Unit-1 Overview and Word Level Analysis

Introduction to Natural Language Processing

- Humans communicate with each other using words and text. The way that humans convey information to each other is called Natural Language.
- Every day humans share a large quality of information with each other in various languages as speech or text.
- However, computers cannot interpret this data, which is in natural language, as they communicate in 1s and 0s.
- The data produced is precious and can offer valuable insights. Hence, you need computers to be able to understand, emulate and respond intelligently to human speech.
- Natural Language Processing or NLP refers to the branch of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages

- NLP combines the field of linguistics and computer science to decipher language structure and guidelines and to make models which can comprehend, break down and separate significant details from text and speech

Definition:

- NLP is a branch of artificial intelligence that deals with analyzing, understanding, and generating the languages that humans use naturally in order to interface with computers in both written and spoken contexts using natural human languages instead of computer languages.
- Natural language simply refers to the way we communicate with each other: speech and text.
- Processing refers to making natural language usable for computational tasks.

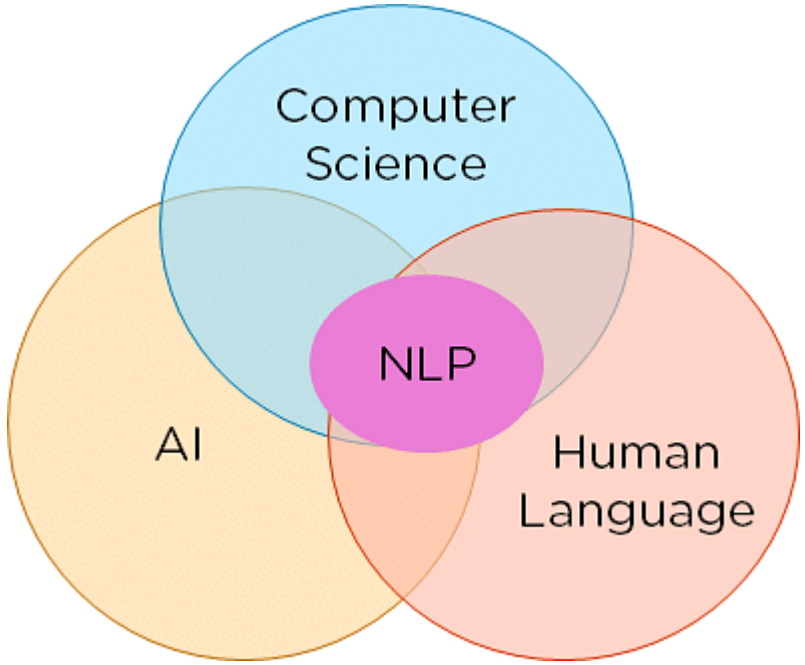


Figure 1: Constituents of NLP

1. Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on the interaction between computers and humans using natural language. It involves analyzing, understanding, and generating human language data, such as text and speech.
2. NLP has a wide range of applications, including sentiment analysis, machine translation, text summarization, chatbots, and more. Some common tasks in NLP include:
    1. Text Classification: Classifying text into different categories based on their content, such as spam filtering, sentiment analysis, and topic modeling.
    2. Named Entity Recognition (NER): Identifying and categorizing named entities in text, such as people, organizations, and locations.
    3. Part-of-Speech (POS) Tagging: Assigning a part of speech to each word in a sentence, such as noun, verb, adjective, and adverb.
    4. Sentiment Analysis: Analyzing the sentiment of a piece of text, such as positive, negative, or neutral.
    5. Machine Translation: Translating text from one language to another.

Applications of NLP

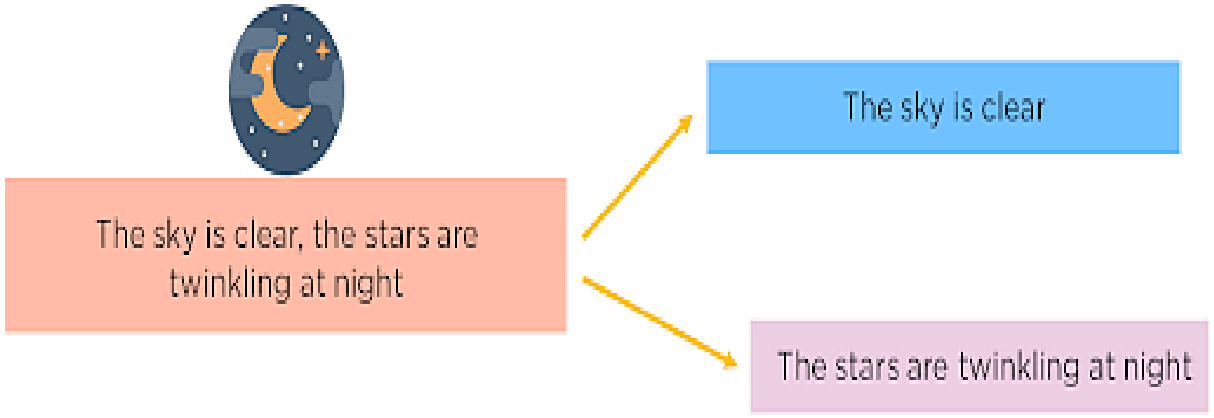
- Machine translation(Google Translate)
- Natural language generation
- Web Search
- Spam filters
- Sentiment Analysis
- Chatbots

How to Perform NLP?

- The steps to perform preprocessing of data in NLP include:

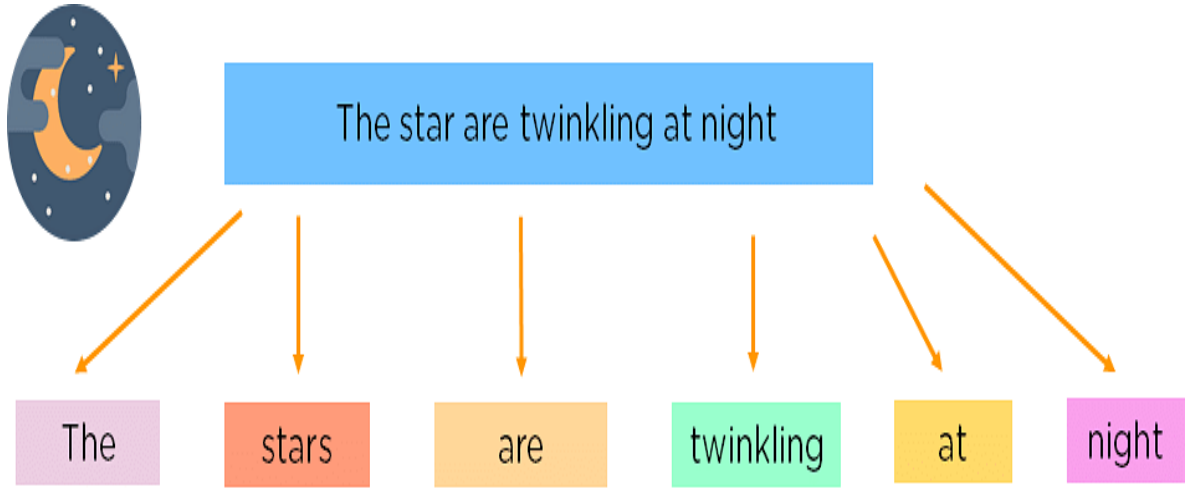
Segmentation:

- You first need to break the entire document down into its constituent sentences. You can do this by segmenting the article along with its punctuations like full stops and commas.



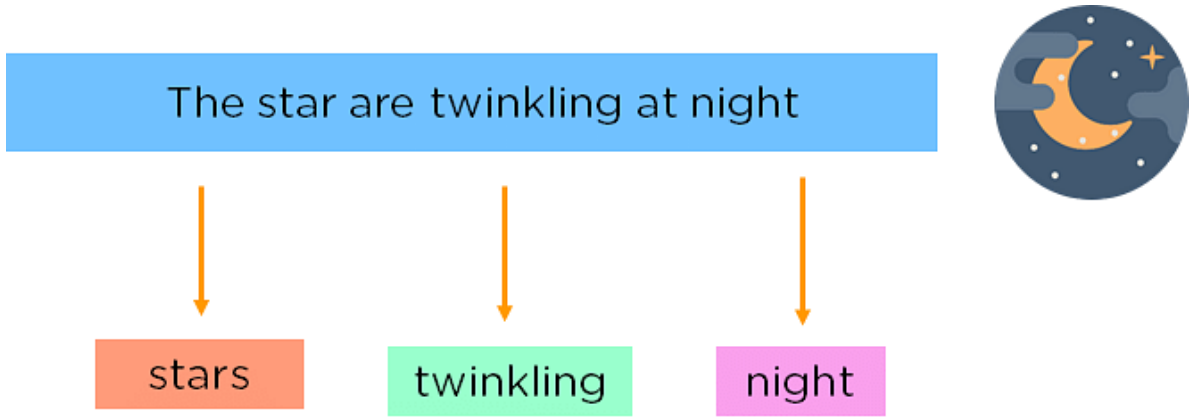
Tokenizing:

- For the algorithm to understand these sentences, you need to get the words in a sentence and explain them individually to our algorithm. So, you break down your sentence into its constituent words and store them. This is called tokenizing, and each world is called a token.



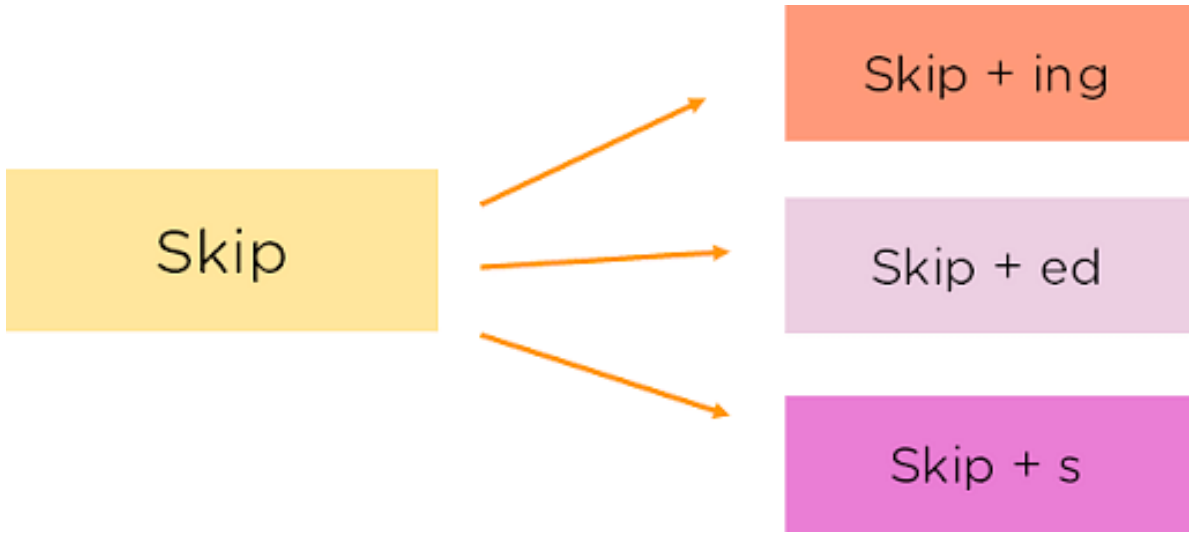
Removing Stop Words:

- Can make the learning process faster by getting rid of non-essential words, which add little meaning to our statement and are just there to make our statement sound more cohesive. Words such as was, in, is, and, the, are called stop words and can be removed.



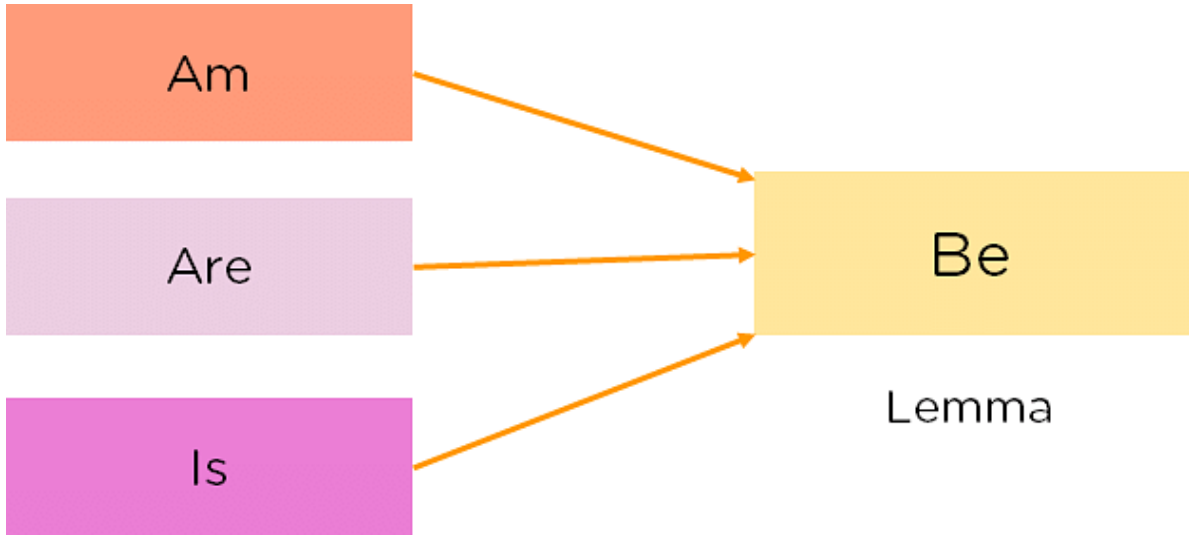
Stemming:

- It is the process of obtaining the Word Stem of a word. Word Stem gives new words upon adding affixes to them



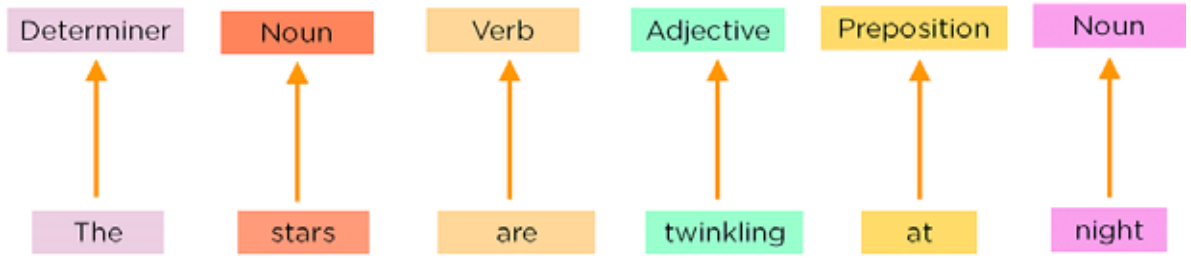
Lemmatization:

- The process of obtaining the Root Stem of a word. Root Stem gives the new base form of a word that is present in the dictionary and from which the word is derived. You can also identify the base words for different words based on the tense, mood, gender,etc.

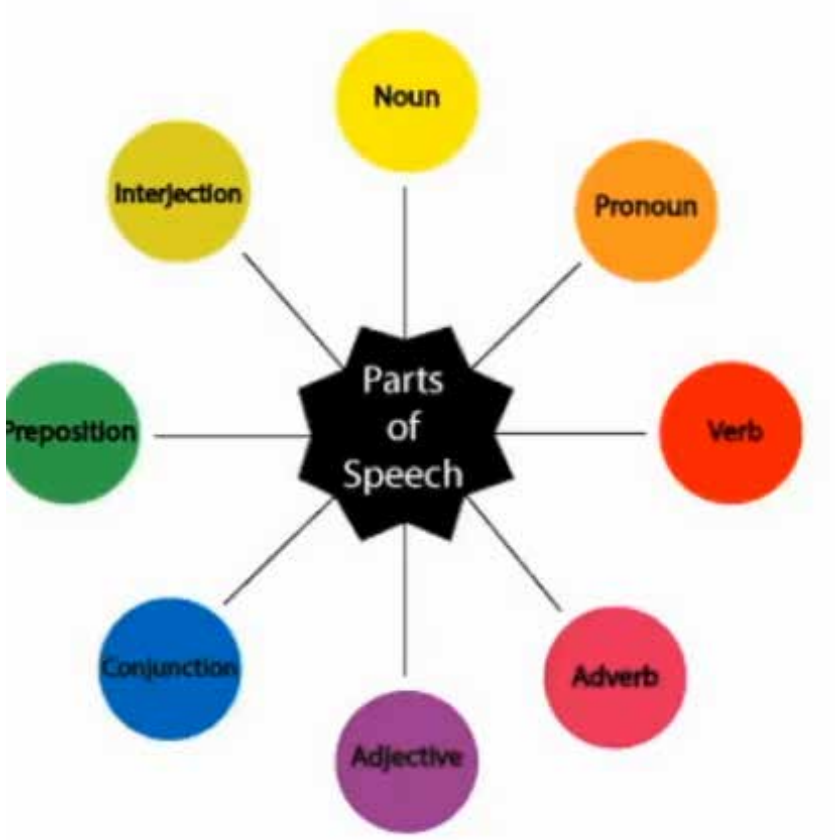


Part of Speech Tagging:

- Now, you must explain the concept of nouns, verbs, articles, and other parts of speech to the machine by adding these tags to our words. This is called ‘part of’.



- Parts of speech tags are the properties of the words, which define their main context, functions, and usage in a sentence. Some of the commonly used parts of speech tags are

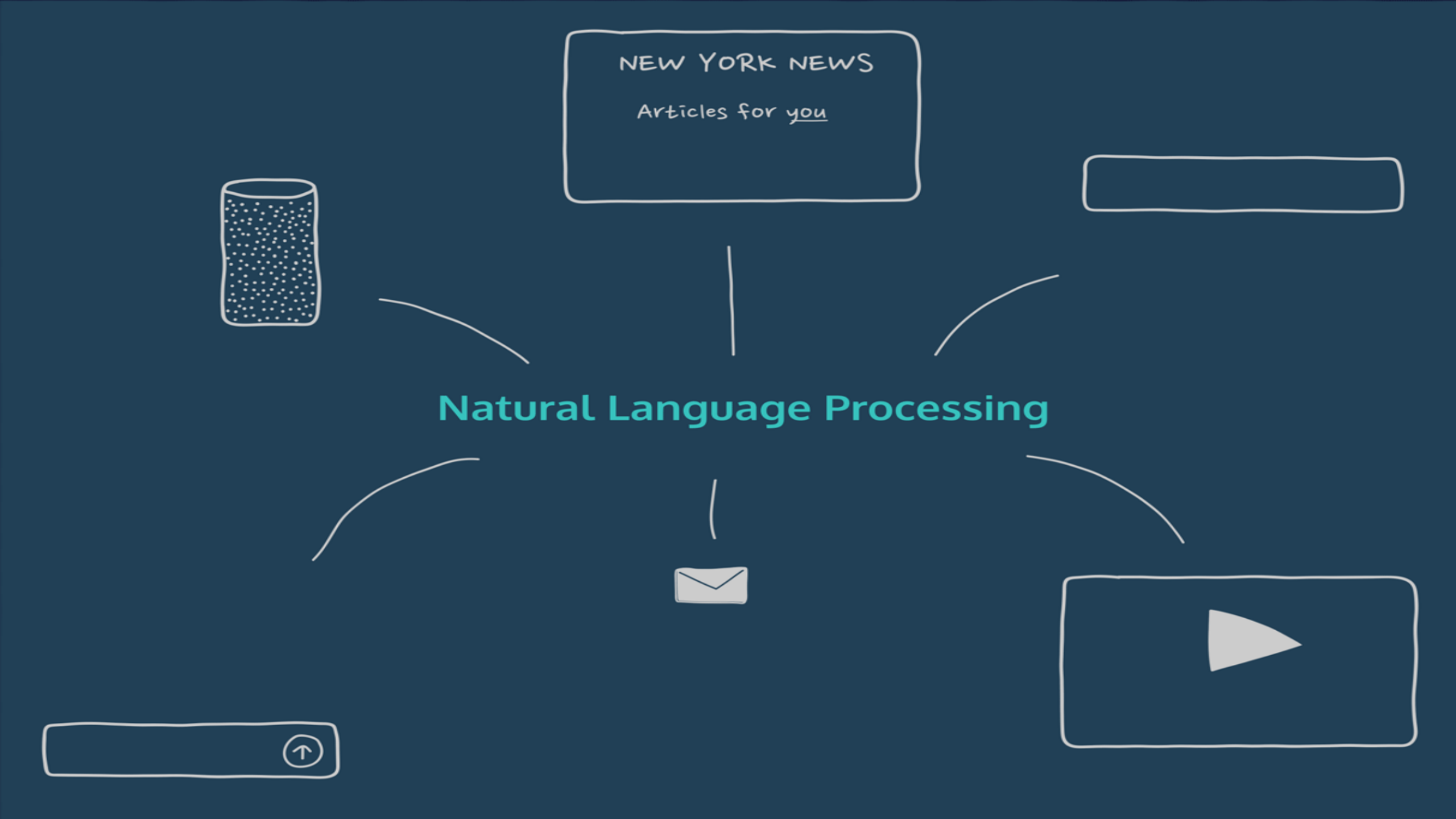


Named Entity Tagging:

- Next, introduce your machine to pop culture references and everyday names by flagging names of movies, important personalities or locations, etc that may occur in the document. You do this by classifying the words into subcategories. This helps you find any keywords in a sentence. The subcategories are person, location, monetary value, quantity, organization, movie.
- After performing the preprocessing steps, you then give your resultant data to a machine learning algorithm like Naive Bayes, etc., to create your NLP application.

Different Levels of NLP

- Natural language processing, also referred to as text analytics, plays a very vital role in today's era because of the sheer volume of text data that users generate around the world on digital channels such as social media apps, e-commerce websites, blog posts, etc. Natural Language Processing works on multiple levels and most often, these different areas synergize well with each other.


NEW YORK NEWS
Articles for you
Natural Language Processing

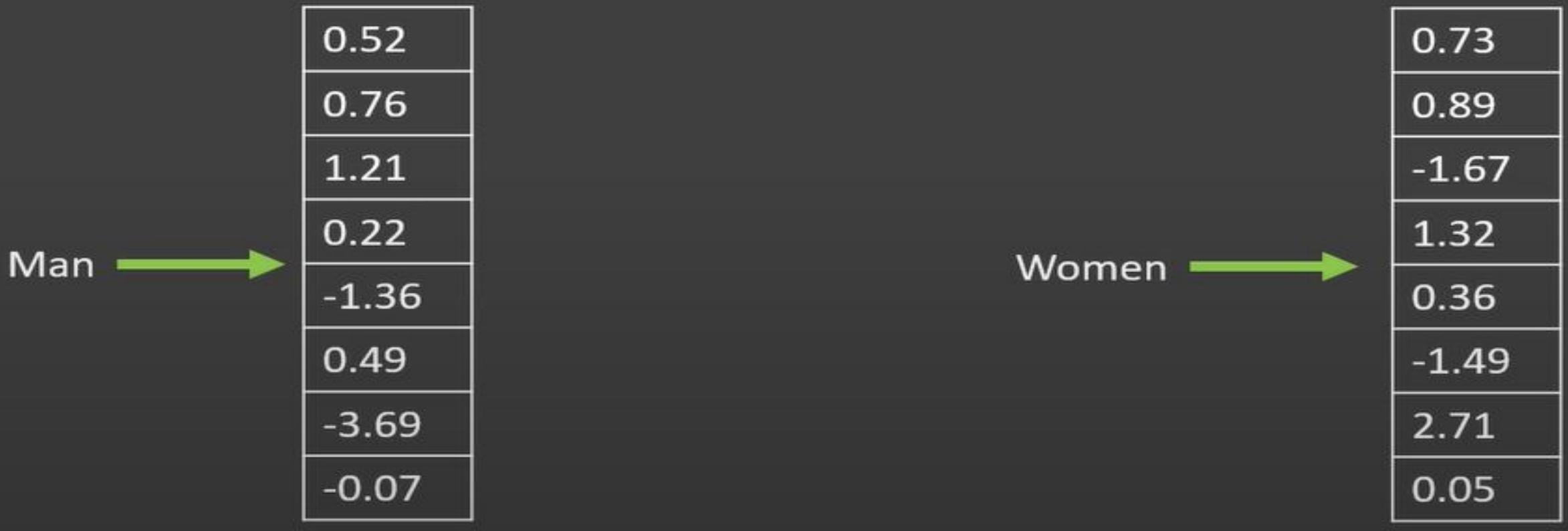
Word Embedding

What is Word2Vec?

- Word2Vec creates vectors of the words that are distributed numerical representations of word features – these word features could comprise of words that represent the context of the individual words present in our vocabulary.
- Word embeddings eventually help in establishing the association of a word with another similar meaning word through the created vectors.

What is Word2Vec ?

- A two layer neural network to generate word embeddings given a text corpus.
- Word Embeddings – Mapping of words in a vector space.



Working of word2Vec

- The word2vec objective function causes the words that occur in similar contexts to have similar embeddings.

Example: The **kid** said he would grow up to be superman.
The **child** said he would grow up to be superman.

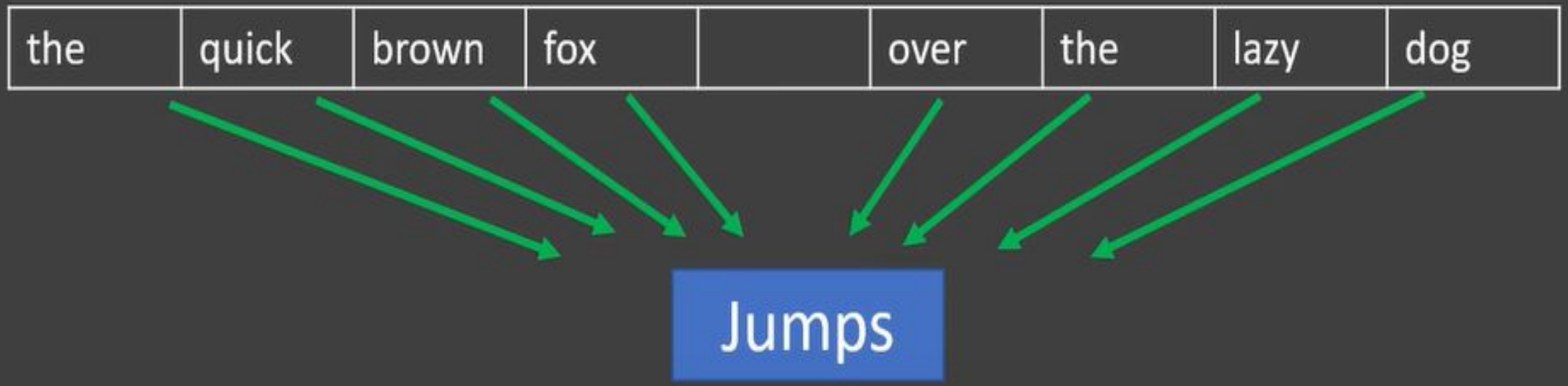
The words kid and child will have similar word vectors due to a similar context.

Types:

- Two different model architectures that can be used by Word2Vec to create the word embeddings are the Continuous Bag of Words (CBOW) model & the Skip-Gram model.

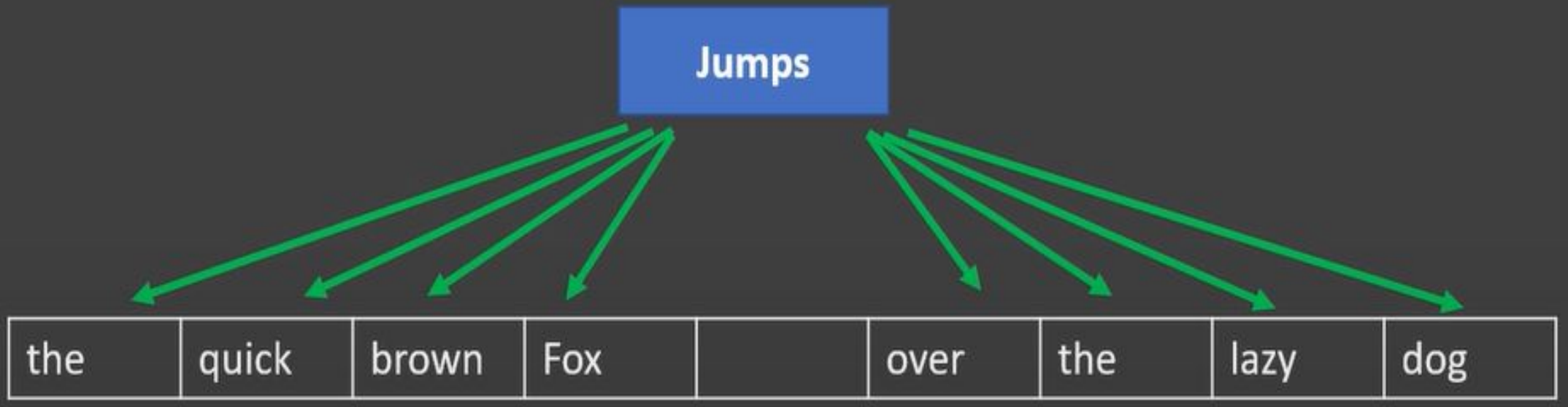
CBOW

- Predict the target word from the context.



Skip Gram

- Predict the context words from target.

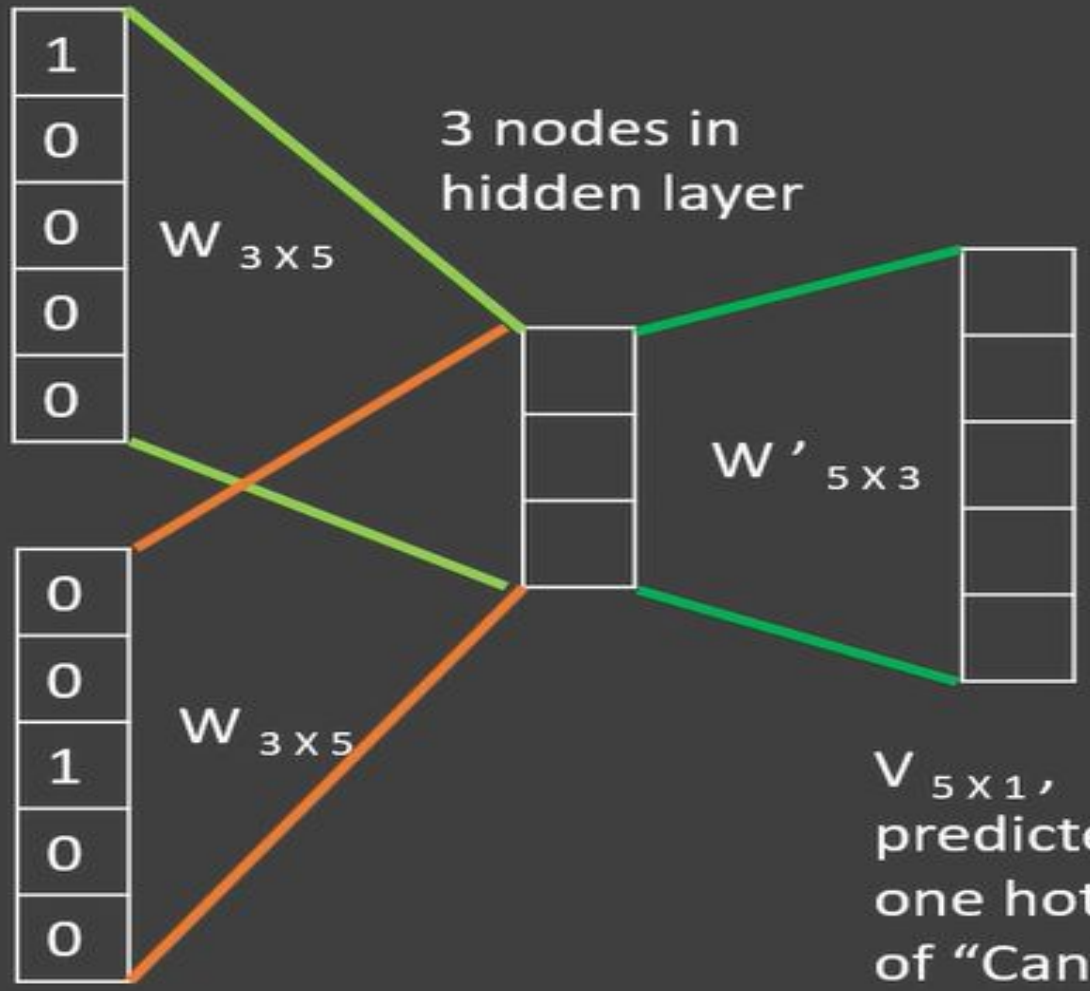


CBOW - Working

Hope can set you free.

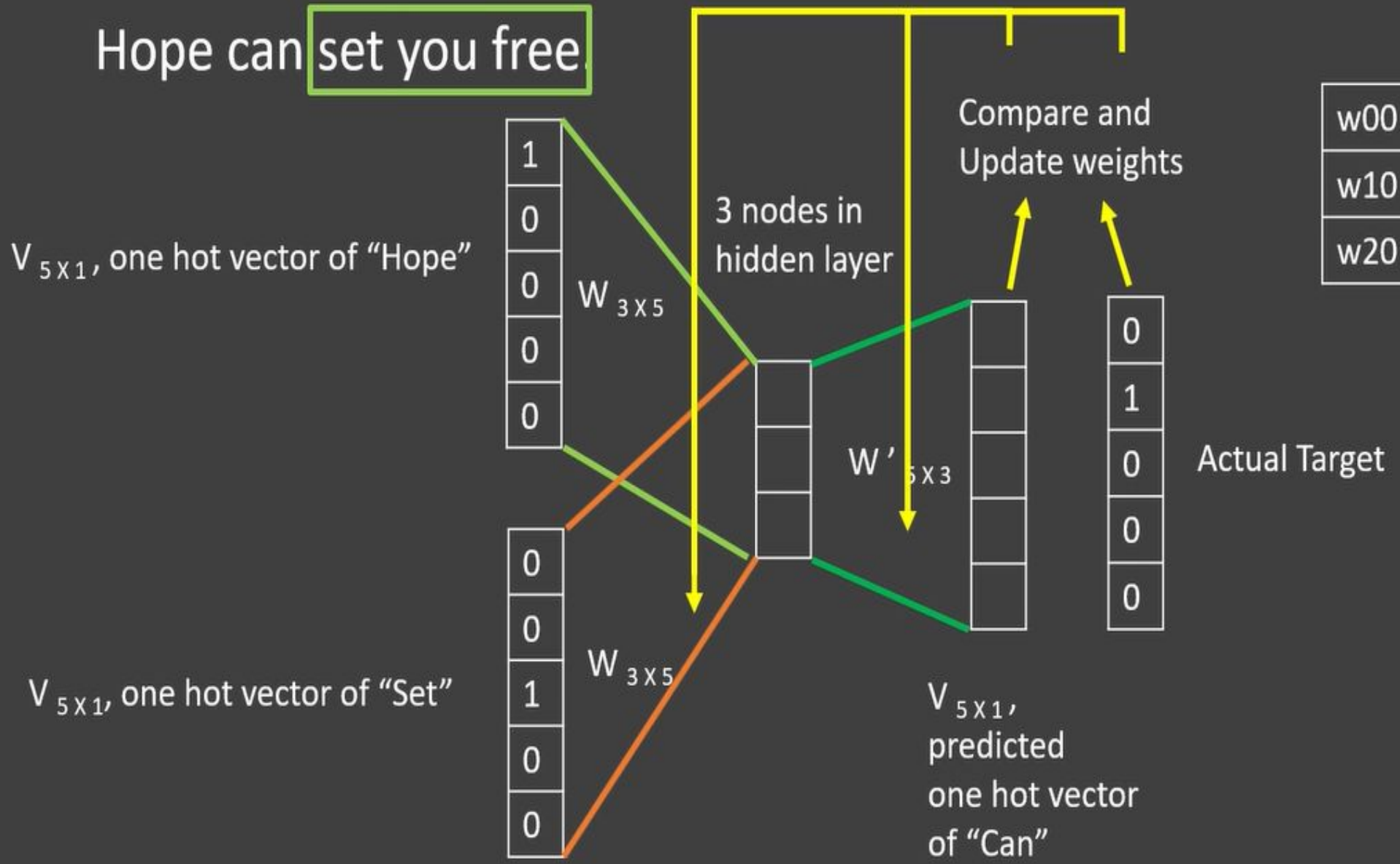
$V_{5 \times 1}$, one hot vector of "Hope"

$V_{5 \times 1}$, one hot vector of "Set"



CBOW - Working

Hope can set you free

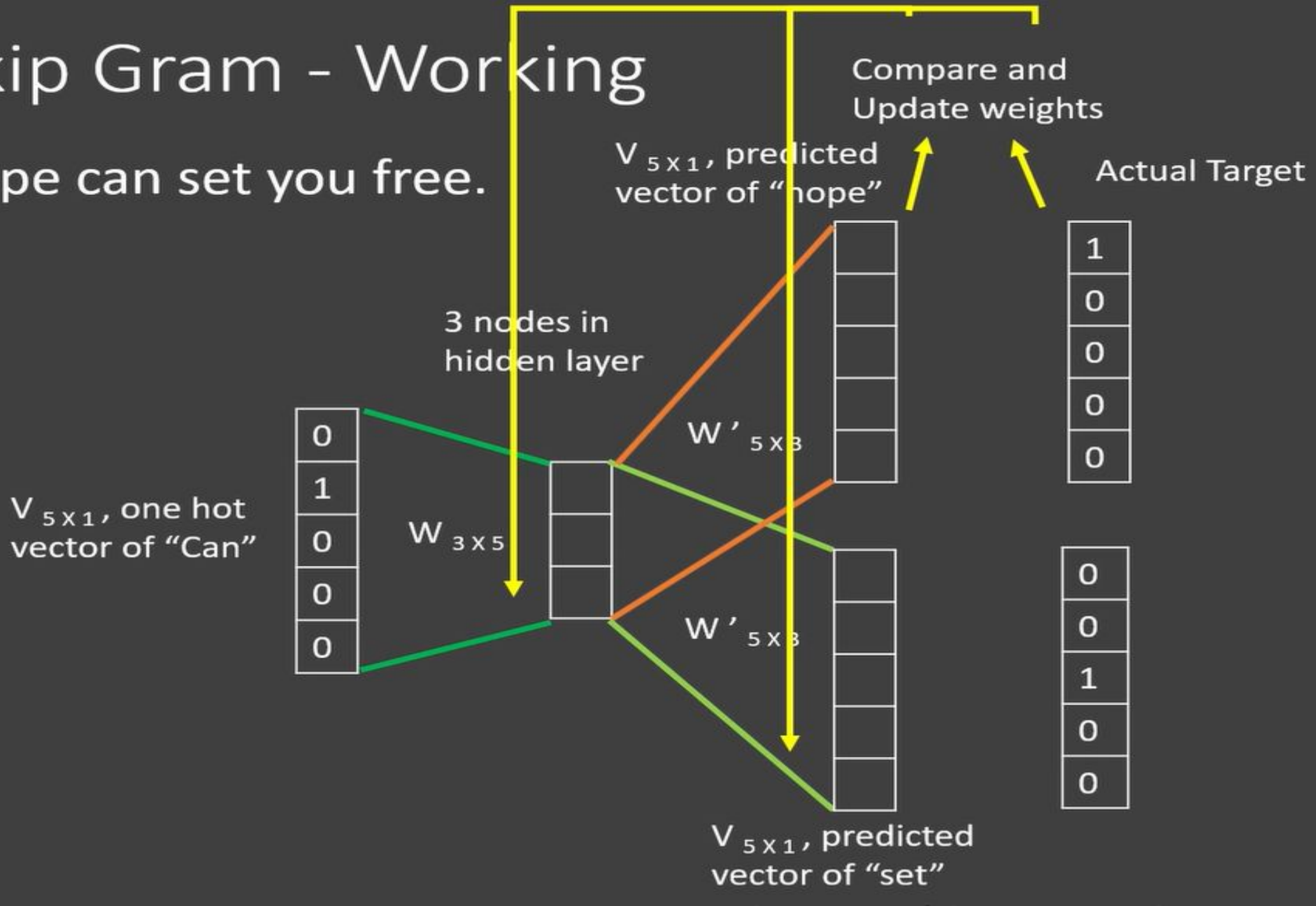


w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

W 3 X 5

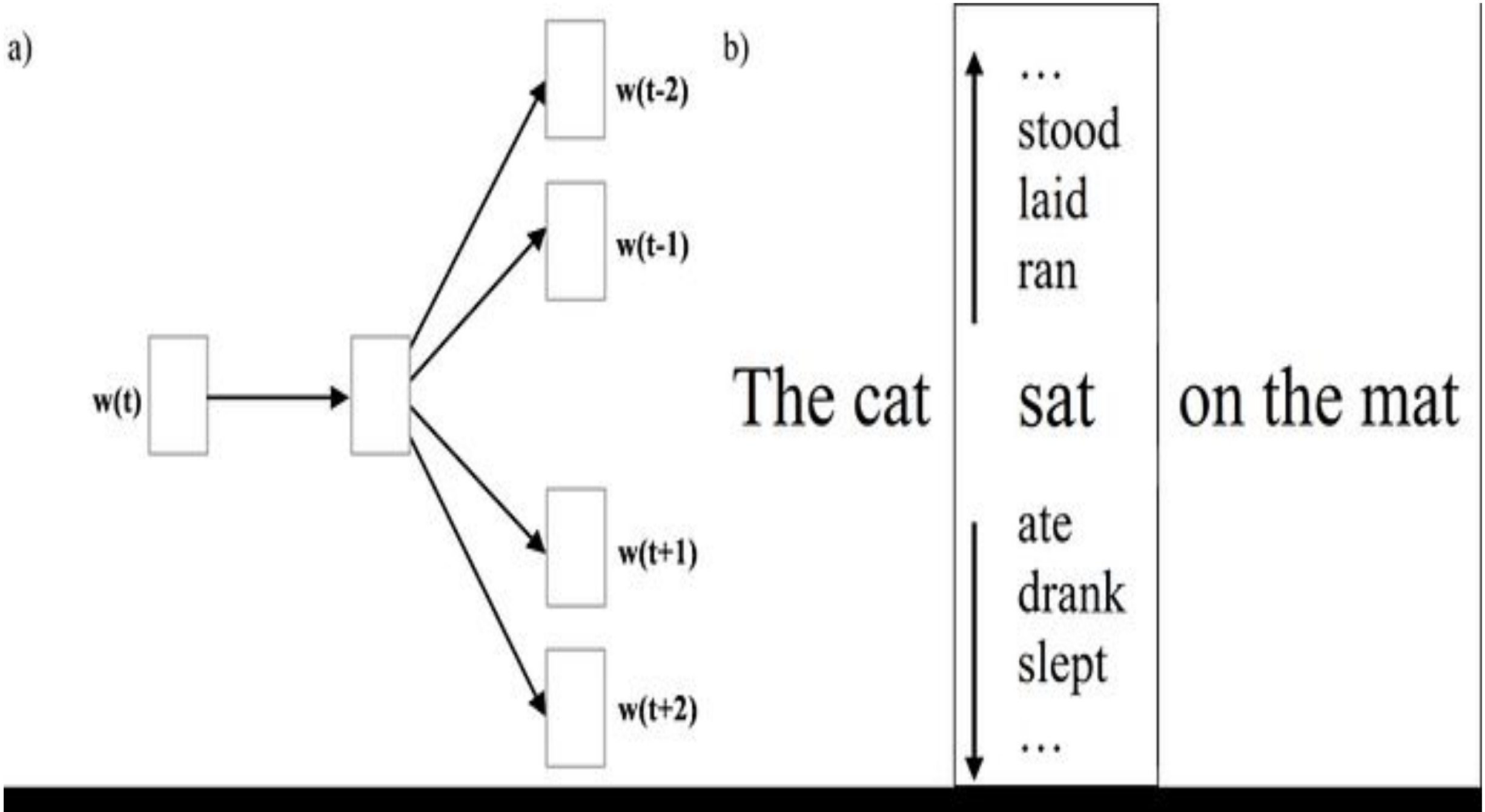
Skip Gram - Working

Hope can set you free.



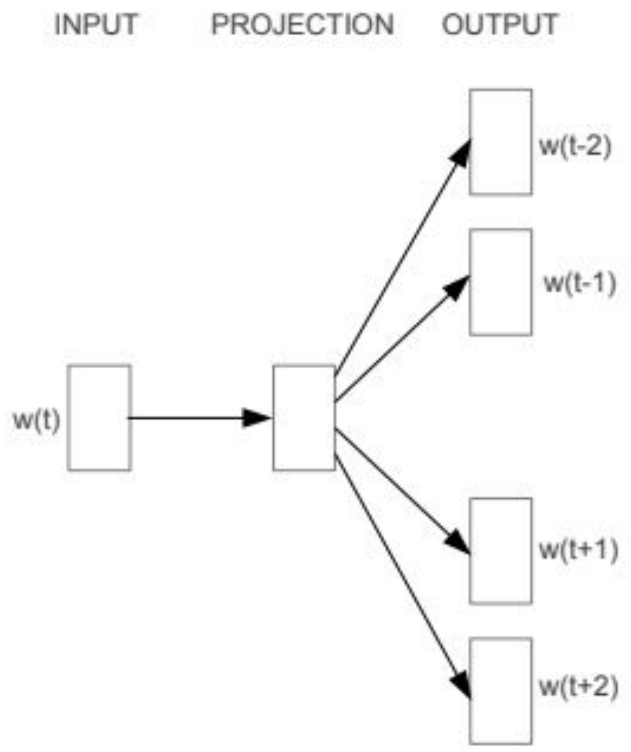
Skip gram:

- Skip-gram is used to predict the context word for a given target word. It’s reverse of CBOW algorithm. Here, target word is input while context words are output.


a)
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
b)
The cat
sat
on the mat
…
stood
laid
ran
ate
drank
slept
…

- The word sat will be given and we’ll try to predict words cat, mat at position -1 and 3 respectively given sat is at position 0 . We do not predict common or stop words such as the .

Architecture



Skip-gram

As we can see w(t) is the target word or input given. There is one hidden layer which performs the dot product between the weight matrix and the input vector w(t).

No activation function is used in the hidden layer. Now the result of the dot product at the hidden layer is passed to the output layer.

Output layer computes the dot product between the output vector of the hidden layer and the weight matrix of the output layer.

Then we apply the softmax activation function to compute the probability of words appearing to be in the context of w(t) at given context location.

https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c#:~:text=Skip%2Dgram%20is%20one%20of,while%20context%20words%20are%20output.

Discourse and Coherence

- **Discourse**: Discourse refers to how sentences or utterances are connected and organized to convey meaning in a conversation or text. It's about understanding the larger structure and flow of communication beyond just individual sentences. In NLP, analyzing discourse helps in understanding how different parts of a text relate to each other and contribute to its overall meaning.
- **Coherence**: Coherence is about the logical and meaningful connection between ideas or sentences within a text. It's the quality that makes a text easy to understand because the ideas flow smoothly and logically from one to the next. Coherent texts are organized and structured in a way that makes sense to the reader or listener.

Example:

- Imagine you're reading a paragraph about the process of making a sandwich:
- *First, you take two slices of bread. Then, you spread peanut butter on one slice and jelly on the other. Next, you put the two slices together to form a sandwich.*
- In this example:
- **Discourse**: It's about understanding the sequence of events or steps involved in making a sandwich. Each sentence builds upon the previous one to describe the process.
- **Coherence**: The sentences are logically connected, with each step following naturally from the one before it. The reader can easily understand the progression from taking the bread to assembling the sandwich because the ideas are coherent and flow in a logical order.

"The Tin Woodman went to the Emerald City to see the Wizard of Oz and ask for a heart. After he asked for it, the Woodman waited for the Wizard's response"

- What do pronouns such as he and it denote?
- No doubt the reader had little trouble figuring out that he denotes the Tin Woodman and not the Wizard of Oz, and that it denotes the heart and not the Emerald City.
- Furthermore, it is clear to the reader that the Wizard is the same entity as the Wizard of Oz, and the Woodman is the same as the Tin Woodman. But doing this disambiguation automatically is a difficult task.
- This goal of deciding what pronouns and other noun phrases refer to is called **coreference resolution**.

- Coreference resolution is important for **information extraction**, **summarization**, and for **conversational agent**
- Consider the task of summarizing the following news passage:

(21.2) First Union Corp is continuing to wrestle with severe problems. According to industry insiders at Paine Webber, their president, John R. Georgius, is planning to announce his retirement tomorrow.

We might want to extract a summary like the following extract a summary like the following:

(21.3) First Union President John R. Georgius is planning to announce his retirement tomorrow.

- In order to build such a summary, we need to know that the second sentence is the more important of the two, and that the first sentence is subordinate to it, just giving background information. Relationships of this sort between sentences in a discourse are called **coherence relations**, and determining the coherence structures between discourse sentences is an important discourse task.

Concept of Coherence

- **Coherence** in terms of Discourse in NLP means making sense of the utterances or making meaningful connections and correlations.
- There is a lot of connection between the coherence and the discourse structure

What are coherent discourse texts?

- If we read a paragraph from a newspaper, we can see that the entire paragraph is interrelated;
- Hence we can say that the discourse is coherence, but if we only combine the newspaper headlines consecutively, then it is not a discourse, it is just a group of sentences that are also non-coherence.

Coherence Relation Between Utterances

- When we say that the discourses are coherent, then it simply means that the discourse has some sort of meaningful connection. The **coherent relation** tells us that there is some sort of connection present between the utterances.
- **Relationship Between Entities**
- If there is some kind of relationship between the entities, then we can also say that the discourse in NLP is coherent. So, the coherence between the entities is known as **entity-based coherence**.

Discourse Structure

- The structure of the discourse depends on the type of segmentation applied to the discourse.

DISCOURSE SEGMENTATION

- When we determine the types of structures for a large discourse, we term its segmentation. The segmentation is a difficult thing to implement, but it is very necessary as discourse segmentation is used in fields like :
  - Information Retrieval,
  - Text summarization,
  - Information Extraction, etc.

Discourse Segmentation

- Separating a document into linear sequence of sub topics
- Discourse Segment: Stretches of discourse in which the sentences are addressing the same topic.
- A sequence of clauses that display local coherence
- A sequence of clauses possibly interrupted by sub segments forming a hierarchical structure
- Each segment is a coherent unit, Should have the following properties:
  - Some technique based on recency should be usable in referential analysis handling ellipses
  - A simple progression in time and location
  - A fixed set of speakers and hearers
  - A fixed set of background assumptions

E: So you have the engine assembly finished?

Now attach the pull rope to the top of the engine.

By the way, did you buy gasoline today?

A: *Yes. I got some when I bought the new lawnmower wheel.*

I forgot to take the gas can with me, so I bought a new one.

E: Did it cost much?

A: *No, and we could use another anyway to keep with the tractor.*

E: OK, how far have you got?

Did you get it attached?

Algorithms for Discourse Segmentation

- Unsupervised Discourse Segmentation
- Supervised Discourse Segmentation

Unsupervised Discourse Segmentation

- The class of unsupervised segmentation is also termed or represented as linear segmentation. Let us take an example to understand this discourse segmentation better.
- Suppose we have a text with us, and the task is to segment the text into various units of multi-paragraphs. In the multi-paragraphs, a single unit is going to represent a passage of the text.
- Now the algorithm will take the help of cohesion and the algorithm will classify the dependent texts and tie them together using some linguistic devices.
- In simpler terms, unsupervised discourse segmentation means the classification and grouping up of similar texts with the help of coherent discourse in NLP.
- The **unsupervised discourse segmentation** can also be performed with the help of lexicon cohesion. The lexicon cohesion indicates the relationship among similar units, for example, synonyms.

- **Coherence** and **cohesion** are often confused; let's review the difference.
- **Cohesion** refers to the way textual units are tied or linked together.
- A cohesive relation is like a kind of glue grouping together two units into a single unit.
- **Coherence** refers to the *meaning* relation between the two units.
- A coherence relation explains how the meaning of different textual units can combine to jointly build a discourse meaning for the larger unit.

Example:

- **Lexical cohesion**

It is the cohesion indicated by relations between words in the two units, such as use of an identical word, a synonym, or a hypernym.

For example the fact that the words *house*, *shingled*, and *I* occur in both of the two sentences in (21.8ab), is a cue that the two are tied together as a discourse:

(21.8) • Before winter **I** built a chimney, and **shingled** the sides of my **house**...

• **I** have thus a tight **shingled** and plastered **house**

In Ex. (21.9), lexical cohesion between the two sentences is indicated by the hypernym relation between *fruit* and the words *pears* and *apples*.

(21.9) Peel, core and slice **the pears and the apples**. Add **the fruit** to the skillet.

Example:

- "John had always loved camping. He would often spend weekends in the wilderness, surrounded by towering trees and the sounds of nature. However, this time was different. The weather forecast had predicted heavy rain, and John was concerned about his tent's ability to withstand the storm. Despite his worries, he decided to go ahead with the trip. As he set up camp, dark clouds loomed overhead, and soon enough, the rain started pouring down."

Using these cues, the example text could be segmented into the following clusters:

1. Segment 1:
    1. "John had always loved camping."
    2. "He would often spend weekends in the wilderness, surrounded by towering trees and the sounds of nature."
2. Segment 2:
    1. "However, this time was different."
    2. "The weather forecast had predicted heavy rain, and John was concerned about his tent's ability to withstand the storm."
    3. "Despite his worries, he decided to go ahead with the trip."
    4. "As he set up camp, dark clouds loomed overhead, and soon enough, the rain started pouring down."

SEG1
a) Jack and Sue went to buy a new lawn mower
b) since their old one was stolen.

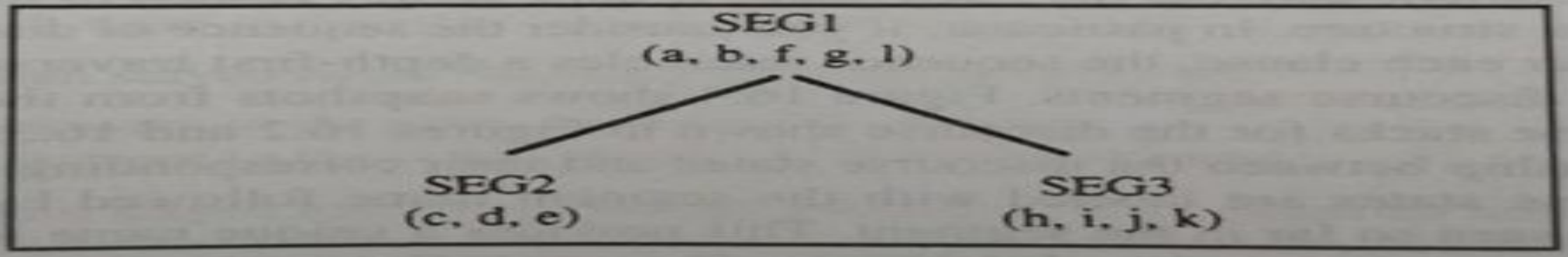
SEG2
c) Sue had seen the men who took it and
d) she had chased them down the street,
e) but they'd driven away in a truck.

f) After looking in the store,
g) they realized they couldn't afford a new one.

SEG3
h) By the way, Jack lost his job last month
i) so he's been short of cash recently.
j) He has been looking for a new one,
k) but so far hasn't had any luck.

l) Anyway, they finally found a used one at a garage sale.

Figure 16.2 The segment hierarchy represented by boxing



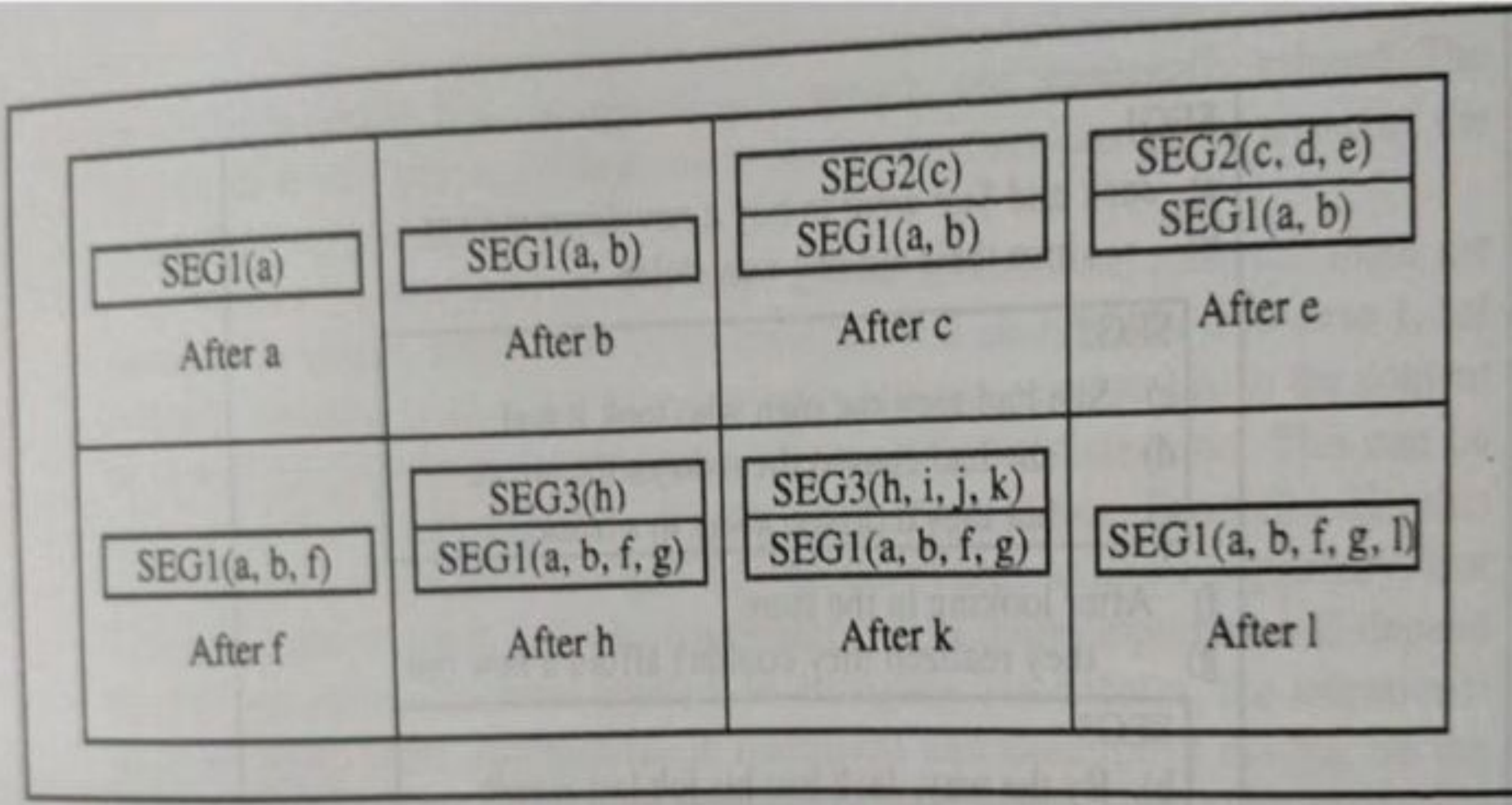


Figure 16.4 Part of the sequence of discourse stacks for the same discourse

Supervised Discourse Segmentation

- In the previous segmentation, there was no certain labeled segment boundary to separate the discourse segments.
- But in the supervised discourse segmentation, we only deal with the training data set having a labeled boundary.
- To differentiate or structure the discourse segments, we make use of cue words or discourse makers.
- These cue words or discourse maker works to signal the discourse structure.
- As there can be varied domains of discourse in NLP so, the cue words or discourse makers are domain specific.
- A key additional feature that is often used for supervised segmentation is the presence of discourse markers or cue words. A discourse marker is a word or phrase that functions to signal discourse structure.

Example:

- "John had always loved camping. He would often spend weekends in the wilderness, surrounded by towering trees and the sounds of nature. However, this time was different. The weather forecast had predicted heavy rain, and John was concerned about his tent's ability to withstand the storm. Despite his worries, he decided to go ahead with the trip. As he set up camp, dark clouds loomed overhead, and soon enough, the rain started pouring down."

Segmented Text:

1. John had always loved camping.
2. He would often spend weekends in the wilderness, surrounded by towering trees and the sounds of nature.
3. However, this time was different.
4. The weather forecast had predicted heavy rain, and John was concerned about his tent's ability to withstand the storm.
5. Despite his worries, he decided to go ahead with the trip.
6. As he set up camp, dark clouds loomed overhead, and soon enough, the rain started pouring down.

- **Cues and Markers:** Cues and markers are linguistic features or patterns that signal the beginning or end of a discourse segment. In the provided example, some cues and markers include:

1. **Transition Words and Phrases:** Words like "however," "despite," and "as" indicate shifts in the narrative or introduction of new information.
2. **Punctuation:** Periods (.) are often used to denote the end of a sentence and consequently, the end of a segment. However, not all periods signify segment boundaries, as some may occur within sentences.
3. **Temporal and Spatial Indicators:** Phrases like "this time," "soon enough," and "as he" provide temporal or spatial context, which can often signal the beginning or continuation of a new segment.
4. **Contextual Cues:** In this example, the change in topic from John's love for camping to his concern about the weather serves as a significant cue for segment boundary.

Text Coherence

- Lexical repetition is a way to find the structure in a discourse, but it does not satisfy the requirement of being coherent discourse.
- To achieve the coherent discourse, we must focus on coherence relations in specific.
- As we know that coherence relation defines the possible connection between utterances in a discourse.

Result

- We are taking two terms $\mathbf{S_0}$ and $\mathbf{S_1}$ to represent the meaning
- It infers that the state asserted by term $\mathbf{S_0}$ could cause the state asserted by $\mathbf{S_1}$.
- For example, two statements show the relationship result:

"Ram was caught in the fire. His skin burned."

Explanation

- It infers that the state asserted by $\mathbf{S_1}$ could cause the state asserted by $\mathbf{S_0}$.
- For example, two statements show the relationship –

  "Ram fought with Shyam's friend. He was drunk."

Parallel

- It infers p(a1,a2,…) from assertion of $\mathbf{S_0}$ and p(b1,b2,…) from assertion $\mathbf{S_1}$. Here ai and bi are similar for all i.
- For example, two statements are parallel –

  “Ram wanted car. Shyam wanted money.”

Elaboration

- It infers the same proposition P from both the assertions – $\mathbf{S_0}$ and $\mathbf{S_1}$
- For example, two statements show the relation elaboration:

  "Ram was from Chandigarh. Shyam was from Kerala."

Occasion

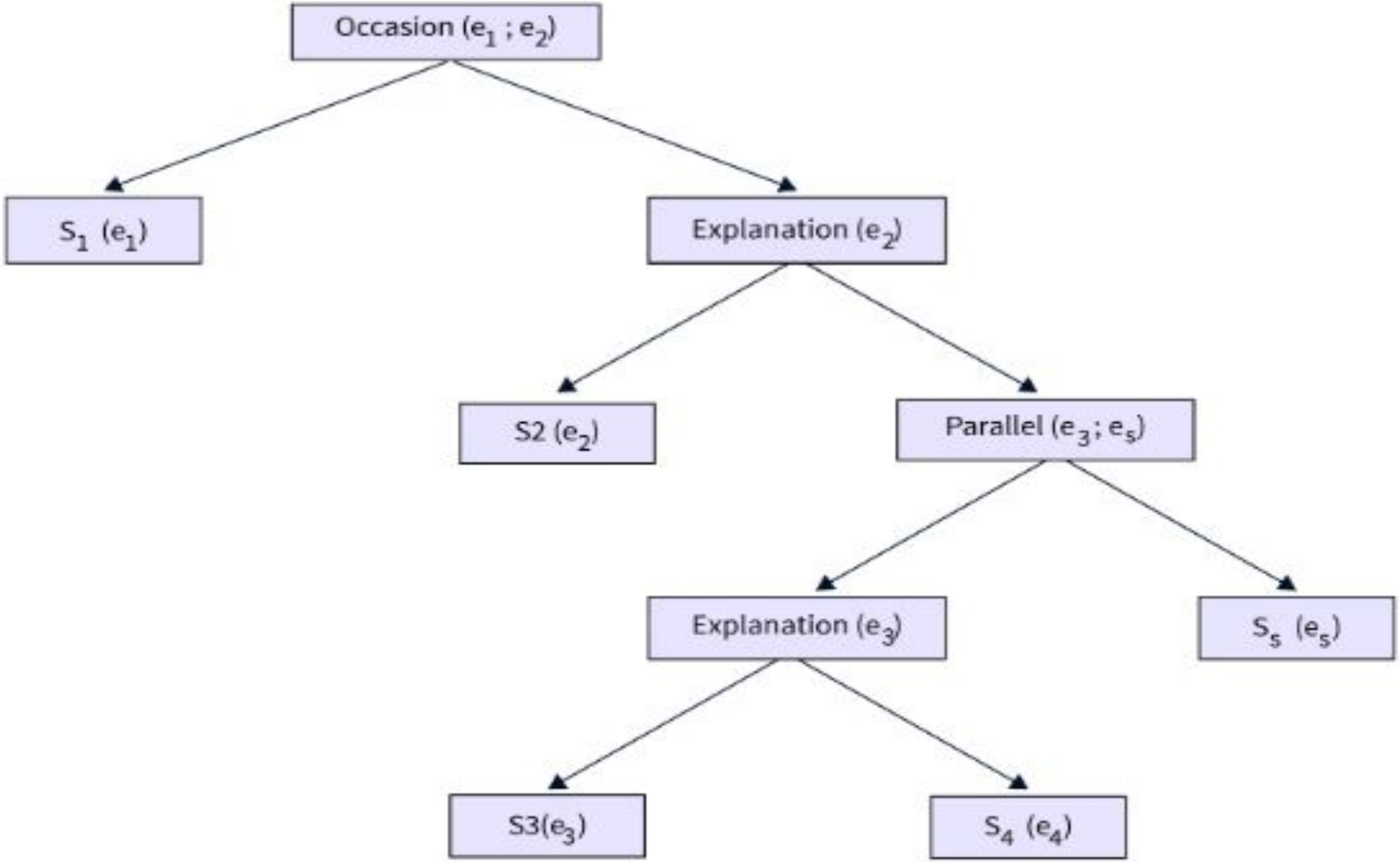
- It happens when a change of state can be inferred from the assertion of $\mathbf{S_0}$, final state of which can be inferred from $\mathbf{S_1}$ and vice-versa.
- For example, the two statements show the relation occasion:

"Ram picked up the book. He gave it to Shyam."

Building Hierarchical Discourse Structure

- In the previous section, we discussed how text coherence takes place. Let us now try to build a hierarchal discourse structure with the help of a group of statements. We generally create the hierarchical structure among the coherence relations to get the entire discourse in NLP.

- Let us consider the following phrases and serially number them.
- **S1:**
  Rahul went to the bank to deposit money.
- **S2:**
  He then went to Rohan's shop.
- **S3 :**
  He wanted a phone.
- **S4 :**
  He did not have a phone.
- **S5:**
  He also wanted to buy a laptop from Rohan's shop.


Occasion (e_1 ; e_2)
S_1 (e_1)
Explanation (e_2)
S2 (e_2)
Parallel (e_3 ; e_5)
Explanation (e_3)
S_5 (e_5)
S3(e_3)
S_4 (e_4)

We can also talk about the coherence of an entire discourse, by considering the hierarchical structure between coherence relations. Consider passage (21.21).

(21.21) John went to the bank to deposit his paycheck. (S1)
He then took a train to Bill's car dealership. (S2)
He needed to buy a car. (S3)
The company he works for now isn't near any public transportation. (S4)
He also wanted to talk to Bill about their softball league. (S5)

Intuitively, the structure of passage (21.21) is not linear. The discourse seems to be primarily about the sequence of events described in sentences S1 and S2, whereas sentences S3 and S5 are related most directly to S2, and S4 is related most directly to S3. The coherence relationships between these sentences result in the discourse structure shown in Figure 21.3.

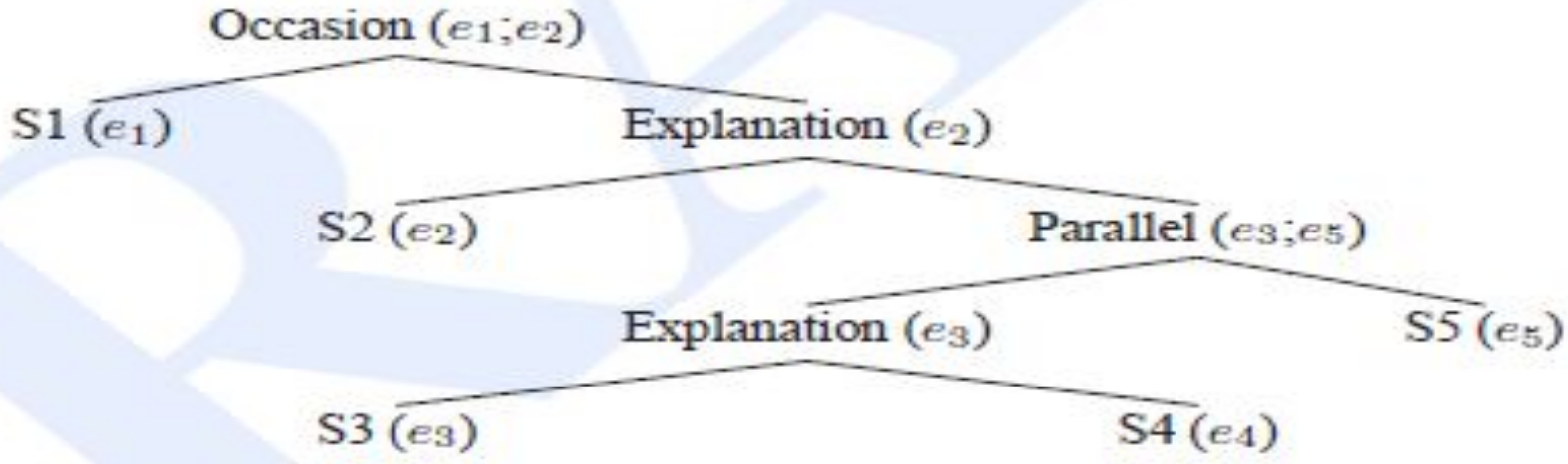


Figure 21.3 The discourse structure of passage (21.21).

Reference Resolution

- The extraction of the meaning or interpretation of the sentences of discourse is one of the most important tasks in natural language processing, and to do so, we first need to know what or who is the entity that we are talking about.
- Reference resolution means understanding the type of entity that is being talked about.
- By the term **reference**, we mean the linguistic expression that is used to denote an individual or an entity. For example, look at the below sentences.
- Rahul went to the farm.
- He cooked food.
- His farm was very big.
- In the above sentences, Rahul, He, and His references. So, we can simply define the reference resolution as the task of determination of the entities that are being referred to by the linguistic expressions.

Terminology Used in Reference Resolution

- **Referring expression:**
  The NLP expression that performs the reference is termed a referring expression.
- For example, the passage that we have talked about in the above section is an example of the referring expression.
- **Referent:**
  Referent is the entity we have referred to.
- For example, in the above passage, Rahul is the referent.
- **Co-refer:**
  As the name suggests, Co-refer is a term used for an entity if two or more expressions are referring to the same entity.
- For example, Rahul and He is used for the same entity, i.e., Rahul.
- **Antecedent:**
  The term that has been licensed to use another term is termed antecedent.
- For example, in the above passage, Rahul is the antecedent of the reference He.
- **Anaphora & Anaphoric:**
  The referring expression is termed anaphoric.
- **Anaphora** & **Anaphoric** can be said to be the term or reference used for an entity that has previously been introduced in the same sentence.

- Reference
  - Reference is a means to link a referring expression to another referring expression in the surrounding text
    - Eg: Suha bought a printer. It costs her Rs. 20,000.
    - Her refers to Suha and it refers to printer
    - Also called anaphoric reference – reference to something before
  - Five types of reference – Indefinite, definite, pronominal, demonstrative, ordinal
  - 1. Indefinite reference – introduces a new object to the discourse context- most commonly with a/an/some
    - Eg: *Some* printers make noise while printing
    - I met *this* girl earlier in a conference

- 2. Definite Reference
  - Refers to an object that already exists in the discourse context
    - Eg: I bought a printer today. *The* printer didn't work properly.
- 3. Pronominal Reference
  - References that use a pronoun to refer to some entity
    - Eg: I bought a printer today. On installation, *it* didn't work properly
    - Zuha forgot *her* pendrive in the lab
  - Usually, pronominal references refer to an object introduced in the previous one or two sentences, whereas definite noun phrases can refer to objects further back
- 4. Demonstrative reference
  - A word that directly indicates a person/thing or few people and few things. The demonstrative words are *that, those, this,* and *these*.

    Eg: This is my book, Give me that blue water bottle

Definite Reference:

1. Definite reference is used when you are talking about a particular, specific thing that both the speaker and the listener know about or can easily identify. It's like saying "the" something, indicating there's a specific instance of that thing being referred to.
2. **Example:** "I saw **the** cat sitting on **the** fence." In this sentence, "the cat" refers to a specific cat that both the speaker and the listener are aware of or can easily understand from the context. Similarly, "the fence" refers to a particular fence that is also known or identifiable in the context of the conversation.

Indefinite Reference:

1. Indefinite reference is used when you are talking about something in a more general or nonspecific way. It's like saying "a" or "an" something, indicating that you're referring to any instance of that thing, not a particular one.
2. **Example:** "I want to buy **a** book." In this sentence, "a book" refers to any book in general, not a specific one. The speaker is expressing a desire to purchase any book, without specifying which book they have in mind.

Demonstrative reference

- Imagine you're in a park with a friend, and you see two dogs playing. You want to refer to one of the dogs. You might say:
- "That dog is chasing the ball."
- In this sentence, "that" is a demonstrative reference that points to a specific dog in the park. It indicates that the dog you're referring to is not close to you but is still visible enough for both you and your friend to identify. The use of "that" helps distinguish the dog you're talking about from other objects or animals in the park.
- Let's break it down:
- "That": This word points to the specific dog you're referring to. It suggests that the dog is a bit distant from both you and your friend but still visible.
- "Dog": This is the noun being referred to, the object of the sentence.
- "Is chasing the ball": This part of the sentence describes what the dog is doing. It provides additional information about the action being performed by the dog.

Example – pronominal reference

- Original sentence: "John went to the store. John bought some groceries."
- In this original sentence, the name "John" is repeated, which can sound repetitive. To avoid this repetition, we can use pronouns for pronominal reference:
- Revised sentence with pronominal reference: "John went to the store. He bought some groceries."
- In this revised sentence, the pronoun "he" is used to refer back to the previously mentioned noun "John." By using the pronoun, we maintain clarity and coherence while avoiding unnecessary repetition.
- Pronominal reference is a fundamental aspect of language that allows speakers and writers to communicate efficiently and effectively by referring back to previously mentioned entities in a sentence or discourse.

- 5. Quantifier or Ordinal Reference
  - Uses an ordinal like first, one etc
  - Eg: I visited a computer store to buy a printer. I have seen many and now I have to select *one*.

- Inferables
  - Refers to entities that can be inferred from other entities.
  - Eg: I bought a printer today. On opening the package, I found the *paper tray* broken.

- Generic reference
  - Refers to a whole class instead of a specific entity.
  - Eg: I saw two laser printers in a shop. *They* were the fastest in the market.

- Ellipsis
  - Refers to the phenomenon where a part of the sentence is omitted or left unpronounced.
  - Eg: Do you like fish? Yes I do
    - Instead of Yes I do like fish, the elided verb phrase is understood from the previous sentence

- Reference Resolution
  - 1. Constraints and Preferences
    - Constraints that rule out certain referents can be checked
    - Constraints of number, gender, case
    - Semantic constraints can also be used to identify a preferred referent

    a) Person agreement

    Referent and referring expressions must agree in person

    Eg: Zuha and I bought a camera. We like capturing nature scenes

    (We = I and Zuha)

    Zuha and Prabha bought a camera. We like capturing nature scenes

    Resolving We into Zuha and Prabha incorrect

- Gender agreement
  - Zuha bought a printer. She is printing now (She = Zuha, not printer)
  - Zuha bought a printer. It is printing now (It = printer, not Zuha)
- Case agreement
  - The position where a pronoun is used constraints its form. Eg: In the object position we use him, her, them and in subject position we use he, she etc
- Selectional Restrictions
  - Selectional restrictions placed by verbs on their arguments can be used to resolve references
    - Eg: Zuha put an apple on the table. Suha is eating it.
      - It refers to apple and not table since eat can take only an edible object

- Recently introduced references
  - While resolving references, the entities introduced recently are considered more important than ones introduced further back
- Parallelism
  - Zuha went with Suha to the computer shop. Danish went with her to the computer institute.
    - Her refers to Suha and not Zuha, due to structural parallelism
- Repeated Mention
  - Refers to the idea that entities that are focused on in prior discourses are more likely to continue to be focused on subsequent discourses
    - Eg: Lucid was the first among the six women to join the astronaut program. A veteran of 5 space flights, logging 223 days in spaces, **she** hold the international record.....

      In 1998 **she** wrote in The Scientific American that **she** viewed the Mir mission.....

Pronominal Anaphora Resolution:

- Pronominal anaphora resolution specifically deals with resolving pronouns to their antecedents.
- Anaphora refers to the use of a pronoun to refer back to a previously mentioned noun or noun phrase (antecedent).

Example: "The cat sat on the mat. It droned happily."

- Here, "It" is a pronoun referring back to the noun "cat" mentioned earlier. Pronominal anaphora resolution involves identifying the antecedent ("cat") to correctly interpret the meaning of the pronoun ("it").

Coreference Resolution:

- Coreference resolution is a broader concept that encompasses the resolution of not only pronouns but also other types of referring expressions that refer to the same entity in a text. This includes noun phrases, proper names, and more.
- **Example:** "John met Mary at the park. He gave her a book. Mary thanked him."
- In this example, "John" and "He" refer to the same entity, as do "Mary" and "her," and "John" and "him." Coreference resolution involves identifying and linking these expressions to the appropriate antecedents to establish the relationships between them.

REFERENCE RESOLUTION

- (21.34) Victoria Chen, Chief Financial Officer of Megabucks Banking Corp since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old also became the Denver-based financial-services company's president. It has been ten years since she came to Megabucks from rival Lotsabucks.

In this passage, each of the underlined phrases is used by the speaker to denote one person named Victoria Chen. We refer to this use of linguistic expressions like *her* or *Victoria Chen* to denote an entity or individual as **reference**.

- Reference resolution is the task of determining what entities are referred to by which linguistic expressions.
- A natural language expression used to perform reference is called a **referring expression**, and the entity that is referred to is called the **referent**.

Reference Resolution

- **1. Resolution:** Resolution in natural language processing (NLP) refers to the process of resolving linguistic ambiguities or references within a text. This includes resolving pronouns, references to entities, or resolving syntactic ambiguities. It helps in understanding the intended meaning of a sentence or a discourse.
- **Example:** "John visited Mary at her house. He brought her a gift."
- In this example, the word "her" in the second sentence is ambiguous. It could refer to Mary or to another female entity previously mentioned or implied. Resolution involves determining which entity the pronoun "her" refers to. In this case, resolution would clarify that "her" refers to Mary, resulting in a coherent understanding of the sentence.

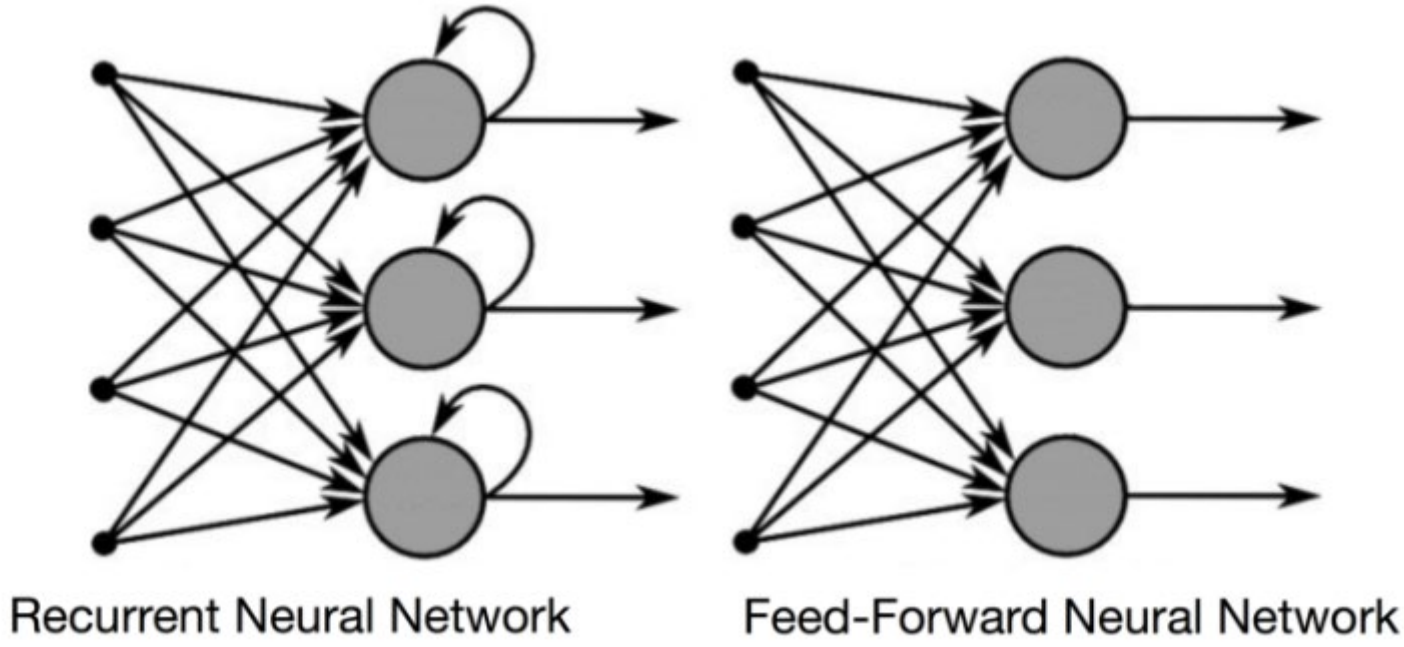
Unit-4 - Language Models

Issues in the Feed forward networks

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

- The solution to these issues is the RNN. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

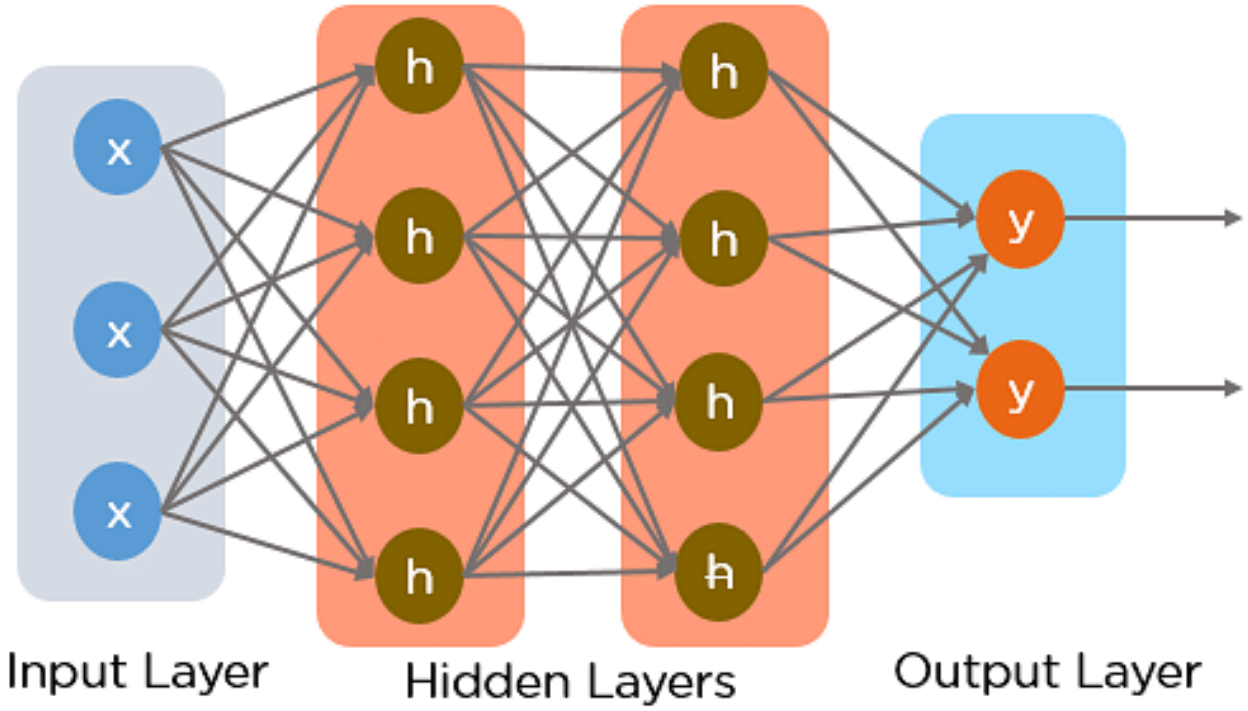
Difference between RNN and FFNN

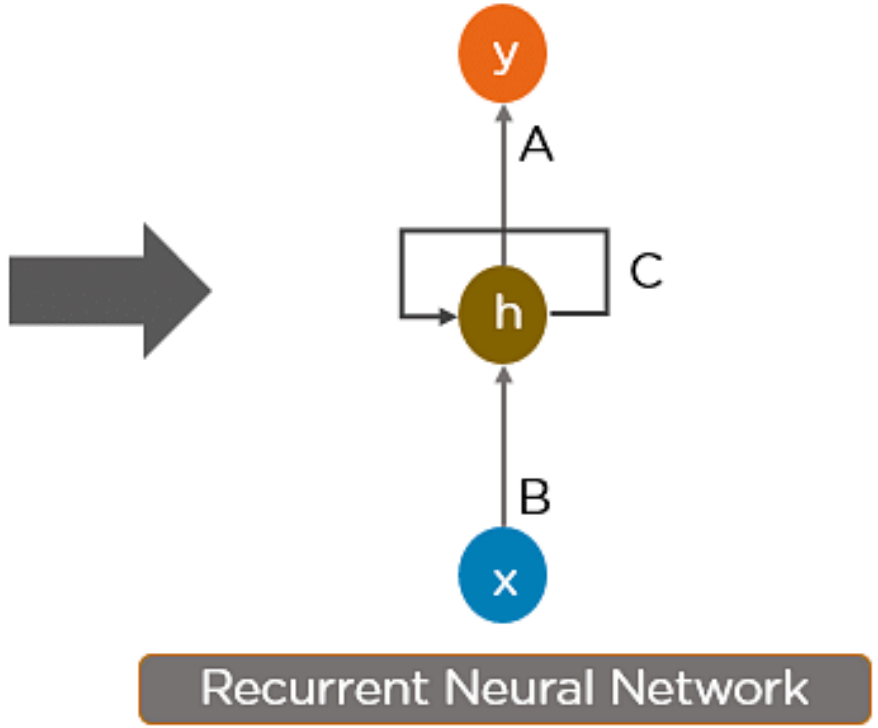


Recurrent Neural Networks (RNN)

- Recurrent neural networks (RNNs) are a class of neural network that are helpful in modeling sequence data.
- Derived from feedforward networks, RNNs exhibit similar behavior to how human brains function.
- Recurrent neural networks produce predictive results in sequential data that other algorithms can’t.
- A usual RNN has a short-term memory.

- Simple Recurrent Neural Network





- 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.
- The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.

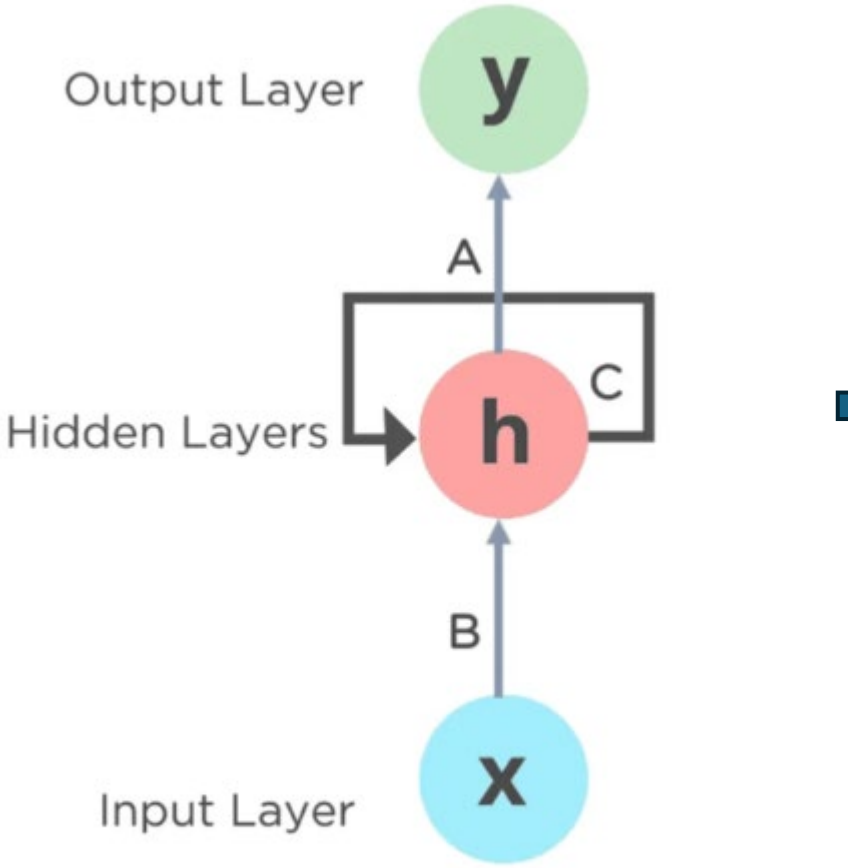


Fig: Fully connected Recurrent Neural Network

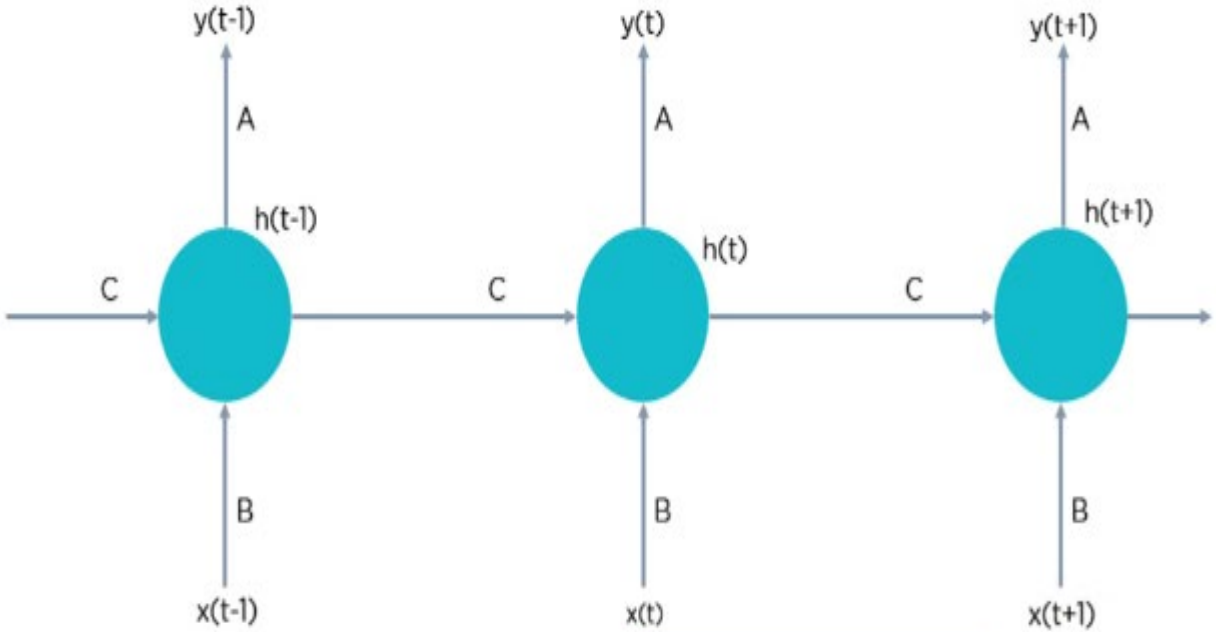


Fig: Fully connected Recurrent Neural Network

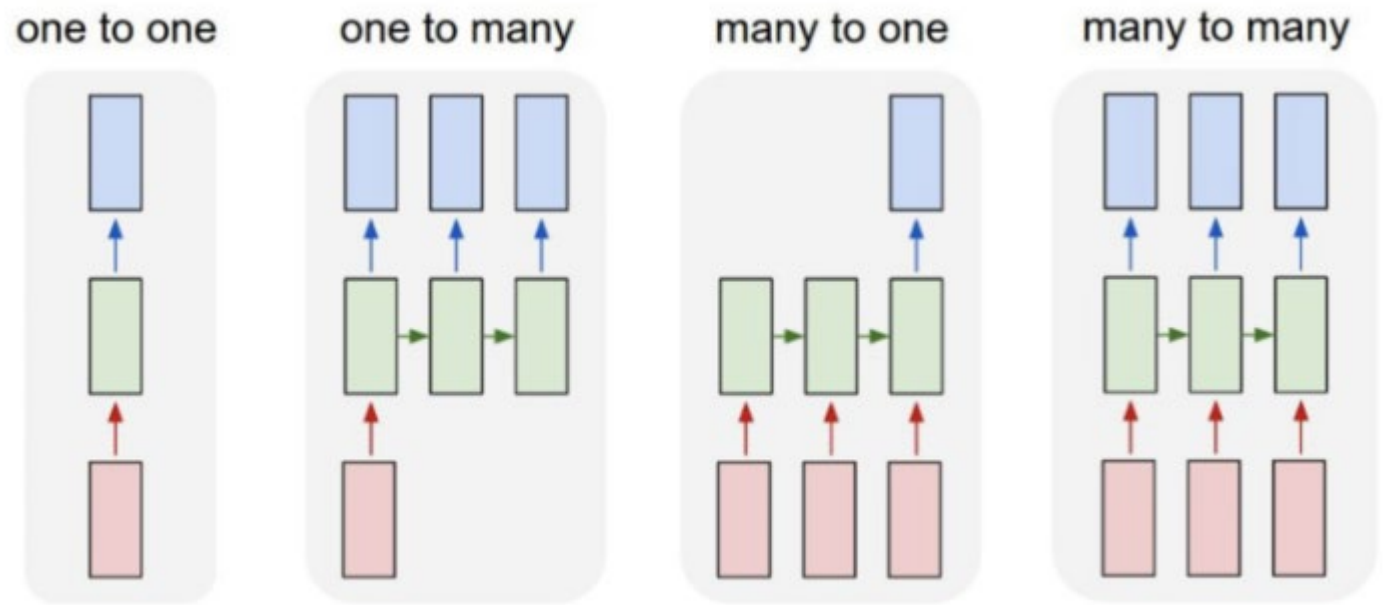
- Here, “x” is the input layer, “h” is the hidden layer, and “y” is the output layer.
- A, B, and C are the network parameters used to improve the output of the model.
- At any given time t, the current input is a combination of input at x(t) and x(t-1).
- The output at any given time is fetched back to the network to improve on the output.

How Does Recurrent Neural Networks Work?

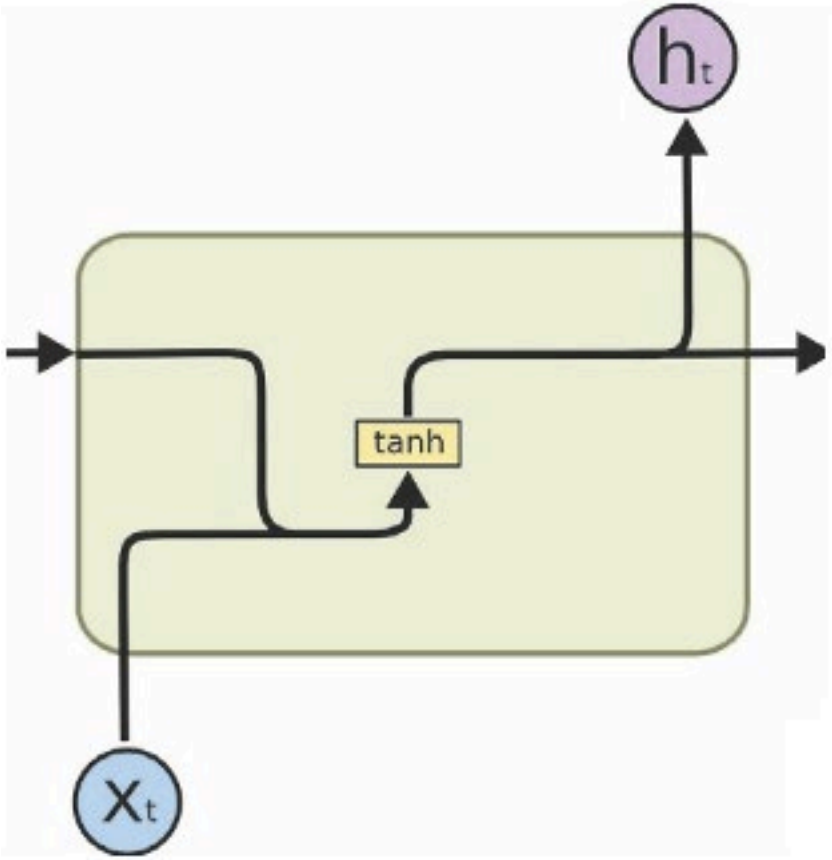
- The input layer ‘x’ takes in the input to the neural network and processes it and passes it onto the middle layer.
- The middle layer ‘h’ can consist of multiple hidden layers, each with its own activation functions and weights and biases.
- If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer,
- ie: the neural network does not have memory, then you can use a recurrent neural network.
- The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters.
- Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.

Types of Recurrent Neural Networks

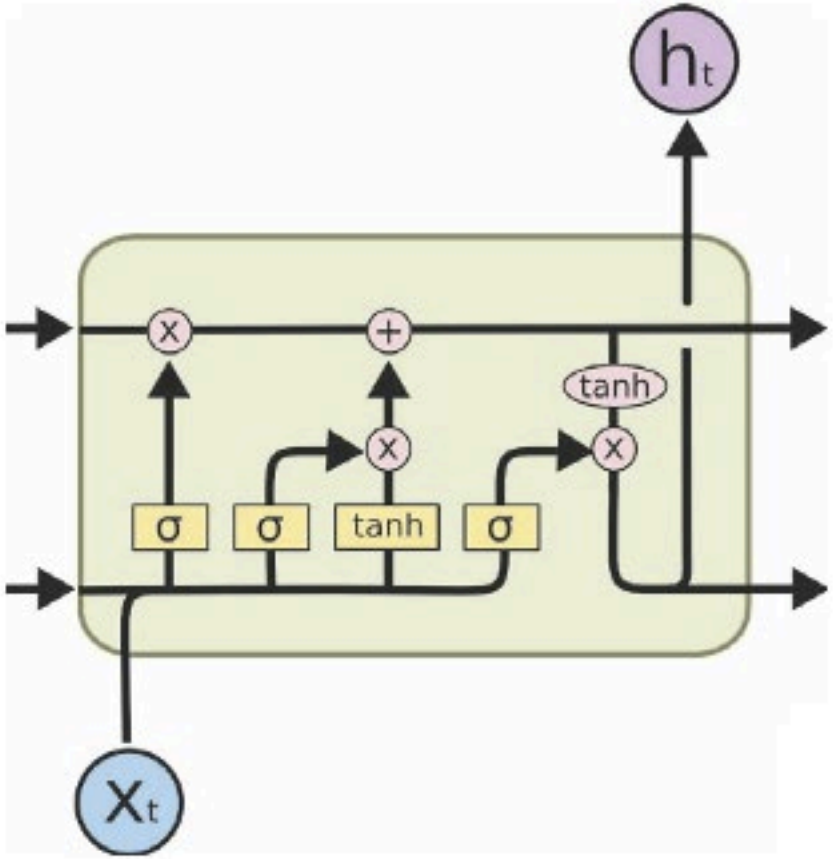
- One to One
- One to Many
- Many to One
- Many to Many



RNN vs LSTM



(a) RNN



(b) LSTM

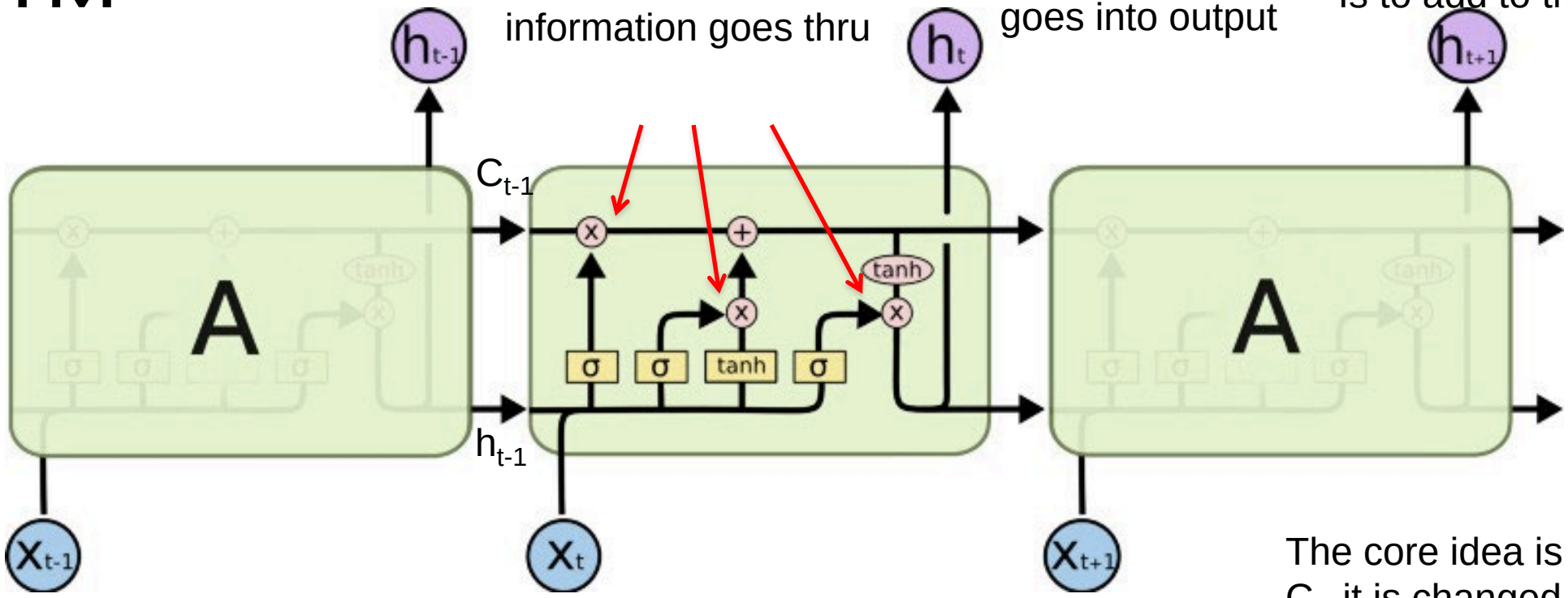
Long Short-Term Memory (LSTM)

LSTM

This sigmoid gate determines how much information goes thru

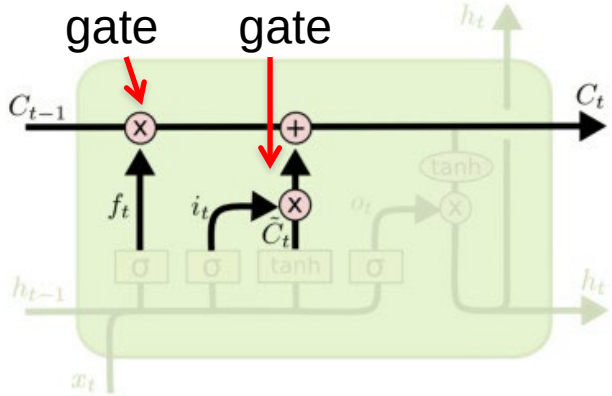
Output gate Controls what goes into output

This decides what info Is to add to the cell state



Forget gate

input gate



The core idea is this cell state C_t, it is changed slowly, with only minor linear interactions. It is very easy for information to flow along it unchanged.

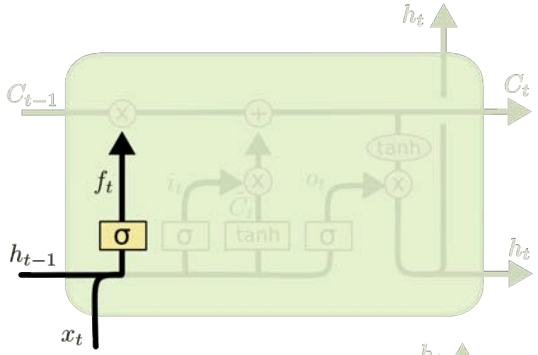
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Why sigmoid or tanh:

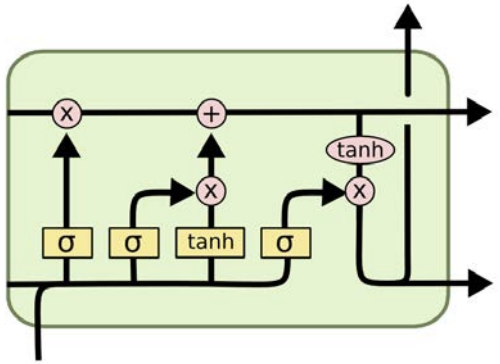
Sigmoid: 0,1 gating as switch.

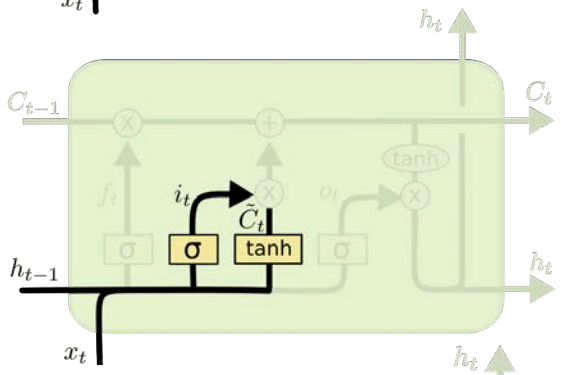
Vanishing gradient problem in LSTM is handled already.

ReLU replaces tanh ok?



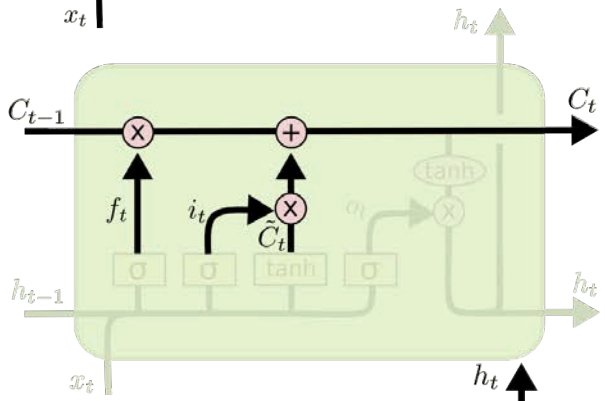
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$





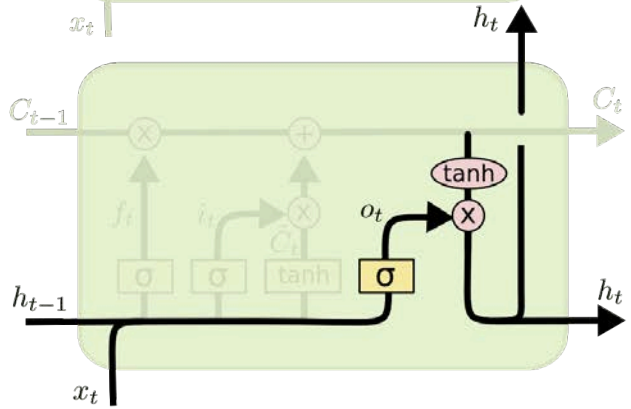
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t decides what component is to be updated.
C'_t provides change contents



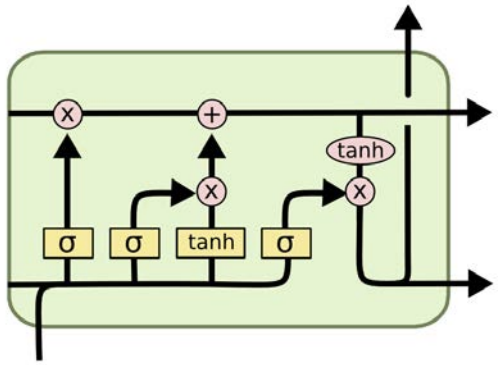
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Updating the cell state



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what part of the cell state to output



These 4 matrix computation should be done concurrently.

c^{t-1}

$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

Controls forget gate

Controls input gate

Updating information

Controls Output gate

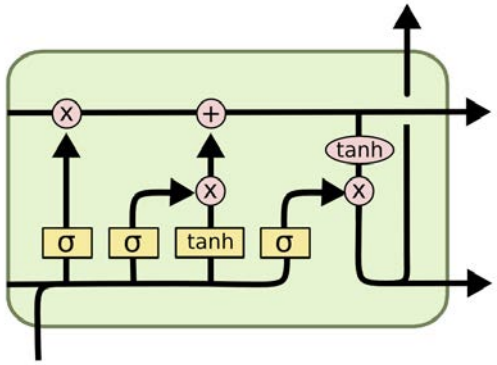
z^f z^i z z^o

h^{t-1} x^t

$$z^f = \sigma\left(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

Information flow of LSTM



z =tanh(W x^t h^{t-1} c^{t-1})

diagonal

c^{t-1}

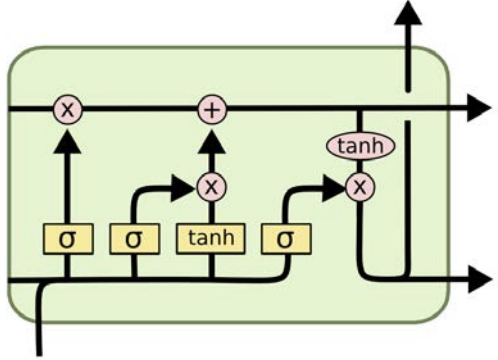
"peephole"

z^o z^f z^i obtained by the same way

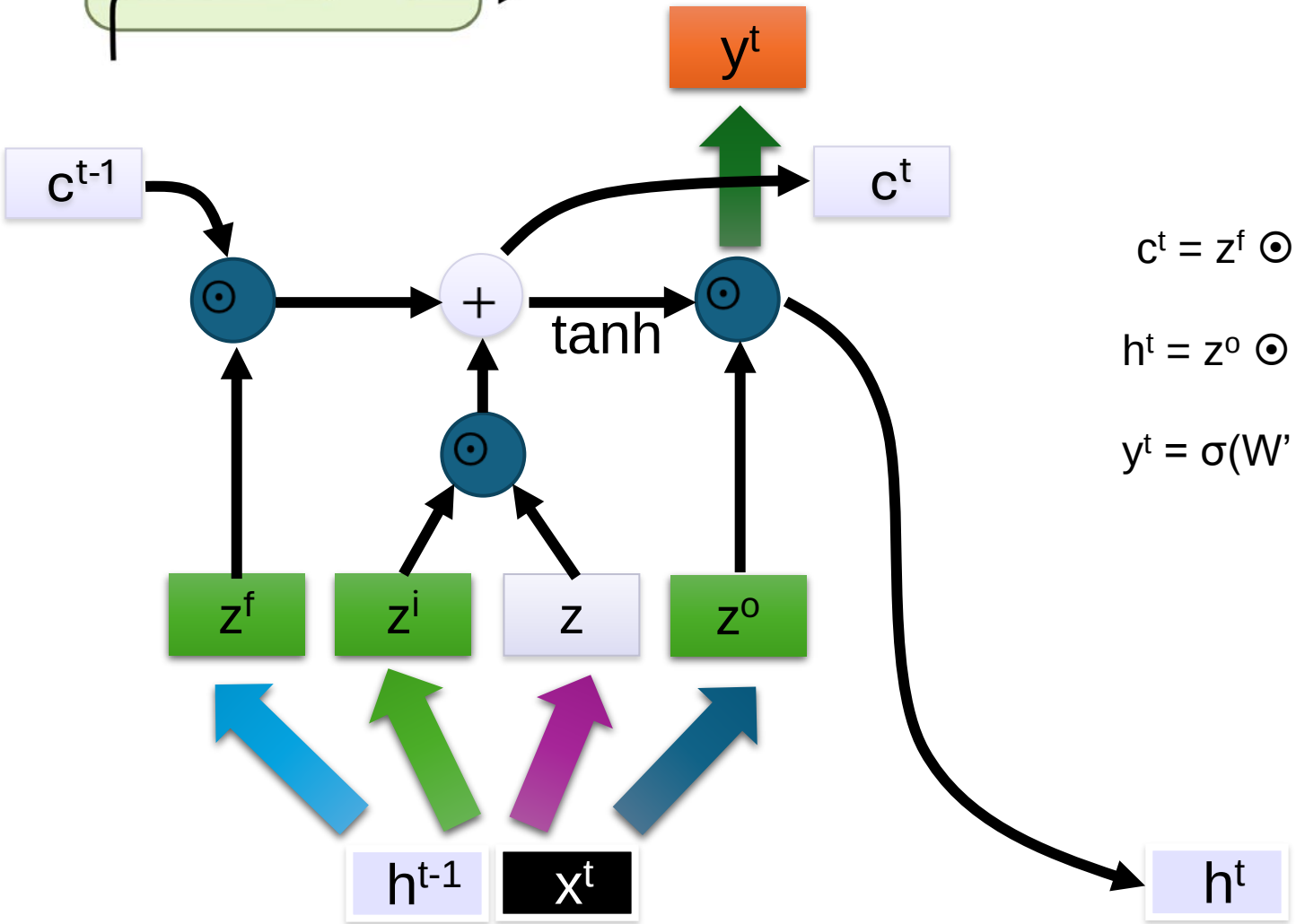
z^f z^i z z^o

h^{t-1} x^t

Information flow of LSTM



Element-wise multiply



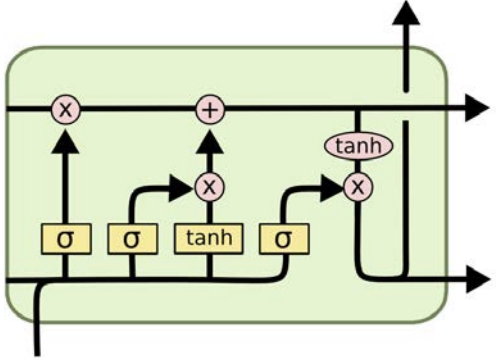
$c^t = z^f \odot c^{t-1} + z^i \odot z$

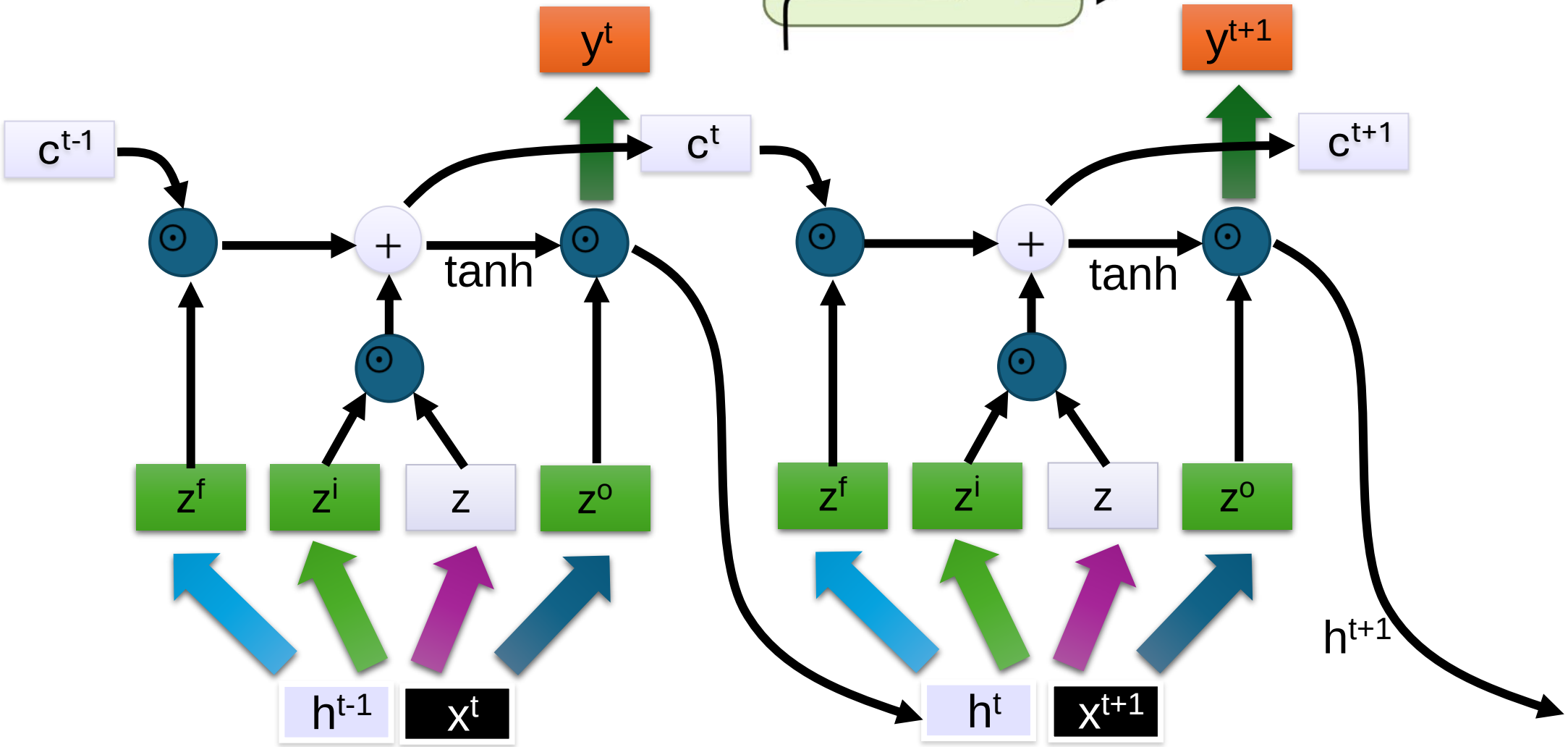
$h^t = z^o \odot \tanh(c^t)$

$y^t = \sigma(W' h^t)$

Information flow of LSTM

LSTM information flow





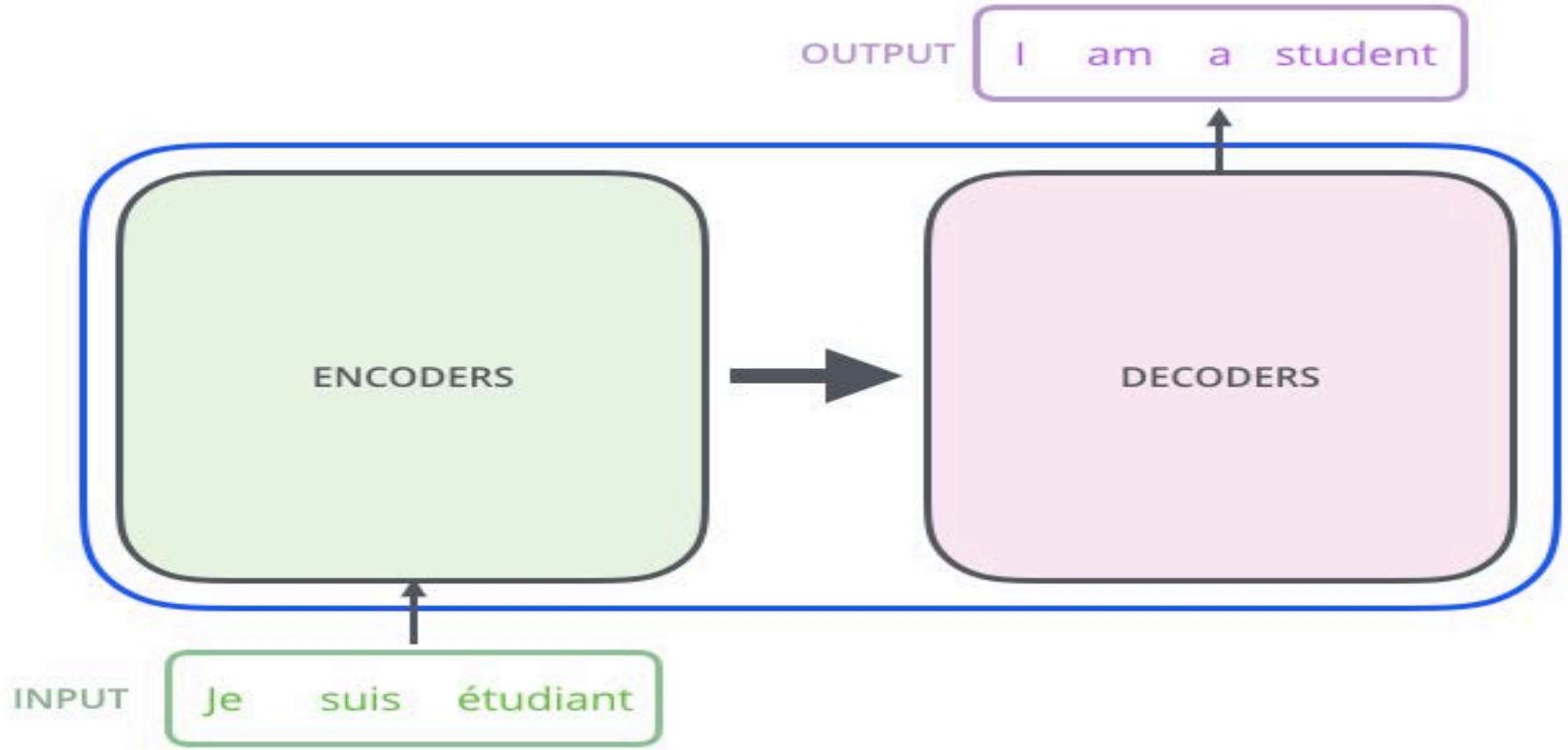
Information flow of LSTM

Transformer

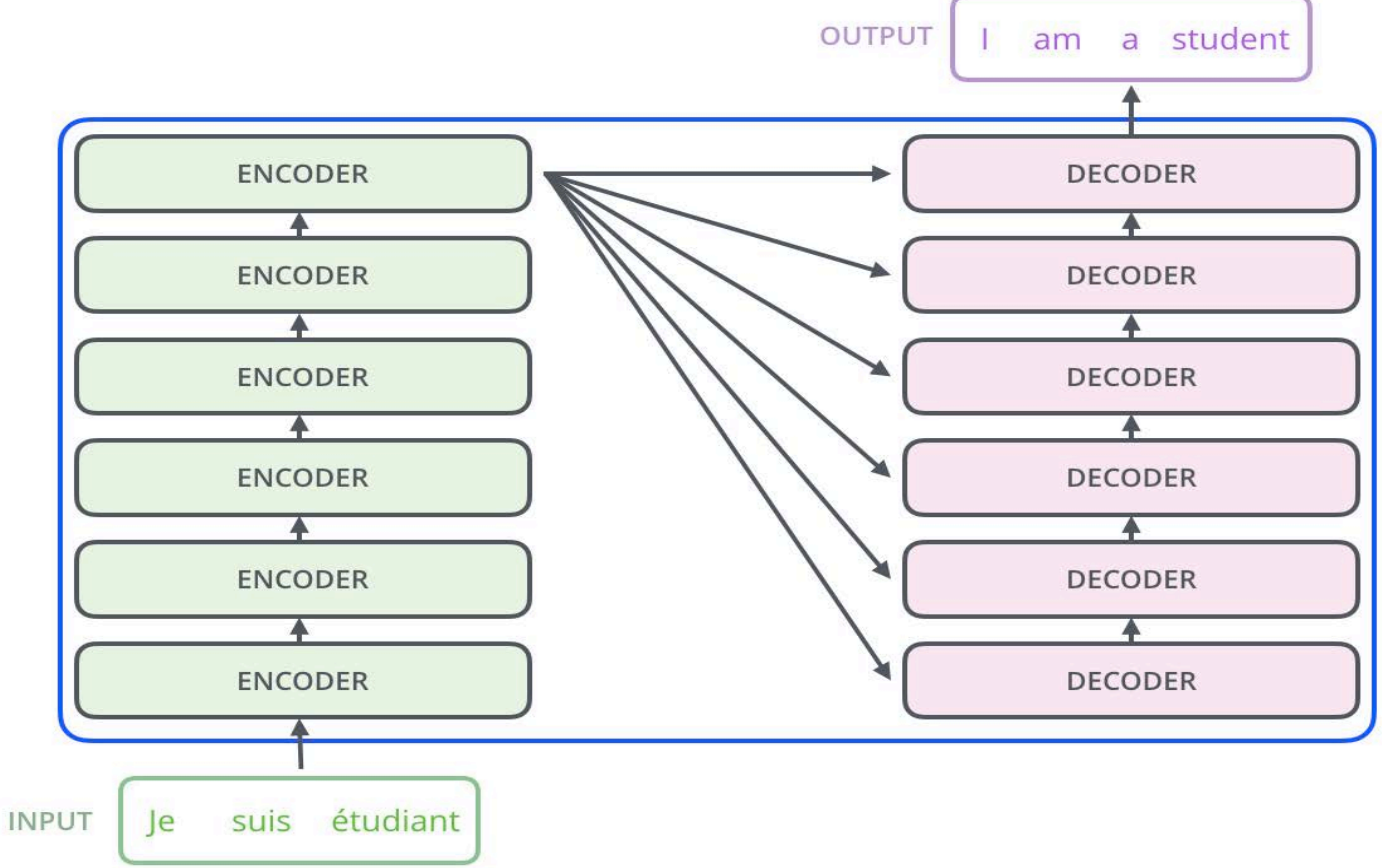
- **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformer outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

- The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

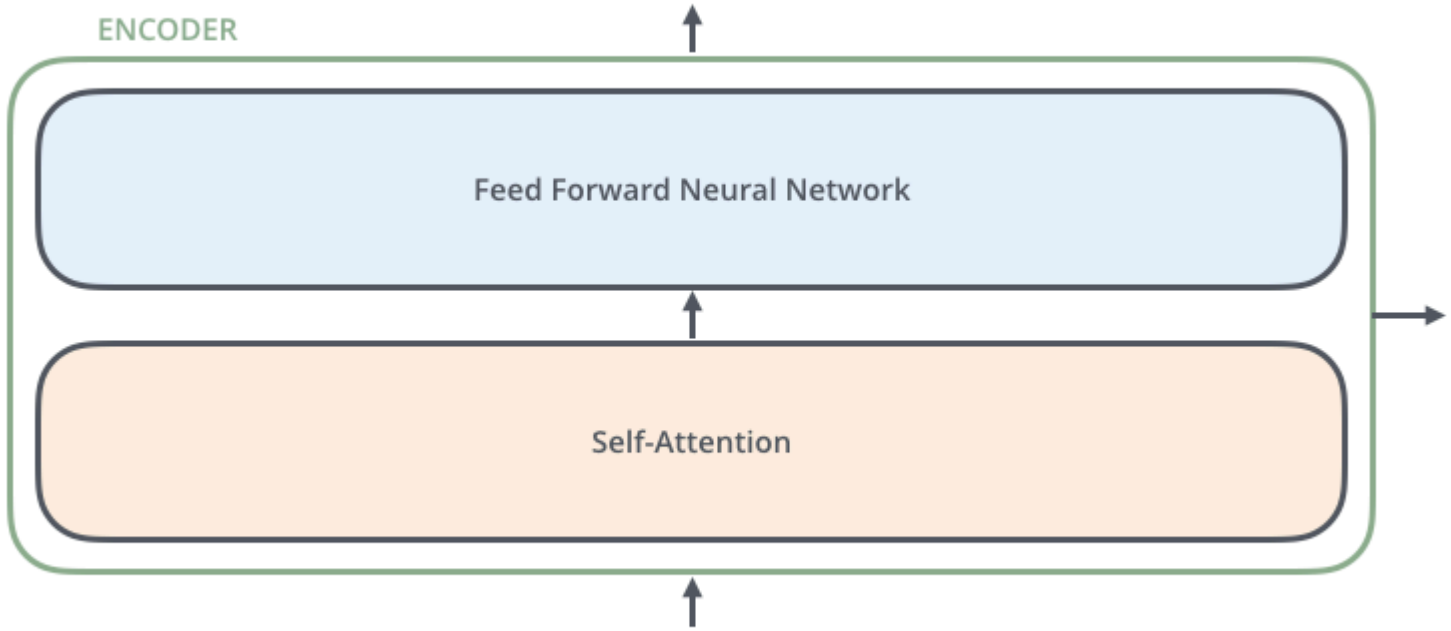
In a machine translation application, it would take a sentence in one language, and output its translation in another.



An encoding component, a decoding component, and connections between them.

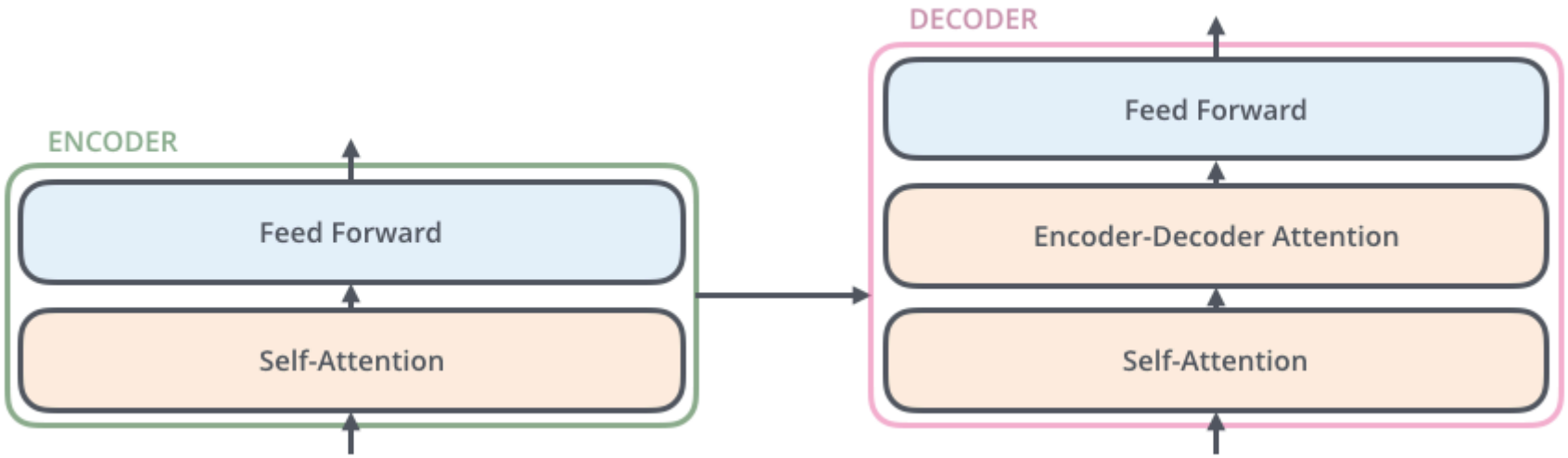


The encoding component is a stack of encoders (the paper stacks six of them on top of each other – there's nothing magical about the number six, one can definitely experiment with other arrangements). The decoding component is a stack of decoders of the same number.



The encoders are all identical in structure (yet they do not share weights). Each one is broken down into two sub-layers:

- The encoder’s inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word.
- The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position.
- The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence.



BERT Model

- **BERT, an acronym for Bidirectional Encoder Representations from Transformers**, stands as an open-source **machine learning framework** designed for the realm of **natural language processing (NLP)**. Originating in 2018, this framework was crafted by researchers from Google AI Language. The article aims to explore the **architecture, working and applications of BERT**.

- BERT (Bidirectional Encoder Representations from Transformers) leverages a transformer-based neural network to understand and generate human-like language. BERT employs an encoder-only architecture. In the original Transformer architecture, there are both encoder and decoder modules. The decision to use an encoder-only architecture in BERT suggests a primary emphasis on understanding input sequences rather than generating output sequences.

Bidirectional Approach of BERT

- Traditional language models process text sequentially, either from left to right or right to left.
- This method limits the model’s awareness to the immediate context preceding the target word.
- BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

Example: "The bank is situated on the _____ of the river."

- *In a unidirectional model, the understanding of the blank would heavily depend on the preceding words, and the model might struggle to discern whether "bank" refers to a financial institution or the side of the river.*
- *BERT, being bidirectional, simultaneously considers both the left ("The bank is situated on the") and right context ("of the river"), enabling a more nuanced understanding. It comprehends that the missing word is likely related to the geographical location of the bank, demonstrating the contextual richness that the bidirectional approach brings.*

Pre-training and Fine-tuning

- The BERT model undergoes a two-step process:

1. Pre-training on Large amounts of unlabeled text to learn contextual embeddings.
2. Fine-tuning on labeled data for specific NLP tasks.

- **Pre-Training on Large Data**
- BERT is pre-trained on large amount of unlabeled text data. The model learns contextual embeddings, which are the representations of words that take into account their surrounding context in a sentence.
- BERT engages in various unsupervised pre-training tasks. For instance, it might learn to predict missing words in a sentence (Masked Language Model or MLM task), understand the relationship between two sentences, or predict the next sentence in a pair.
- **Fine-Tuning on Labeled Data**
- After the pre-training phase, the BERT model, armed with its contextual embeddings, is then fine-tuned for specific natural language processing (NLP) tasks. This step tailors the model to more targeted applications by adapting its general language understanding to the nuances of the particular task.
- BERT is fine-tuned using labeled data specific to the downstream tasks of interest. These tasks could include sentiment analysis, question-answering, named entity recognition, or any other NLP application. The model's parameters are adjusted to optimize its performance for the particular requirements of the task at hand.

How BERT work?

- BERT is designed to generate a language model so, only the encoder mechanism is used. Sequence of tokens are fed to the Transformer encoder.
- These tokens are first embedded into vectors and then processed in the neural network.
- The output is a sequence of vectors, each corresponding to an input token, providing contextualized representations.
- When training language models, defining a prediction goal is a challenge. Many models predict the next word in a sequence, which is a directional approach and may limit context learning.
- BERT addresses this challenge with two innovative training strategies:

1.Masked Language Model (MLM)

2.Next Sentence Prediction (NSP)

- In BERT’s pre-training process, a portion of words in each input sequence is masked and the model is trained to predict the original values of these masked words based on the context provided by the surrounding words.

1. **Masking words:** Before BERT learns from sentences, it hides some words (about 15%) and replaces them with a special symbol, like [MASK].
2. **Guessing Hidden Words:** BERT’s job is to figure out what these hidden words are by looking at the words around them. It’s like a game of guessing where some words are missing, and BERT tries to fill in the blanks.
3. **How BERT learns:**
    1. BERT adds a special layer on top of its learning system to make these guesses. It then checks how close its guesses are to the actual hidden words.
    2. It does this by converting its guesses into probabilities, saying, “I think this word is X, and I’m this much sure about it.”
4. **Special Attention to Hidden Words**
    3. BERT’s main focus during training is on getting these hidden words right. It cares less about predicting the words that are not hidden.
    4. This is because the real challenge is figuring out the missing parts, and this strategy helps BERT become really good at understanding the meaning and context of words.

- In technical terms,

1. BERT adds a classification layer on top of the output from the encoder. This layer is crucial for predicting the masked words.
2. The output vectors from the classification layer are multiplied by the embedding matrix, transforming them into the vocabulary dimension. This step helps align the predicted representations with the vocabulary space.
3. The probability of each word in the vocabulary is calculated using the SoftMax activation function. This step generates a probability distribution over the entire vocabulary for each masked position.
4. The loss function used during training considers only the prediction of the masked values. The model is penalized for the deviation between its predictions and the actual values of the masked words.
5. The model converges slower than directional models. This is because, during training, BERT is only concerned with predicting the masked values, ignoring the prediction of the non-masked words. The increased context awareness achieved through this strategy compensates for the slower convergence.

2. Next Sentence Prediction (NSP)

- BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

1. In the training process, BERT learns to understand the relationship between pairs of sentences, predicting if the second sentence follows the first in the original document.
2. 50% of the input pairs have the second sentence as the subsequent sentence in the original document, and the other 50% have a randomly chosen sentence.
3. To help the model distinguish between connected and disconnected sentence pairs. The input is processed before entering the model:
    3. A [CLS] token is inserted at the beginning of the first sentence, and a [SEP] token is added at the end of each sentence.
    4. A sentence embedding indicating Sentence A or Sentence B is added to each token.
    5. A positional embedding indicates the position of each token in the sequence.
4. BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

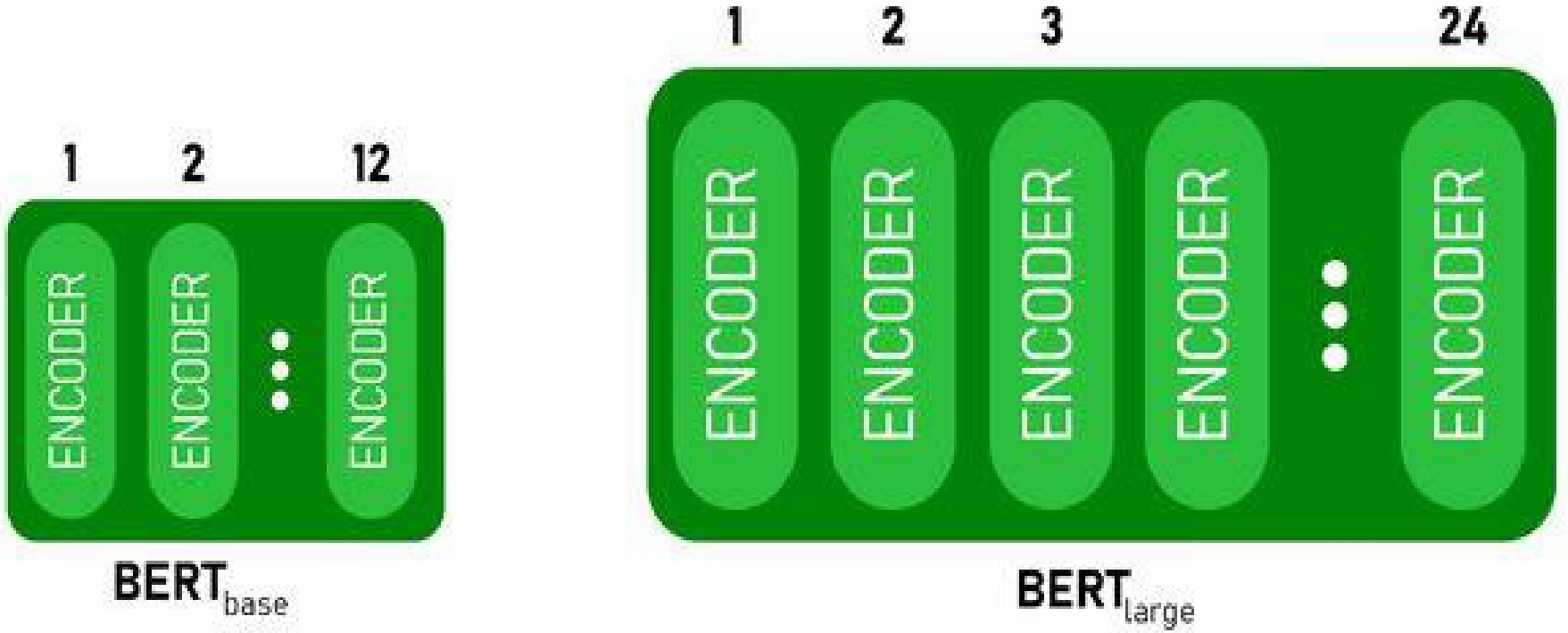
- During the training of BERT model, the Masked LM and Next Sentence Prediction are trained together.
- The model aims to minimize the combined loss function of the Masked LM and Next Sentence Prediction, leading to a robust language model with enhanced capabilities in understanding context within sentences and relationships between sentences.

BERT Architectures

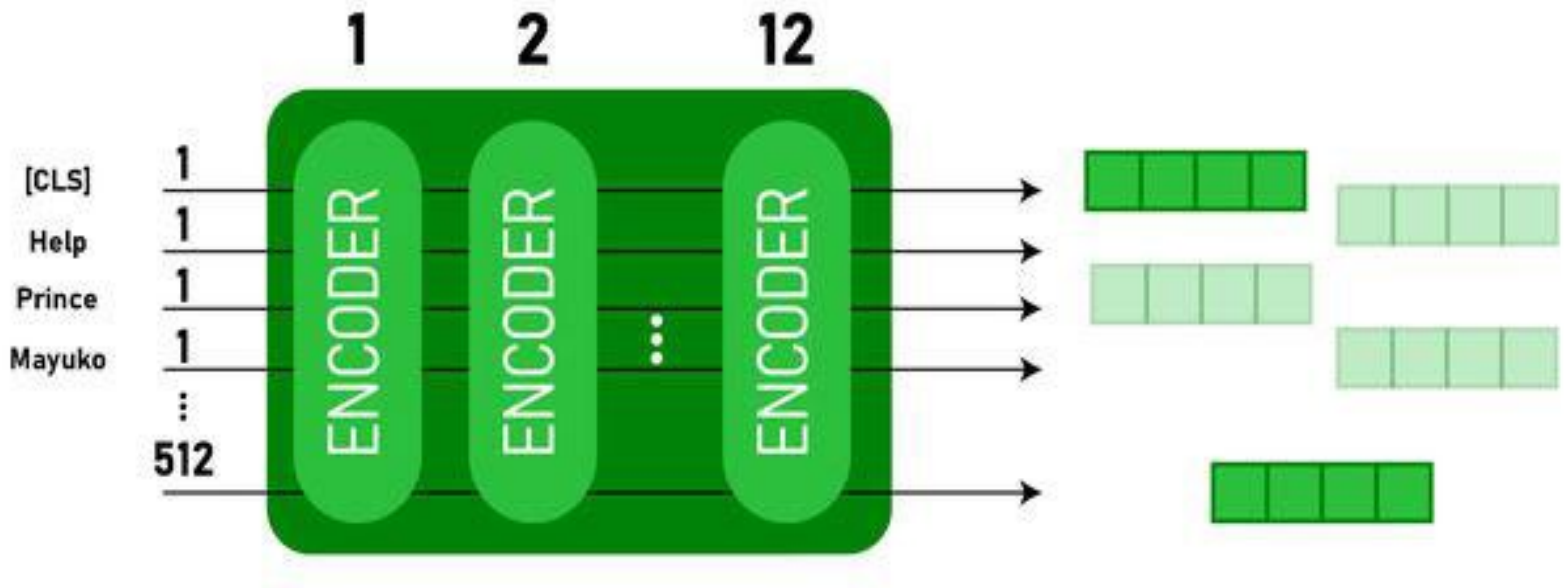
- The architecture of BERT is a multilayer bidirectional transformer encoder which is quite similar to the transformer model. A transformer architecture is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side.

1. $BERT_{BASE}$ has 12 layers in the Encoder stack while $BERT_{LARGE}$ has 24 layers in the Encoder stack. These are more than the Transformer architecture described in the original paper (6 encoder layers).
2. BERT architectures (BASE and LARGE) also have larger feedforward networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the Transformer architecture suggested in the original paper. It contains 512 hidden units and 8 attention heads.
3. $BERT_{BASE}$ contains 110M parameters while $BERT_{LARGE}$ has 340M parameters.

BERT BASE and BERT LARGE architecture.



- This model takes the **CLS** token as input first, then it is followed by a sequence of words as input. Here CLS is a classification token.
- It then passes the input to the above layers. Each layer applies self-attention and passes the result through a feedforward network after then it hands off to the next encoder.
- The model outputs a vector of hidden size (*768* for BERT BASE).
- If we want to output a classifier from this model we can take the output corresponding to the CLS token.



Now, this trained vector can be used to perform a number of tasks such as classification, translation, etc. For Example, the paper achieves great results just by using a single layer Neural Network on the BERT model in the classification task.

RoBERTa

- Despite the excellent performance of BERT, researchers still continued experimenting with its configuration in hopes of achieving even better metrics.
- Fortunately, they succeeded with it and presented a new model called **RoBERTa** — Robustly Optimised BERT Approach.
- RoBERTa (short for “Robustly Optimized BERT Approach”) is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model, which was developed by researchers at Facebook AI.
- Like BERT, RoBERTa is a transformer-based language model that uses self-attention to process input sequences and generate contextualized representations of words in a sentence.

- One key difference between RoBERTa and BERT is that RoBERTa was trained on a much larger dataset and using a more effective training procedure.
- In particular, RoBERTa was trained on a dataset of 160GB of text, which is more than 10 times larger than the dataset used to train BERT.
- Additionally, RoBERTa uses a dynamic masking technique during training that helps the model learn more robust and generalizable representations of words.
- RoBERTa has been shown to outperform BERT and other state-of-the-art models on a variety of natural language processing tasks, including language translation, text classification, and question answering.
- It has also been used as a base model for many other successful NLP models and has become a popular choice for research and industry applications.
- Overall, RoBERTa is a powerful and effective language model that has made significant contributions to the field of NLP and has helped to drive progress in a wide range of applications.

- RoBERTa stands for Robustly Optimized BERT Pre-training Approach. It was presented by researchers at Facebook and Washington University.
- The goal of this paper was to optimize the training of BERT architecture in order to take lesser time during pre-training.

Modifications to BERT:

- RoBERTa has almost similar architecture as compare to BERT, but in order to improve the results on BERT architecture, the authors made some simple design changes in its architecture and training procedure. These changes are:
- **Removing the Next Sentence Prediction (NSP) objective**: In the next sentence prediction, the model is trained to predict whether the observed document segments come from the same or distinct documents via an auxiliary Next Sentence Prediction (NSP) loss. The authors experimented with removing/adding of NSP loss to different versions and concluded that removing the NSP loss matches or slightly improves downstream task performance

- **Training with bigger batch sizes & longer sequences:** Originally BERT is trained for 1M steps with a batch size of 256 sequences. In this paper, the authors trained the model with 125 steps of 2K sequences and 31K steps with 8k sequences of batch size. This has two advantages, the large batches improves perplexity on masked language modelling objective and as well as end-task accuracy. Large batches are also easier to parallelize via distributed parallel training.
- **Dynamically changing the masking pattern**: In BERT architecture, the masking is performed once during data preprocessing, resulting in a single static mask. To avoid using the single static mask, training data is duplicated and masked 10 times, each time with a different mask strategy over 40 epochs thus having 4 epochs with the same mask. This strategy is compared with dynamic masking in which different masking is generated every time we pass data into the model.

Datasets Used:

- The following are the datasets used to train ROBERTa model:
- **BOOK CORPUS and English Wikipedia dataset:** This data also used for training BERT architecture, this data contains 16GB of text.
- **CC-NEWS**. This data contains 63 million English news articles crawled between September 2016 and February 2019. The size of this dataset is 76 GB after filtering.
- **OPENWEBTEXT:** This dataset contains web content extracted from the URLs shared on Reddit with at least 3 upvotes. The size of this dataset is 38 GB.
- **STORIES:** This dataset contains a subset of Common Crawl data filtered to match the story-like style of Winograd NLP task. This dataset contains 31 GB of text.

Fine-tuning For Downstream Tasks

- Adapting BERT for downstream tasks entails utilizing the pre-trained BERT model and customizing it for a particular task by adding a layer on top and training it on the target task.
- This technique allows the model to learn dependent on the task details from the data used for training while drawing on the knowledge of broad language expression of the pre-trained BERT model. Use the hugging face transformers package in Python to fine-tune BERT.
- Describe your training data, incorporating input text and labels. Fine-tuning the pre-trained BERT model for downstream tasks according to your data using the fit() function from the BertForSequenceClassification class.

How BERT Undergoes Fine-Tuning?

- Fine-tuning BERT adapts a pre-trained model with training data from the desired job to a specific downstream task by training a new layer.
- This process empowers the model to gain task-specific knowledge and enhance its performance on the target task.

Primary steps in the fine-tuning process for BERT

1: Utilize the hugging face transformers library to load the pre-trained BERT model and tokenizer.

```
import torch

# Choose the appropriate device based on availability (CUDA or CPU)
gpu_available = torch.cuda.is_available()
device = torch.device("cuda" if gpu_available else "cpu")

# Utilize a different tokenizer
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')

# Load the model using a custom function
from transformers import AutoModelForSequenceClassification
model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased')

model.to(device)
```

2: Specify the training data for the specific target task,encompassing the input text and their corresponding labels

```
# Specify the input text and the corresponding labels
input_text = "This is a sample input text"
labels = [1]
```

3: Utilize the BERT tokenizer to tokenize the input text

```
# Tokenize the input text
input_ids =torch.tensor(tokenizer.encode(input_text)).unsqueeze(0)
```

4: Put the model in training mode.

```
# Set the model to training mode
model.train()
```

- **Step 5:** For obtaining fine-tuning of the pre-trained BERT model, we use the method of Bert For Sequence Classification class. it includes training a new layer of pre-trained BERT model with the target task's training data.

```
# Set up your dataset, batch size, and other training hyperparameters
dataset_train = ...
lot_size = 32
num_epochs = 3
learning_rate = 2e-5
# Create the data loader for the training set
train_dataloader = torch.
utils.data.
DataLoader(dataset_train,
batch_size=lot_size)
model.fit(train_dataloader,num_epochs=num_epochs,
learning_rate=learning_rate)
```

- **Step 6:** Investigate the fine-tuned BERT model's illustration on the specific target task.

```
# Switch the model to evaluation mode
model.eval()
# Calculate the logits (unnormalized probabilities) for the input text
with torch.no_grad():
    logits = model(input_ids)
# Use the logits to generate predictions for the input text
predictions = logits.argmax(dim=-1)
accuracy = ...
```

- These represent the primary steps involved in fine-tuning BERT for a downstream task. You can utilize this as a foundation and customize it according to your specific use case.
- Fine-tuning BERT enables the model to acquire task-specific information, enhancing its performance on the target task. It proves particularly valuable when the target task involves a relatively small dataset, as fine-tuning w ith the small dataset allows the model to learn task-specific information that might not be attainable from the pre-trained BERT model alone.

Which Layers Undergo Modifications During Fine-tuning?

- During fine-tuning, solely the weights of the supplementary layer appended to the pre-trained BERT model undergo updates. The weights of the pre-trained BERT model remain fixed. Thus only the added layer experiences modifications throughout the fine-tuning process.
- Typically, the attached layer functions as a classification layer proceeds the pre-trained BERT model results, and generates logits for each class in the end task. The target task's training data trains the added layer, enabling it to acquire task-specific information and improve the model's performance on the target task.
- To sum up, during fine-tuning, the added layer above the pre-trained BERT model undergoes modifications. The pre-trained BERT model maintains fixed weights. Thus, only the added layer is subject to updates during the training process.

Downstream Tasks

- Downstream tasks include a variety of natural language processing (NLP) operations that use pre-trained language reconstruction models such as BERT.
- Text Classification
- Text classification involves the assignment of a text to predefined categories or labels. For instance, one can train a text classification model to categorize movie reviews as positive or negative.
- Use the BertForSequenceClassification library to alter BERT for text classification. This class uses input data, such as words or paragraphs, to generate logits for every class.

UNIT 5
NLP Applications

What is Chatbot?

- A chatbot (conversational interface, AI agent) is a computer program that can understand human language and converse with a user via a website or a messaging app.
- Chatbots can handle various tasks online — from answering simple questions and scheduling calls to gathering customer feedback. Brands use bots to automate their business processes, speed up customer service, and lower support costs.
- Not all chatbots are equipped with artificial intelligence (AI), but modern chatbots increasingly use conversational AI techniques such as natural language processing (NLP) to understand user questions and automate responses to them.

The Evolution of Chatbots

- The evolution of chatbots represents a charming journey via the annals of AI and human-computer interplay.
- Over the years, chatbots have undergone a remarkable transition, evolving from its basic text-based programs to sophisticated digital assistants with natural language and context recognition.
- This history reflects the convergence of scientific discoveries, technology advancements, and practical worldwide programs, and it echoes the unrelenting quest to replicate human-like conversational abilities in machines

1. **Eliza**
2. **Parry**
3. **The Jabberwacky**
4. **A. L. I. C. E**
5. **SmarterChild**
6. **Google Assistance**
8. **ChatGPT**

Types of Chatbots

Limitations of Chatbots

- **Lack of Understanding:** Chatbots may also struggle to recognize complicated queries or nuances in language, which leads to misunderstandings.
- **Limited Scope:** They are powerful as their programmed abilties, restricting their ability to handle unforeseen data.
- **Impersonal Interactions:** Some customers may also decide on human interplay over interacting with a system, leading to dissatisfaction.
- **Technical Issues:** Chatbots are prone to technical system faults and errors, that can disrupt consumer reports.
- **Dependency:** Overreliance on chatbots may result in reduced human interaction and lack of interpersonal connections.
- **Initial Investment:** Developing and imposing chatbots requires preliminary investment in phrases of time, resources, and technical kmowledge.
- **Maintenance:** Chatbots require ongoing protection and updates to stay powerful and relevant through the years.

What is Information Extraction?

- Text data contains a plethora of information, yet not all of it may be relevant to your needs. Some may seek to extract specific relationships between entities, utilizing information extraction NLP techniques. Our intentions vary based on individual requirements and objectives.
- Imagine having to go through all the legal documents to find legal precedence to validate your current case. Or having to go through all the research papers to find relevant information to cure a disease. There are many more examples like resume harvesting, media analysis, email scanning, etc.
- But just imagine having to manually go through all of the textual data and extracting the most relevant information. Clearly, it is an uphill battle and you might even end up skipping some important information.

- For anyone trying to analyze textual data, the difficult task is not of finding the right documents, but of finding the right information from these documents. Understanding the relationship between entities, understanding how the events have unfolded, or just simply finding hidden gems of information, is clearly what anyone is looking for when they go through a piece of text.
- Therefore, coming up with an automated way of extracting the information from textual data and presenting it in a structured manner will help us reap a lot of benefits and tremendously reduce the amount of time we have to spend time skimming through text documents. This is precisely what information extraction strives to achieve.

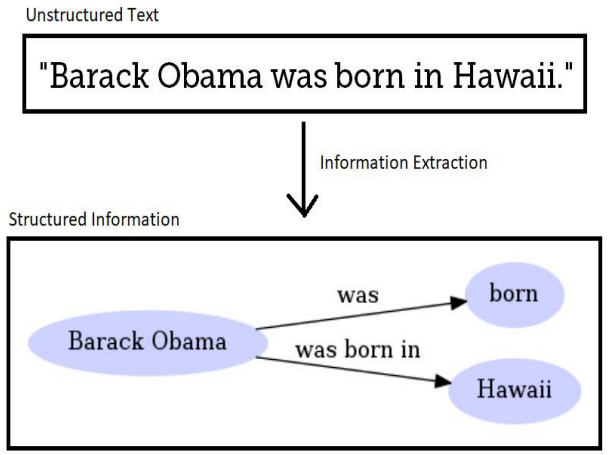
- The task of Information Extraction (IE) involves extracting meaningful information from unstructured text data and presenting it in a structured format.
- Using information extraction nlp, we can retrieve pre-defined information such as the name of a person, location of an organization, or identify a relation between entities, and save this information in a structured format such as a database.
- Let me show you another example I've taken from a cricket news article:
- Indian captain Virat Kohli was dismissed cheaply for just 2 in Wellington on Friday by debutant Kyle Jamieson extending a rare lull in the batsman's stellar career. Throughout the ongoing New Zealand tour, Kohli has managed to score just a single fifty across 8 innings in all 3 international formats.

- We can extract the following information from the text:
- Country – India, Captain – Virat Kohli
- Batsman – Virat Kohli, Runs – 2
- Bowler – Kyle Jamieson
- Match venue – Wellington
- Match series – New Zealand
- Series highlight – single fifty, 8 innings, 3 formats
- This enables us to reap the benefits of powerful query tools like SQL for further analysis. Creating such structured data using information extraction will not only help us in analyzing the documents better but also help us in understanding the hidden relationships in the text.

How Does Information Extraction Work?

- Given the capricious nature of text data that changes depending on the author or the context, Information Extraction seems like a daunting task. But it doesn't have to be that way!
- We all know that sentences are made up of words belonging to different **Parts of Speech (POS)**. There are eight different **POS** in the English language: **noun, pronoun, verb, adjective, adverb, preposition, conjunction,** and **intersection**.
- The POS determines how a specific word functions in meaning in a given sentence. For example, take the word "right". In the sentence, "The boy was awarded chocolate for giving the right answer", "right" is used as an adjective. Whereas, in the sentence, "You have the right to say whatever you want", "right" is treated as a noun.
- This goes to show that the POS tag of a word carries a lot of significance when it comes to understanding the meaning of a sentence. And we can leverage it to extract meaningful information from our text.

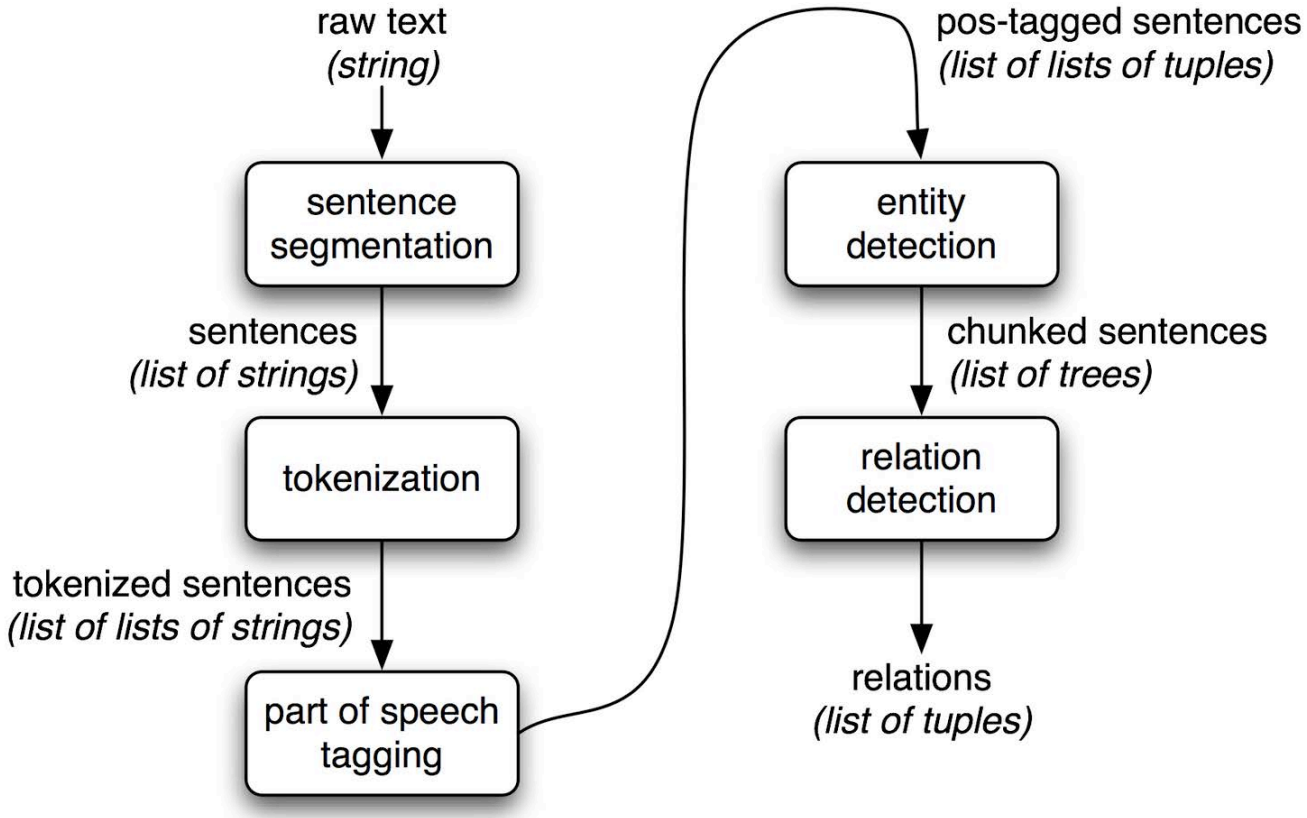
Information extraction (IE)



The task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents and other electronically represented sources is known as **information extraction** (IE).
When there are a significant number of customers, manually assessing Customer Feedback, for example, can be tedious, error-prone, and time-consuming. There's a good chance we'll overlook a dissatisfied consumer. Fortunately, sentiment analysis can aid in the improvement of customer support interactions' speed and efficacy. By doing sentiment analysis on all the incoming tickets and prioritizing them above the others, one can quickly identify the most dissatisfied customers or the most important issues. One might also allocate tickets to the appropriate individual or team to handle them. As a result, Consumer satisfaction will improve dramatically.

General Pipeline of the Information Extraction Process

- *The following steps are often involved in extracting structured information from unstructured texts*:
  1. Initial processing.
  2. Proper names identification.
  3. Parsing.
  4. Extraction of events and relations.
  5. Anaphora resolution.
  6. Output result generation.



1. Initial processing

- The first step is to break down a text into fragments such as zones, phrases, segments, and tokens. This function can be performed by tokenizers, text zoners, segmenters, and splitters, among other components. In the initial processing stage, part-of-speech tagging, and phrasal unit identification (noun or verb phrases) are usually the next tasks.

2. Proper names identification

One of the most important stages in the information extraction chain is the identification of various classes of proper names, such as names of people or organizations, dates, monetary amounts, places, addresses, and so on. They may be found in practically any sort of text and are widely used in the extraction process. Regular expressions, which are a collection of patterns, are used to recognize these names.

3. Parsing

The syntactic analysis of the sentences in the texts is done at this step. After recognizing the fundamental entities in the previous stage, the sentences are processed to find the noun groups that surround some of those entities and verb groups. At the pattern matching step, the noun and verb groupings are utilized as sections to begin working on.

4. Extraction of events and relations

This stage establishes relations between the extracted ideas. This is accomplished by developing and implementing extraction rules that describe various patterns. The text is compared to certain patterns, and if a match is discovered, the text element is labeled and retrieved later.

5. Coreference or Anaphora resolution

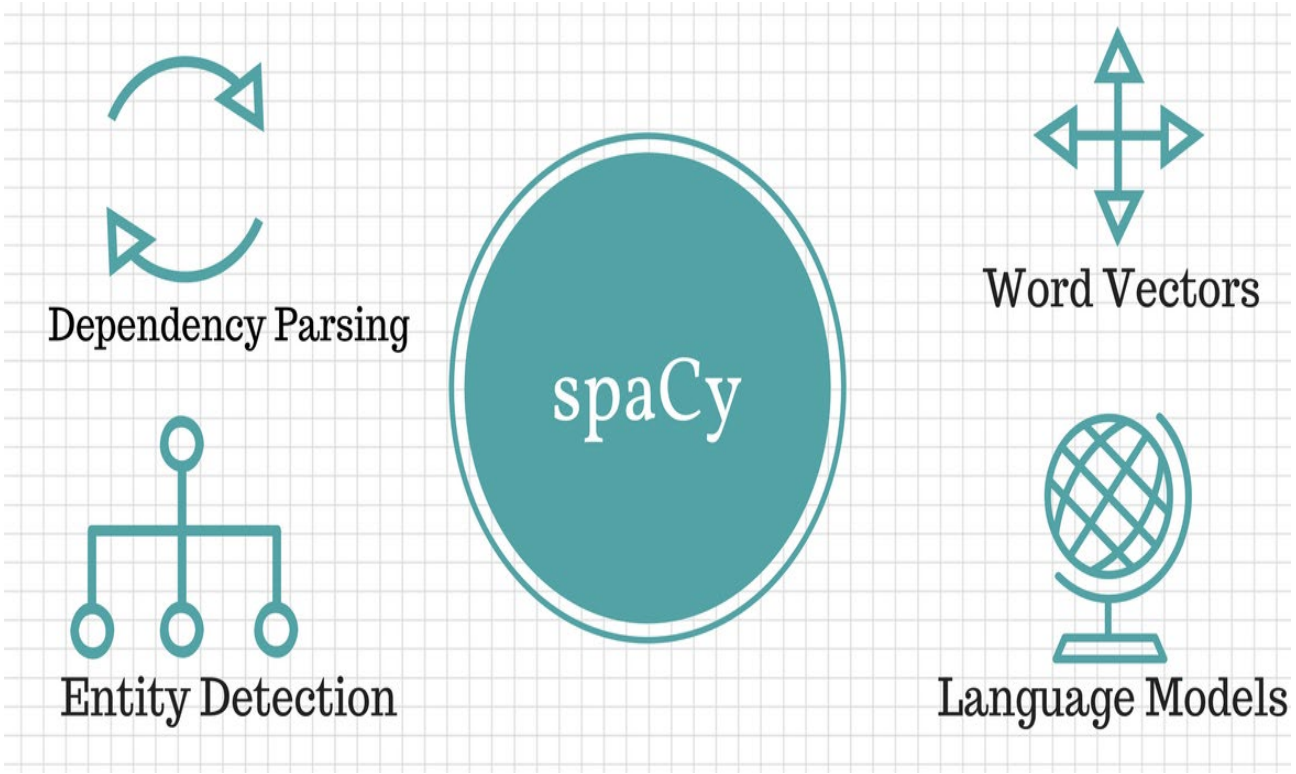
Coreference resolution is used to identify all of the ways the entity is named throughout the text. The step where noun phrases are decided if they relate to the same entity or not is called *coreference or anaphora resolution*.

6. Output results generation

This stage entails converting the structures collected during the preceding processes into output templates that follow the format defined by the user. It might comprise a variety of normalization processes.

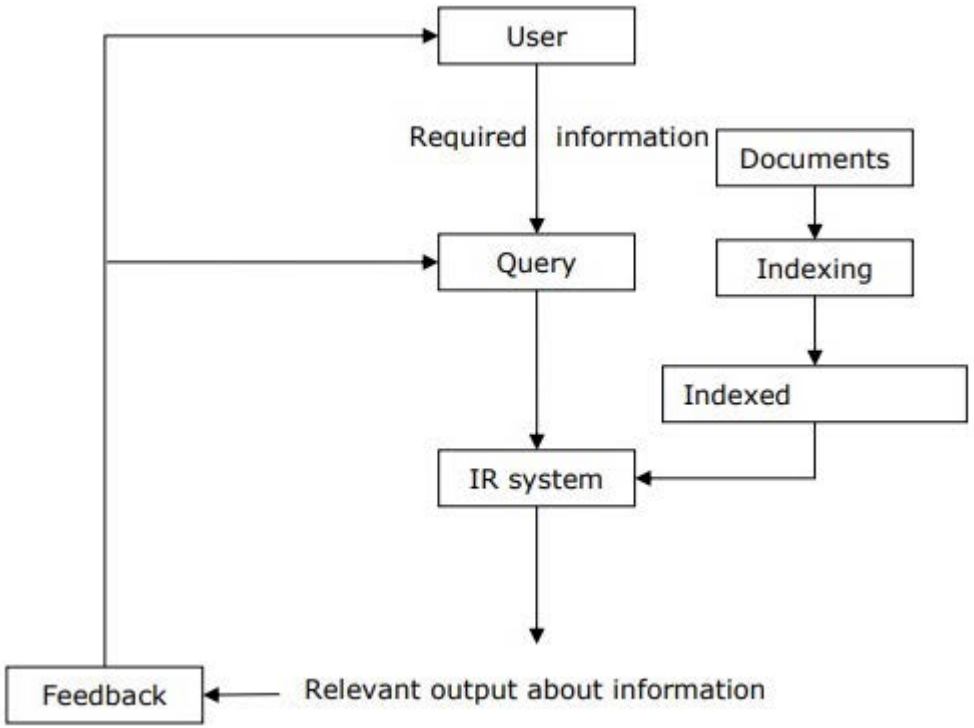
Information Extraction Techniques Using Natural Language Processing

1. Regular Expression.
2. Part-of-speech tagging.
3. Named Entity Recognition.
4. Topic Modeling.
5. Rule-Based Matching.



NLP - Information Retrieval

- Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information.
- The system assists users in finding the information they require but it does not explicitly return the answers of the questions. It informs the existence and location of documents that might consist of the required information.
- The documents that satisfy user’s requirement are called relevant documents.
- A perfect IR system will retrieve only relevant documents.



It is clear from the diagram that a user who needs information will have to formulate a request in the form of query in natural language. Then the IR system will respond by retrieving the relevant output, in the form of documents, about the required information.

Classical Problem in Information Retrieval (IR) System

- The main goal of IR research is to develop a model for retrieving information from the repositories of documents. Here, we are going to discuss a classical problem, named **ad-hoc retrieval problem**, related to the IR system.
- In ad-hoc retrieval, the user must enter a query in natural language that describes the required information. Then the IR system will return the required documents related to the desired information. For example, suppose we are searching something on the Internet and it gives some exact pages that are relevant as per our requirement but there can be some non-relevant pages too. This is due to the ad-hoc retrieval problem.

Types of Information Retrieval (IR) Model

- Classical IR Model
- It is the simplest and easy to implement IR model. This model is based on mathematical knowledge that was easily recognized and understood as well. Boolean, Vector and Probabilistic are the three classical IR models.
- Non-Classical IR Model
- It is completely opposite to classical IR model. Such kind of IR models are based on principles other than similarity, probability, Boolean operations. Information logic model, situation theory model and interaction models are the examples of non-classical IR model.
- Alternative IR Model
- It is the enhancement of classical IR model making use of some specific techniques from some other fields. Cluster model, fuzzy model and latent semantic indexing (LSI) models are the example of alternative IR model.

Design features of Information retrieval (IR) systems

Inverted Index

- The primary data structure of most of the IR systems is in the form of inverted index. We can define an inverted index as a data structure that list, for every word, all documents that contain it and frequency of the occurrences in document. It makes it easy to search for ‘hits’ of a query word.

Stop Word Elimination

- Stop words are those high frequency words that are deemed unlikely to be useful for searching. They have less semantic weights. All such kind of words are in a list called stop list. For example, articles “a”, “an”, “the” and prepositions like “in”, “of”, “for”, “at” etc. are the examples of stop words. The size of the inverted index can be significantly reduced by stop list. As per Zipf’s law, a stop list covering a few dozen words reduces the size of inverted index by almost half. On the other hand, sometimes the elimination of stop word may cause elimination of the term that is useful for searching. For example, if we eliminate the alphabet “A” from “Vitamin A” then it would have no significance.

Stemming

- Stemming, the simplified form of morphological analysis, is the heuristic process of extracting the base form of words by chopping off the ends of words. For example, the words laughing, laughs, laughed would be stemmed

Semantic Search and Evaluation

- Semantic search means understanding the intent behind the query and representing the “knowledge in a way suitable for meaningful retrieval,” according to Towards Data Science.
- Semantic search is a search technique that uses natural language processing (NLP) and machine learning (ML) to understand the context and meaning behind a user's search query.

What is a Semantic Search Engine?

- A Semantic Search Engine (sometimes called a Vector Database) is specifically designed to conduct a semantic similarity search.
- Semantic Search Engines will use a specific index algorithm to build an index of a set of vector embeddings. Milvus has 11 different Index options, but most Semantic Search Engines only have one (typically HNSW).
- With the Index and similarity metrics, users can query for similar items with the Semantic Search Engine.

Keyword Search Vs Semantic Search

- At first, search engines were lexical: the search engine looked for literal matches of the query words, without understanding of the query's meaning and only returning links that contained the exact query.
- By using regular keyword search, a document either contains the given word or not, and there is no middle ground
- On the other hand, "Semantic Search" can simplify query building, because it is supported by automated natural language processing programs i.e. using Latent Semantic Indexing — a concept that search engines use to discover how a keyword and content work together to mean the same thing.

How to Implement a Semantic Search Engine?

- There are several options to implement Semantic Search. Here are a few options
- Python Semantic Search Engine. You can build a custom Semantic Search on your own corpus of data using Python, a machine model, and a Vector Index Algorithm like FAISS, HNSW, or even ANNOY. Here is a tutorial to walk you through how to implement Semantic Search with Facebook AI Similarity Search (FAISS).
- Traditional keyword based Search Engines like ElasticSearch also have added Vector Search capabilities. The benefit is that you can easily add vector search to a solution already using Elasticsearch.
- Popular database solutions like PostgreSQL have added extensions like Pgvector to support vector search. Here is a tutorial to walk you through how to get started using Pgvector.
- Vector Databases Another great option is to use a Vector Database to implement Semantic Search. With a vector database, you store and index the vector embeddings that you generate with your chosen machine learning algorithm. With most vector databases, you use HSNW to generate the index, with Milvus, you can choose from 11 different index types to best fit your use case. When you start your search, you will convert to a vector embedding then do a query against your dataset to find the most similar items.

Types of Semantic Search

1. **Entity Recognition and Linking**: This type recognizes entities within the text (like names, locations, products) and links them to a knowledge base or ontology. This helps in understanding the context and relations among entities.
2. **Concept Search**: Goes beyond matching keywords to understanding concepts within the text. For example, searching for "cold symptoms" might also return results related to flu or pneumonia without those exact words being present.
3. **Lexical Affinity**: Determines the association and co-occurrence frequencies of words within a corpus to infer the context and meaning, thus aiding in retrieving more relevant results.
4. **Query Expansion**: Automatically expands a user's search query by adding synonyms or related terms to fetch broader results that still align with the user's intent.

Semantic search utilizes advanced NLP techniques such as:

- **Word Embeddings**: Vector representations of words that capture their meanings, relations, and context within a dimensional space. Tools like Word2Vec, GloVe, or BERT can be used to analyze semantic similarities between different terms.
- **Ontologies and Knowledge Graphs**: Structured frameworks that organize information and define relationships between concepts. They help in understanding how various entities are interconnected.
- **Contextual Analysis**: Interprets the context in which words or phrases are used. This might involve looking at surrounding text to gauge sentiment, usage, and intent.
- **Machine Learning and Deep Learning**: Algorithms that learn from data to make inferences about the semantics of a query or a document. They can adapt to new patterns in language use, improving the search experience over time.