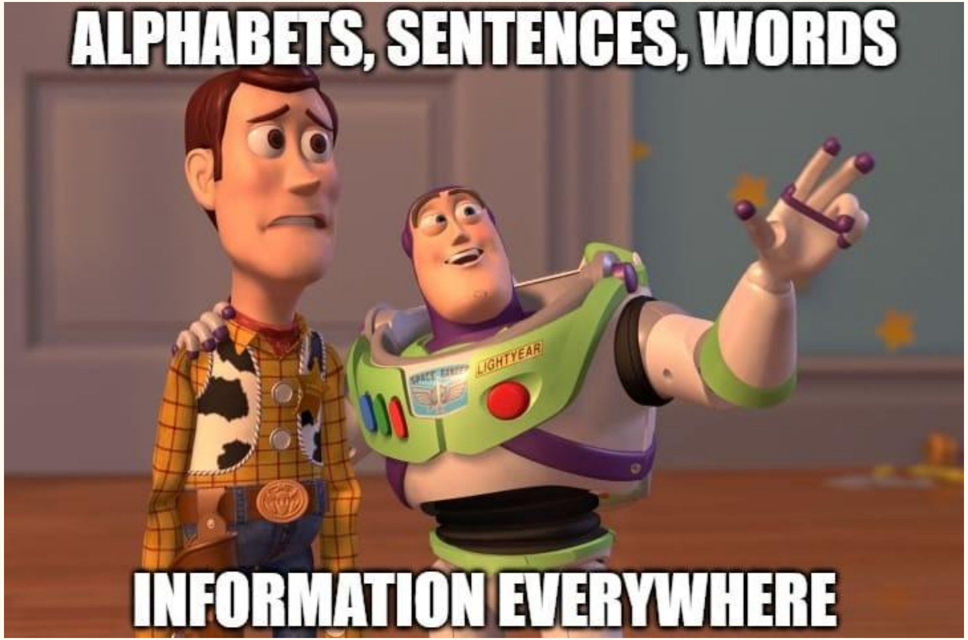
Natural Language Processing

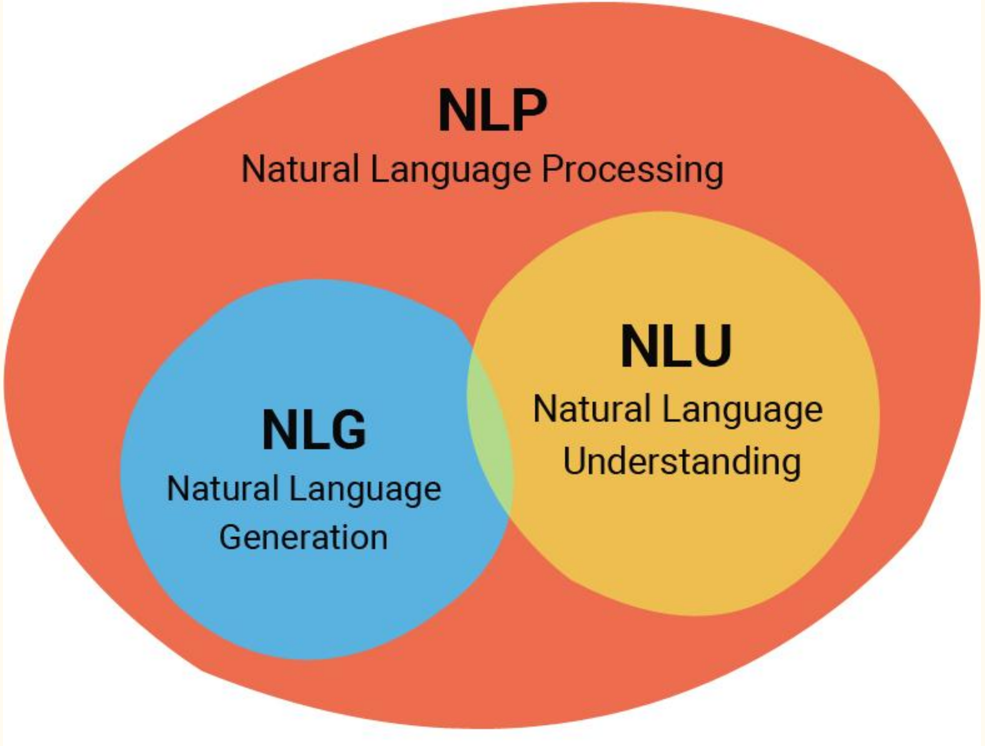
Tinkerers' Lab IITH

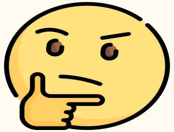
Welcome to the World of Natural Language Processing!



Natural Language Processing

- Subfield of AI, specializing in understanding and generating texts that “makes sense” to humans.
- It has 2 sub categories: Natural Language Understanding (NLU) and Natural Language Generation (NLG)
- Wide range of applications like: Text-to-Speech, Generative Text, Chatbots, Language Translation, etc.



But Why is NLP so important? 

- ★ Smart Assistants such as **Siri**, **Alexa** and **Google Assistant**
- ★ Suggesting **replies/autocorrect** in messaging apps
- ★ **Language translation** from one language to another
- ★ **Spam Filtering**
- ★ Content Summarization: **Inshorts**
- ★ **Voice-to-Text** conversion
- ★ **Chatbots**
- ★ ….. and many more!!

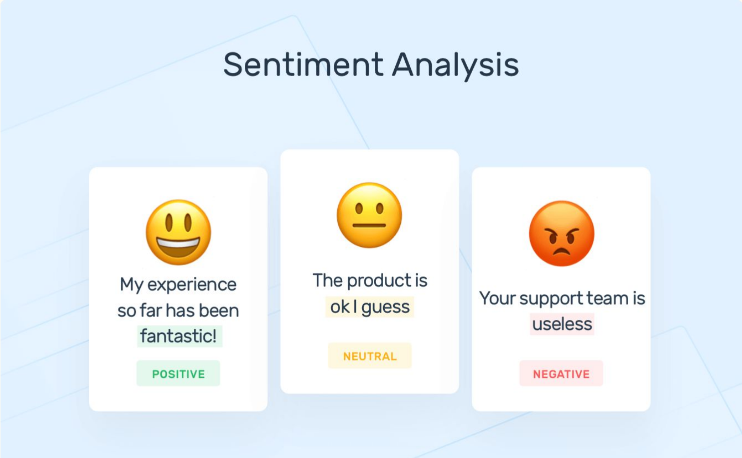
Some NLP related Tools to Play With

- Typeset.io is an AI chatbot/summarizer for scientific Papers which can be used to summarize a research paper and can also be used to ask questions related to that research paper.
- MonkeyLearn is an online text analysis tool. This can be used to predict the sentiment of a text.

Domains in NLP

Sentiment Analysis: Analyzing and determining the sentiment expressed by a text.

Applications: Customer reviews, social media monitoring



Named Entity Recognition: Identifying and classifying entities in text (such as names, locations, etc)

Applications: Information extraction, knowledge graph construction

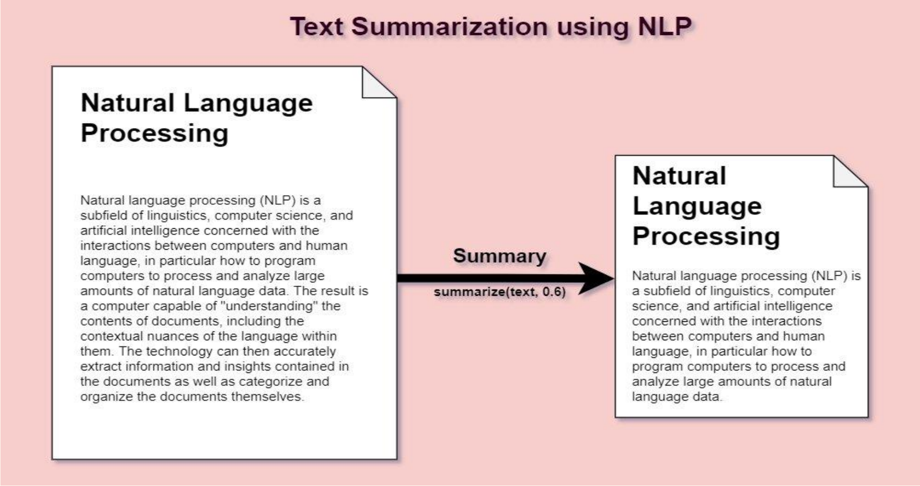
Domains in NLP

Chatbots: Building systems that can engage in natural language conversations.

Applications: Customer service, online support

Text Summarization: Generating concise summaries of longer texts while retaining essential information.

Applications: News summarization, document summarization



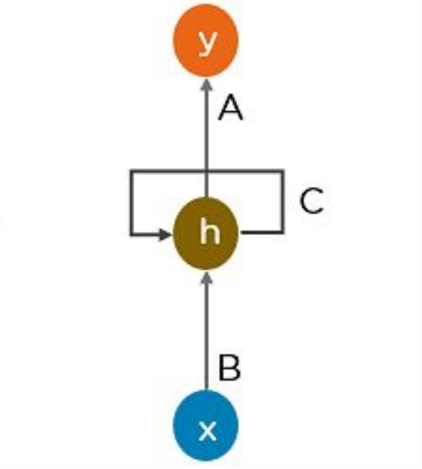
Sequential Data

- Whenever the points in the dataset are dependent on the other points in the dataset the data is said to be Sequential data.
- A common example of this is a Time Series such as a day-to-day weather forecast.

Day 1 Day 2 Day 3 Day 4 Day 5

RNNs

How do RNNs actually handle sequential input?

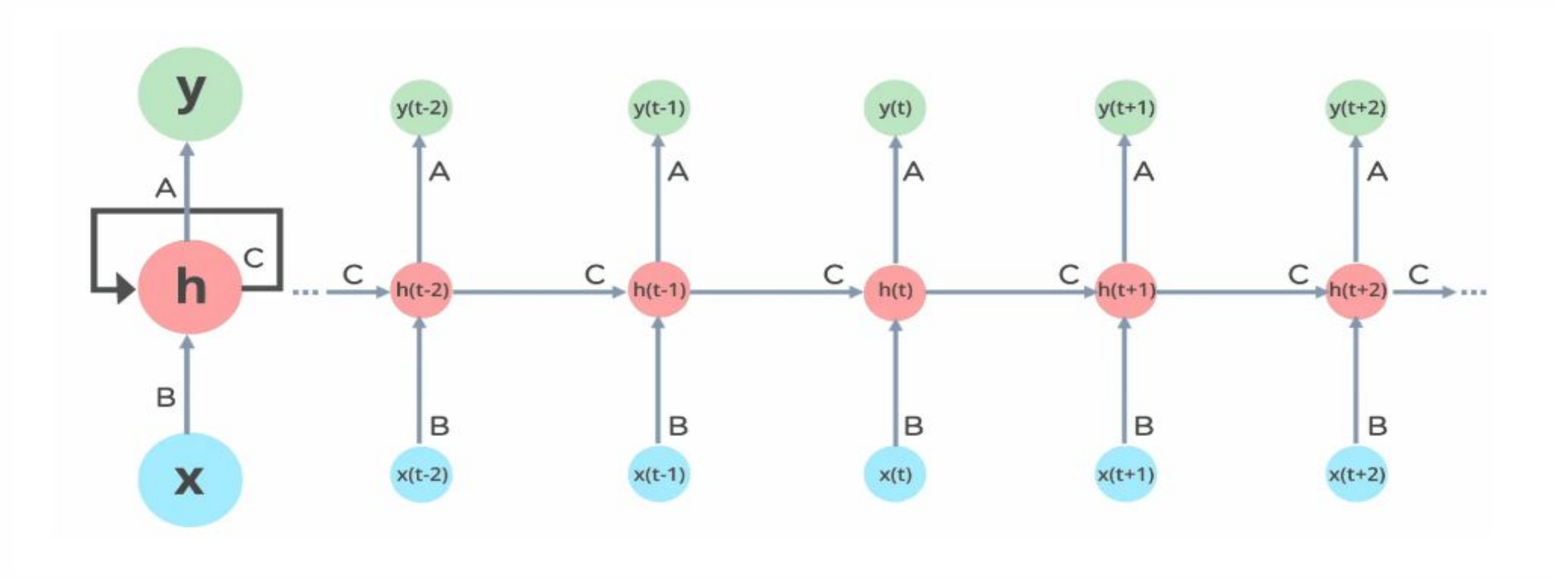


$$h(t) = f_c(h(t-1), x(t))$$

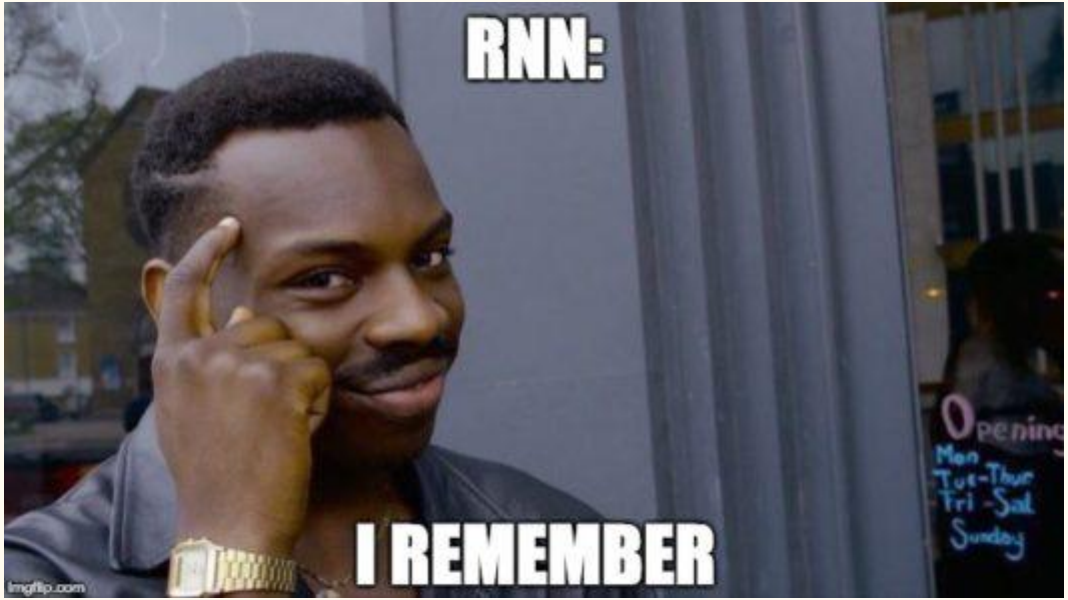
where,
$h(t)$ is the new state function
f_c is a function with parameter c
$h(t-1)$ is the previous state
$x(t)$ is the input vector at time-stamp t

RNNs

How do RNNs actually handle sequential input?



RNNs



Model Long-Term Dependence

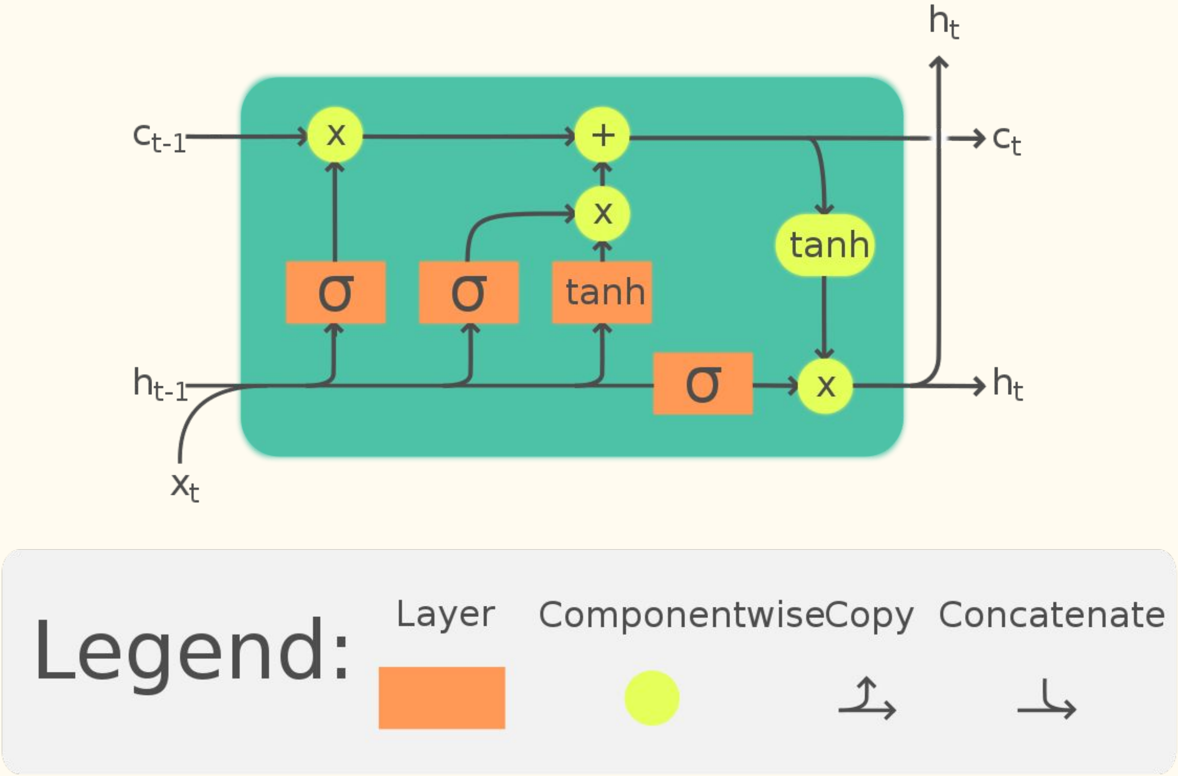
“I was born in **France** but I live in **Spain.** I speak fluent _____.”

French or **Spanish?**

In-order to get useful information we need to fix weights for both **long-term memorise** and **short-term memories**

This is what **LSTM(Long-Short Term Memory)** units are for!!!

LSTMs



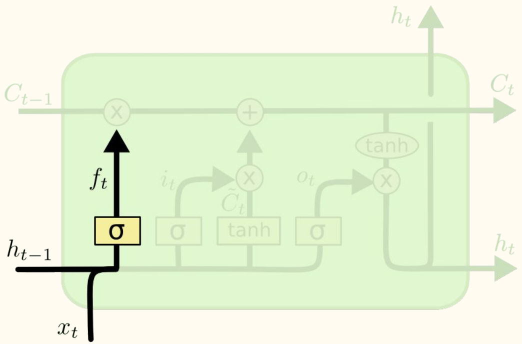
x_t Input Vector

h_t Cell Output

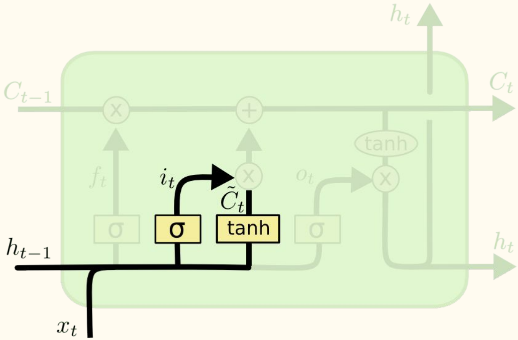
c_t Cell State

LSTMs

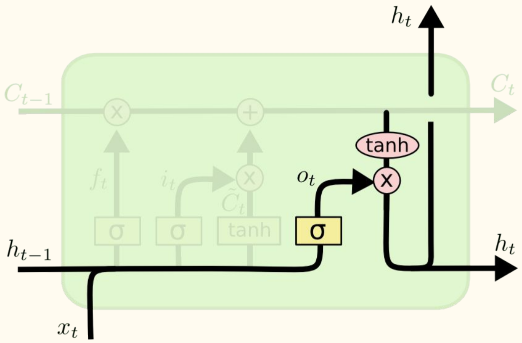
- The key to LSTMs is the cell state.
- LSTMs have the ability to remove or add information to the cell state
- The dataflow in LSTMs can be divided into 3 steps.



Forget Step

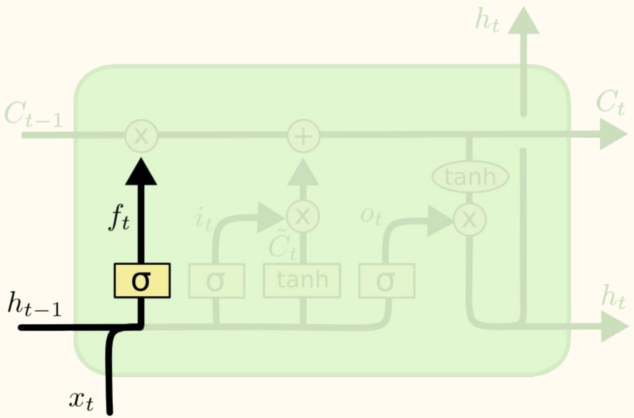


Input/Store Step



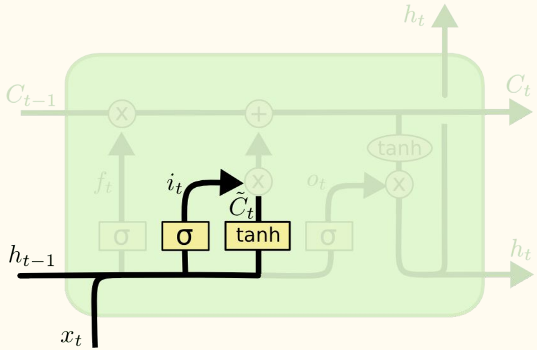
Output Step

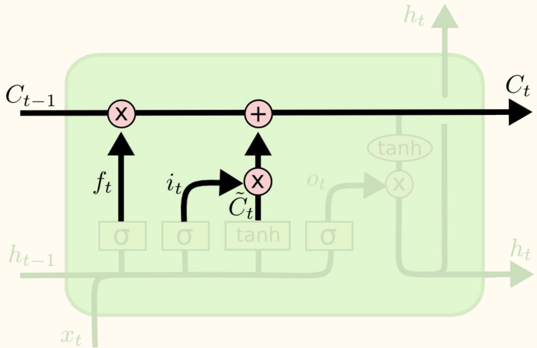
LSTMs: Forget Step



- The first step is to decide which information we're going to throw away from the **cell state.**
- This decision is taken by looking at h_{t-1} and x_t.

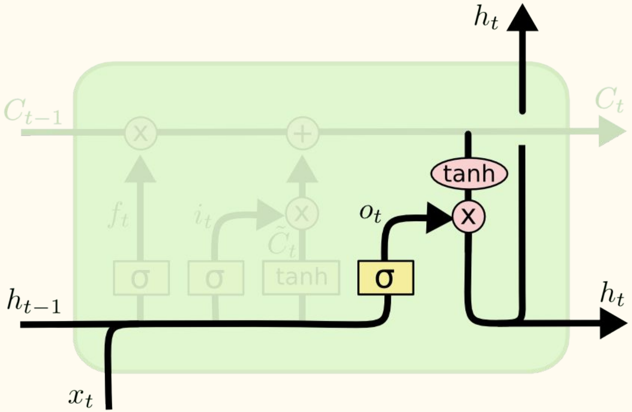
LSTMs: Input/Store Step





- The next step is to decide what new information we're going to store in the new cell state.
- This has two parts
- First the input layer we decide which values we'll update using h_{t-1} and x_t.
- Next we update the **previous cell state** using the output of the previous state $\tilde{C}_t$ and C_{t-1}.

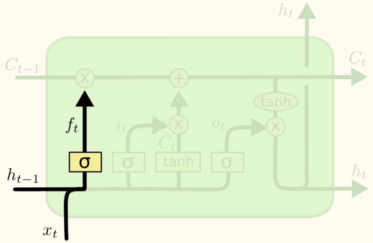
LSTMs: Output Step



- Finally our unit needs to give an output which is based on our cell state and the input.

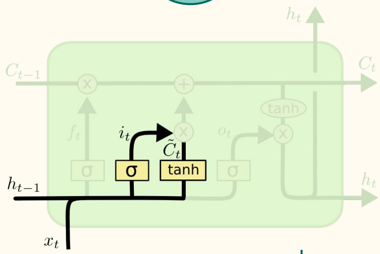
LSTMs: Maths :)

1



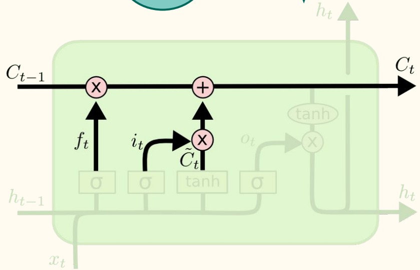
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

2



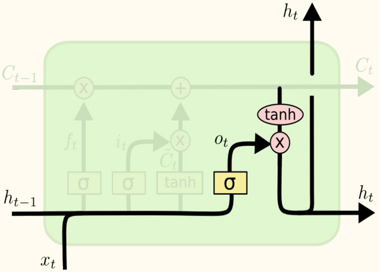
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

3



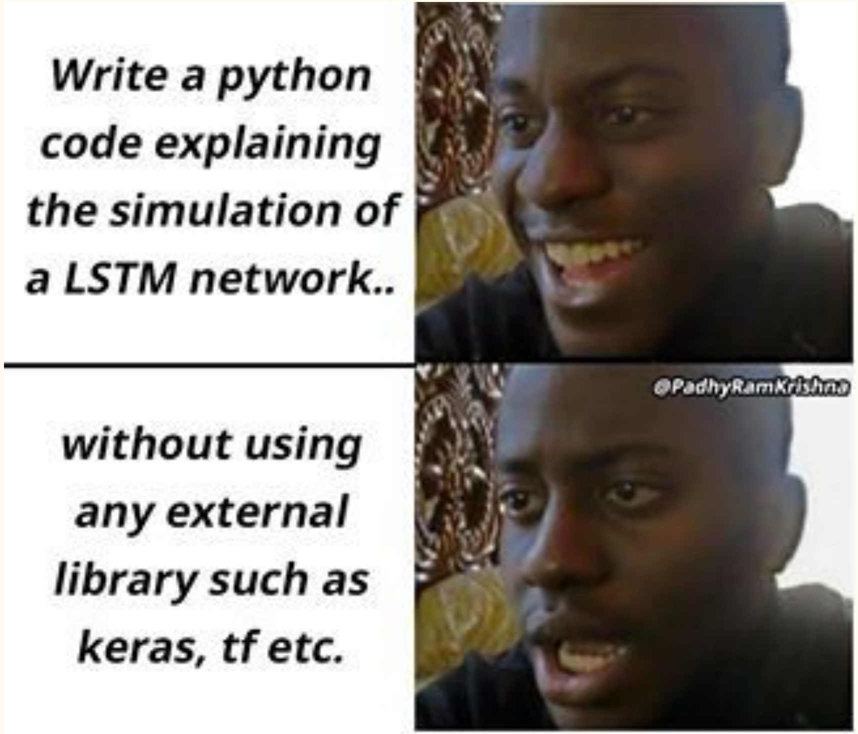
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4



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

LSTMs


Write a python code explaining the simulation of a LSTM network..
@PadhyRamKrishna
without using any external library such as keras, tf etc.

How to Process Language? Basic Units

Challenges of Natural Language:

- Words as such are not numbers to process directly.
- Sequence of words is very important unlike classification/recognition tasks.
- Necessity for means to remember sequence of words that followed before.
- Same meaning conveyed in different forms.
  Ex: 1. Dogs are barking because of cold.

  2. Chilly weather is making the dogs bark.

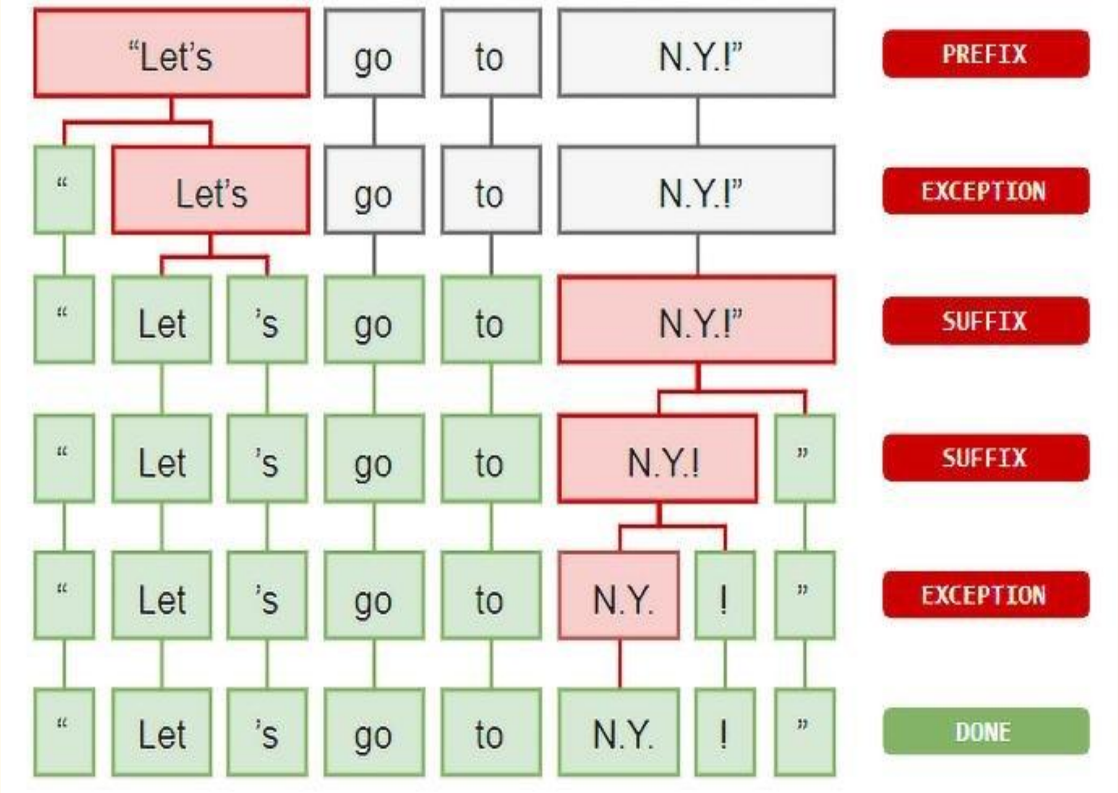
- Grammatical rules to be followed for a certain language.

Solution:

- Tokenization converts sentences to small chunks of words.
- Word-to-Vec can convert similar words into real vector.
- RNNs like LSTM or GRU which can have memory for solving the problem of sequencing.
- Huge dataset that can provide enough examples of variation of sentence structure for same meaning.
- We call them LLMs (Large Language Models)

Tokenization

- It is the process of **converting a large sentence into small chunks** of words.
- But why break a sentence into chunks?
- **High level Answer:** This helps to **assign numbers** to the chunks because numbers is what computer understands.



Types of Tokenization

Character Tokenization: Breaks text into individual characters

Original text: ‘Character Tokenization’

Tokenized: [‘C’, ‘h’, ‘a’, ‘r’, ‘a’, ‘c’, ‘t’, ‘e’, ‘r’, ‘T’, ‘o’, ‘k’, ‘e’, ‘n’, ‘i’, ‘z’, ‘a’, ‘t’, ‘i’, ‘o’, ‘n’]

Sentence Tokenization: Splits text into sentences

Original text: ‘Sentence tokenization splits text into sentences. Here is an example.’

Tokenized: [‘Sentence tokenization splits text into sentences’, ‘Here is an example’]

Word Tokenization: Splits text into words

Original Text: ‘Word tokenization is an important step in NLP’

Tokenized : [‘Word’, ‘tokenization’, ‘is’, ‘an’, ‘important’, ‘step’, ‘in’, ‘NLP’]

Subword Tokenization: Splits words into sub-words. Helps with out-of-vocabulary tokens.

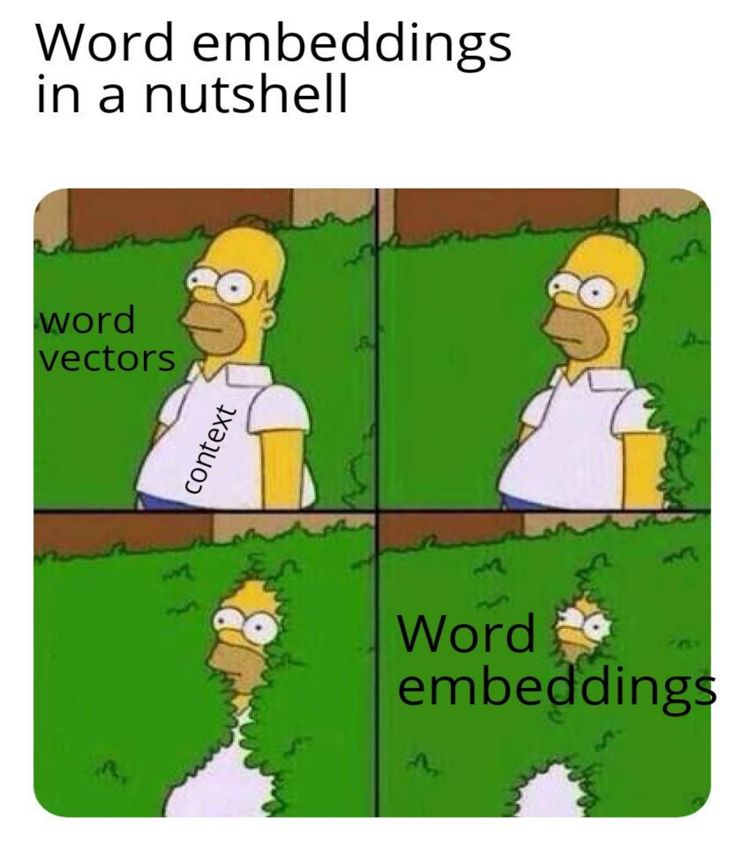
Original Text: ‘Tokenization’

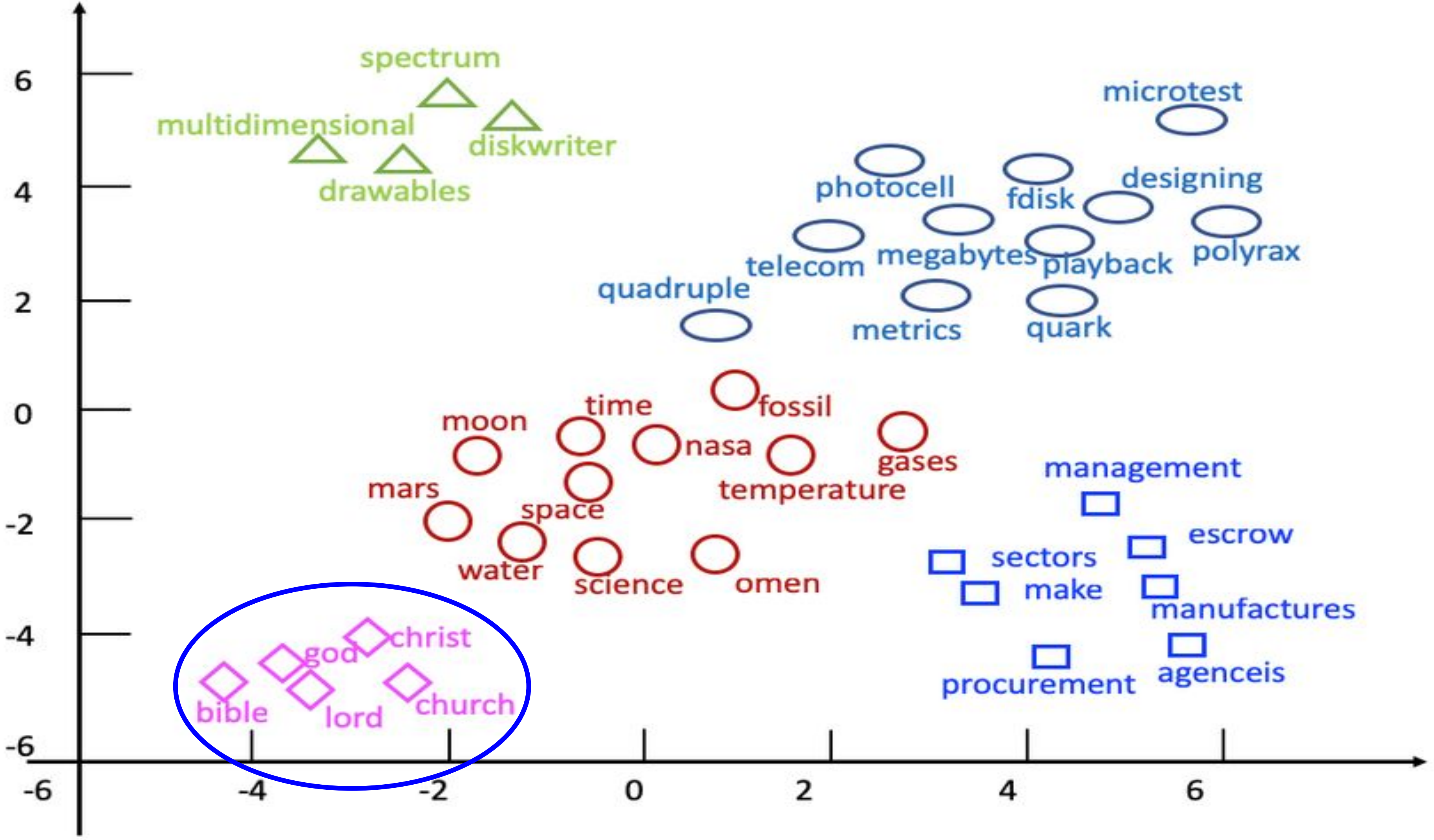
Tokenized: [‘Token’, ‘ization’]

Word Embedding

- It is the process of converting tokens into numerical data because character data is multi-dimensional!

- Word embeddings allows us to reduce data dimensionality and capture the importance of the word in the sentence based on its neighbours and context in the sentence.




spectrum
multidimensional
diskwriter
drawables
microtest
photocell
fdisk
designing
telecom
megabytes
playback
polyrax
quadruple
metrics
quark
time
fossil
moon
nasa
gases
mars
temperature
space
water
science
omen
management
escrow
sectors
make
manufactures
procurement
agenceis
christ
god
bible
lord
church
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6

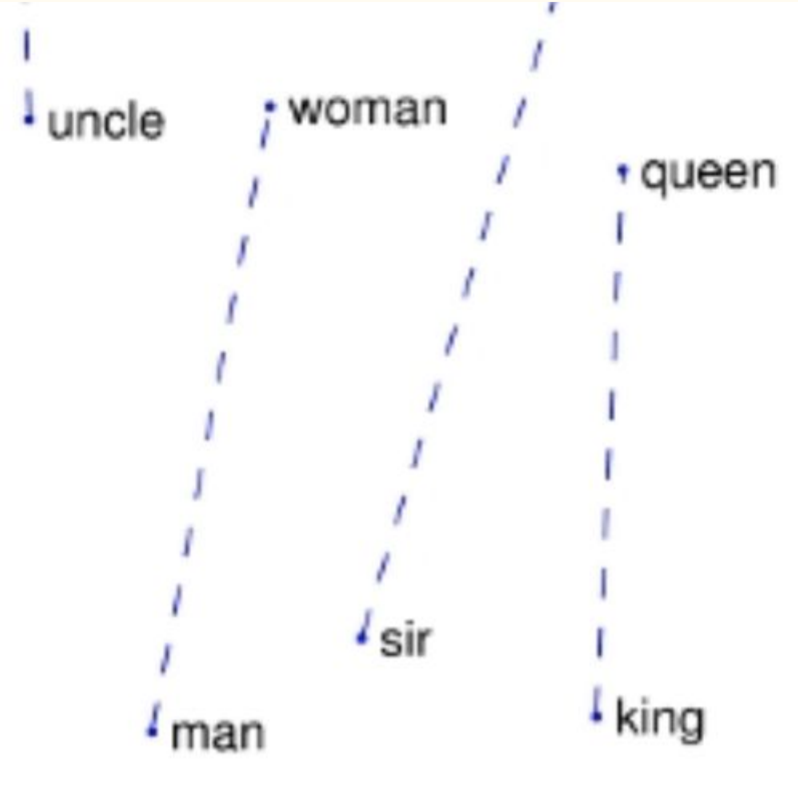
Obvious Question

- In general, word embedding can be thought of assigning “similar” words a “near-same” real value.
- Look at the picture, in the last slide and then at the blue marker at the bottom-left
- You may see that “God”, “Christ”, “Lord” are placed near each other because they possess almost same meaning.
- What word-embedding does is the same job at a much higher dimension.

- Well, you don't need to it exactly. Spacy and many other libraries are there for you!
- The techniques used are of many types, but the most common one is TF-IDF, GloVE, BERT and Word2Vec
- These are some mind-boggling jargons which you don't need to worry about now.

GloVe (Global Vector for Word Representation)

GloVe is a word vector technique that encode the co-occurrence probability ratio between two words as vector differences. GloVe uses a weighted least squares objective that minimizes the difference between the dot product of the vectors of two words and the logarithm of their number of co-occurrences.
GloVe is capable of finding linear sub-structures and nearest-neighbours in its embedding vector space.



Further Readings

- **Text Generation With RNNs:**
  - https://blog.paperspace.com/recurrent-neural-networks-part-1-2/
  - https://karpathy.github.io/2015/05/21/rnn-effectiveness/?ref=blog.paperspace.com
- **LSTMs:**
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/?ref=blog.paperspace.com
- **Roadmap for NLP:**
  - https://medium.com/aimonks/roadmap-to-learn-natural-language-processing-in-2023-6e3a9372b8cc

Thank You
For attending...
Follow us :)