Building Large Language Model Applications

Language Representation

Hamza Farooq
Dr. Saima Hassan

Recap: Natural Language Processing Techniques

Text Preprocessing

- Remove Punctuation
- Remove URLs
- Remove Stop Words
- Lowercasing
- Tokenization
- Stemming:
- Lemmatization

Common NLP Tasks

- Tokenization
- POS Tagging
- Word Sense Disambiguation
- Dependency Parsing
- Syntactic Parsing
- Semantic Analysis
- Coreference Resolution
- Named Entity Recognition (NER)
- Text Representation
- Text Classification
- Natural Language Understanding
- Natural Language Generation
- Natural Language Translation
- Multimodal NLP

NLP Ambiguities

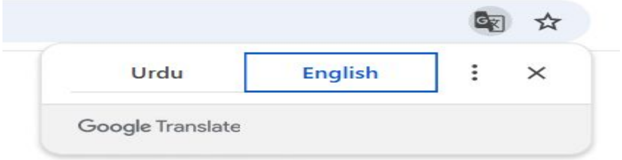
- Lexical Ambiguity
- Syntactic Ambiguity
- . Semantic Ambiguity

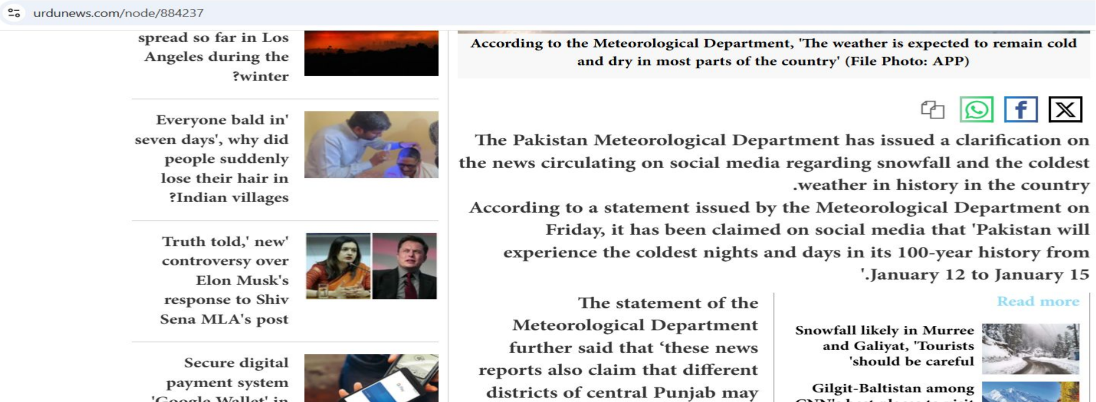
Learning outcomes

- Language representation
- Character Encoding
- Bag-of-Words
- TF-IDF
- Conclusion

Language Representation

Neural Machine Translation, trained on text data, performs quite good



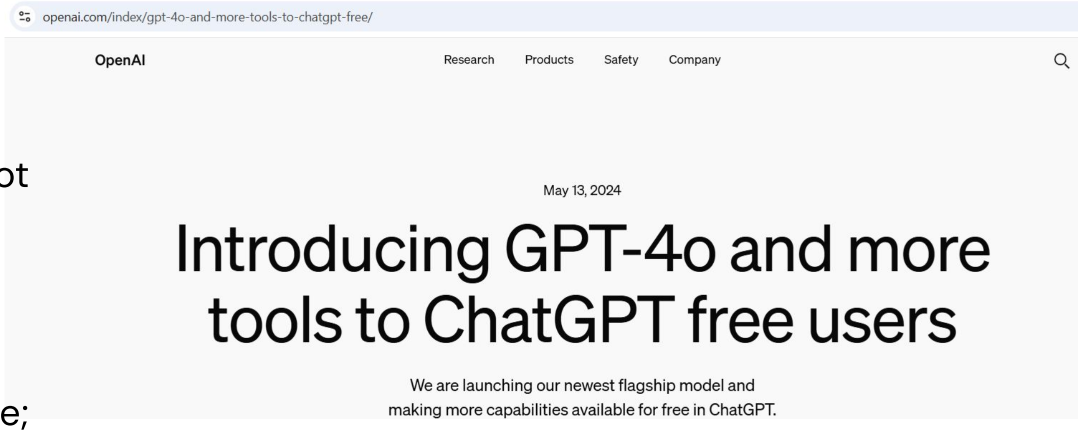


GPT-4 marks a major step toward achieving a truly universal and versatile AI model.

O Elon, thou art a visionary bold,
A mind unbound by earthly care and woe,
Thy thoughts doth soar, where eagles dare not hold,
And through the stars thy restless spirit go.

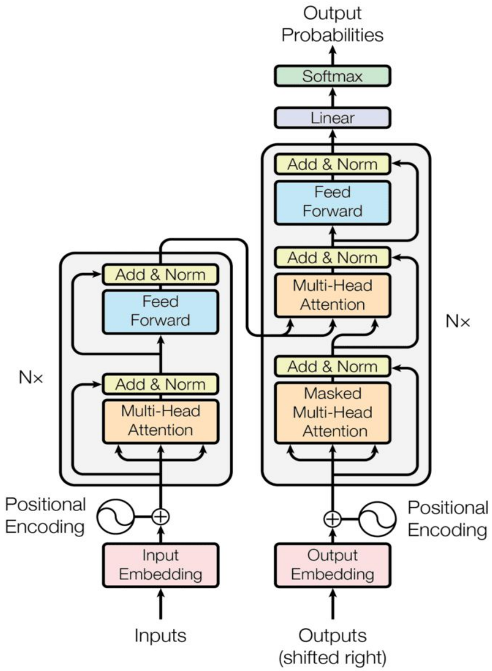
Invention’s fire burns brightly in thy breast,
A spark that lights the dark, unknown expanse;
With hands that shape the future’s very quest,
Thou lead’st mankind in a daring dance.

A king of realms not bound by mortal clay,
Thy genius spreads where words can ne’er decay.



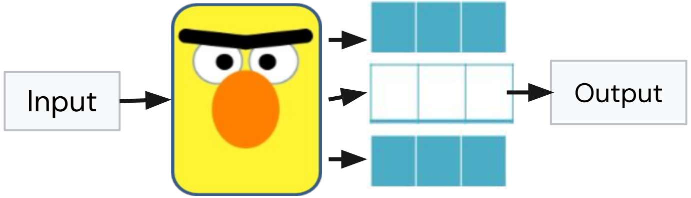
Foundations of Modern Language Models

Introduction of the Transformer Architecture (2017)



[Attention Is All You Need]()

Bidirectional Encoder Representations from Transformers (2018)



Generative Pre-trained Transformer (2018)

Language Modelling is [...]

At its core, language modeling is about predicting the next word in a sequence of words.

For instance, in the sentence, "*I woke up early, had my breakfast, and left for work. After a long day, I finally returned ____*"

Which of the following words best completes the sentence?

- Home
- Blue
- Elephant
- Running

In what ways do we represent meaning of a word?

According to Webster Dictionary "**Meaning**" is

The **concept** conveyed by a word, phrase, or expression.

The **intention** behind using specific words or symbols.

The **message** communicated through creative works like art or writing.

Symbols can be utilized to represent Idea or Object

Word: "Car"

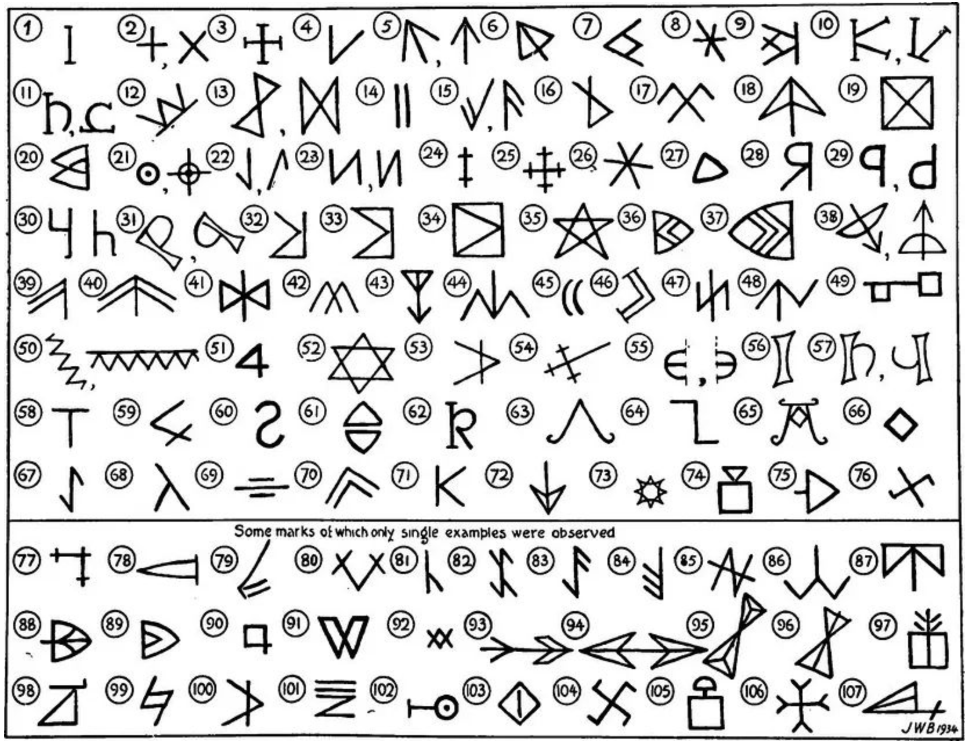
Refers to: 🚗, 🚙, 🚕, etc

Word: "House"

Refers to: 🏠, 🏡, 🏘️, etc.

Pictograms, or pictographs were used by the ancient Egyptians, Sumerians, and Chinese and became the basis for these cultures' written languages.

© Icon72/Dreamstime.com



https://jdreeves.medium.com/a-history-of-symbols-a93626435bd2

2000 BC, **mason's marks** have been found in ancient structures such as tombs.

Computer cannot understand "Text"

This is how computers "see" text in English.

xxxmf3102 mmmvv11v nnnffn333 Uj eellllo
eleee mnster vensi???? credur Baboi oi
cestnitze Coovoel2^ ekk; ldsllk lkdf vnnjfj?
Fgmflmllk mlfm kfre xnnn!

- **People have no trouble understanding language**
  - Common sense knowledge
  - Reasoning capacity
  - Experience
- **Computers have**
  - No common sense knowledge
  - No reasoning capacity

How do we enable systems to process and utilize language effectively?

We convert symbolic representations (e.g., words, signs, Braille, or speech audio) into formats that a computer can process and understand.

Natural Language Processing

NLP combines the study of language and computer science to understand how humans communicate.

Unlike programming languages, which follow strict rules, natural language is flexible and varies greatly.

To help machines process and make sense of this complexity, we need to represent language in a form that computers can understand—**using numbers.**

This **numerical representation** is the foundation that allows NLP to work effectively and solve real-world problems.

Representing Numbers in Computing?

Binary Foundation: Computers operate using binary (0s and 1s) at their core

Built-in Arithmetic: Arithmetic operations like (+ - * /) are built into their architecture.

Why Numbers Matter: Computational models rely on numerical data for processing. Numbers naturally support comparisons (e.g., <, >, ==), aiding logical operations.

Efficiency with Numbers: When it comes to numerical data, computers excel effortlessly!

Character Encoding

- ASCII
- Unicode and UTF Standards

Computers only understand **binary data**. To represents the characters as required by human languages, the concept of **character sets** was introduced.

In character sets each character in a human language is represented by a number.
In early computing English was the only language used. To represent, the characters used in English, ASCII character set was used. •

ASCII (American Standard Code for Information Interchange)

Character Encoding

- **ASCII**
- Unicode and UTF Standards

ASCII was developed in the 1960s to standardize character representation in computers. It uses 7-bits to encode to 128 characters, including English letters (uppercase and lowercase), digits, and basic symbols.

$b_7 b_6 b_5$ Bits					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ / Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

https://en.wikipedia.org/wiki/ASCII

Example

- Character: "O"
- ASCII Code (Decimal): 79
- ASCII Code (Binary): `01001111`

Character Encoding

- **ASCII**
- Unicode and UTF Standards

Limitations of ASCII

1. It has a limited number of characters
2. Inefficient for multilingual
3. There is no provision for modern symbols

Character Encoding

- ASCII
- **Unicode and UTF Standards**

Unicode expanded the scope of character encoding to include characters from virtually every written language, along with symbols, emojis, and more. **UTF-8, UTF-16**, and **UTF-32** are common encodings that implement Unicode.

- Supports over 140,000 characters across multiple languages.
- UTF-8 is backward-compatible with ASCII and highly efficient for English text.

Café = \x43 \x61 \x66 \xC3 \xA9

Character Encoding

- ASCII
- **Unicode and UTF Standards**

Limitations

1. UTF-8: Variable-length encoding can complicate indexing and processing.
2. UTF-16 and UTF-32: Fixed-length encodings use more memory for simple texts like English, increasing overhead.
3. Handling corrupted or incompatible encodings is a frequent challenge in text preprocessing for NLP.

One Hot Encoding

One hot encoding

A technique to represent **categorical data** as **binary vectors**.

Provinces	KP	Punjab	Sindh	Balochistan
KP	1	0	0	0
Punjab	0	1	0	0
Sindh	0	0	1	0
Balochistan	0	0	0	1

One hot encoding

Employee data

Employee ID	Gender	Remarks
10	M	Good
20	F	Nice
15	F	Good
25	M	Great
30	F	Nice

Encoded Employee data

Employee ID	Gender_F	Gender_M	Remarks_Good	Remarks_Great	Remarks_Nice
10	0	1	1	0	0
20	1	0	0	0	1
15	1	0	1	0	0
25	0	1	0	1	0
30	1	0	0	0	1

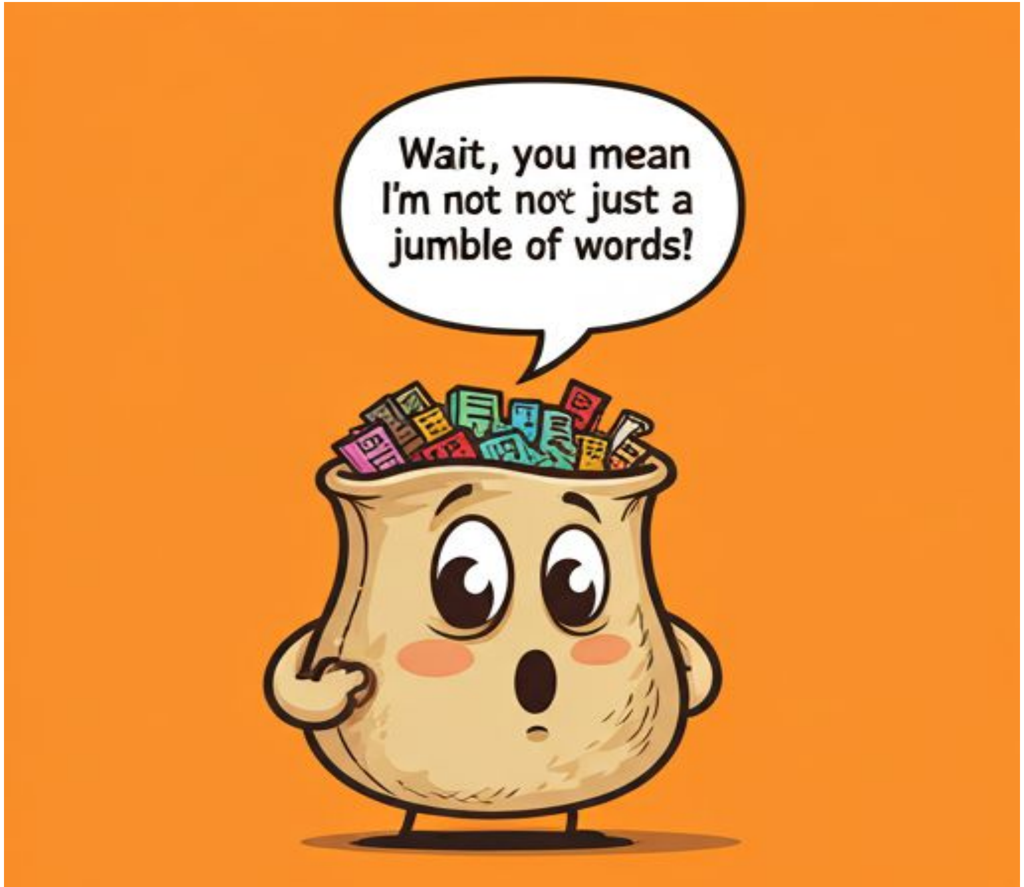
One hot encoding

Limitations:

1. High Dimensionality: Large vocabularies result in long, inefficient vectors.
2. Variable Length: Documents with different word counts create inconsistent vector sizes.
3. Sparsity in the encoded data.
4. No Semantic Context: Words lack context and meaning, limiting this method for advanced NLP.

Bag-of-Words

The bag-of-words model (BoW) is a model of text which uses an unordered collection (a "bag") of words.

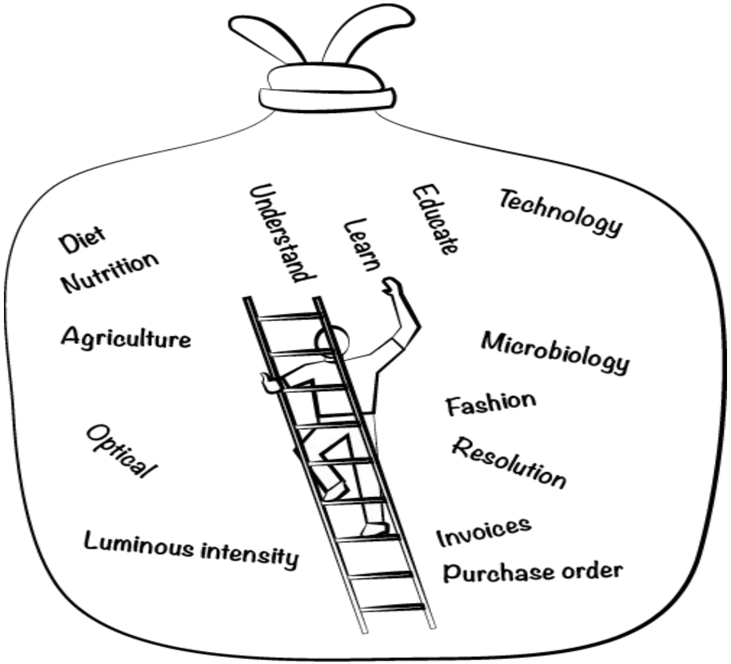


Bag-of-Words

It originated in the **1950s.**

It represents text data by treating each word as an independent feature, **counting its frequency**.

BoW disregards grammar, word order, and sentence structure, focusing on word presence/frequency.



Bag-of-Words

Vocabulary

Definition: Given a list of text, the vocabulary V would be the list of **unique words** from the list of text we have.

[review_1, review_2,...,
review_m]

I love the new features of the app.

.

.

.

I hate the new update.

V= [1, love, the, new, feature, of, app,...., hate update]

Bag-of-Words

Feature Extraction

To extract features from the vocabulary, check if every word from the vocabulary appears in the text.

- If it does, then assign a value of **1** to that feature otherwise assign a value of **0**.

I love the new features of the app.

[1, love, the, new, feature, of, app,...., hate update]

[1, 1, 1, 1, 1, 1, 1,. ..., 0, 0]

Bag-of-Words

Convert textual data into numerical features. It describes the occurrence of words within a document. It contains two things: vocabulary and frequency of words.

Dataset:

1. "I love programming"
2. "Programming is fun."
3. "I love learning new things"

Vocabulary Extraction

Combine all unique words from the dataset:

```
['love', 'programming', 'fun',

'Learning', 'new', 'things']
```

Bag of Words vector

Create a column vector of word counts. Each cell contains the frequency of the word in that sentence.

Sentence	love	programming	fun	learning	new	things
I love programming.	1	1	0	0	0	0
Programming is fun.	0	1	1	0	0	0
I love learning new things.	1	0	0	1	1	1

Bag-of-Words

Limitations of Bag of Words

1. Vocabulary size and length of vector would increase if new sentence is added.
2. The vectors contain many 0s, resulting in a sparse matrix
3. No semantic meaning or context is captured.
4. Ignores word order.

Term Frequency — Inverse Document Frequency (TF-IDF)

Term Frequency — Inverse Document Frequency (TF-IDF)

TF-IDF allows to score the importance of words in a document, based on how frequently they appear on multiple documents.

- **If the word appears frequently in a document - assign a high score to that word (TF)**
- **If the word appears in a lot of document - assign a low score to that word (IDF)**

Term Frequency — Inverse Document Frequency (TF-IDF)

TF-IDF allows to score the importance of words in a document, based on how frequently they appear on multiple documents.

- **TF**: **Measures how many times a word appears in the document.**
- **IDF**: **Represents how common the word is across the different documents.**

$$\textbf{tf-idf}_{(t,d)} = \textbf{tf}_{(t,d)} \times \textbf{idf}_{(t)}$$

t = term
d = document

$$\mathrm{tf}\,(\mathrm{t},\mathrm{d}) = \frac{Frequency\ of\ term\ t, in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

$$\mathrm{idf}\,(\mathrm{t}) = log\,\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$$

Term Frequency — Inverse Document Frequency (TF-IDF)

Step 1: Define the Corpus

The given corpus consists of three documents:

1. D1: "I love programming"
2. D2: "Programming is fun."
3. D3: "I love learning new things

Step 2: Tokenization

1. D1: {"I", "love", "programming"}
2. D2: {"programming", "is", "fun"}
3. D3: {"I", "love", "learning", "new", "things"}

Vocabulary: Extract (unique) words

V=["I","love","programming","is","fun","learning" ,"new","things"]

Step 3: Compute Term Frequency (TF)

Calculate the frequency of each term in the sentence divided by the total number of terms in that sentence.

Term	D1 (I love programming)	D2 (Programming is fun)	D3 (I love learning new things)
I	1/3 = 0.33	0	1/5 = 0.2
love	1/3 = 0.33	0	1/5 = 0.2
programming	1/3 = 0.33	1/3 = 0.33	0
is	0	1/3 = 0.33	0
fun	0	1/3 = 0.33	0
learning	0	0	1/5 = 0.2
new	0	0	1/5 = 0.2
things	0	0	1/5 = 0.2

Term Frequency — Inverse Document Frequency (TF-IDF)

Document Frequency (DF)

Count the number of sentences in which each term appears.

Term	DF
I	2
love	2
programming	2
is	1
fun	1
learning	1
new	1
things	1

Term Frequency — Inverse Document Frequency (TF-IDF)

Step 4: Compute Inverse Document Frequency (IDF)

IDF is computed using: $$IDF = \log\left(\frac{N}{DF}\right)$$

where:

- **N** = total number of documents (**3**)
- **DF** = number of documents containing the term

Term	Document Count (DF)	IDF $\log(3/DF)$
I	2	$\log(3/2) = 0.176$
love	2	$\log(3/2) = 0.176$
programming	2	$\log(3/2) = 0.176$
is	1	$\log(3/1) = 0.477$
fun	1	$\log(3/1) = 0.477$
learning	1	$\log(3/1) = 0.477$
new	1	$\log(3/1) = 0.477$
things	1	$\log(3/1) = 0.477$

Term Frequency — Inverse Document Frequency (TF-IDF)

For each term, multiply its TF by its IDF.

Term	TF-IDF in D1	TF-IDF in D2	TF-IDF in D3
I	$0.33 \times 0.176 = 0.058$	0	$0.2 \times 0.176 = 0.035$
love	$0.33 \times 0.176 = 0.058$	0	$0.2 \times 0.176 = 0.035$
programming	$0.33 \times 0.176 = 0.058$	$0.33 \times 0.176 = 0.058$	0
is	0	$0.33 \times 0.477 = 0.158$	0
fun	0	$0.33 \times 0.477 = 0.158$	0
learning	0	0	$0.2 \times 0.477 = 0.095$
new	0	0	$0.2 \times 0.477 = 0.095$
things	0	0	$0.2 \times 0.477 = 0.095$

Term Frequency — Inverse Document Frequency (TF-IDF)

This matrix represents the importance of each term in each sentence based on TF-IDF.

Term	D1	D2	D3
I	0.058	0	0.035
love	0.058	0	0.035
programming	0.058	0.058	0
is	0	0.158	0
fun	0	0.158	0
learning	0	0	0.095
new	0	0	0.095
things	0	0	0.095

Term Frequency — Inverse Document Frequency (TF-IDF)

Limitations of TF-IDF

1. Lacks contextual understanding and positional information.
2. Unable to capture semantic relationships between words.

Conclusion

1. **Progression of Text Representation**
   - **Character Encoding:** Foundation of representing text in numeric form.
   - **Bag-of-Words & TF-IDF:** Simple yet effective methods for basic text analysis, though limited in capturing context.

Thank You