is evil”.
- It is very clear, in the first sentence, we are talking about Bob, and as soon as we encounter the full stop(.), we started talking about Dan.
- As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Bob.
- Now our subject is Dan. Here, the Forget gate of the network allows it to forget about it.

Forget Gate

- In a cell of the LSTM neural network, the first step is to decide whether we should keep the information from the previous time step or forget it.
- Here is the equation for forget gate.

Forget Gate:

- $f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$

Forget Gate

- Let's try to understand the equation, here
- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with the hidden state
- Later, a sigmoid function is applied to it. That will make ft a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp, as shown below.

$C_{t-1} * f_t = 0$ ...if $f_t = 0$ (forget everything)

$C_{t-1} * f_t = C_{t-1}$ ...if $f_t = 1$ (forget nothing)

Input Gate

- “Bob knows swimming. He told me over the phone that he had served the navy for four long years.”
- So, in both these sentences, we are talking about Bob.
- However, both give different kinds of information about Bob.
- In the first sentence, we get the information that he knows swimming. Whereas the second sentence tells, he uses the phone and served in the navy for four years.

Input Gate

- The input gate is used to quantify the importance of the new information carried by the input.
- Here is the equation of the input gate

Input Gate:

- $i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$

Here,
•Xt: Input at the current timestamp t
•Ui: weight matrix of input
•Ht-1: A hidden state at the previous timestamp
•Wi: Weight matrix of input associated with hidden state
Again we have applied the sigmoid function over it.
As a result, the value of I at timestamp t will be between 0 and 1.

Output Gate

- “Bob single-handedly fought the enemy and died for his country. For his contributions, brave_____.”
- During this task, we have to complete the second sentence.
- Now, the minute we see the word brave, we know that we are talking about a person.
- In the sentence, only Bob is brave, we can not say the enemy is brave, or the country is brave. So based on the current expectation, we have to give a relevant word to fill in the blank. That word is our output, and this is the function of our Output gate.

Output Gate

- Here is the equation of the Output gate, which is pretty similar to the two previous gates.

Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

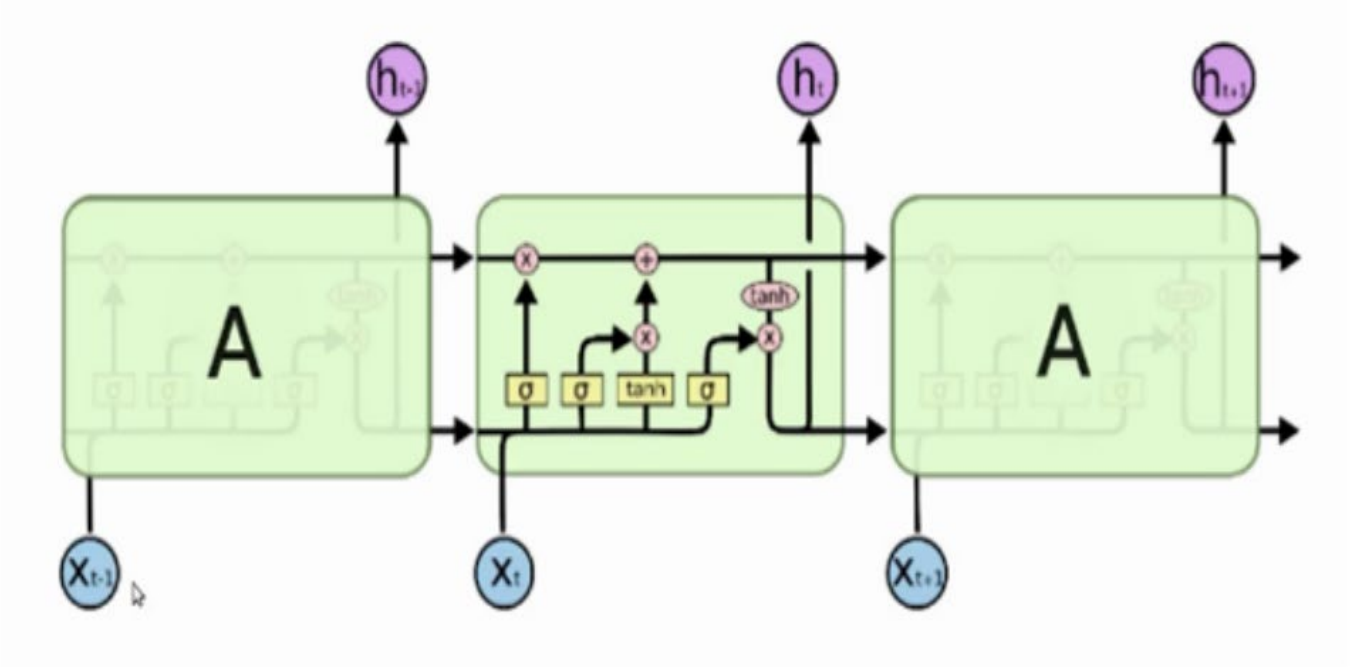
Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state, we will use Ot and tanh of the updated cell state. As shown below.

$$H_t = o_t * \tanh(C_t)$$

It turns out that the hidden state is a function of Long term memory (Ct) and the current output. If you need to take the output of the current timestamp, just apply the SoftMax activation on hidden state Ht.

$$\text{Output} = \text{Softmax}(H_t)$$

More intuitive diagram of the LSTM network



LTSM vs RNN

Aspect	LSTM (Long Short-Term Memory)	RNN (Recurrent Neural Network)
Architecture	A type of RNN with additional memory cells	A basic type of RNN
Memory Retention	Handles long-term dependencies and prevents vanishing gradient problem	Struggles with long-term dependencies and vanishing gradient problem
Cell Structure	Complex cell structure with input, output, and forget gates	Simple cell structure with only hidden state
Handling Sequences	Suitable for processing sequential data	Also designed for sequential data, but limited memory
Training Efficiency	Slower training process due to increased complexity	Faster training process due to simpler architecture
Performance on Long Sequences	Performs better on long sequences	Struggles to retain information on long sequences
Usage	Best suited for tasks requiring long-term memory, such as language translation and sentiment analysis	Appropriate for simple sequential tasks, such as time series forecasting
Vanishing Gradient Problem	Addresses the vanishing gradient problem	Prone to the vanishing gradient problem