

Basics of NLP And Research Avenues in NLP

Dr. Pruthwik Mishra
Department of AI, SVNIT Surat
18.12.2025

Session in FDP on “Artificial Intelligence and Data Science:
Foundations, Pedagogy, Tools, and Research Trends”
for the Faculty Members of P. P. Savani University, Surat organized by the
Department of AI, SVNIT, Surat

How is NLP different from other domains?

- ML and Neural Approaches need data/features to train
 - The data is in numeric format
- NLP (Natural Language Processing) deals with text
 - Text are not numbers
 - What is the minimum unit for processing?
 - How do we convert them into a numeric format?

Numeric Conversion of Text

- What are the possible options?
 - Can we just give a separate index to a word in a piece of text?
 - Sentence: I am in Surat .
 - Index Sequence: 11 32 107 1377 32 [Assuming we have 10000 unique words in a corpus]
 - One-Hot-Encoding
 - (Assuming we have 10000 unique words in a corpus) Each word will be of 10000 dimensions
 - The index of the word will be 1 others will be 0
 - Term Frequency
 - Count/Frequency of each word in the corpus
 - Can also be normalized
 - TF-IDF
 - Term Frequency and Inverse Document Frequency
 - One of the most widely used representations
 - $TF_IDF = TF * IDF$
 - Reference: <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

More on TF-IDF

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	0, 1
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \{d \in D : t \in d\} $)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log \left(\frac{N}{1 + n_t} \right) + 1$
inverse document frequency max	$\log \left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t} \right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

But, Tokenization matters

- Tokenization
 - Splitting/Dividing text into *meaningful?* tokens
 - Based on Tokens
 - Representations will vary
 - What is a token?
 - Words?
 - Punctuations?
 - Emails?
 - URL?
 - Phone Number?
 - The priority is to reduce the sparsity
 - Sparse means no representation at all

Lemmatization/Stemming/Morph Analysis

- Languages are beautiful
- A root can have multiple word forms
 - walk, walks, walked, walking for *walk*
 - go, goes, went, going, gone for *go*
 - child, children for *child*
- Can we represent all these into a single root form?
 - This can be done using Lemmatization (finding lemma) or Morph Analysis
 - Usually linguistic root
- Stemming
 - Reduces a word to its base form (may not be linguistic)
 - Usually done using basic rules of affixation
- Other Preprocessing
 - Case Normalization
- Indian Languages are much more complex
 - Single root can have thousands of word forms in Dravidian languages

Basic Units of Processing

- Word n-grams
 - Text: I am in Surat .
 - Unigrams (n=1): [I, am, in, Surat, .]
 - Bigrams (n=2): [I am, am in, in Surat, Surat .]
 - Trigrams (n=3): [I am in, am in Surat, in Surat .]
 - So on
- Character n-grams
 - Text: I am in Surat .
 - Unigrams: [I,SPACE, a, m, i, n, S, u, r, t, .]
 - Bigrams: [ISPACE, SPACEa, am, mSPACE, SPACEi, in, nSPACE, SPACES, Su, ur, ra, at, tSPACE, SPACE.]
 - So on
- Multiple n-grams at Word level and Character level can be combined
- Both can also be combined
- The current words use subwords as the input units found through different subword algorithms such as Byte-Pair Encoding, WordPiece, SentencePiece

Types of NLP Tasks

- Text/Sequence Classification
 - Sentiment Analysis
 - Language Identification
- Token Classification
 - Part-of-Speech (POS) Tagging
 - Chunking
 - Named Entity Recognition
- Text/Sequence Generation
 - Machine Translation
 - Summarization
 - Question and Answering
 - Autocompletion
- We will have demo sessions for all of the tasks.

Text Classification and/or Categorization

1. Text Categorization and Text Classification are mostly used interchangeably in the NLP Community
2. But in recent years there has been attempts to delineate the two
3. Text Classification refers to the identification of a text into one of predefined classes
 - a. Only one class will be assigned
 - b. Can be modeled as a Multi-Class/Binary Classification Task
4. Text Categorization goes another step further allowing not only categories, but also sub-categories
 - a. Allows Hierarchical Categorization
 - b. Can be modeled as a Multi-Label or Hierarchical Classification Task
5. Both are Natural Language Processing Tasks
6. We will use the definition mentioned in 1

Classification Tasks in NLP

- Text Classification
 - Author Identification
 - **Sentiment Classification/Detection**
 - Aspect Detection
 - Hate Speech Detection
 - Dialect Identification
 - Offensive Language Identification
 - Fake News Detection
 - Query Classification
 - **Language Identification**
 - Customer Complaint Identification
- Usually done using Supervised Techniques
- In case of scarce data, Semi-Supervised Techniques are also explored

Unsupervised Learning

- Clustering
 - Hate Speech Detection

A word cloud visualization showing clusters of English hashtags related to political hate speech. The words are colored blue or orange, indicating different clusters identified by a clustering algorithm. The most prominent words include 'you', 'trump', 't', 'realdonaldtrump', 'co', 'https', 'trumpisatraitor', and 'fucktrump' in blue; and 'doctorsfightback', 'coctors', 'world', 'iccc', 't', 'co', 'rain', 'shameonicc', 'https', and 'dhonikeepstheglove' in orange.

A word cloud visualization showing clusters of Hindi/Urdu hashtags related to political hate speech. The words are colored blue or orange, indicating different clusters identified by a clustering algorithm. The most prominent words include 'अंग्रेजों', 'करना', 'जीत', '%', 'ये', 'इस', 'नहीं', 'हो', 'देश', 'एक', 'कुमार', 'भारत', 'हार', 'साथ', 'जी', 'ही', and 'र'.

Neural Network Approaches

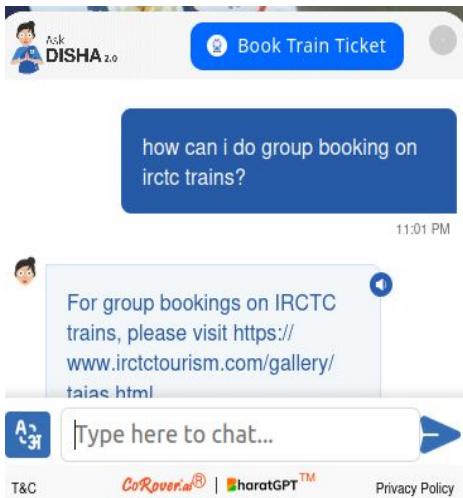
- Multi-Layer Perceptrons
- Recurrent Neural Networks and its Variants
 - Long-Term Short Term Memory (LSTM)
 - Gated Recurrent Unit (GRU)
- Recursive Neural Networks
- Transformers
 - Encoder-Only Models
 - BERT and its Variants
 - Decoder-Only Models
 - Large Language Models
 - Instruction Tuned Models

Representations

- The efficiency of the Neural Networks depend on the representations of the units
- The units can be:
 - Characters (Character Embedding)
 - Subwords/Character N-grams (Subword Embeddings)
 - Most of the current state-of-the-art models use this
 - Words (Word Embedding)
 - Sentences (Sentence Embedding)
 - Paragraphs/Discourse (Discourse Embedding)
 - Documents (Document Embedding)
- You can see a hierarchical structure here.

What is Text Generation?

- The automatic creation of human-like text by a computer.
- Why is it important?
 - Several Applications
 - chatbots, content creation, summarization, translation, creative writing, coding assistance
 - Humans have always wanted machines to “think” and “write”.



A screenshot of a web-based AI playground. At the top, it shows "All Models > Llama-3.3-70B-Instruct". There are buttons for "LLM", "FP8", and "Popular". The main area is titled "Playground". A user input box contains the text "Write a python code for finding factorial of a number.". Below it, the model has generated the following Python code:

```
def factorial(n):
    """
    Calculate the factorial of a given number.

    Args:
        n (int): The number to calculate the factorial of.

    Returns:
        int: The factorial of the given number.

    Raises:
        ValueError: If the input number is negative.
    """
    if n < 0:
```

Below the code, there's a note: "Here's a simple Python function to calculate the factorial of a given number:".

Example

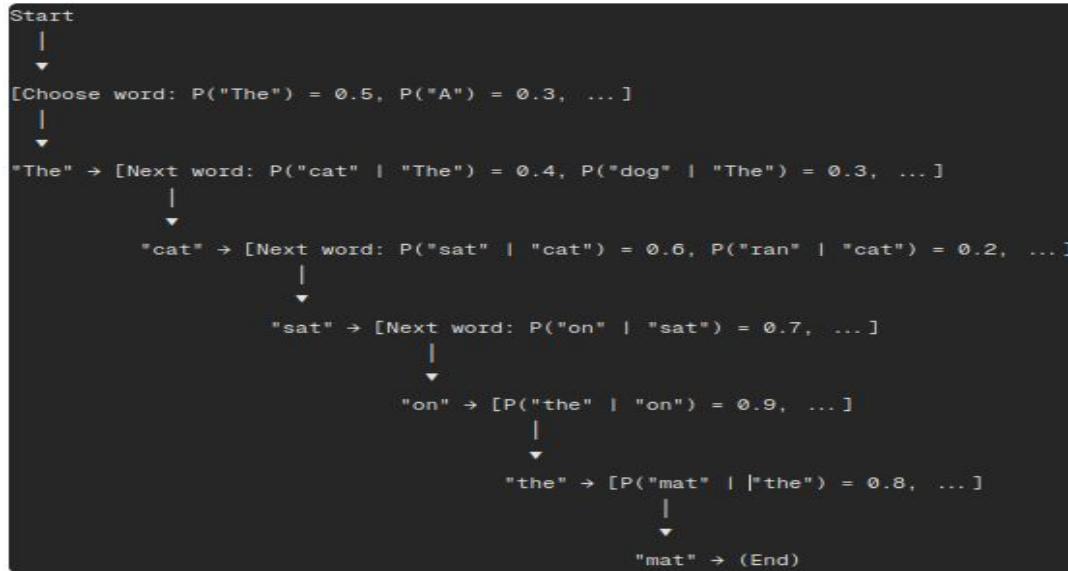
- There are 4 sentences, choose the best one.
 - he briefed to reporters on the chief contents of the statement
 - he briefed reporters on the chief contents of the statement
 - he briefed to reporters on the main contents of the statement
 - he briefed reporters on the main contents of the statement

Which is the best one and why?

- **he briefed reporters on the main contents of the statement**
 - This is the best sentence
 - *brief* is a transitive verb and requires an object
 - *main contents* is more **fluent** than *chief contents*
- “You shall know a word by the company it keeps”
 - Famous quote attributed to British linguist *J.R. Firth*
- This is the basis of language models

The Foundation: Language Models (LMs)

- What is a Language Model?
 - A model that assigns a probability to a sequence of words.
 - Using chain rule of Probability,
 - $P(w_1, w_2, \dots, w_n) = P(w_1)*P(w_2|w_1)*P(w_3|w_1, w_2)*\dots*P(w_n|w_1, \dots, w_{n-1})$
 - Core idea for generation: Predict the next word given the previous words.



Early Approaches: N-gram Language Models

- The probability of the next word in a sequence of n words depends only on the previous “n-1” words
- Markov Assumption
 - First Order: the probability of the next word depends only on the previous word.
 - $P(W_n | W_{n-1})$
 - Second Order: the probability of the next word depends only on the previous word.
 - $P(W_n | W_{n-2}, W_{n-1})$
- Using the Markov assumption, the probability of a sequence of n words can be further simplified
 - $P(w_1, w_2, \dots, w_n) = P(w_1) * P(w_2 / w_1) * P(w_3 / w_1, w_2) * \dots * P(w_n / w_1, \dots, w_{n-1})$ – (No Markov Assumption)
 - $P(w_1, w_2, \dots, w_n) = P(w_1) * P(w_2 / w_1) * P(w_3 / w_2) * \dots * P(w_n / w_{n-1})$ – (With First Order Markov Assumption)
 - $P(w_1, w_2, \dots, w_n) = P(w_1) * P(w_2 / w_1) * P(w_3 / w_1, w_2) * P(w_4 / w_2, w_3) * \dots * P(w_n / w_{n-2}, w_{n-1})$ – (With Second Order Markov Assumption)

Types of Statistical LMs

- Unigram Model
 - Composed of unigrams or single tokens
 - $P(w_1, w_2, \dots, w_n) = P(w_1)*P(w_2)*P(w_3)*\dots*P(w_n)$
 - All the words are independent of each other
 - Zeroth Order Markov Assumption
 - Very strong and weak assumption
- Bigram Model
 - Composed of bigrams or strings consisting of two tokens each
 - First Order Markov Assumption
 - $P(w_1, w_2, \dots, w_n) = P(w_1)*P(w_2/w_1)*P(w_3/w_2)*\dots*P(w_n/w_{n-1})$
- Trigram Model
 - Composed of trigrams or strings consisting of three tokens each
 - Second Order Markov Assumption
 - $P(w_1, w_2, \dots, w_n) = P(w_1)*P(w_2/w_1)*P(w_3/w_1, w_2)*P(w_4/w_2, w_3)\dots*P(w_n/w_{n-2}, w_{n-1})$

How to Compute the Probabilities?

- Maximum Likelihood Estimation (MLE)
- Use relative frequency
 - $P(w_i | w_{i-1}) = \text{count}(w_{i-1}, w_i) / \text{count}(w_{i-1})$
 - This count is computed on a large corpus of data
 - For reliable estimation of probabilities
- Computationally Inexpensive and Easy
- Suffers from Sparsity Problem
 - Zero probabilities for unseen sequences
 - Smoothing Used to handle this
- Limited context window
 - Can't capture long-range dependencies
- Lack of generalization.

	i	want	to	eat
i	6	828	1	10
want	3	1	609	2
to	3	1	5	687
eat	1	1	3	1

Smoothed bigram counts from BeRP Corpus[1]

	i	want	to	eat
i	0.0015	0.21	0.00025	0.0025
want	0.0013	0.00042	0.26	0.00084
to	0.00078	0.00026	0.0013	0.18
eat	0.00046	0.00046	0.0014	0.00046

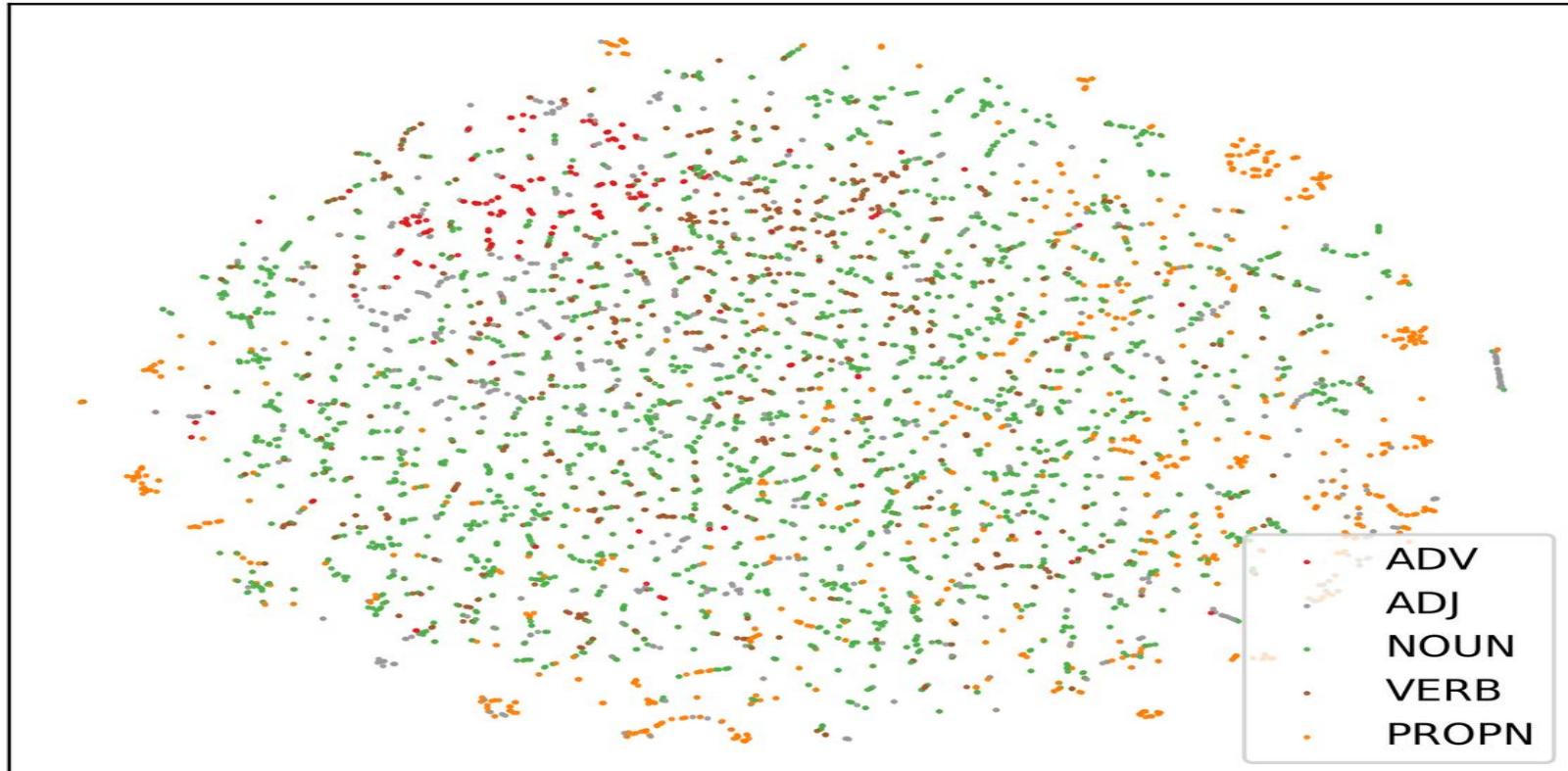
Smoothed bigram probabilities from BeRP Corpus[1]

Beyond N-grams: Introducing Neural Language Models

- Motivation for neural LMs
 - Overcome sparsity and capture longer dependencies
- Word Embeddings: A crucial step
 - Representing words as dense vectors in a continuous space
 - Capturing semantic relationships [2]
 - *India - Sachin + Australia = Pointing* [Word2vec model trained on GoogleNews]
 - *France: Paris:: India: New Delhi*
 - 3 most Similar words to *Computer*
 - *Microcomputer*
 - *Laptop*
 - *Mainframe*
 - Simple feed-forward Neural Networks were used for early word embedding

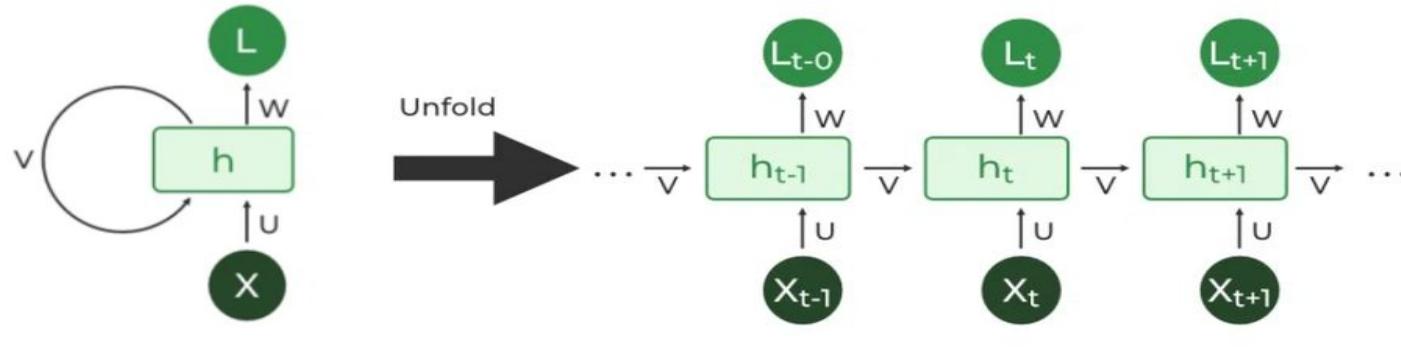
Word Vectors Visualization [2]

5 000 most frequent words in the English Wikipedia model



Recurrent Neural Networks (RNNs) for Text Generation

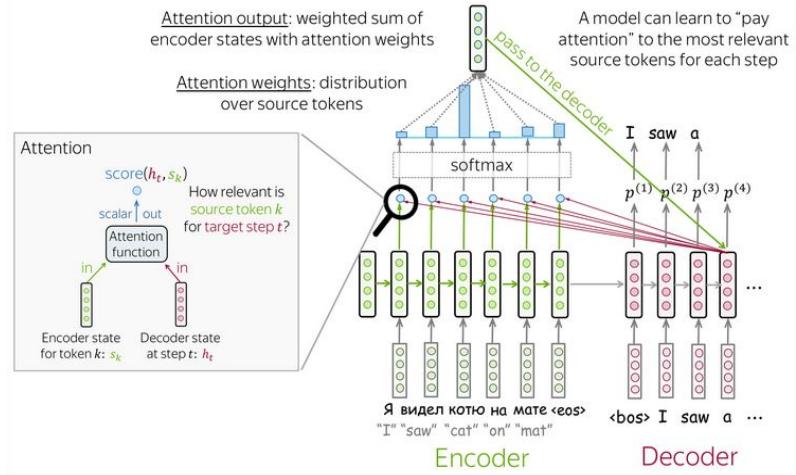
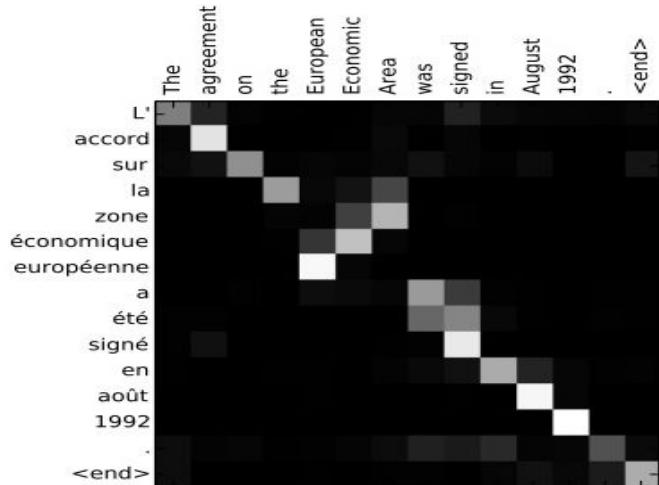
- How RNNs work?
 - Process sequences by maintaining a hidden state that carries information from previous steps



- Vanishing/Exploding gradients
 - Difficulty learning very long-range dependencies
 - To alleviate this problems, LSTM/GRU are proposed using gating mechanism
- Slow training
- Lack of Parallelism

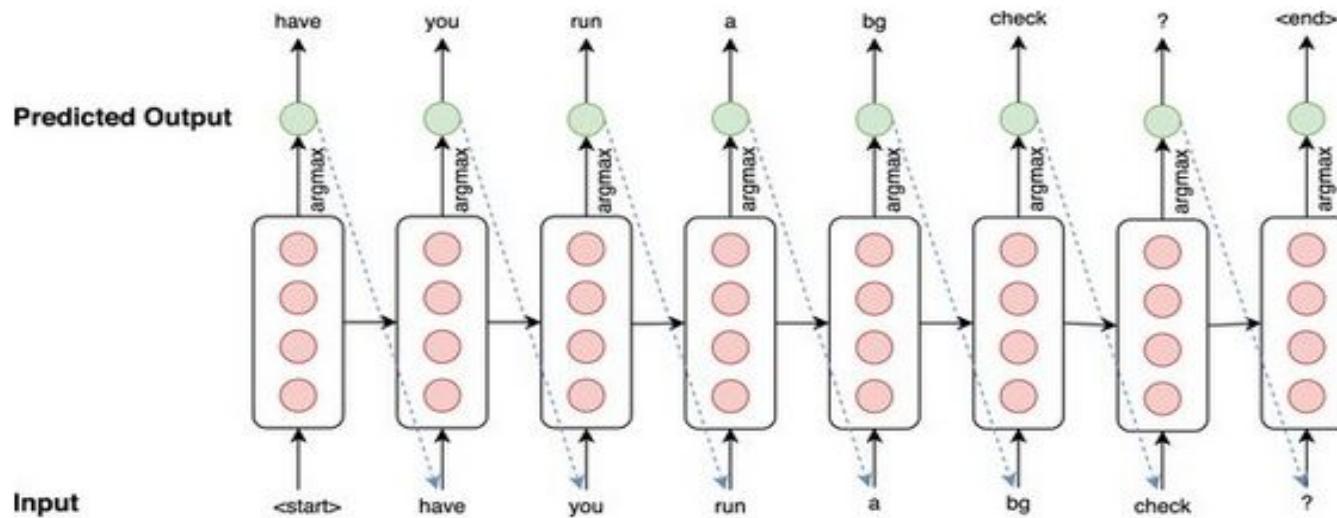
The Breakthrough: The Attention Mechanism

- Allows the model to “*focus*” on relevant parts of the input sequence when generating output
- Encoder-Decoder Architecture with Attention
- Resulted in improved performance in machine translation, summarization, and other sequence-to-sequence tasks



Just a Next Word Prediction Task

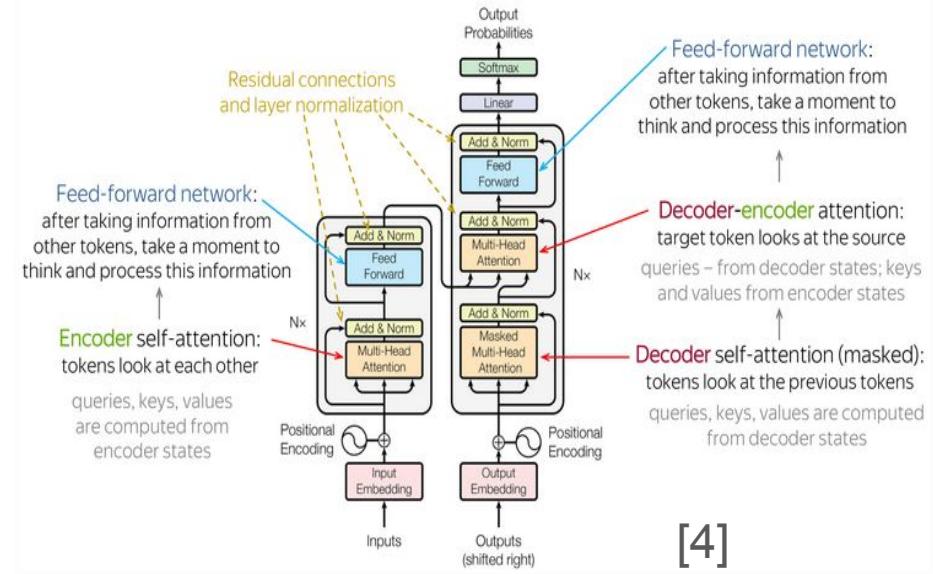
- Text Generation is a next word prediction task in a recursive manner
 - When do you stop generating?
 - After a fixed number of words
 - After the end sentence token is generated



[9]

The Transformer Architecture: The New Paradigm

- Key Innovation: Self-Attention mechanism
 - allowing each word to attend to all other words in the same sequence
- Elimination of recurrence
 - Introduction of Parallelization
- Encoder and Decoder stacks
- Positional Encoding
 - How to retain word order information
- Introduction of Key, Query, Value

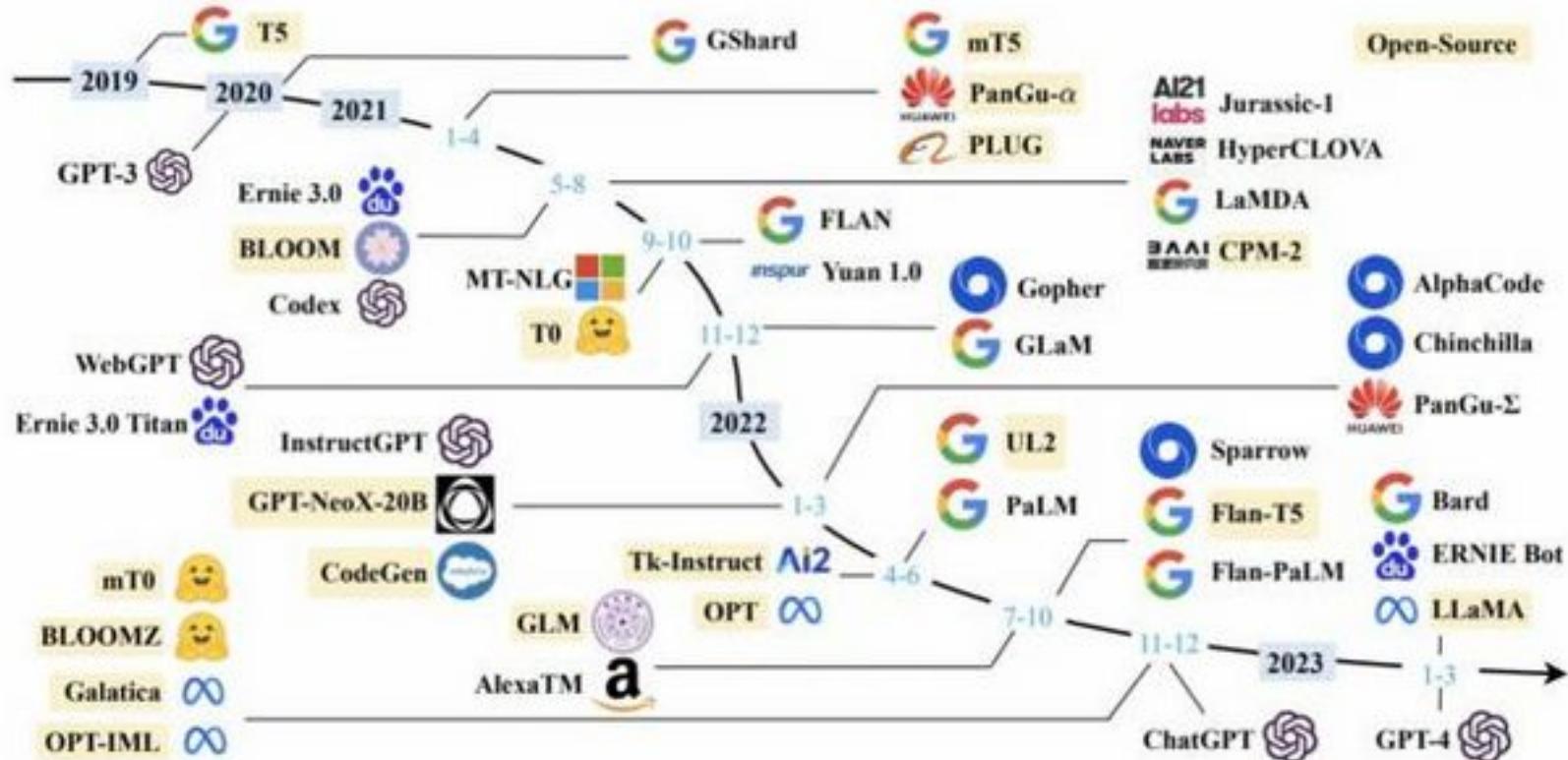


[4]

The Era of Large Language Models (LLMs)

- Scaling Massively Up
 - Massive datasets (Common Crawl, web text, books)
 - Billions and Trillions of parameters
 - Enormous computational resources
- Pre-training and Fine-tuning Paradigm
 - Pre-training: Unsupervised learning on vast text data (next-word prediction, masked language modeling).
 - Fine-tuning: Supervised learning for specific downstream tasks (e.g., question answering, sentiment analysis).
 - Emergent Abilities: Capabilities that arise from scale (e.g., reasoning, code generation, summarization).

Timeline of LLMs [5]



Very Very Competitive Field

Research Avenues in NLP

- Foundational resources and benchmarks
 - Creation of large, clean monolingual and parallel corpora for both major and low-resource Indian languages
- Low-resource and diversity-centric modeling
 - Robust modeling of dialectal variation, spelling non-standardization, and script variation across closely related Indian languages
- Code-mixing, speech, and OCR
 - Modeling code-mixed and code-switched text (e.g., Hinglish, Tanglish) across social media, chat, and speech transcripts
 - Indian-language ASR and TTS with robustness to accent, dialect, noisy environments
 - OCR and handwriting recognition for complex Indic scripts, old prints, and low-quality scans
- Responsible, inclusive, and societal NLP
 - Privacy-preserving and resource-efficient NLP for deployment in Indian settings
 - Bias, fairness, and inclusivity analysis in Indic language models
 - Human-in-the-loop, linguist-in-the-loop methodologies for annotation, evaluation, and documentation of Indian languages
- Applications for Indian domains
 - Machine Translation, Summarization, Q&A, Sentiment Analysis for domains such as governance, healthcare, legal, education

References

1. [Speech and Language Processing, 2nd Edition in PDF format \(complete and parts\) by Daniel Jurafsky, James H. Martin](#)
2. <https://vectors.nlpl.eu/explore/embeddings/en/>
3. <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>
4. https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html
5. <https://www.nextbigfuture.com/2023/04/timeline-of-open-and-proprietary-large-language-models.html>
6. <https://gemini.google.com/>
7. <https://www.iguazio.com/glossary/llm-hallucination/>
8. <https://medium.com/@tehreemyounas/ai-as-a-catalyst-for-creativity-igniting-innovation-within-the-modern-world-36959bef2199>
9. [https://www.researchgate.net/publication/342609939 Developing a Twitter bot that can join a discussion using state-of-the-art architectures](https://www.researchgate.net/publication/342609939_Developing_a_Twitter-bot_that_can_join_a_discussion_using_state-of-the-art_architectures)