LLM (Large Language Model)

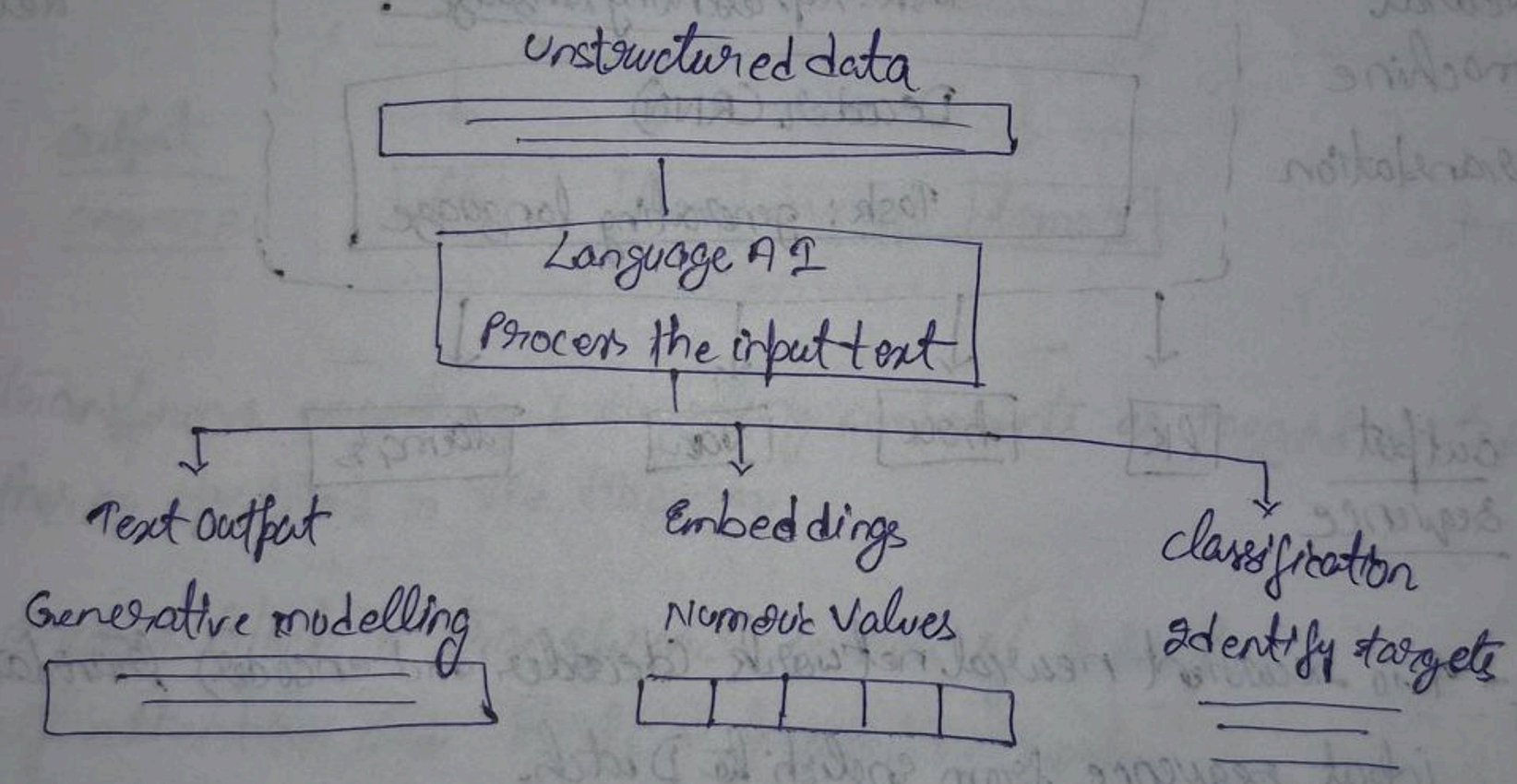
- Deep Learning is a subset of AI.
- Language AI is used to process and generate human language.

* After 2018, the AI models uses transformer (convert one language to another).

* Transformer can be Decoder, encoder or both encoder and decoder.

LLM – The deep learning based model used transformer to train on the natural language like text or image.

- The language AI is the subfield of AI used to understand process and generate the human language.



Bag of words – A method for representing unstructured text.

Steps 1> Tokenization – Splitting the ~~to~~ sentences into individual words or subwords (tokens) is known as tokenization ~~(token)~~.

Input
This is a cute dog
↓ split i/p by white space
"that" "is" "a" "cute" "dog"
↓ create vocabulary
[that] [is] [a] [cute] [dog] [my] [cat]

Input
my cat is cute
↓
"my" "cat" "is" "cute"

Input

My cat is cute

↓

[my] [cat] [is] [cute]

↓

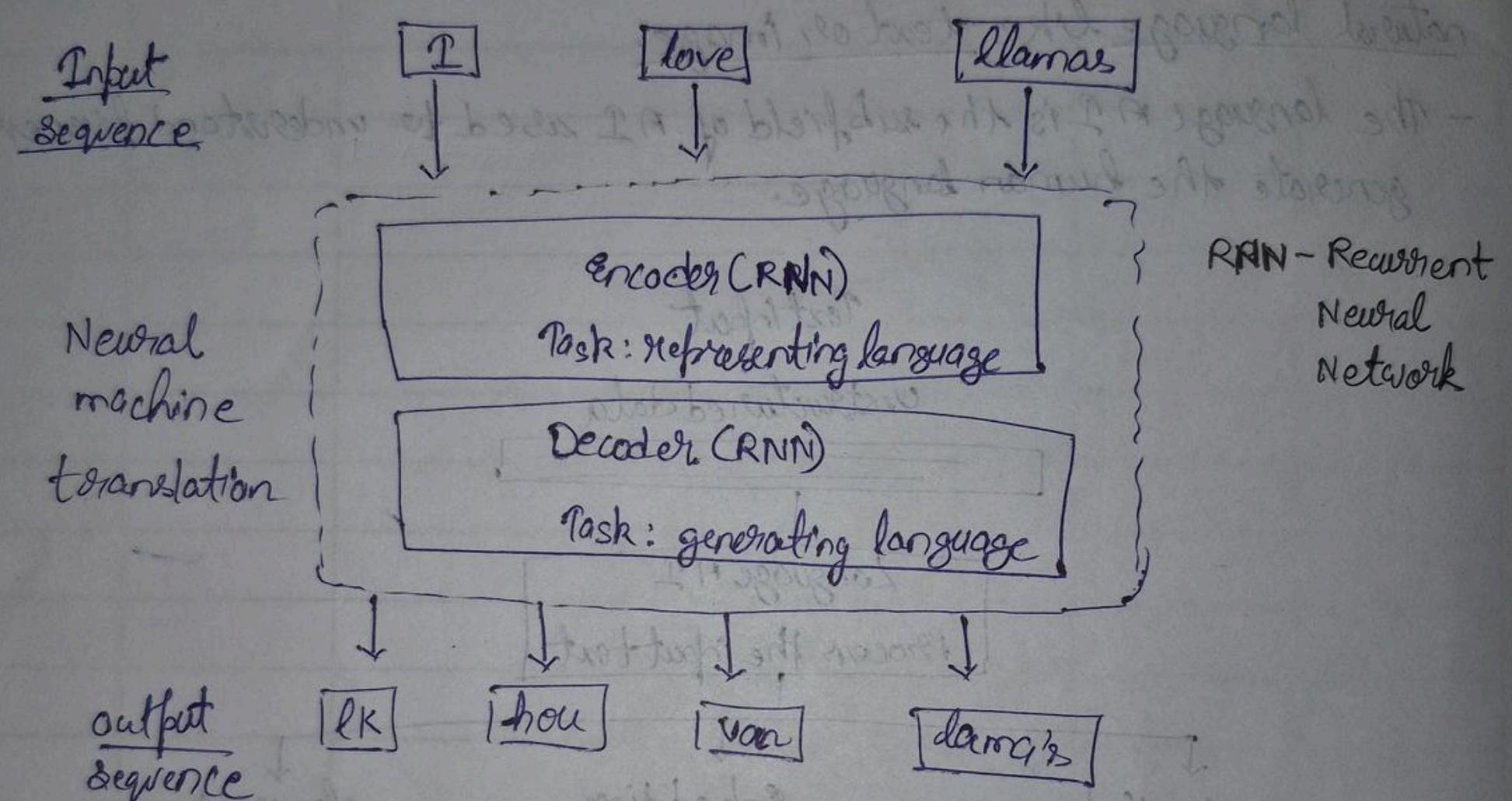
that	is	a	cute	dog	my	cat
0	1	0	1	0	1	1

← vector representation

Bag of words

count individual words

Convert the text from english to dutch language -



RNN - Recurrent Neural Network

Fig - Two recurrent neural network (decoder and encoder) translating an input sequence from english to Dutch.

Embedding - process of converting tokens into vector forms which try to capture its meaning.

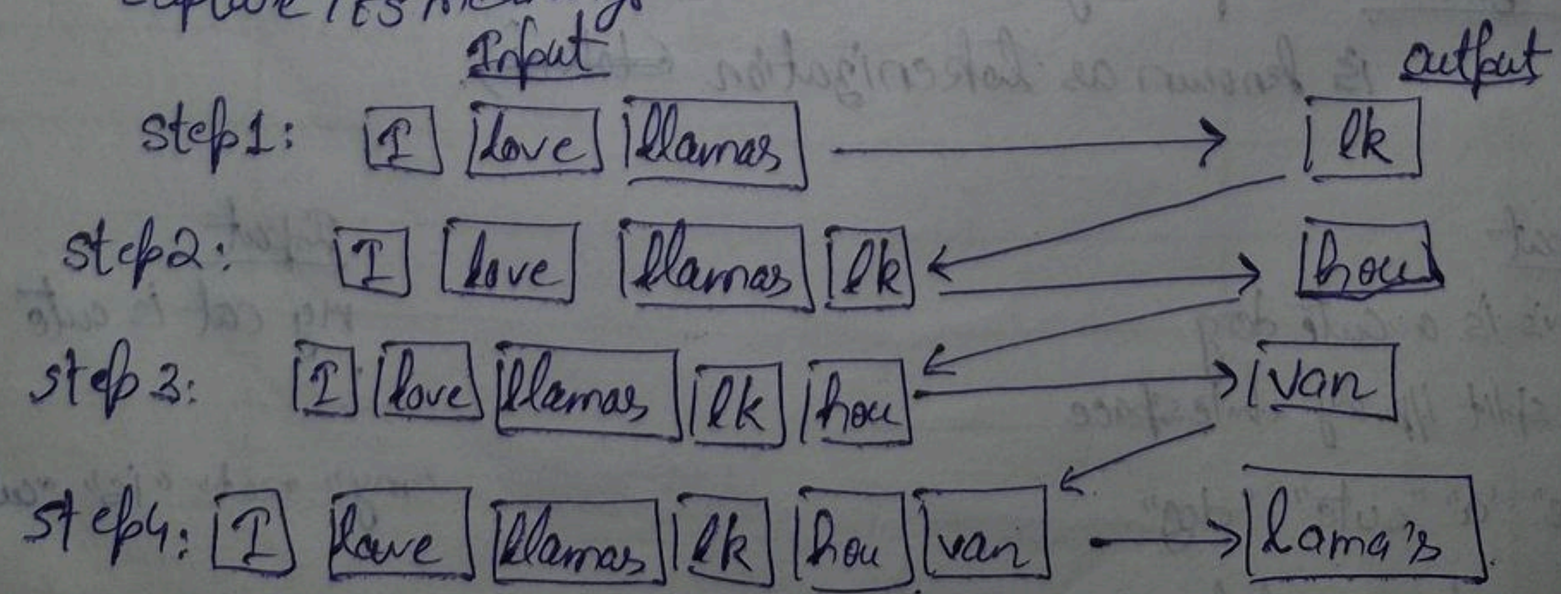


Fig 1.12 → Each previous o/p token is used as i/p to generate next token

Embedding - allows us to measure the semantic similarities b/w 2 words.

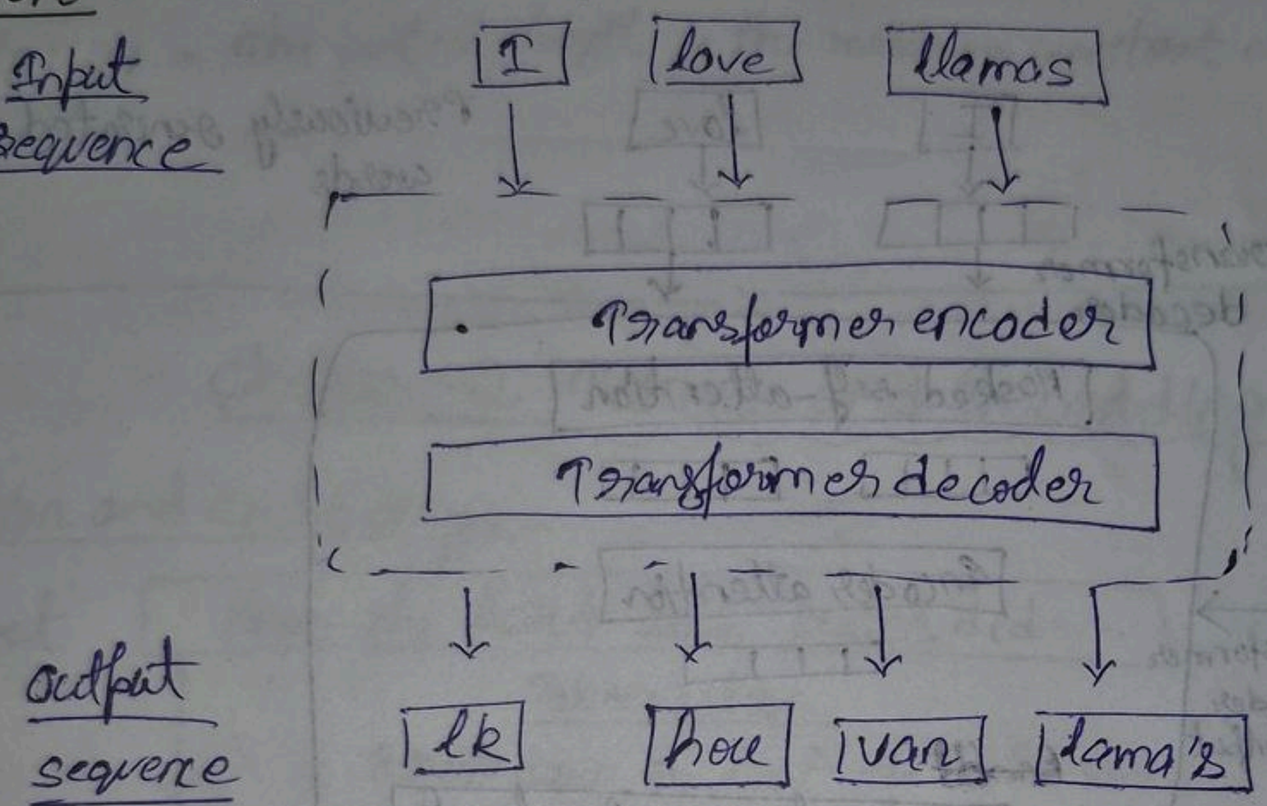
- Using various ~~ma~~ metrics we can judge how close one word to another.

Types of embedding

i) Document embedding ii) Sentance embedding

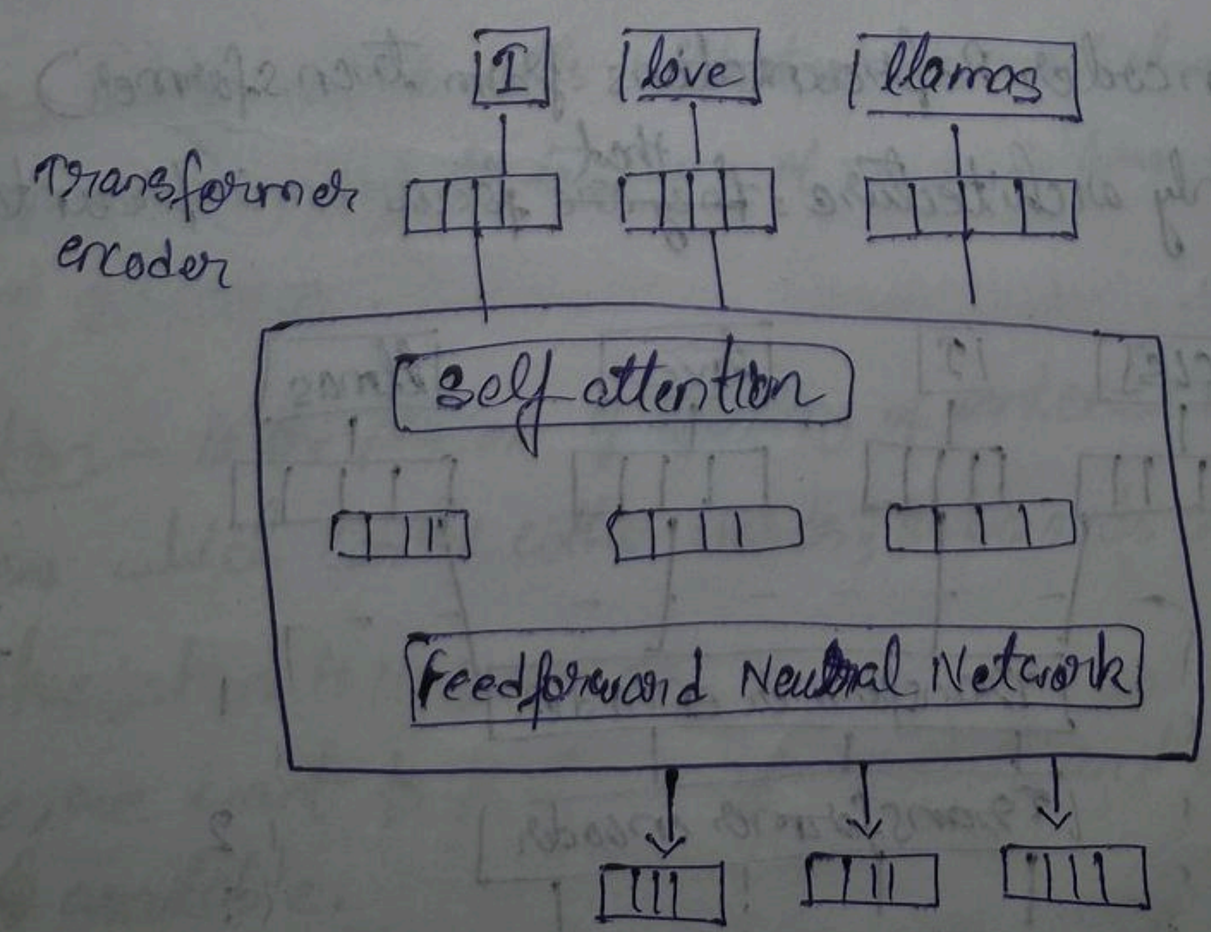
iii) Word embedding iv) Token embedding.

Attention - "Attention is all you need". It allows a model to focus on parts of the input sequence that are relevant to one another and amplify their signals.



In the transformer encoding & decoding components as tagged on top of eachother as specified in the diagram.

- The encoder block in transformer consists of 2 parts -
  i) Self attention ~~and~~ ii) Feed forward network



- The decoder block in transformer ~~consts of~~ consists of
  i) Masked self-attention ii) Encoder attention iii) Encoder neural network.

9/9

- BERT is an encoder only model (Introduced in 2018).
- GPT 1 is an decoder only model. Trained with 7000 books and has 117 million parameters. (Proposed in 2018).

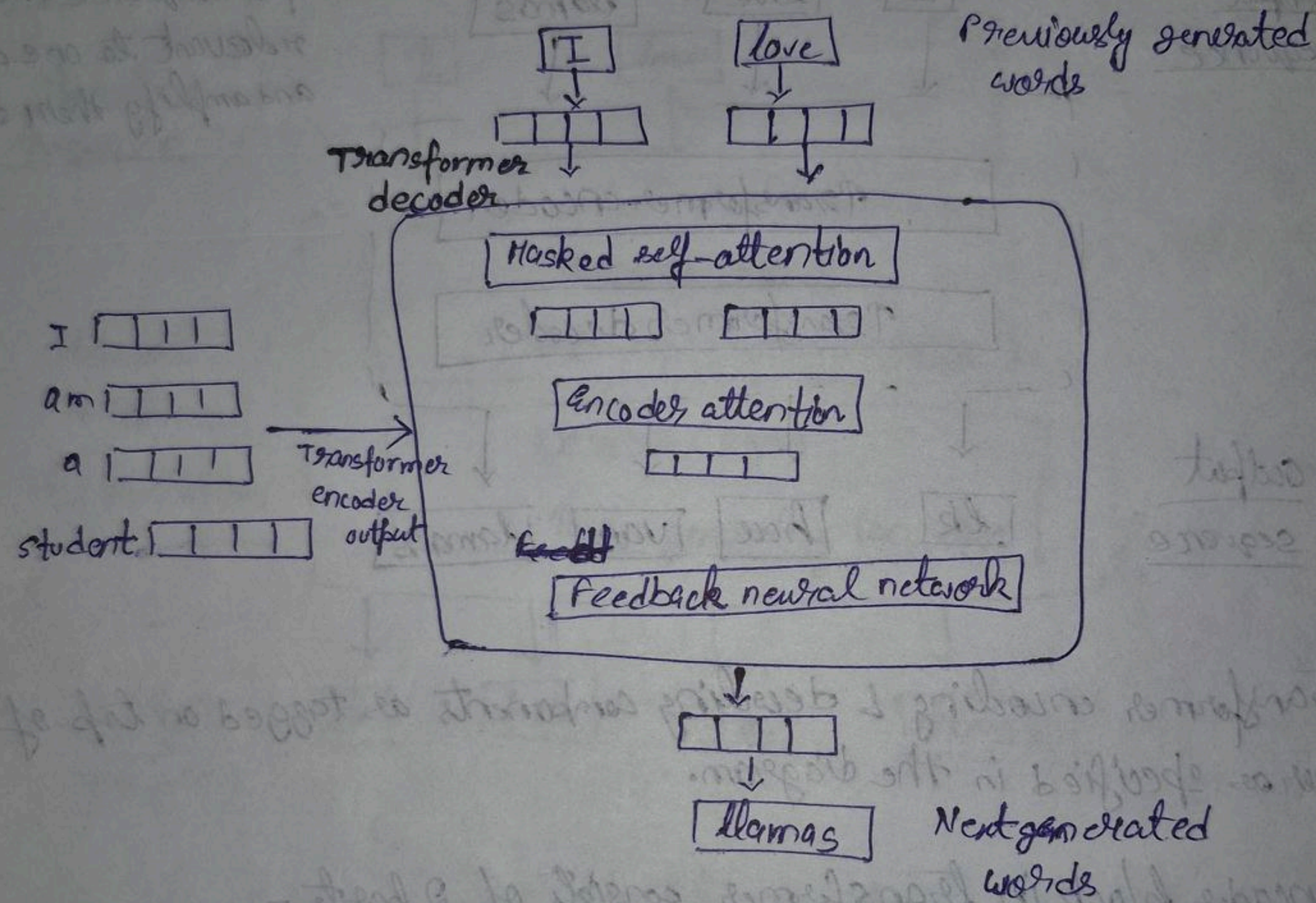


Fig 1.9 → The decoder has an additional attention layer that attends to the o/p of the encoder.

BERT - (Bidirectional encoder Representations from transformer)

- It is an encoder only architecture, ~~they are~~ that focus on representing language.

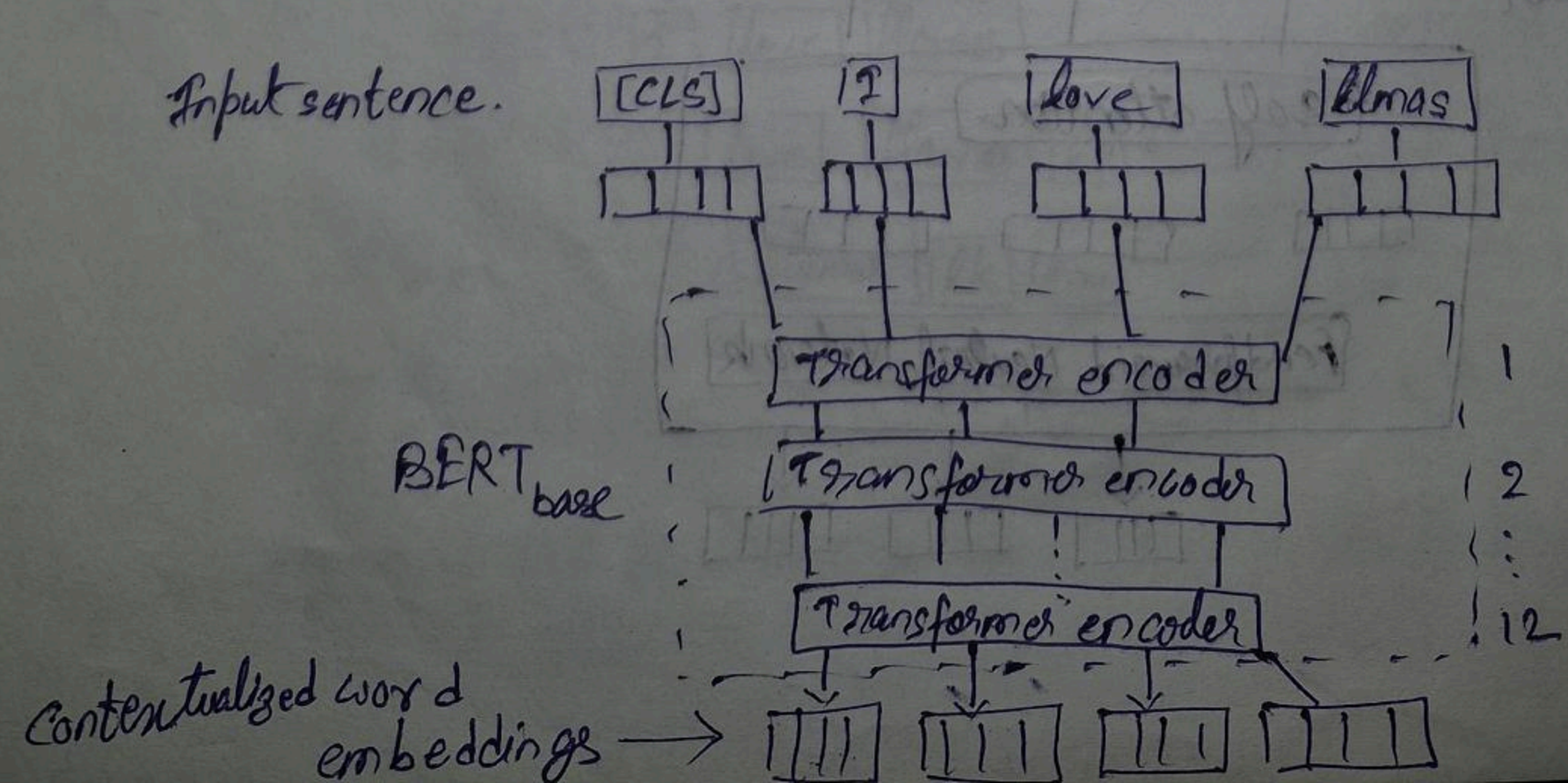
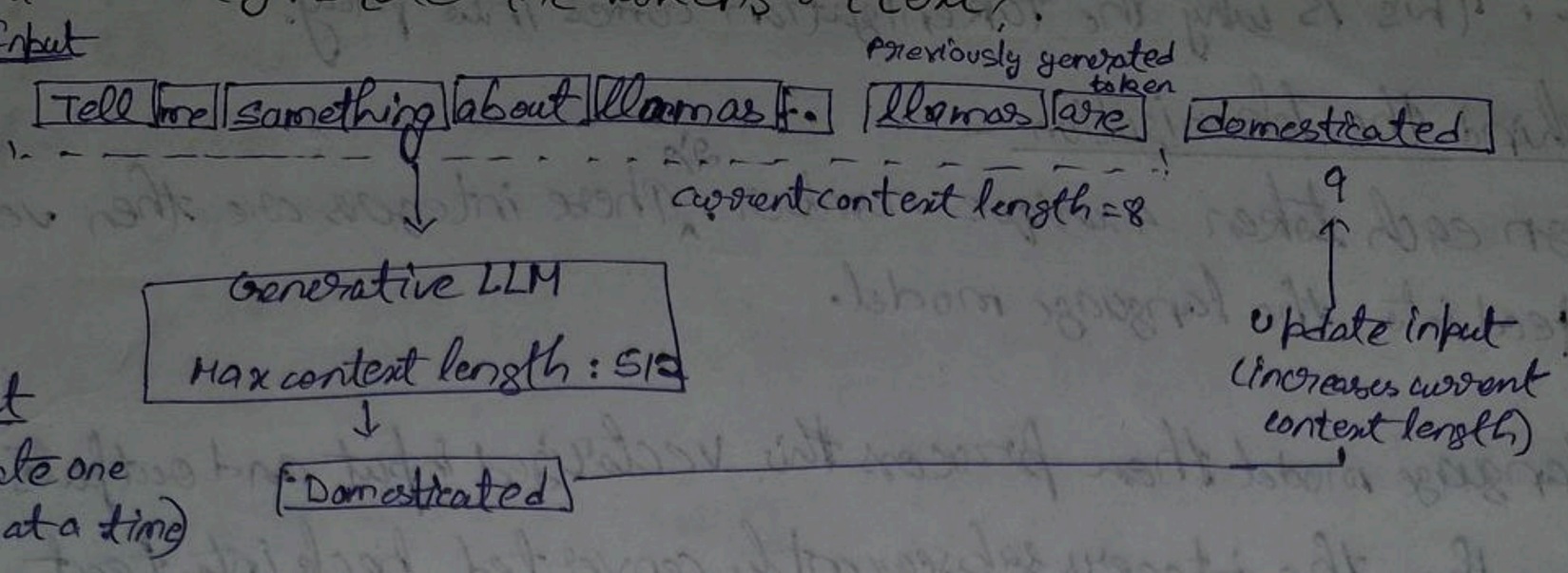


Fig 1.21 The architecture of bert with 12 encoders.

Q> How the LLM generate the tokens or text?

Sol: Input



Output
(generate one token at a time)

Fig 1.27 → The context length is the maximum context an LLM can handle

---

9/9/25

Chapter-2 : Tokens and Embeddings

Tokenization and embeddings -

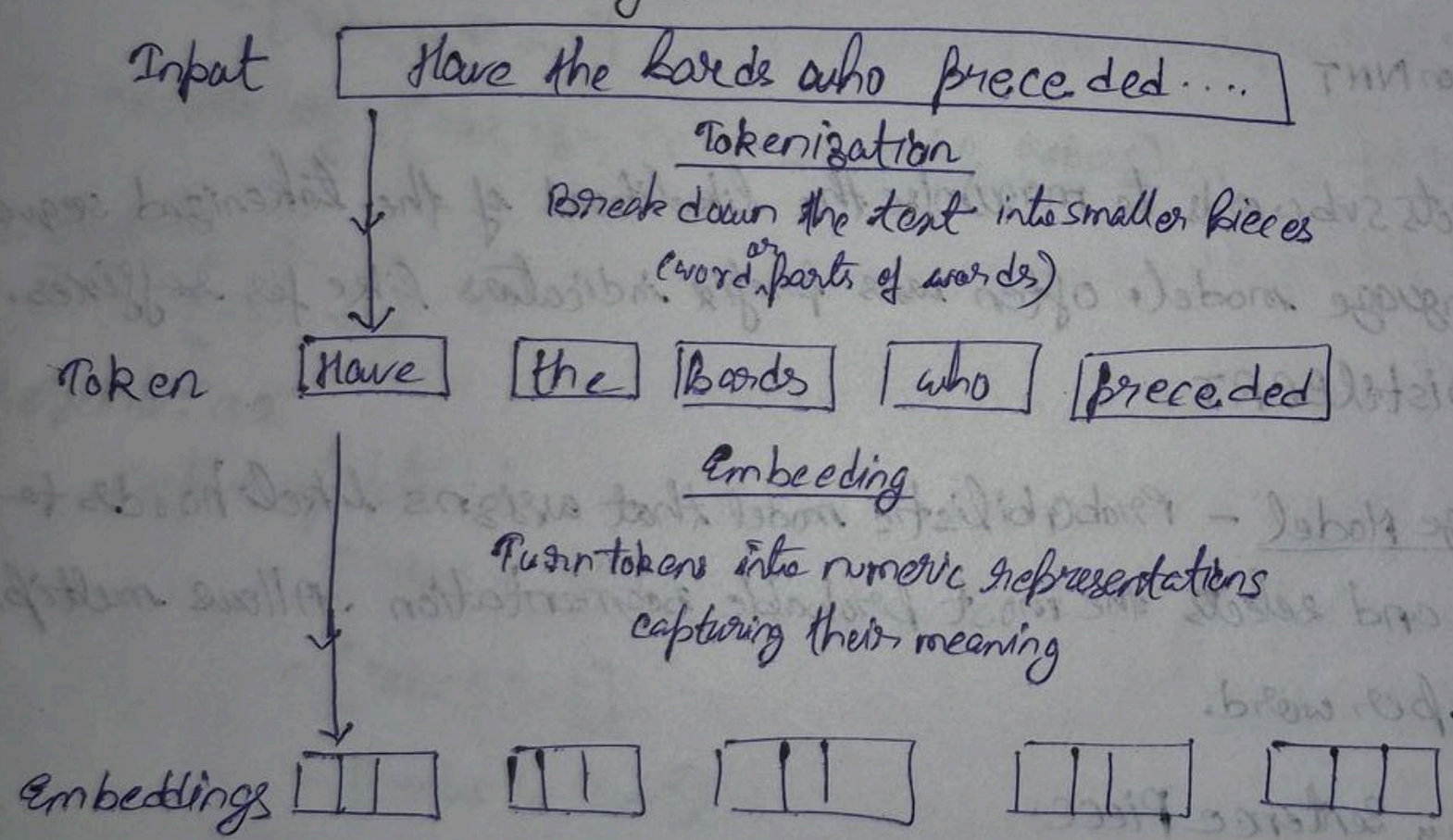


Fig 2.1 → Language models deals with text in small chunks called tokens. For the language model to compute language, it needs to turn tokens into numeric representation called embeddings.

Tokenization - is the process of splitting of sentence into small parts or sequence of tokens which can be either words, subwords or characters.

Q> Why tokenization is required?

Suppose, we want to pretrain our model and we have 10 GB of language data available.

However, the data is in text form but the language model process the data in numbers. Therefore, we need to convert the text into the

numbers. This is why the tokenization comes into play.

Steps behind the tokenization -

1) Assign each token a unique number. 2) These integers are then vectorized and feed into the language model.

3) The language model then process this vectorized input and outputs the integer ~~number~~ again, then the integers subsequently converted back into text. That is the generated output.

Imp
There are 3 types of tokenization -

i) Byte Pair encoding - Greedy merging of frequently adjacent character pairs to form subwords. Vocabulary is built by iteratively combining the most common pairs.

Eg: GPT, OpenNMT

ii) Word Piece - Selects subwords to maximize the likelihood of the tokenized sequence under ~~piece~~ a language model. Often uses prefix indicators like for suffixes.

Eg: BERT, DistilBERT

iii) Unigram Language Model - Probabilistic model that assigns likelihoods to subword units and selects the most probable segmentation. Allows multiple segmentations per word.

Eg: XLNet, T5, Sentence Piece.

Eg: Byte pair coding example: aa abc abc.

Sol: Initial Corpus: ["aa", "abc", "abc"]

Step 1: Preprocessing

"aa": ["a", "a", "_"]

"abc": ["a", "b", "c", "_"]

"abc": ["a", "b", "c", "_"]

Step 2: Iterative merging

1. Iteration 1:

- Most frequent pair : ("a", "b")
- Merge into : "ab"
- corpus updates :
  - ["a", "a", "_"]
  - ["ab", "c", "_"]
  - ["ab", "c", "_"]
- vocabulary : ["a", "b", "c", "_", "ab"]

2. <u>Iteration 2</u>

("ab", "c")

- Merge into: "abc"
  - Corpus Update :
    - ["a", "a", "_"]
    - ["abc", "_"]
    - ["abc", "_"]
  - Vocabulary : ["a", "b", "c", "_", "ab", "abc"]

3. <u>Iteration 3</u>

("a", "a")

- Merge into: "aa"
  - corpus update :
    - ["aa", "_"]
    - ["abc", "_"]
    - ["abc", "_"]
  - vocabulary : ["a", "b", "c", "_", "ab", "abc", "aa"]

<u>Final Tokenized Corpus</u>

- ["aa", "_"]
- ["abc", "_"]
- ["abc", "_"]

Final vocabulary : ["a", "b", "c", "_", "ab", "abc", "aa"]

Word Peice - The word peice tokenization was developed by google that the primary goal is to handle the unseen ~~words~~ words more effectively than traditional word level tokenization.

Suppose the vocabulary is given as [UNK, a, aa, abc, ##b, ##c]

Corpus ["aa", "abc", "abc"]

case 1: abc and ab are in vocabulary

- "aa":
  - Longest prefix match : "aa" ∈ vocab
  - Tokenized : ["aa"]
- "abc":
  - Longest prefix match : "abc" ∈ vocab
  - Tokenized : ["abc"]
- Final tokenized corpus : [["aa"], ["abc"], ["abc"]]

case 2: abc and ab are not in vocabulary

Step by step tokenization of "abc":

1. Start at position 0 : Try longest match from vocab
   - Try "abc" → not in vocab
   - Try "ab" → not in vocab
   - Try "a" → match
   - Emit: "a"
2. Position 1 : Remaining string "bc"
   - Try "##bc" → not in vocab
   - Try "##b" → match
   - Emit "b"
3. Position 2 : Remaining string is "c"
   - Try "##c" → match
   - Emit : "c"

Final Tokenization : ["a", "##b", "##c"]

Note : Word Piece doesn't back track or look ahead to match "##bc" unless it is explicitly in the vocabulary.

Sentence Peice - is language ~~independent~~ independent subword tokenization method developed by google which is designed to train directly on raw text without requiring the pre-tokenization.

- unlike BPE and wordpeice, the sentence peice treats the raw string of unicode characters for tokenization.

- The sentence peice use unigram language model that assigns a probability 'Pi' to each subword 'ti' in the vocabulary to find the total cost.

$$\boxed{Cost(t_i) = -\ln P_i}$$

Eg: Suppose sentence is "abc". Use sentence peice algorithm to tokenize the "abc".

Token	Probability P	cost = $-\ln P_i$
a	0.20	1.61
b	0.20	1.61
c	0.20	1.61
ab	0.15	1.90
bc	0.15	1.90
abc	0.10	2.30

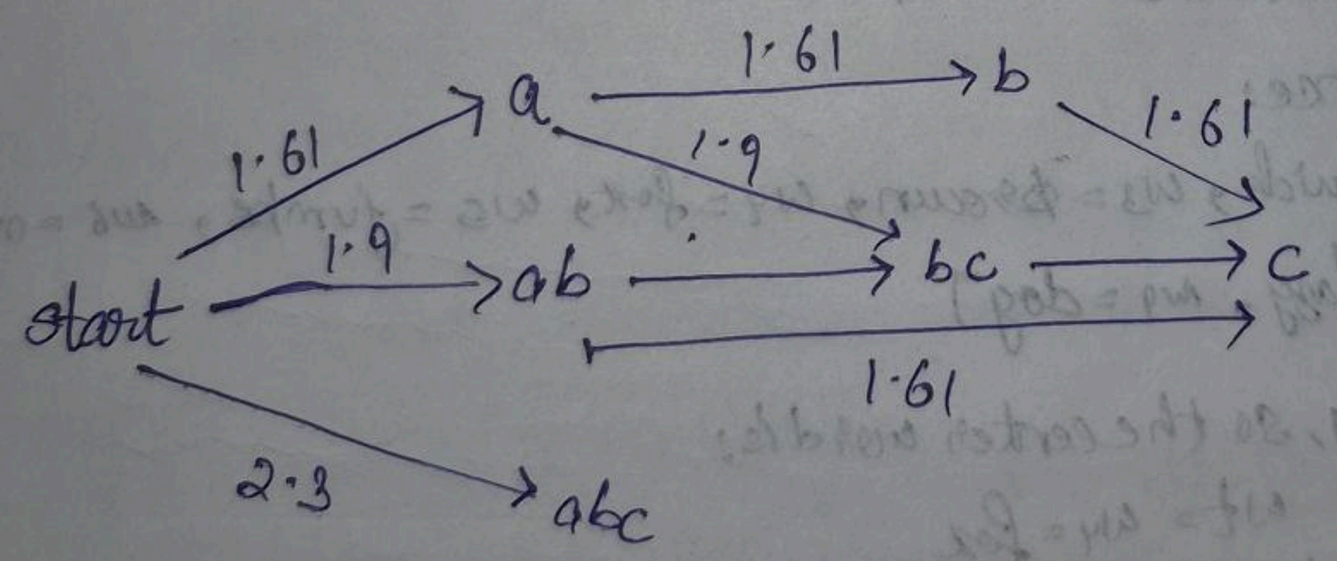


Fig 4: The cost lattice for "abc"

Segmentation: ["a", "b", "c"]
Cost: 1.61 + 1.61 + 1.61 = 4.83

Segmentation: ["abc"]
Cost: 2.30

Segmentation: ["ab", "c"]
Cost: 1.90 + 1.61 = 3.51

Segmentation: ["a", "bc"]
Cost: 1.61 + 1.90 = 3.51

optimal path via Viterbi Decoding

The viterbi algorithm selects the path with minimum total cost.

In this case E["abc"] = 2.30

Best segmentation ["abc"], so total cost = 2.30

Embeeding - It will find the semantic similarity b/w 2 words using various matrices that can judge how close one word to another.

The embeeding techniques has been used to convert the each token in input sequence into the numerical forms called as Input embedding.

- Each token is mapped to a vector using an embedding matrics.

$$\boxed{x = \text{Embedding}(t)}$$

t = token id

x = resulting vector

Word2Vec - It is the most embedding technique uses a sliding window to generate the semantic and syntactic relationship.

Eg: Consider the sentence: "The quick brown fox jumps over the lazy dog."

Let's define the context window and center word

- Tokenized in sequence:

  [w_1 = The, w_2 = quick, w_3 = brown, w_4 = fox, w_5 = jumps, w_6 = over, w_7 = the, w_8 = lazy, w_9 = dog]

- choose position t = 4, so the center word is:

  $w_t = w_4$ = fox

- Let, the context window size be c = 2. Then the context words are

  $w_{t-2} = w_2$ = quick, $w_{t-1} = w_3$ = brown, w_{t+1} = jumps, $w_{t+2} = w_6$ = over.

CBOW objective - Use the context words quick, brown, jumps and over to predict the center word fox.

Embeeding and prediction steps -

1. One hot encoding - Each context word w_i is represented as a one-hot vector

$x_i \in R^V$, where V is the vocabulary size.

2. <u>Embedding lookup</u>: Each one-hot vector is multiplied by the input embedding matrix $W \in R^{V \times N}$ to obtain its dense embedding

$$e_i = W^T x_i$$

3. <u>Context aggregation</u>: The embedding of four context words are averaged:

$$h = \frac{1}{4}(e_{quick} + e_{brown} + e_{jumps} + e_{over})$$

4. <u>Prediction</u>: The hidden vector $h \in R^N$ is passed through the output matrix $U \in R^{N \times V}$ to produce scores:

$$z = U^T h$$

Then softmax is applied: $P(w_t = fox \mid w_{context}) = softmax(z)$

5. <u>Training</u>: The model is trained to maximize the probability of the correct center word fox. The loss is:

$$\boxed{L = -\log P(fox \mid quick, brown, jumps, over)}$$

<u>Result</u>: After training, the rows of W become meaningful word embeddings. Word that appear in similar context (like fox and dog) will have similar embeddings.

16/9

Chapter 3 - Looking Inside of Large Language Models

Autoregressive model - In machine learning, the models that consume their early predictions to make later predictions means the model's first generated ~~the~~ ~~tod~~ token is used to generate the second token are known as "auto regressive models".

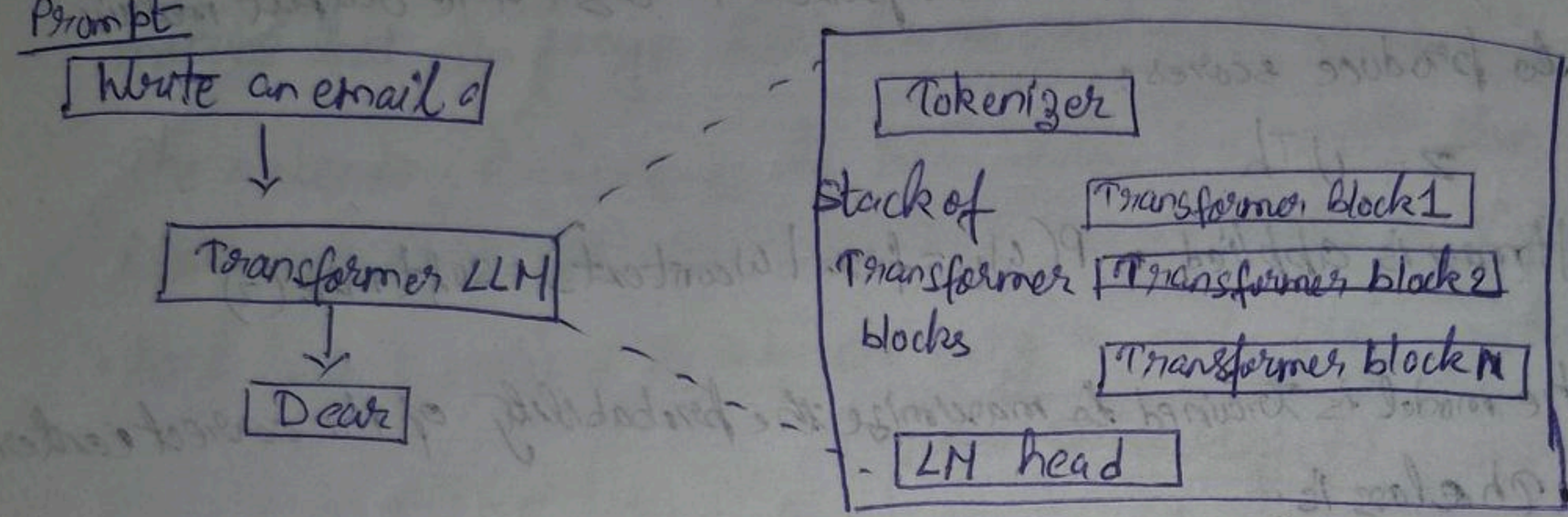


Fig 3.4 - A transformer LLM is made up of a tokenizer, a stack of transformer blocks, and a language modelling head.

Decoding strategy - ~~In the de~~ The method of choosing a single token from the probability distribution means the probability score for each token in the vocabulary is called as "Decoding strategy".

- Choosing the highest scoring token every time is called as "Greedy Decoding".

Eg:

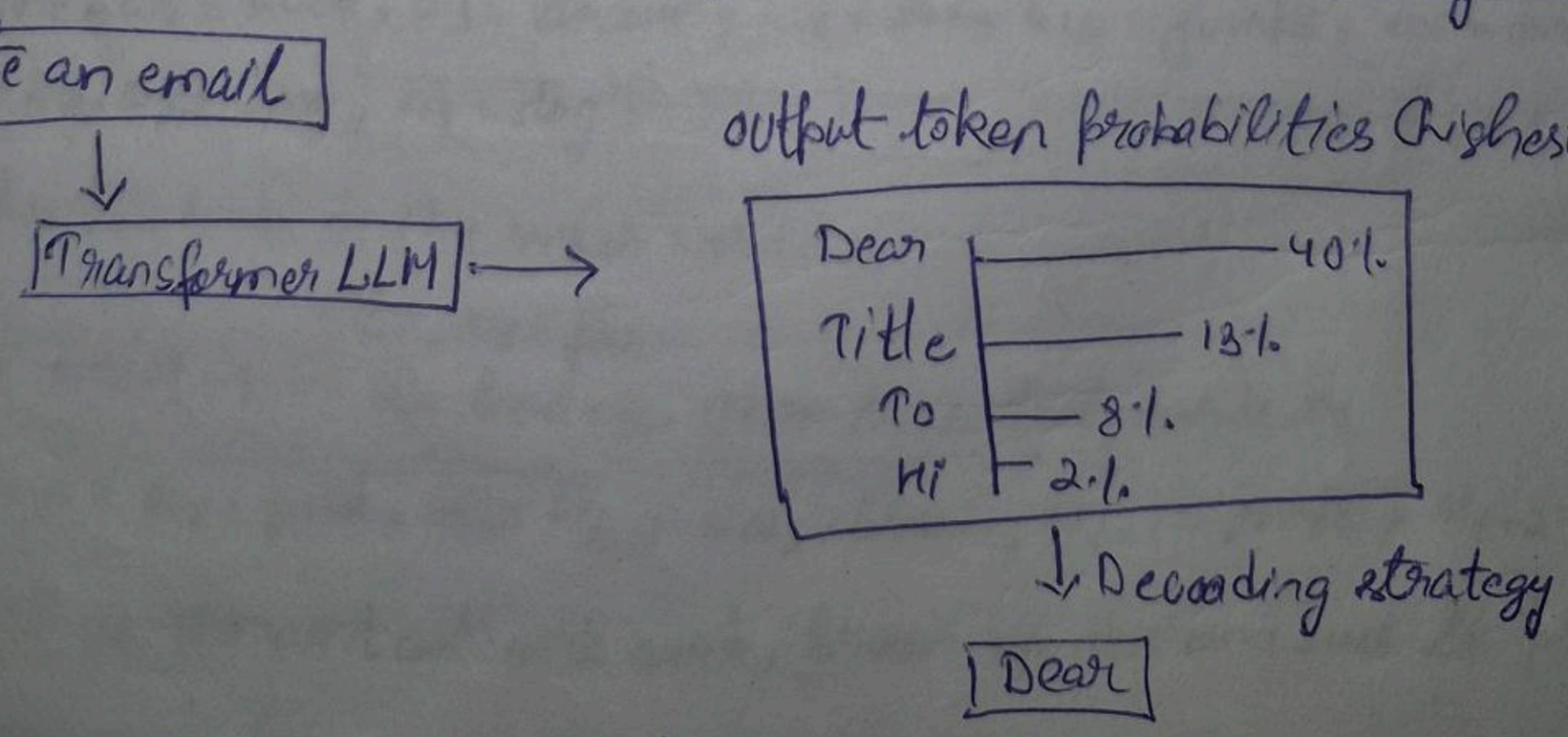


Fig 3.7 - The tokens with the highest probability after the model's forward pass. Our decoding strategy decides which of the tokens to output by sampling based on the probabilities.

Q) How the Transformer LLM works?

Sol: i) First, the tokenizer in transformer LLM creates or converts the tokens which in further goes for token embedding means the vector representation.

ii) Each generated token, the process flows once through each of the transformer Blocks in the stack in order.

iii) ~~In fur~~ That generated output goes to LM_{head} which contains the probability distribution for the next token.

Imp - For text generation only the output result of the last stream is used to predict the next token that output vector is the only input into the LM head as it calculates the probabilities of the next token.

Keys and Values cache memory - When to generate the 2nd token we simply append the output token to the input and do the forward pass to the model. However, if we give the model the ability to ~~catch~~ cache the result of the previous calculation, we no longer need to repeat the calculation of the previous strings. This time the only needed calculation is for the last string. This process is called as "Key Value cache" which speeds of the generation process significantly.

- This is known as the "attention mechanism".

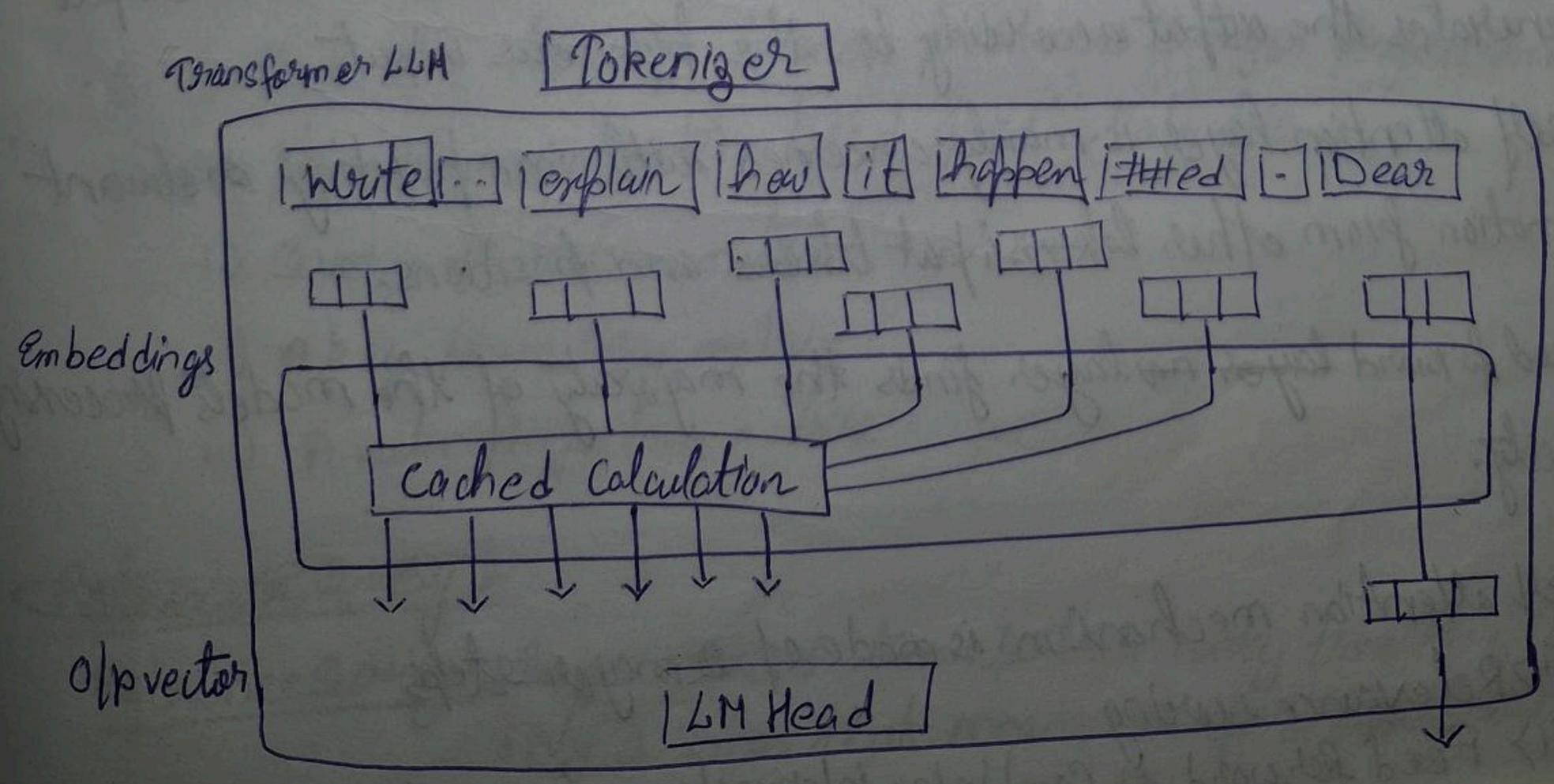


Fig 3.10 - When generating text, its important to ~~cache~~ cache the computational results of previous tokens instead of repeating the same calculation over and over again

<u>Transformer block</u> - The transformer LLMs are composed of series of transformer blocks and each block process its input then passes the results of its processing to the next block.

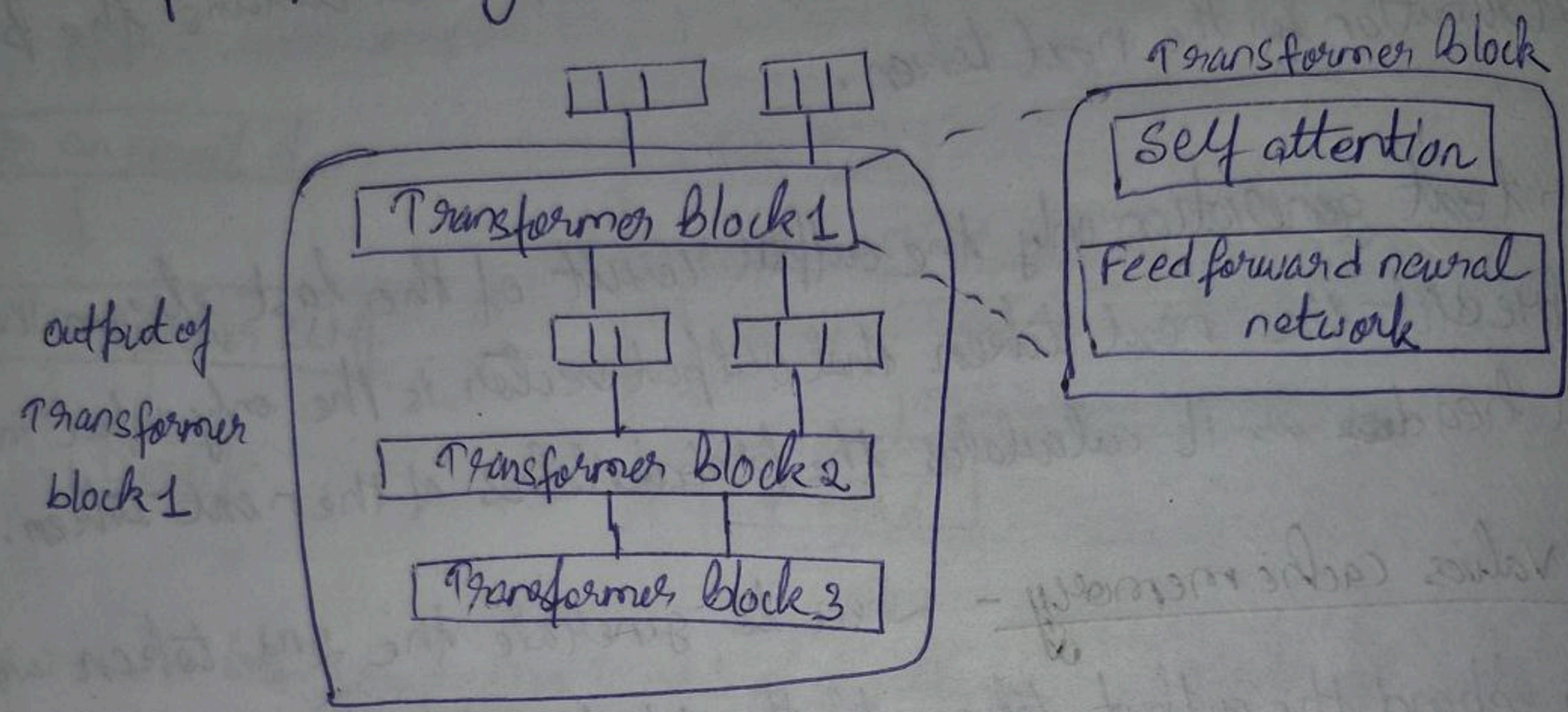


Fig 3.12. A transformer ~~block~~ block is made up of a self-attention layer and a ~~feedback~~ feedforward neural network.

<u>The shawshank</u> - is a ~~movie~~ movie. When we search ~~the~~ "The shawshank" it will automatically complete the whole movie name "The Shawshank ~~Redemp~~ Redemption"

19/9

<u>Attention Layer</u> - Attention is a technique that used to process the input and generates the output according to the previous inputs.

- The self attention layer is mainly concerned with incorporating relevant information from other ~~tokens~~ input tokens and positions.
- The feed forward ~~layer~~ ~~no~~ layer finds the majority of the model's processing capacity.
- The self attention mechanism is made of 2 major steps
  i) Relevance scoring
  ii) Feed forward or Combining information

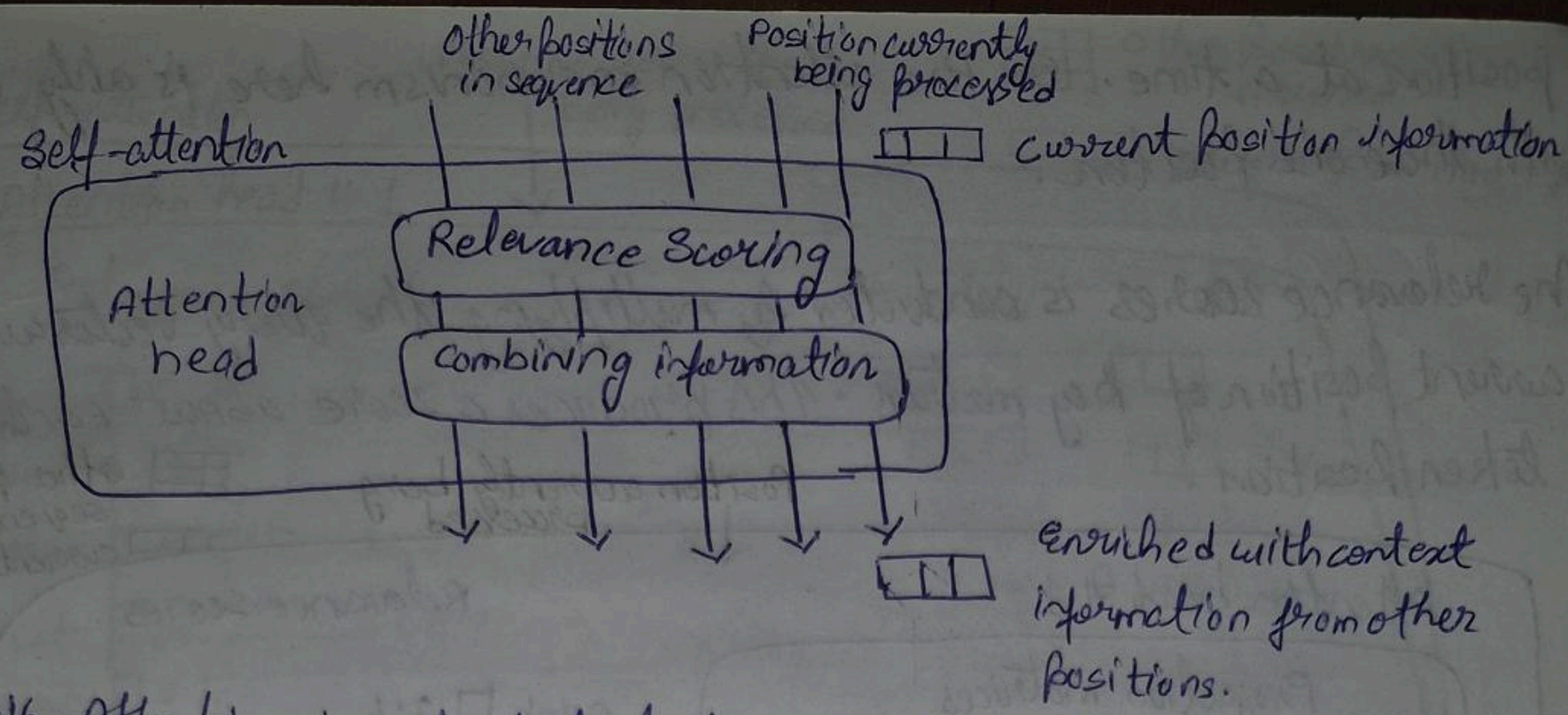


Fig 3.16 · Attention is made ~~up~~ up of 2 major steps: relevance scoring for each position, then a step where we combine the information based on those scores

<u>Attention head</u> - contains all the parallel applications of attention into the self attention layer to give the transformer more extensive attention capability.

Q> How attention is calculated?

- The attention layer in generative LLM used for processing the attention for a single position.
- The inputs to the layer are
  i> Vector representation of the <u>current position information</u>.
  ii> Vector representation of <u>previous tokens</u>.
- The goal here is to produce a new representation of current position. that incorporates ~~relativ~~ relevant information from the previous tokens.
- In further the training process produces 3 major projection matrix to interrupt with the attention head for the calculations.
  i) Query projection matrix
  ii) A key projection matrix
  iii) A value projection matrix.

~~Relevance Query~~

<u>Relevance Scoring</u> - In generated transformer every time it will generate only 1 token that means the processing will be always

1 position at a time. So, the attention mechanism here is only concerned with these one position.

- The relevance scores is conducted by multiplying the query vector with the current position of key matrix. This produces a score about each previous token position.

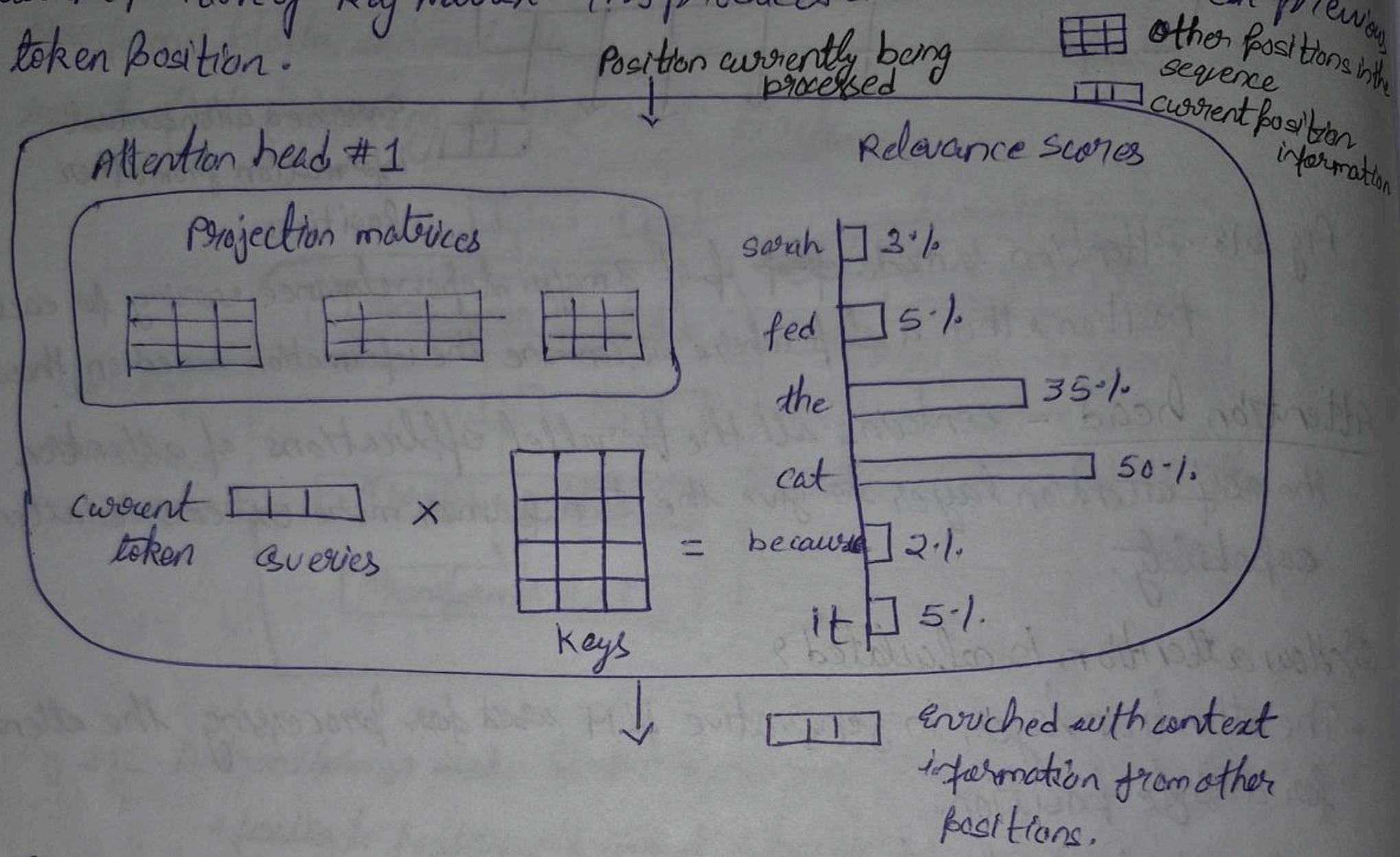


Fig 3.20 → Scoring the ~~relevance~~ relevance of previous tokens is accomplished by multiplying the query associated with the current position with the keys matrix.

- Passing these scores by a ~~submorge fun~~ ~~summer~~ softmax function normalizes these ~~sco~~ scores by ~~so~~ sum of ~~two~~ 1 to 1.

<u>Combining information</u> - After performing the relevance scores the attention mechanism ~~multiply with~~ multiplied the scores with the value vectors for each token that means.

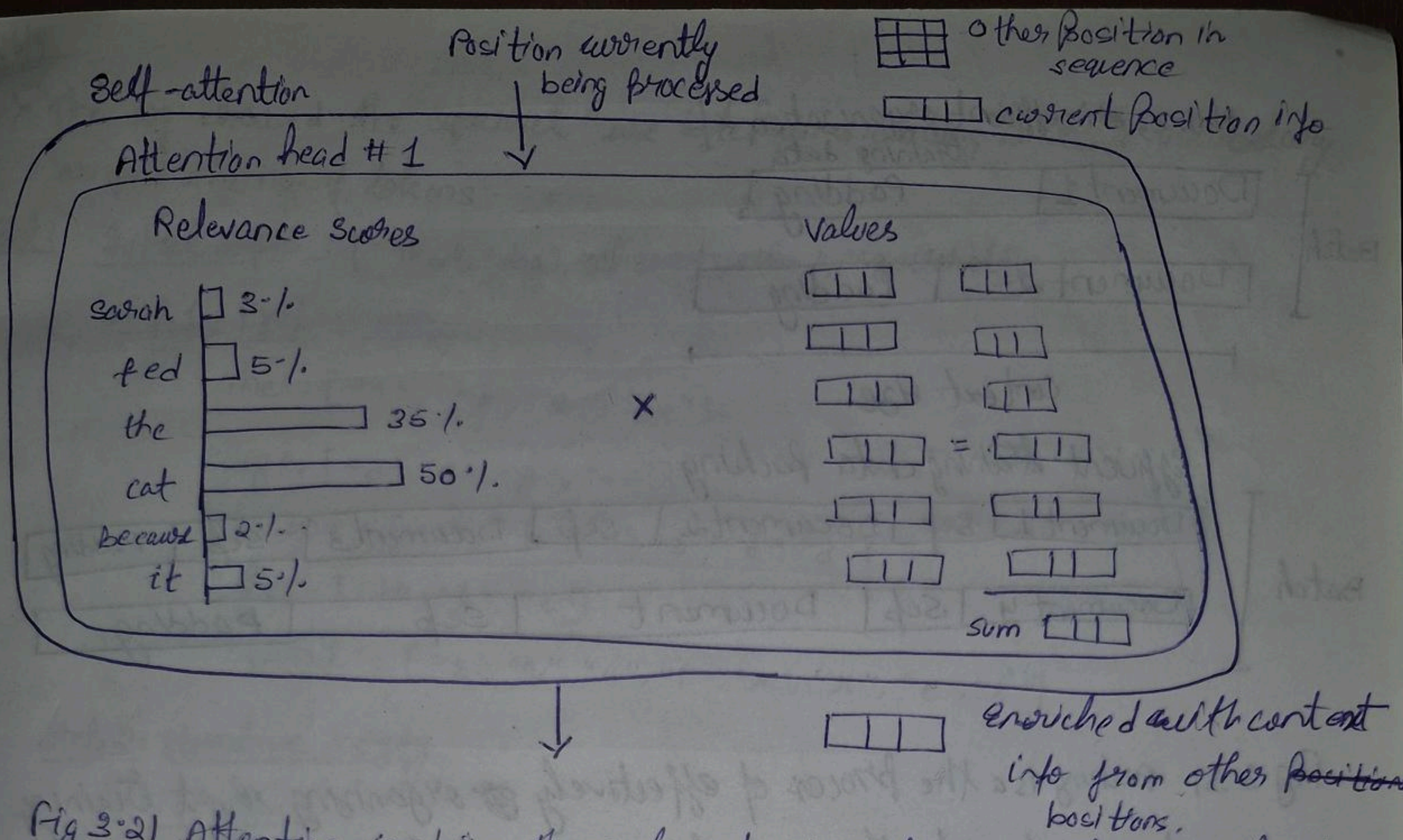


Fig 3.21 Attention combines the relevant info of previous positions by multiplying their relevance scores by their respective value vectors.

<u>Local/sparse attention</u> - To make more efficient attention, the local or sparse attention mechanism limits the context of previous tokens by only paying attention to a small no. of previous tokens / positions.

<u>Positional embedding</u> - also called as RoPE (Rotary Position Embedding).

- The RoPE is a key component in transformer LLM.

Q) Why positional embedding is required?

<u>Sol</u>: Let us take an example.

There are lot of documents are trained using language model, however the training set is much shorter than that context

- That means it would be inefficient to allocate the entire content.

Eg: 4K into a 10 word sequence.

- So during model training the documents are packed together into each context in the training batch, that is known as <u>packing</u>.

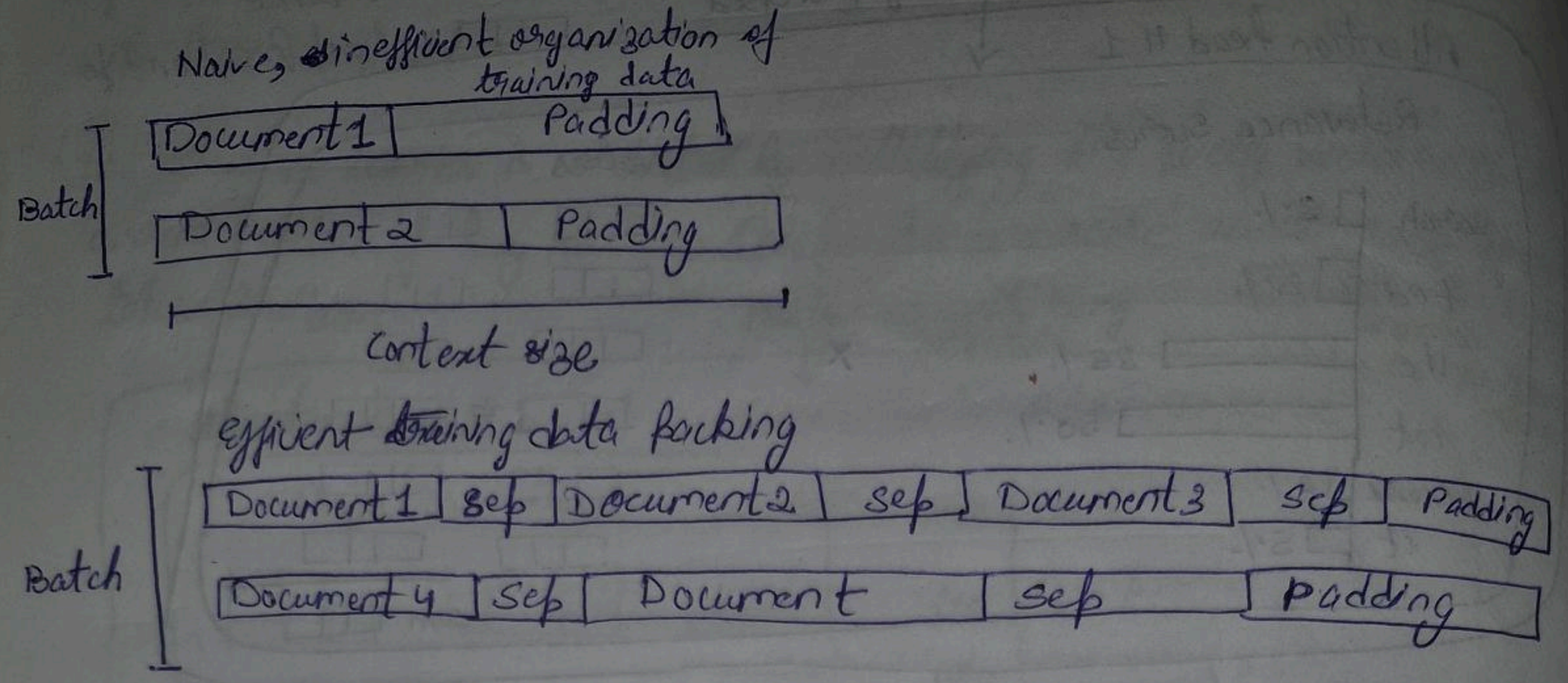


Fig 3-31 Packing is a the process of effectively organizing short training documents into the context. It includes grouping multiple documents in a single context while minimizing the padding at the end of context.

Rotary Embeddings -

Rotary embeddings are a method to encode positional information in a way that captures absolute and relative token position information. It is based on the idea of rotary vectors in their embedding.

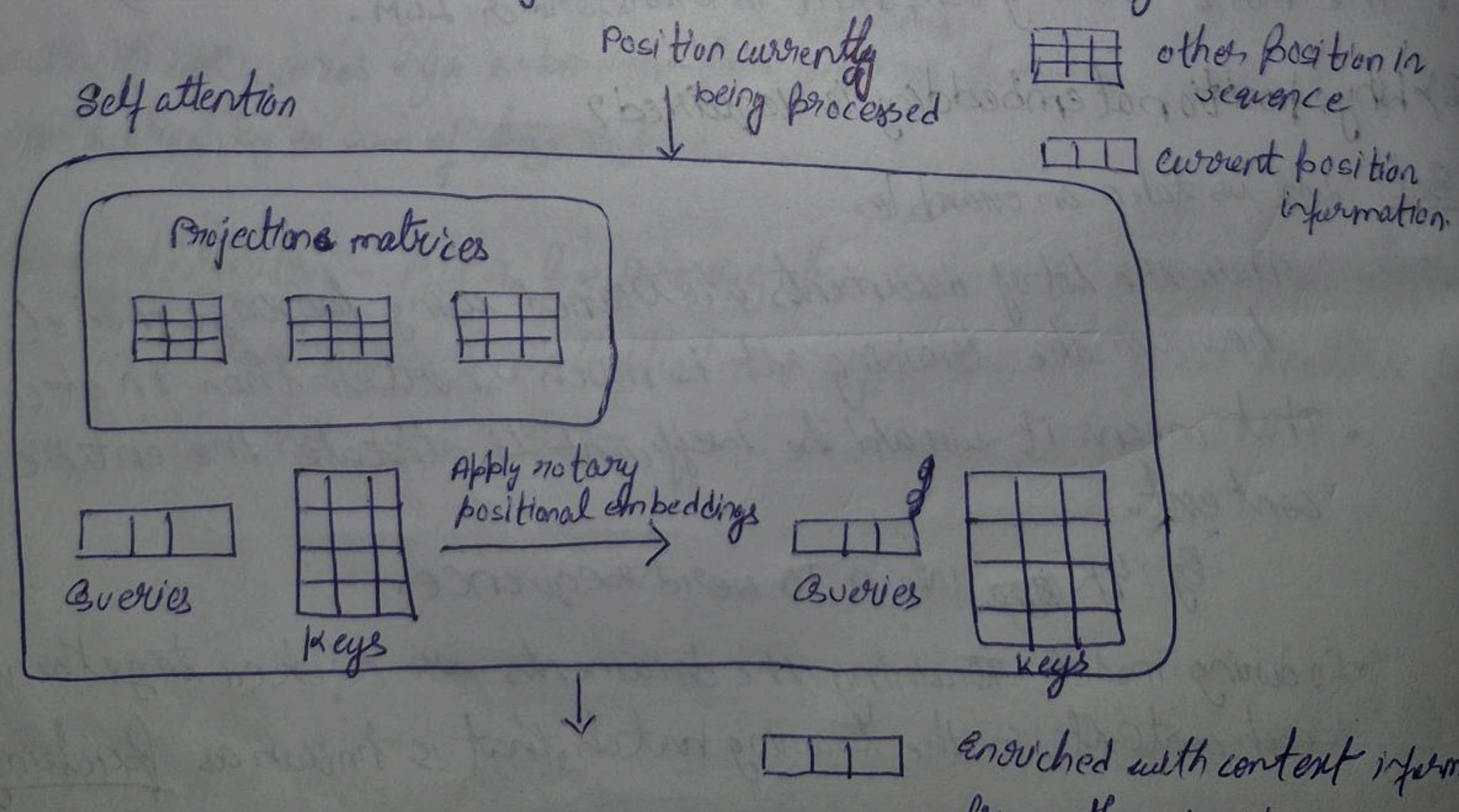


Fig 3.33. Rotary positional embeddings are added to the representation of tokens just before the relevance scoring step in self-attention.

26/9

Q> "The dog chased the squirrel" use bytepair encoding to create vocabulary or find the no. of tokens.

Sol: Initial corpus : ["The", "dog", "chased", "the", "squirrel"]

Step 1: Preprocessing

"The": ["T", "h", "e", "_"]

"dog": ["d", "o", "g", "_"]

"chased": ["c", "h", "a", "s", "e", "d"]

"the": ["t", "h", "e"]

"squirrel": ["s", "q", "u", "i", "r", "r", "e", "l"]

Step 2: Iterative merging

1. Iteration 1

Most frequent pairs: ("t", "h", "e")

Merge into : "The"

corpus updates: ["The", "_"]

["d", "o", "g"]

["c", "h", "a", "s", "e", "d"]

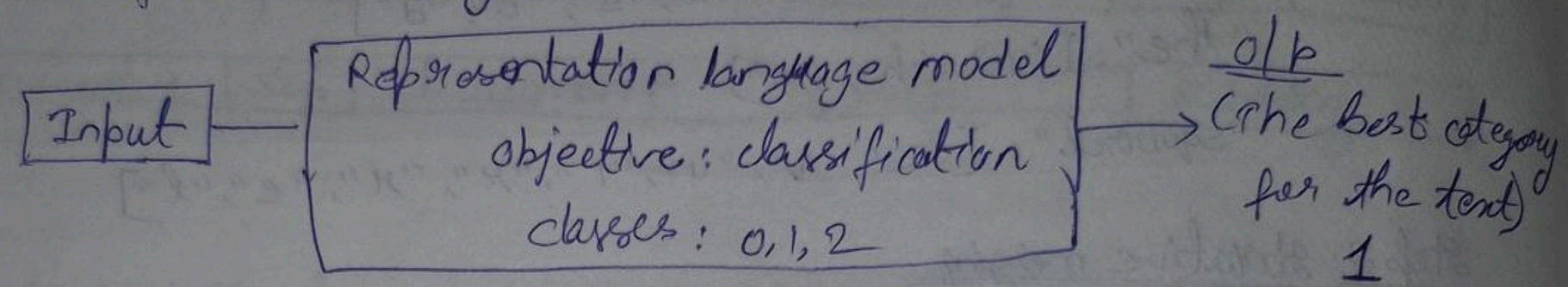
7/10/25

CH-4 : Text Classification

Text classification - classification of text is used to extract the meaningful insights like entities and detecting the language.

- The text classification in the field of language models can be represented using :
  i) text classification with representation model
  ii) Text classification using generative model

Eg -



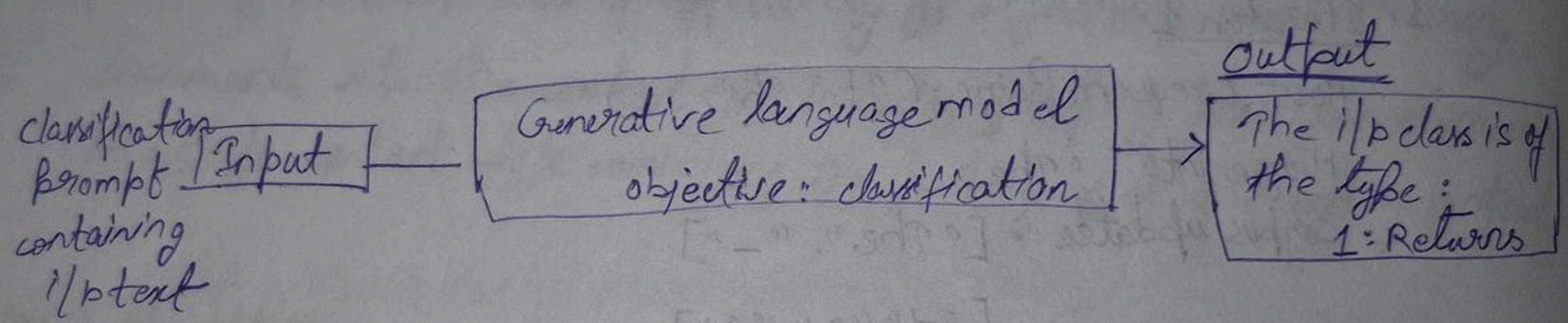


Fig 4:2 . Although both representation and generative models can be used for classification, their approaches differ.

Text classification with representation models -

1) Task specific model
2) Embedding model

- The Task specific model and embedding model are used for the text classification where the task specific uses tokenizer and token embedding to ~~generate~~ generate the word embedding.
- On the other hand the embedding model uses the supervised ML techniques for the text classification.

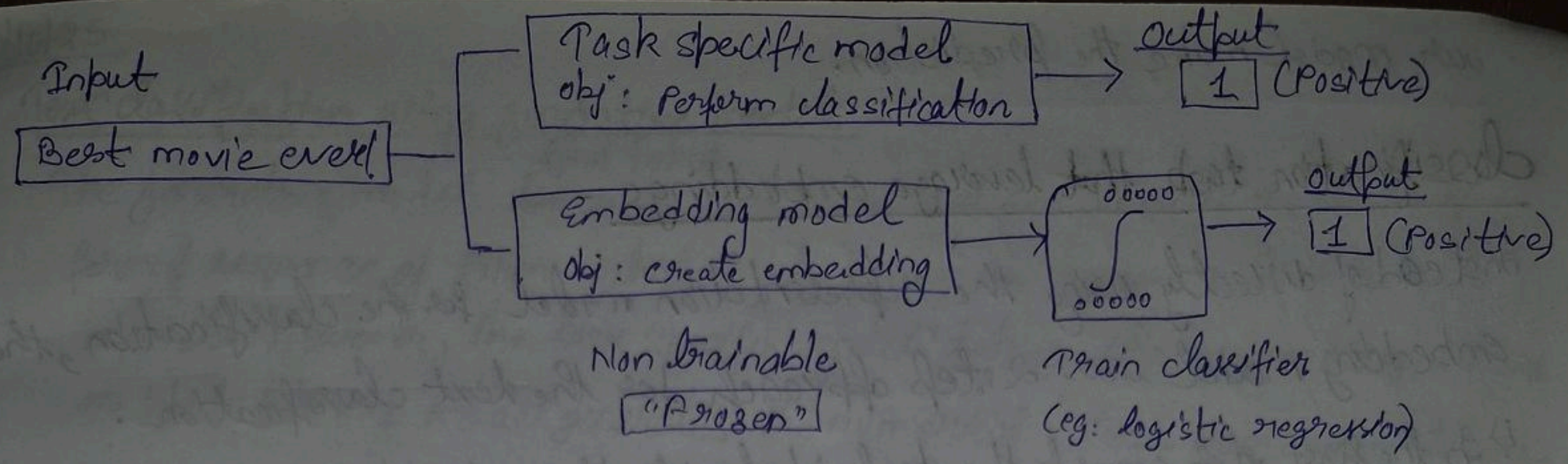


Fig 4.4. Perform classification directly with a task-specific model or indirectly with general purpose embeddings.

Confusion matrix

		Actual values	
		P	N
Predicted values	P	TP	FP
	N	FN	TN

FP → incorrectly classified as positive

FN → incorrectly classified as negative

- The confusion matrix shows ~~whether~~ whether the model correctly or incorrectly predicted result correspond to the actual values
- It represents by using 4 combinations of true and false values i.e TP, TN, FP, FN

Precision $= \dfrac{TP}{TP + FP}$

- It specifies how many true values identified actually.

Recall $= \dfrac{TP}{TP + FN}$

- It defines out of all values which are actually predicted true.

Accuracy $= \dfrac{TP + TN}{TP + FP + FN + TN}$

- Accuracy find how many are true out of all predictions.

F1 score $= \dfrac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$

- It combines both recall and precision scores which measures how effectively

our model make the prediction.

classification task that leverage embeddings -

Instead of directly using the representation model for the classification, the embedding model uses 2 step approach for the text classification.

i) In the first step convert the text input into the embeddings using embedding model.

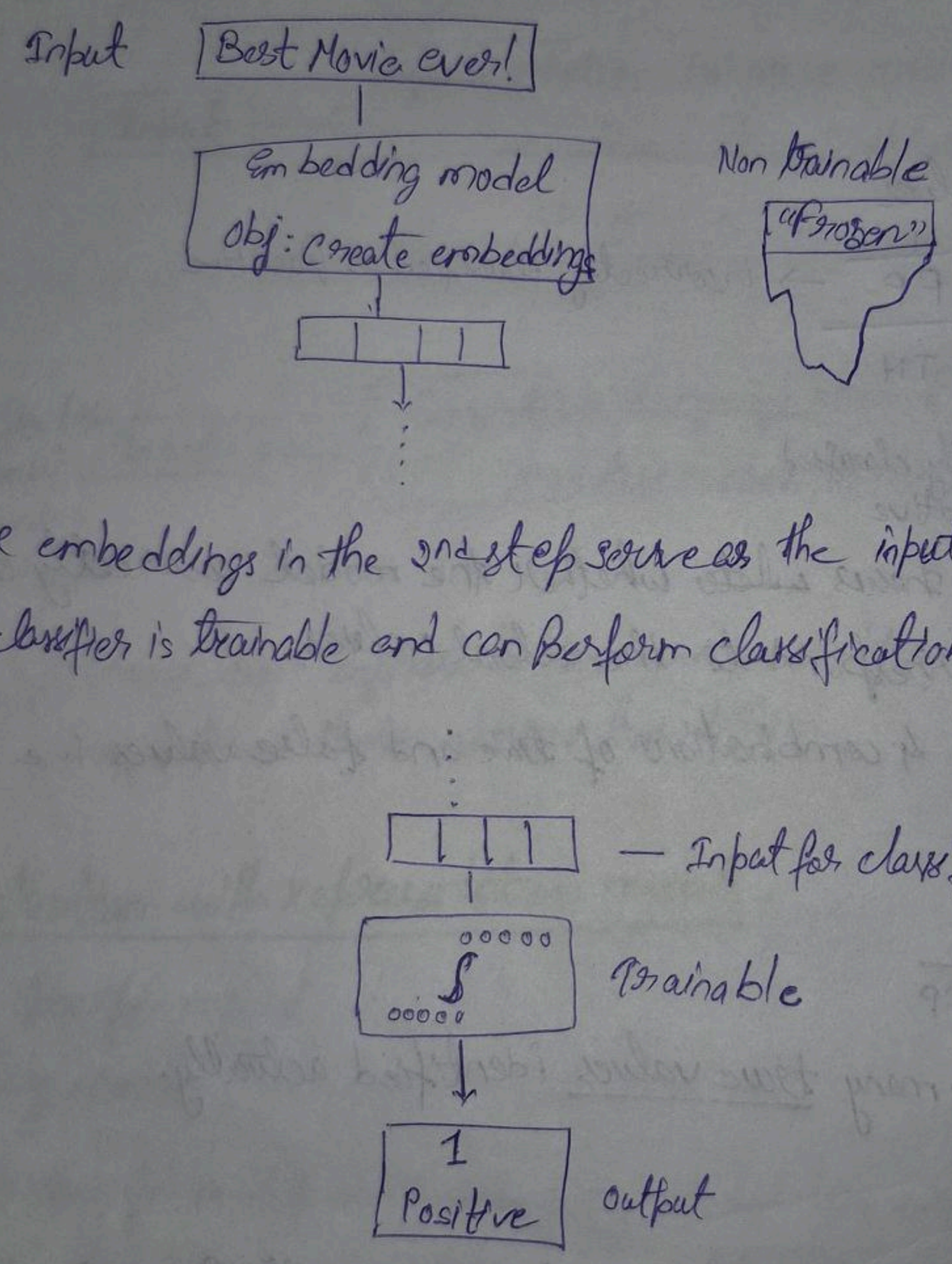


Fig 4.12

ii) These embeddings in the 2nd step serve as the input features to the classifier. The classifier is trainable and can perform classification.

Benefit of this method is that -

We didn't need to fine tune our embedding model which can be costly.

10/10/25

Text classification using generative models -

The generative models ~~takes~~ ~~text~~ takes as input some text and generate text in the form of sequence of tokens which is also known as sequence to sequence model. However, the task specific model takes the sequence of tokens as input but the model generates the numerical values instead of tokens.

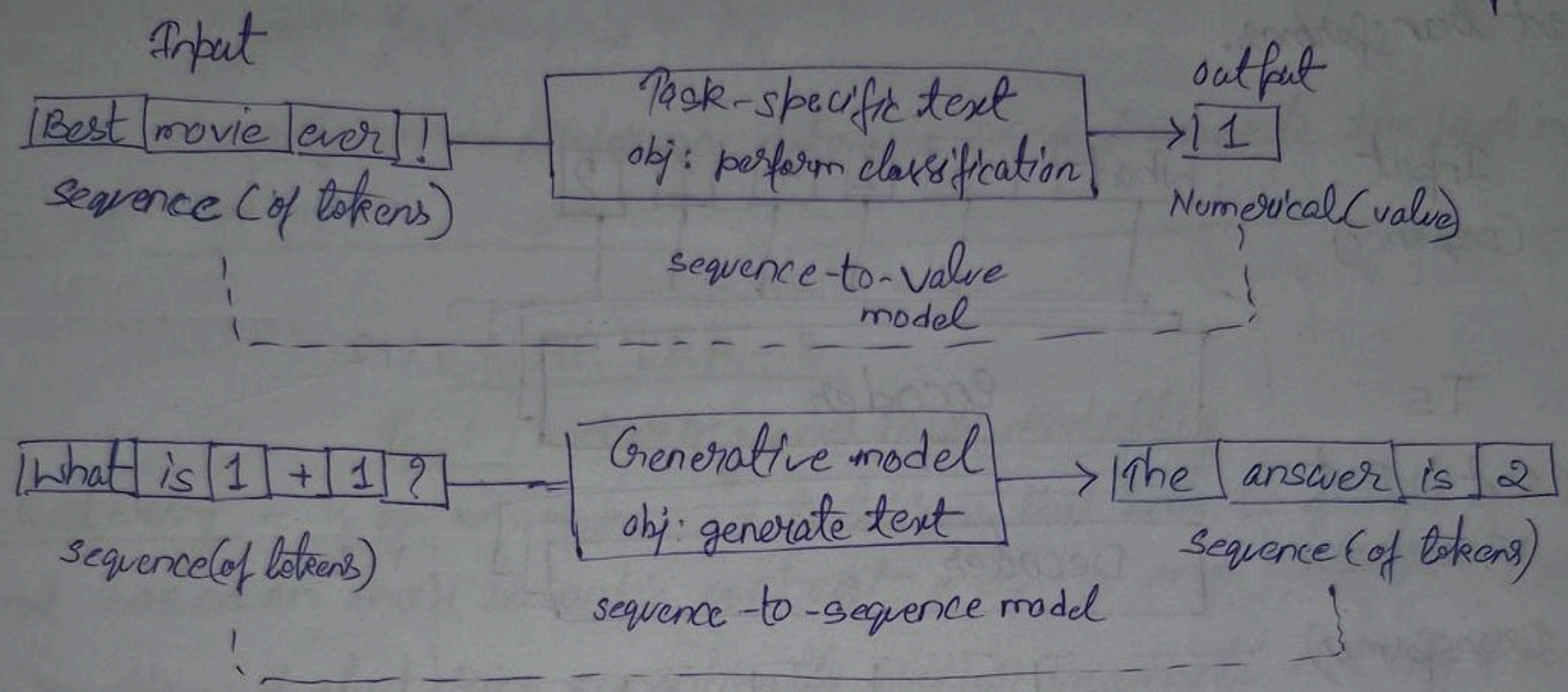


Fig 4:17 A task specific model generates numerical values from sequences of tokens while a generative model generates sequences of tokens from sequence of tokens.

- These generative models are generally trained on a wide variety of task and usually don't perform out of box.

For eg: If we give a generative model a movie review without any context then it has no idea ~~what~~ what to do.

- This guiding process is done mainly through the instruction or known as "prompt" that we provided to the model.

- Iteratively improving the prompt to achieve the ~~a result~~ preferred result is called as prompt engineering.

- Prompt engineering allows prompts to be updated to improve the output generated by the model.

~~Text to Text~~

Text to text transfer transformer

- Instead of encoder only model (BERT) and decoder only model (ChatGPT) ~~that~~ the text-to-text transfer ~~m~~ model (T5) contains both encoder and decoder for the classification of text.
- T5 model contains 12 encoder and 12 decoders that work for text-to-text transformer.

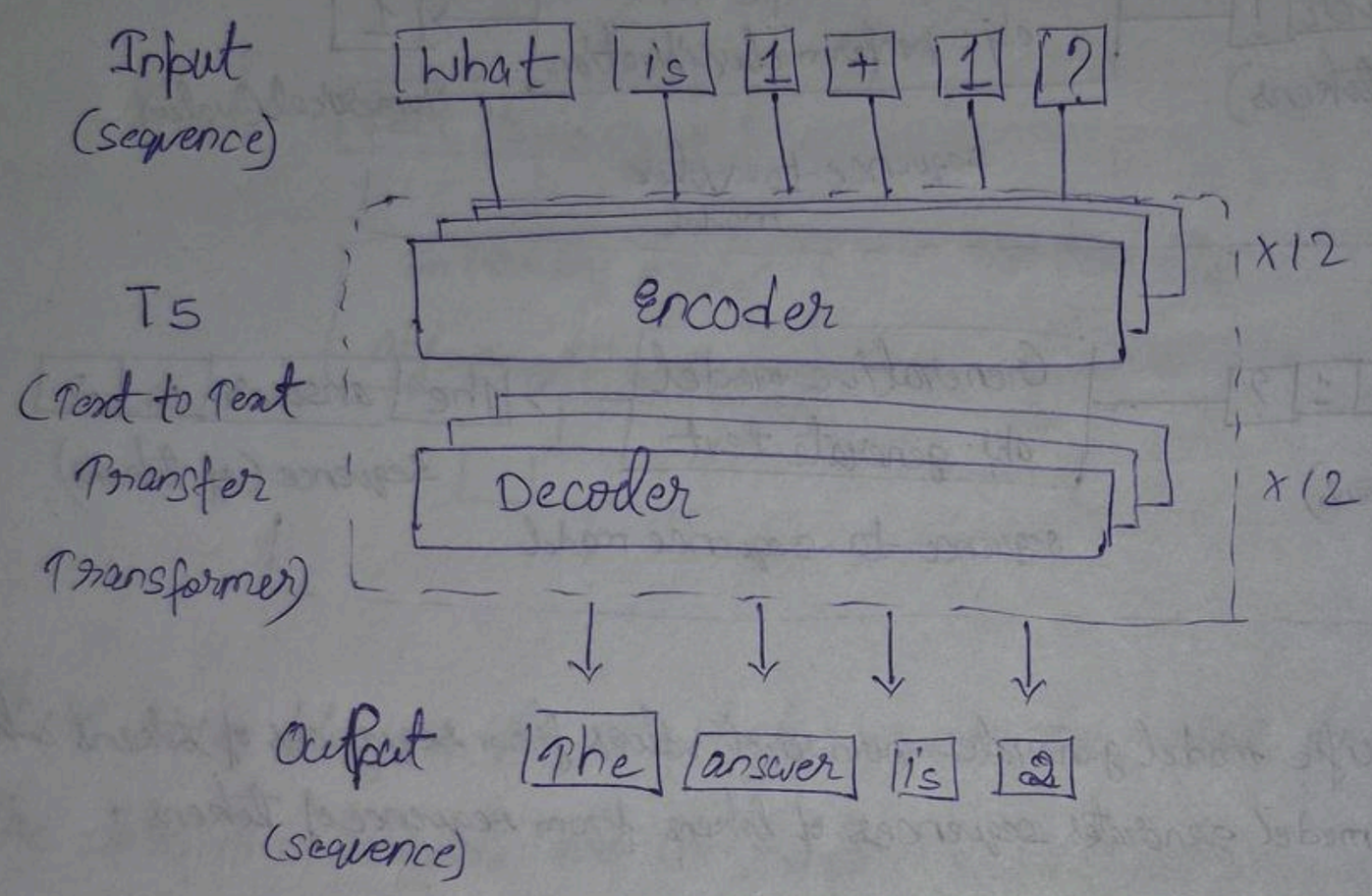


Fig: 4.19 The T5 architecture ~~to the of~~ is similar to the original transformer model, a decoder-encoder architecture.

In 1st step,

- ~~In beginning step~~ The models first pre trained with set of tokens known as ~~masg~~ masked language modelling.
- In 2nd step ~~of~~ training, the model is fine tuned with variety of task.

ChatGPT for classification -

- The ChatGPT uses open AI during the training process that generate the desired output to an input data in the prompt. Every time it checks for the preference and final output will be accordingly preference-tunning.

For eg:

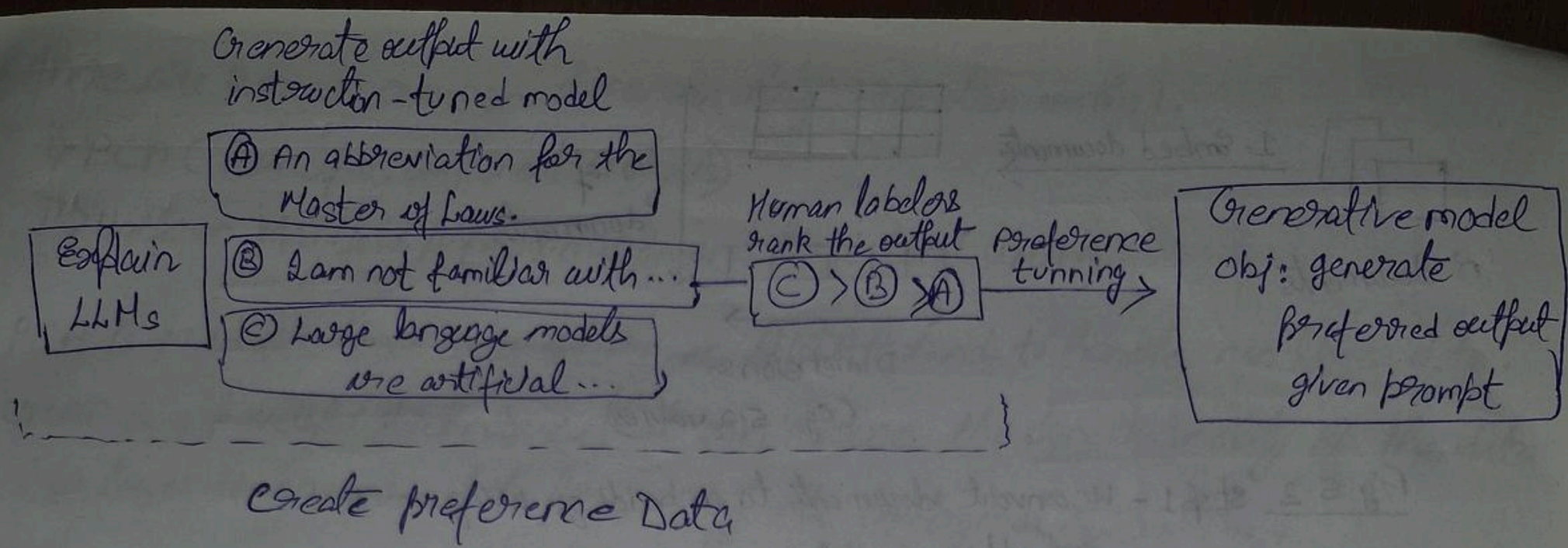


Fig: 4.23. Manually ranked preference data was used to generate the final model, chat GPT.

14/10

CHAPTER - 5

Text clustering and topic modelling

Text clustering - is an unsupervised technique that ~~ends~~ aims to groups similar text based on their semantic content meaning and relationships.

- The resulting of clustering semantically similar documents not only fascinate efficient categorization of large volume of ~~data~~ unstructured text but also allow for quick explanatory data analysis. (EDA)

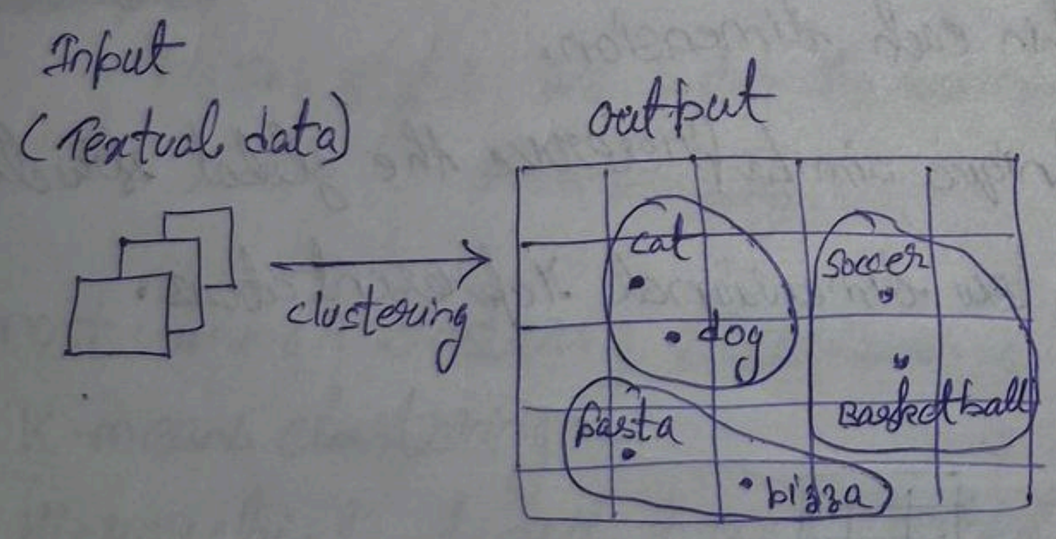


Fig 5.1 clustering unstructured textual data

A common pipeline for text clustering -

- Text clustering displays non linear patterns in the data which may contain the complexity in the analysis.
- To apply the text clustering from the graph based neural network to centroid based clustering technique with use 3 common steps and algorithms.

steps

1) Convert the input documents to embeddings with an embedding model.

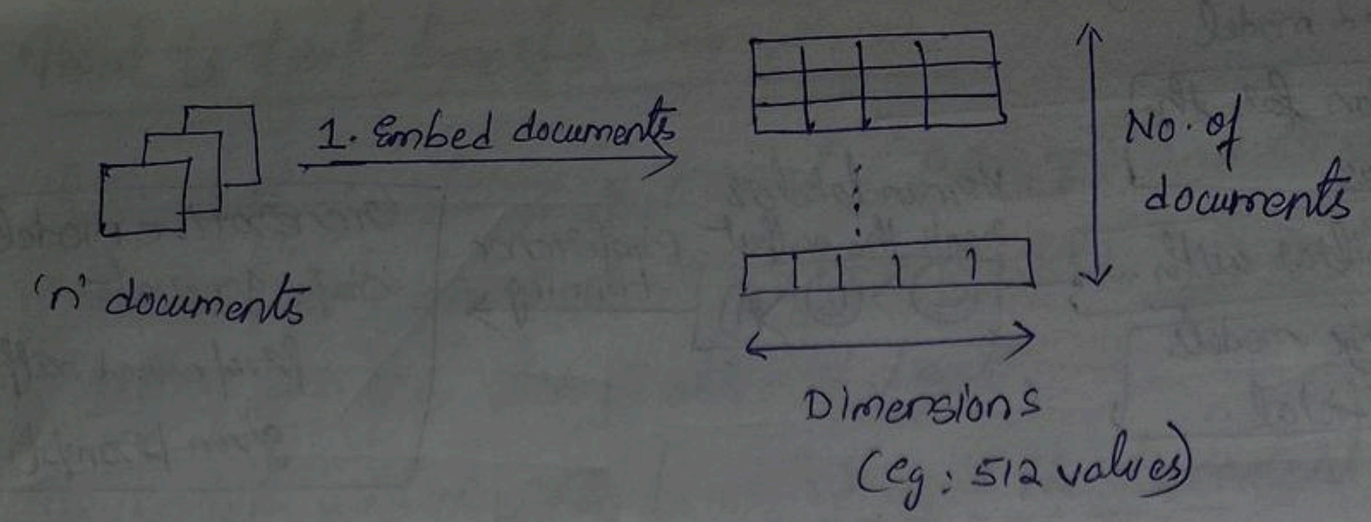


Fig 5.3 step 1 - we convert documents to embeddings using an embedding model.

- Here the textual data converting into embeddings means the numerical representations of text that attend to capture ~~at~~ its meaning.

Step 2) Reduce the dimensionality of embeddings using dimensionality reduction model.

- The dimensionality reduction process can be troublesome for many clustering techniques when there have larger volume of data as it gets more difficult to identify the meaningful clusters.
- As the dimensions in terms of no. increases there is an exponential growth in the no. of possible values within each dimension.
- So, the dimension reduction technique aim to preserve the global structure of high dimension data by finding low dimensional representations.

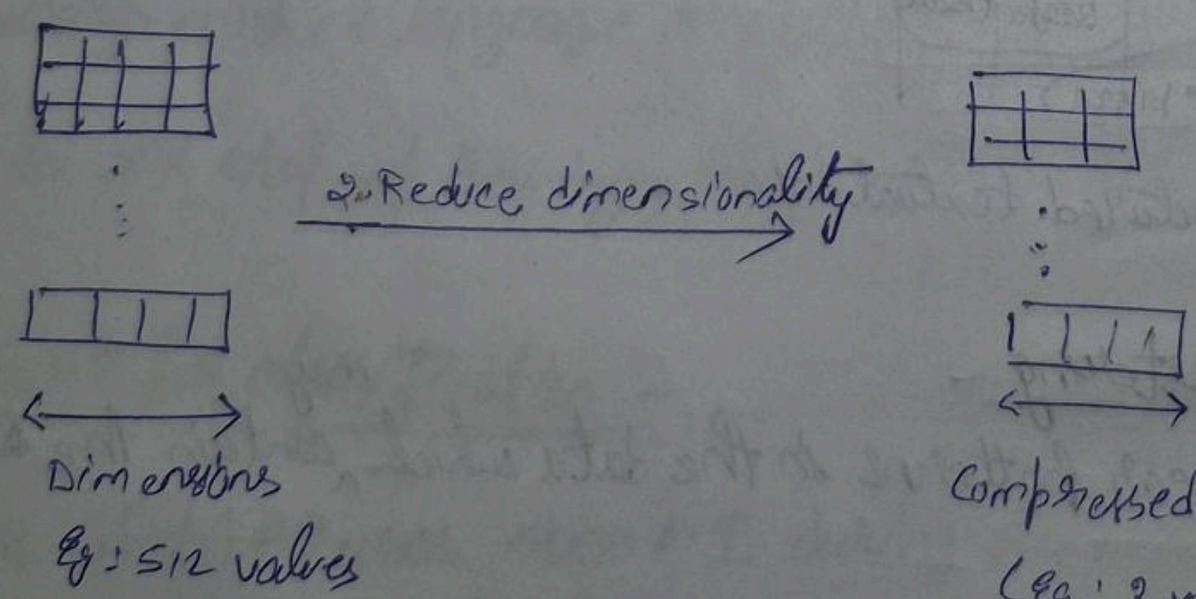


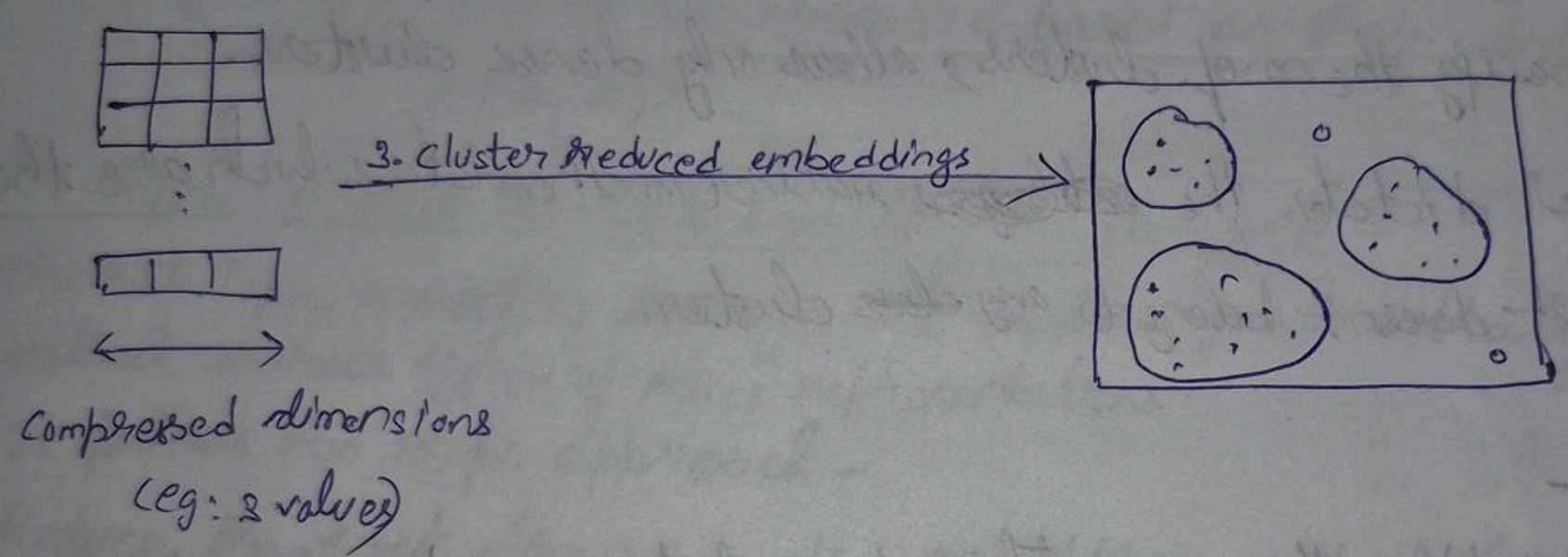
Fig: 5. step 2: The embeddings are reduced to a lower-dimensional space using dimensionality reduction.

- There are 2 most common dimensionality reduction method.
  i) PCA (Principal Component Analysis)
  ii) Uniform Manifold Approximation and Projection (UMAP)

* PCA is used for linear data where as UMAP tends to handle non linear data.
* UMAP is a powerful technique that can reduce the dimensionality of the data into lower dimensions while preserving its topological structure.

Step 3) Find the groups of semantically similar ~~models~~ documents with a cluster model.

- Clustering of the documents requires a set of clusters to be generated.



~~Fig 5.6 Step 3~~

Fig 5.6 step 3: We cluster the documents using the embeddings with reduced dimensionality.

2 most common clustering techniques as
i) K-means clustering
ii) Hierarchical density based spatial clustering of applications with noise (HDBSCAN)

* K-means clusterings - mechanism is a centroid based algorithm used to find the similar data into the clusters.
  - It's widely used when clusters are ~~sper~~ spherical and of similar size.
* HDBSCAN - It is a hierarchical density based clustering method that calculates the no. of cluster to be found without ~~forch~~ forcing all datapoints to be part of a cluster.

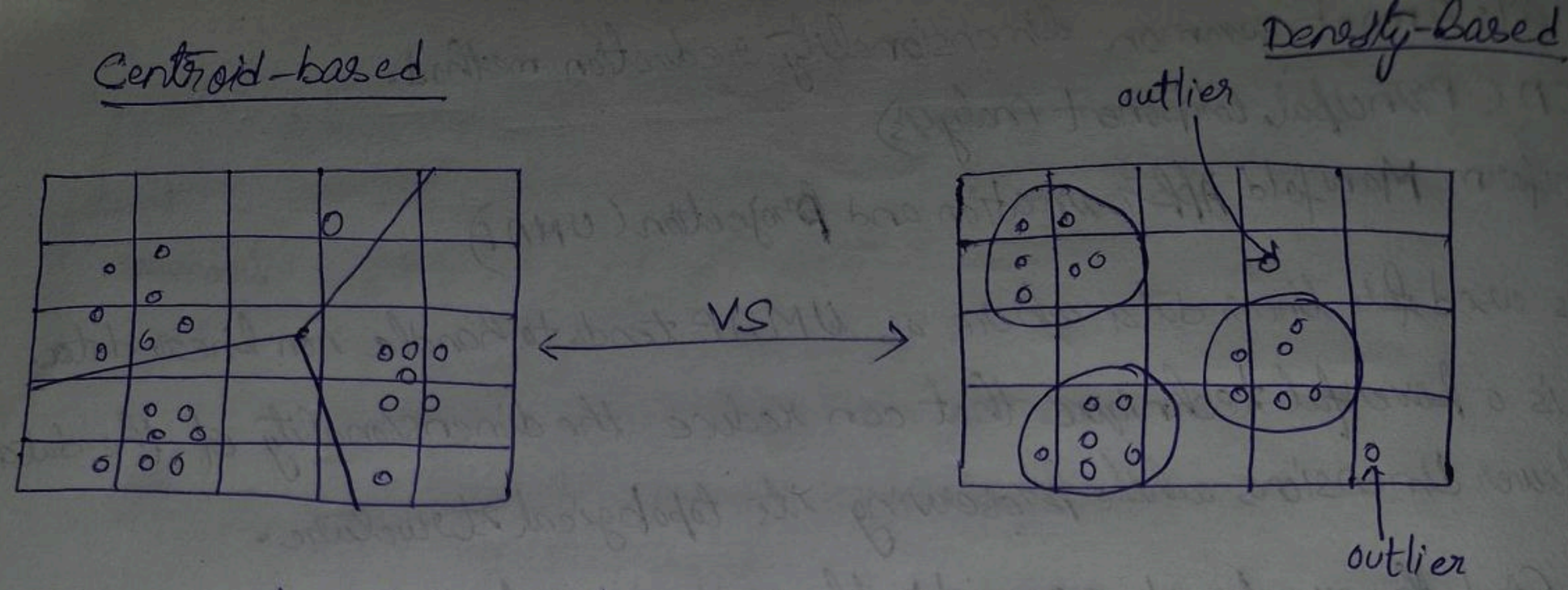


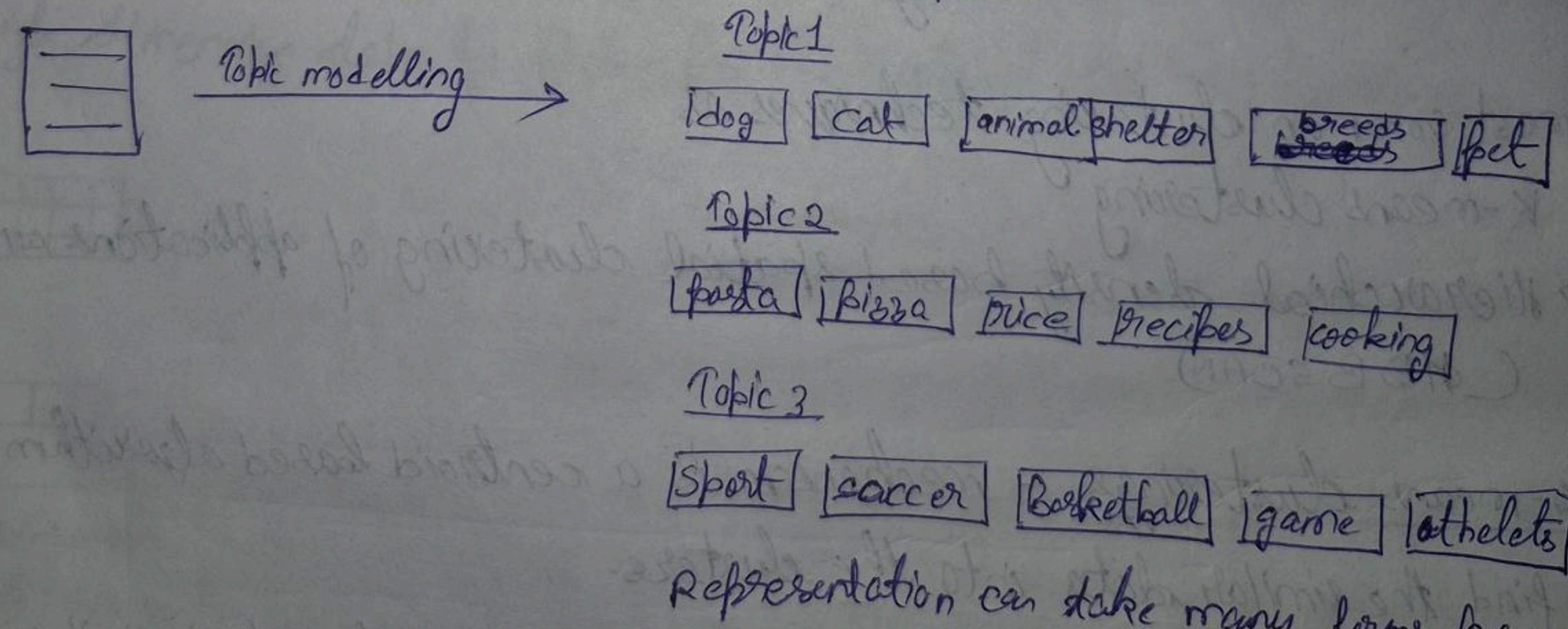
Fig: 5-7 · The clustering algorithm not only impacts how clusters are generated but also how they are viewed.

- The HDBSCAN algorithm allows dense clusters to be found without having to explicitly specify the no. of clusters, allows only dense clusters.
- It also ~~decte~~ dictates the ~~outliers~~ outlier in the data which are the datapoints that doesn't belong to any ~~class~~ cluster.

21/10

Topic modelling -

The process of providing themes on topics to a cluster according to their idea of finding themes each known as topic modelling.



Representation can take many forms keywords bi-grams, labels etc.

Fig: 5-9: Traditionally topics are represented by a number of keywords but can take another forms.

- Each topic is characterized by a probability distribution of words in a corpus vocabulary.

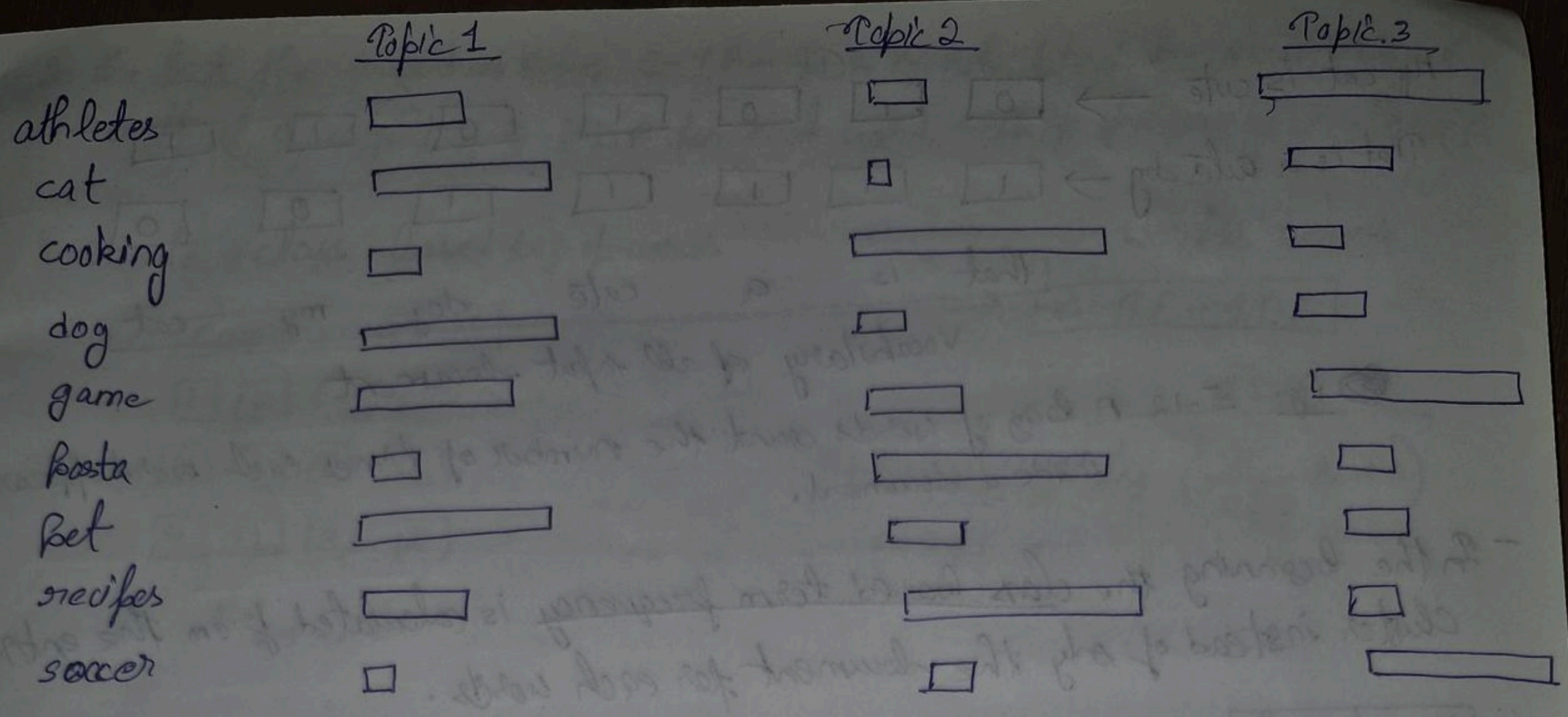


Fig: 5-10: Keywords are extracted based on their distribution over a single topic.

BERT Topic : A Modular Topic Modelling Framework -

- The BERTopic modeling liverage the clusters of semantically similar text to extract various types of topics representations.
- It follows two steps approach -

① Reduce the text dimensions after embeddings and cluster those compressed dimensions to create groups of semantically similar documents.

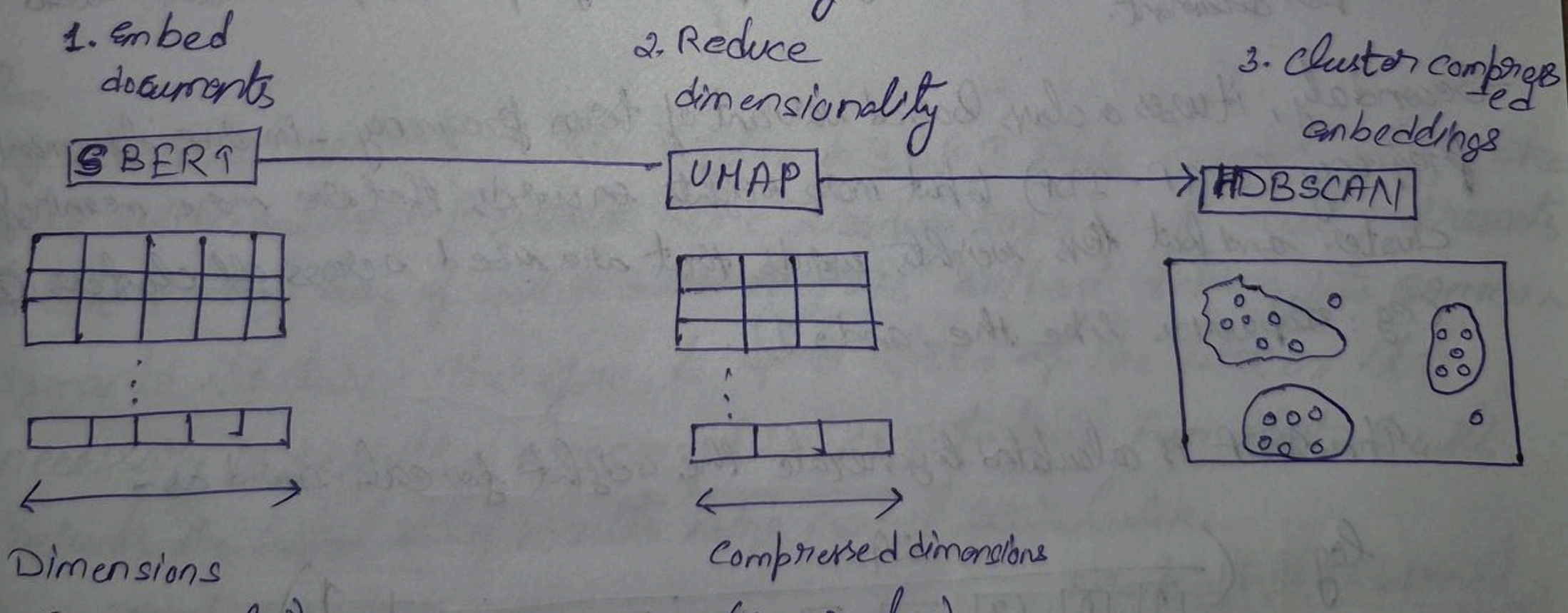


Fig 5-11: The first part of BERTopic's pipeline is to create clusters of semantically similar documents.

② Bertopic model provides a distribution of words in a corpus vocabulary by leveraging a classic method known as Bag Of Words.

My cat is cute → [0] [1] [0] [1] [0] [1] [1]

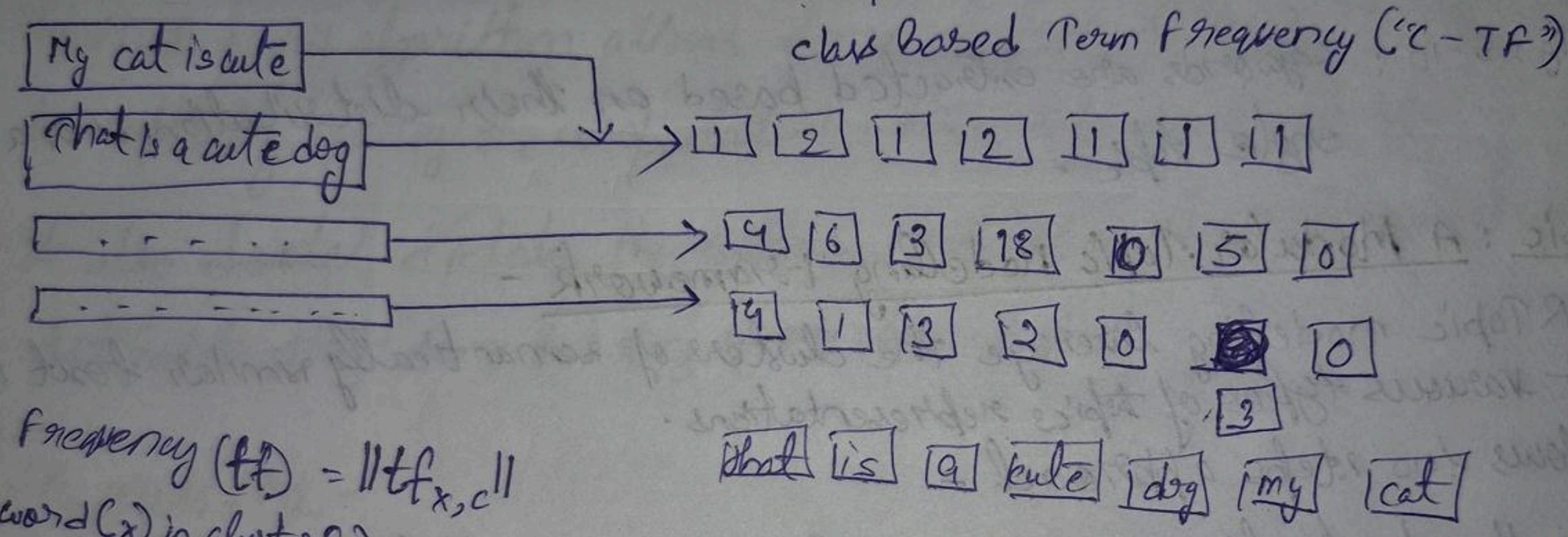
That is a cute dog → [1] [1] [1] [1] [1] [0] [0]

that is a cute dog my cat

Vocabulary of all input document

Fig: 5-12 A Bag of words count the number of times each word appears inside a document.

- In the beginning the class based term frequency is calculated from the entire cluster instead of only the document for each words.



Frequency (tf) = $\|tf_{x,c}\|$
word (x) in cluster (c)

Fig - Generating c-TF by counting the frequency of words per cluster instead of per document.

- Secondaly, it uses a class based variant of term frequency - inverse document frequency (c-TF-IDF) to put more weights on words that are more meaningful to cluster and put less weights words that are used across all clusters (eg: stopwords like the, and, I).

The IDF is calculated to generate the weight for each word as -

$$\log\left(\frac{29}{\underset{\text{that is a cute dog my cat}}{\boxed{4}\ \boxed{9}\ \boxed{7}\ \boxed{22}\ \boxed{1}\ \boxed{9}\ \boxed{1}}} + 1\right)$$

Average frequency (A) divided by the total frequency of each word (cfx)

$$= \log\left(\frac{A}{cfx} + 1\right)$$

Fig 5-14: Creating a weighting scheme

- In the last the weighted value C-TF-IDF is calculated by multiplying the result of a weight i.e IDF for each word with its frequency i.e C-TF.

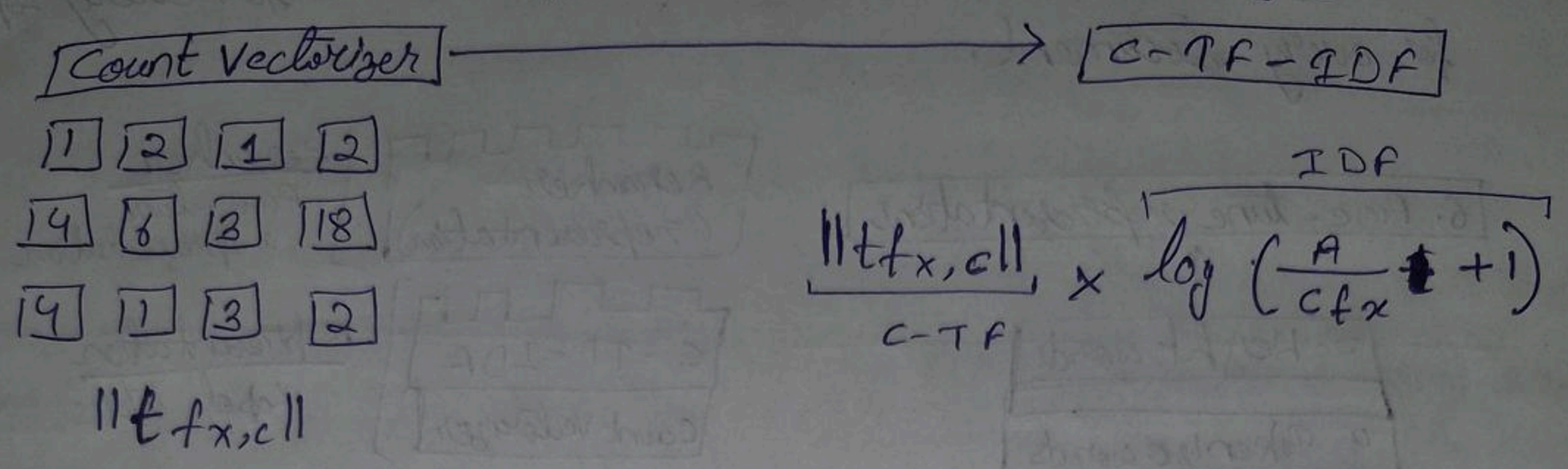
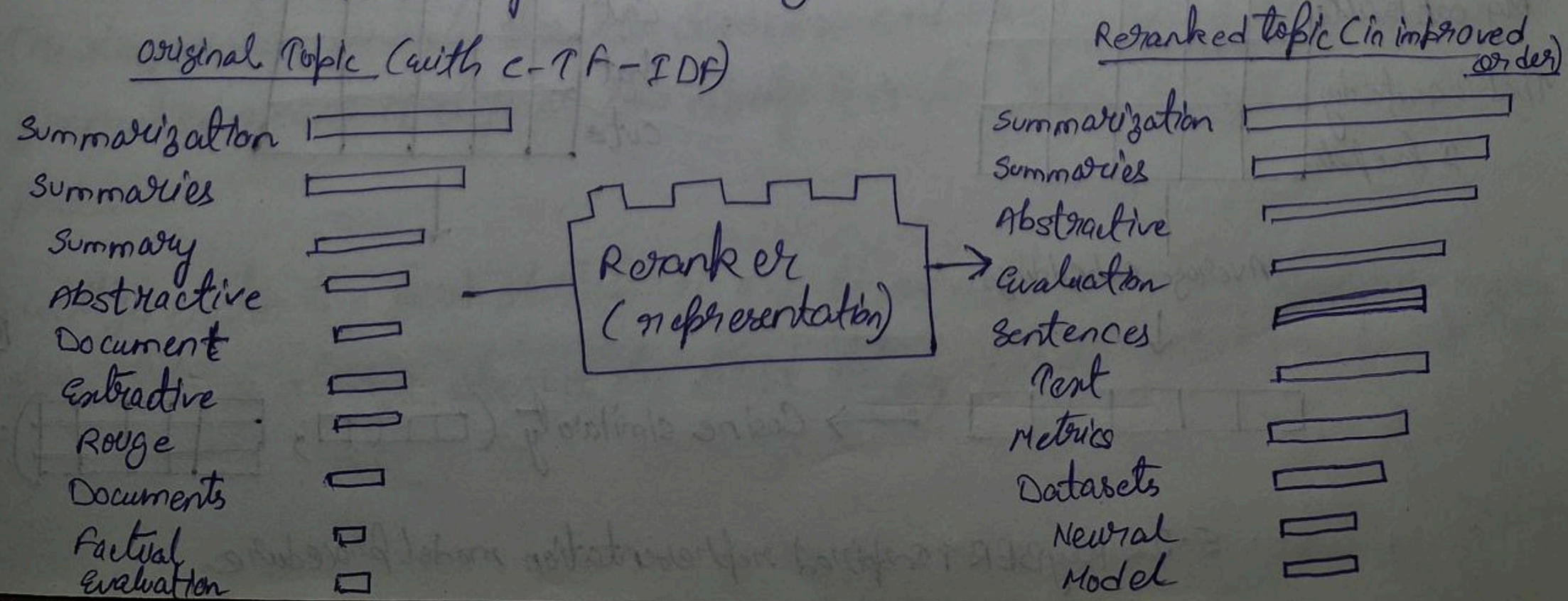


Fig: The second part of BERTopics pipeline is representing the topics, the calculation of the weights of term *X* is a class *C*.

24/10

Adding a special Lego Block - The pipeline in BERT topic has advantage like fast and modular but it follows some disadvantages as it still represents a topic through bag of words ~~with they are~~ without taking into common symantic structure. Therefore, ~~to spad~~ speed up the process it's necessary to generate a meaningful representation. For that, it will rerank the initial set of results using neural search like



- In BERT topic the reranker models are known as <u>representation models</u>.
- <u>For eg:</u> If we have millions of documents and hundred topics then the representation model only needs to be applied once for every topic instead for every document.

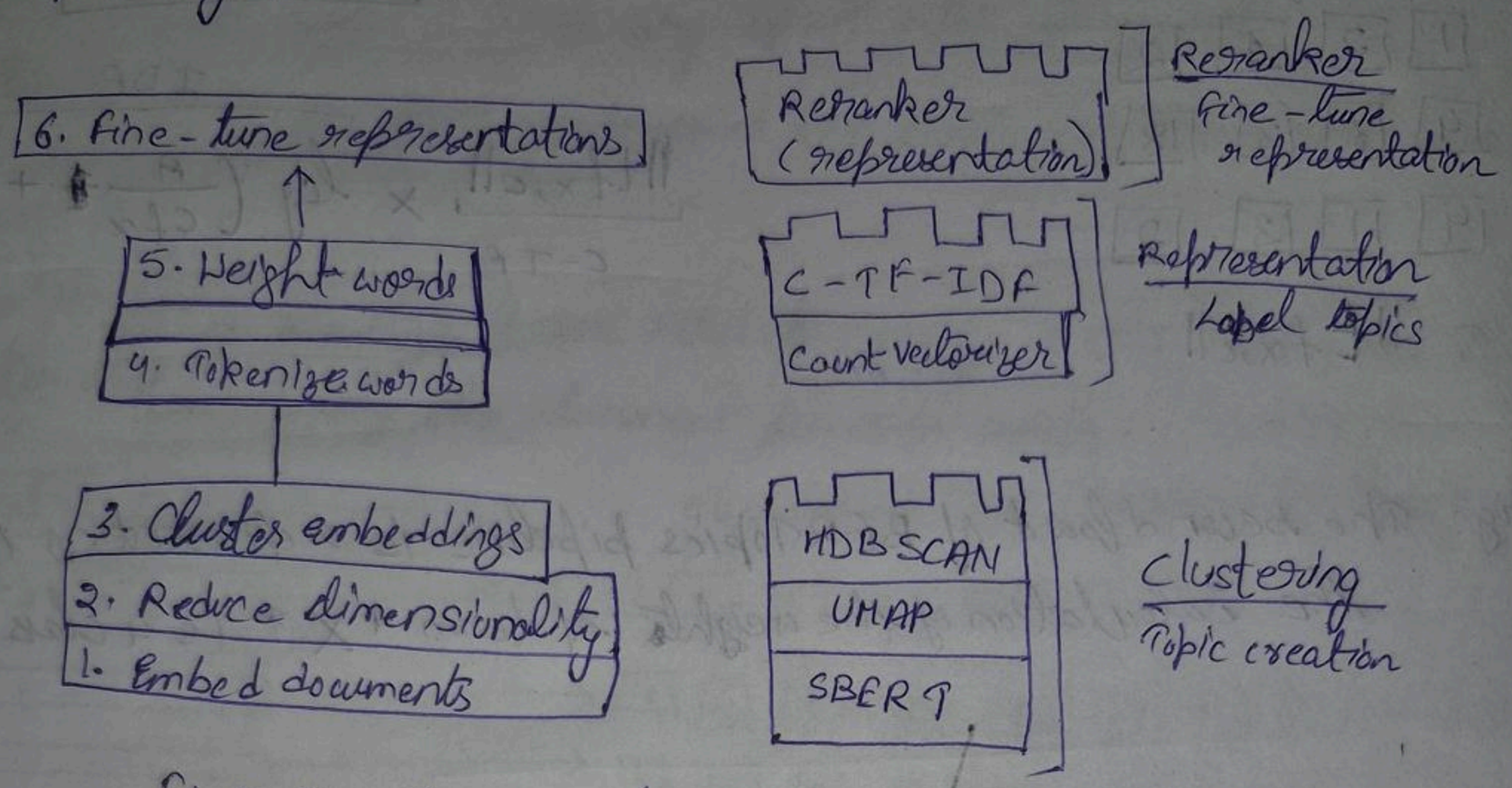
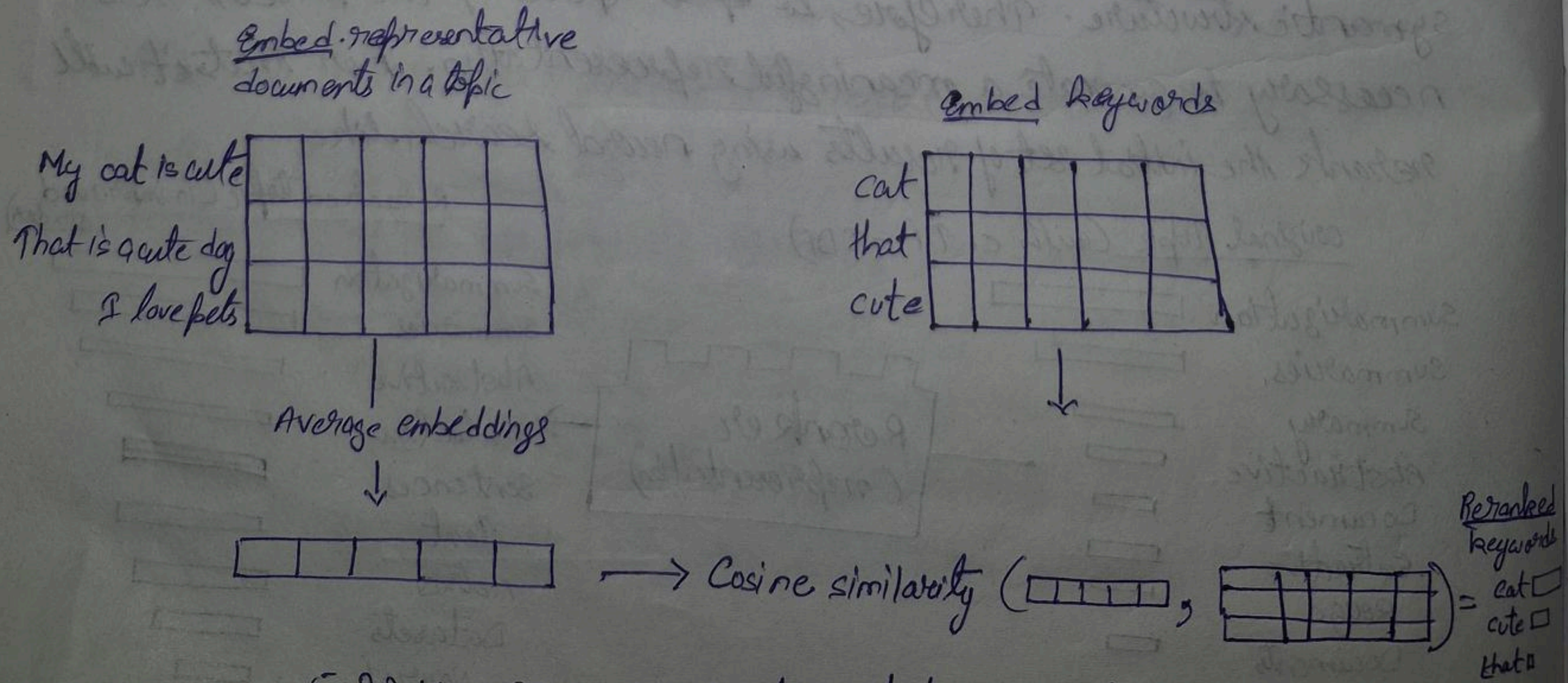


Fig 5.20 - The Reranker (representation) block sits on the top of C-TF-IDF representation.

- A wide variety of representation model have developed ~~for~~.
  Eg: KeyBERT Inspired
- <u>KeyBERT Inspired</u> ~~uses~~ uses C-TF-IDF to extract the most represented documents per topic by calculating the similarity between a documents C-TF-IDF values and those topics related to that keywords.



5.22 KeyBERT Inspired representation model procedure

- In this figure the average document embeddings per topic is calculated and compare to the embeddings of candidate keywords to rerank the keywords.

4/11

Chapter 6: Prompt Engineering

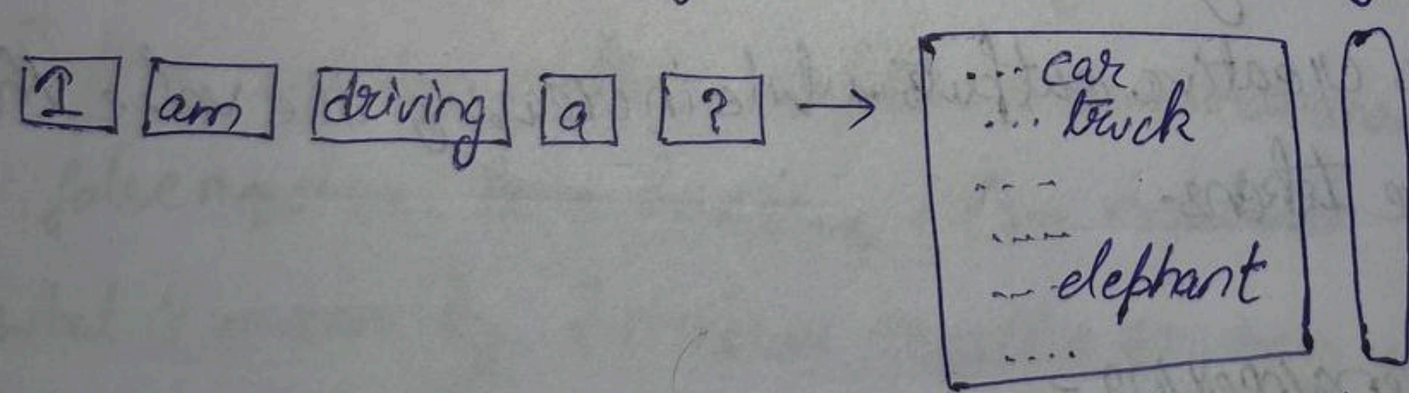
Prompt engineering - The prompt engineering provides the creativity and potential components as per the ~~user~~ user requirements such as in context learning and chain of thoughts.

- The prompt engineering uses generative pre-trained transformer (i.e. gpt) for the text generation.

- There are so many models for the generating text with billion of parameters Eg: Phi-3-mini which has 3.8 billion parameters.

Controlling the model output - In the prompt engineering the output can be generated according to the different response for the exact same prompt. Each time an LLM needs to generate a token.

Eg: There's a sentence given "I am driving a ?"



There are 2 important factors defined in the S function as

i) Temperature

ii) Top_p

- The temperature controls the randomness and creativity of the text that generated means it defines how likely it is to choose tokens that are less probable.

- It will generate the probability of any token with the same response every time lolo it always chooses the most likely word.

Top-p - It is a nucleus sampling method which controls the subset of tokens (nucleus) that the LLM can handle. means it will consider tokens ~~...~~ untill it reaches the cumulative probability.

For eg: If we ~~say~~ set top-p to 0.1, it will consider tokens untill it reaches that value but if we say top-p to 1 then it will consider all tokens.

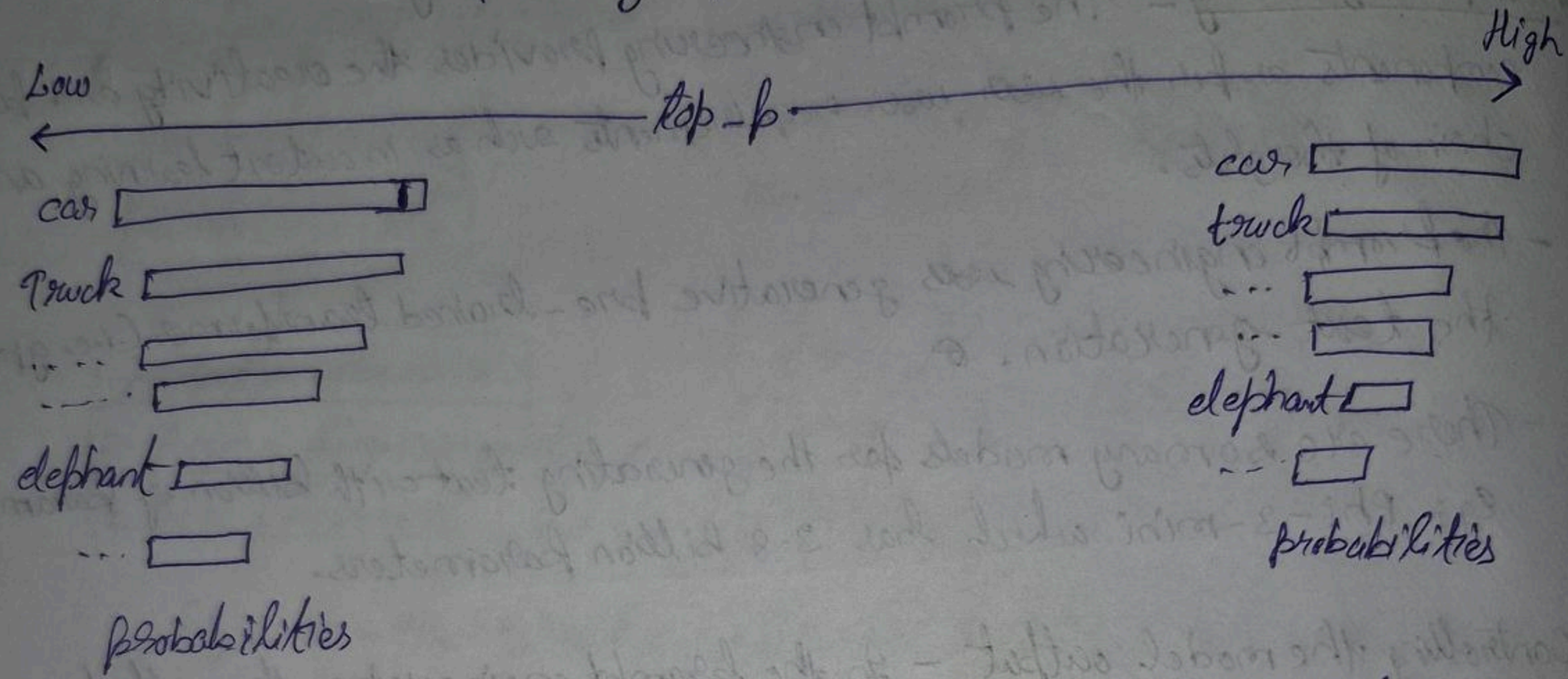


Fig 6.5 - A higher top-p increases the no. of tokens that can be selected to generate and vice versa.

In this figure, the lowering the value it will consider the fewer tokens and generally give less creative outputs while increasing the value the LLM to choose from more tokens.

Introduction to Prompt engineering -

It is a tool that evaluate the output of a model as well as to design ~~sept~~ safety methods. This is known as Prompt ~~en~~ optimization.

- The basic LLM model is a prediction machine that is based on a ~~sub~~ certain inputs where the prompt tries to predict the words that might give the appropriate or ~~relat~~ relevant answers.

- In the prompt box we can ~~contai~~ continue adding or updating the elements untill we can get the response what we are looking for.
- We can add additional examples.

Eg: Use cases in more detail, provide additional context etc to get the result.

7/11

Chapter 3 Question

Q1) What do you mean by the transformer LLM? Explain with diagram.
Q2) Define attention mechanism. What are 2 main steps involved in attention mechanism?
Q3) What is self attention? How you calculate the weight matrix when query, key and value matrix are given?
Q4) How to calculate the relavance score in self attention?
Q5) What is positional embedding? Explain with example.

Chapter 4 Question

Q1) What is text classification and how it works with representation models?
Q2) What is the difference b/w task specific model and embedding model?
Q3) How the text classification works using task specific model?
Q4) Define confusion matrix. Explain 4 combinations i.e false positive, false negative, true positive & true negative.
Q5) What is mean by precision recall, accuracy and F1 score?
Q6) Differentiate b/w supervised and unsupervised classification.
Q7) How text classification works using generative model such as chatgpt?

chapter 5 Question

Q1) What is text clustering? Explain the common pipeline for text clustering?
Q2) What do you mean by dimensionality reduction? How UMAP is different than PCA algorithm?
Q3) What is HDBscan clustering method? How it is different than K-means

clustering - Explain with diagram.

Q4) Write 5 steps in BERTopic modelling method.

Q5) What is topic modelling? How key word inspired provides advantage over BERTopic model?

Date
Page

CH-5

Q1) What is text clustering? Explain the common pipeline for text clustering.

Q2) What do you mean by dimensionality reduction? How UMAP is different than PCA algorithm.

Q3) What is HDBSCAN clustering method. How it is different than K-means clustering? Explain with diagram.

Q4) Write 5 steps in ~~Bert topic~~ BERTopic modeling method

Q5) What is topic modeling. How Key ~~BE~~ word inspired provides advantage over BERTopic model.

Instruction based prompt engineering/prompting

As we know prompting is a mechanism in large language model is ~~performed~~ often used to have the llm answers specific question ~~of~~ or resolve a certain task this is known as instruction based prompting.

Although, this technique ask the llm to summarize a piece of text that will not suddenly result in classification.

- This works as an important role often to ~~sell~~ select the most common use cases based on the instruction.
- ~~Co~~ The use case may be - the structure and the location of ~~the~~ the instruction that can change.

- ~~There are 3 steps~~ To improve the quality of the output the llm uses to follow 3 techniques -

1. Specificity - means accurately describe what we want to achieve
2. ~~Hallucitions~~ Hallucination - Sometimes llm's generate incorrect information confidently which is refer to as ~~Hallu~~ hallucination.
   - To reduce its impact we ask llm ~~&~~ to only to generate the answer if it knows the ~~ans~~ answer.
3. Order - Either in the begin or end the prompt generates the instructions which tends to focus on the order of the information.

Advanced prompt engineering -

Now a days the Llm models brings the advanced technique from the prompt starting from the iteratively work from.
using Llm models to achieve improved results.
- Although the prompt engineering provides some common components-

1. Persona (identify) - it describes what the Llm should taken
2. Instruction - the task itself means summarizes the key finding of the specific instruction more accurately
3. Context (additional information) - describes the context of problems it wants to ~~ask~~ answer.
4. Format - The Llm should follow the format to generate the ~~out~~ output.
5. Audience - The target of the generated text it describes the level of generated output for the specific purposes.
6. Tone - The style of text means - the professional and clear.
7. Data - The summarized form of generated text.

28/

Chain prompting -

- It is the mechanism by splitting of the prompts into a modular components to improve the performance of Llm.
- It works with multiple prompts like a sequential pipeline.

Eg: we take the output of 1 prompt and use it as input for the next thereby creating a continous chain of interaction that solve our problems.

Chain of thoughts -

- The most important part of generating Llm is chain-of-thoughts that aims to have the generative model "think" fast rather than answering the question directly" without any reasoning.
- Adding these reasoning steps allows the model to distribute more computation over the reasoning process. This helps

additional tokens to generate in this reasoning process to allow the llm to more stablize the output.
- It is basically used for mathematical calculations.

~~Zero shot~~ - In generally chain-of-thought is a great method for enhancing the output of a generative model and it always require one or more reasoning of eg in the prompt.
- However, if we simply asks the generative model to provide the reasoning, instead of providing eg it is known as zero shot chain-of-thought.
- But if we provide one or more eg to the model and asks for the question then it is known as few shot.

28/11/25

Exploring the tree of thoughts -
- The idea of tree of thoughts mainly used to improve the output of generative models.
- The tree of thoughts contains sampling of multiple thoughts and make them more to improve the decision process.
- Likewise the questions relevant to multiple reasoning are used by the llm models as -

i) first breaks those into multiple pieces
ii) at each step the generative model is prompted to explore different solution to ~~proble~~ the problem asked by the users.
iii) It then goes for the highest or best solutions and continues to the next steps. In between that the generative models can generate the intermediate thoughts to be related to the output. The most promising thoughts are kept and less promising are ignored.

Output Verification -
1. Structured output
2. Valida output

3. Ethics
4. Accuracy
These are used for validating for the outputs for the generative AI models however, there are 3 ways of controlling the output

i) Examples
ii) Grammar
iii) Fine - tunning

CH-7 Advanced text generation techniques & tools

In this chapter we will extend the capabilities by adding modular components.

1. we first create a simple and reusable chain that connect the LLM with a prompt template
2. We ~~can~~ then expand on the concept by adding memory to the chain which allowed the LLM to remember the conversation.

There are 3 different methods to add memory in the LLM model - Mapped

i) Model I/O - loading quantized models with ~~langchain~~ langchain

Langchain - It is the method or framework that contain four methods for improving the quality of generated text.

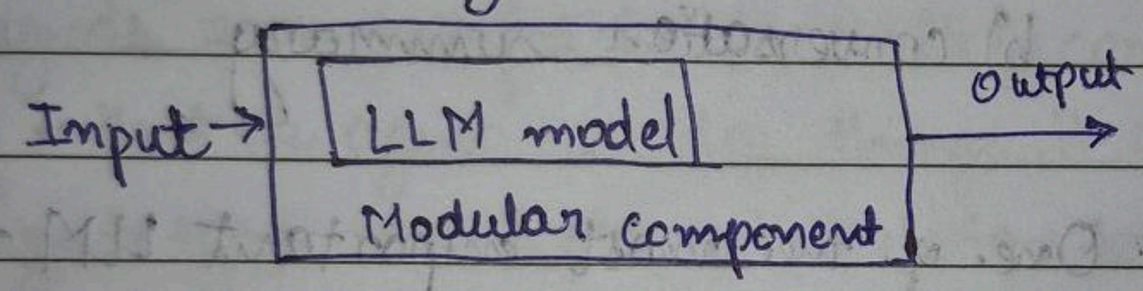
a) model I/O - prompts, LLM, Output parser (mapped)
b) memory - conversion memory, summarization
c) Retrival - Embeddings documents, vector database
d) Agents - autonomous, ~~conversion~~ conversation

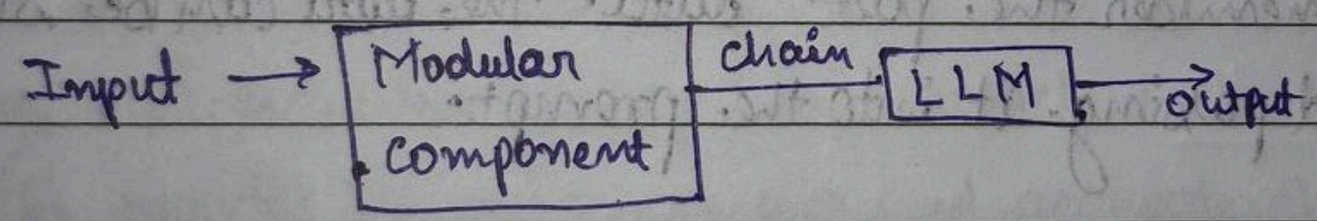
Date
Page

Quantization - Generative model like GGUF represents a compressed versions of its original value through a method called as quantization, which reduces the number of bits needed to represent the parameters of an LLM model. This means reducing the no. of bits while attempting to maintain most of the original information for example - representing the value of pi (π) with 32 bit and 16 bit gives the same result.

Chains - In LLM models the Langchain is used to take many forms of chains with having different complexity additional tools prompts and features.

- There can be the single chain that ~~context~~ contains.

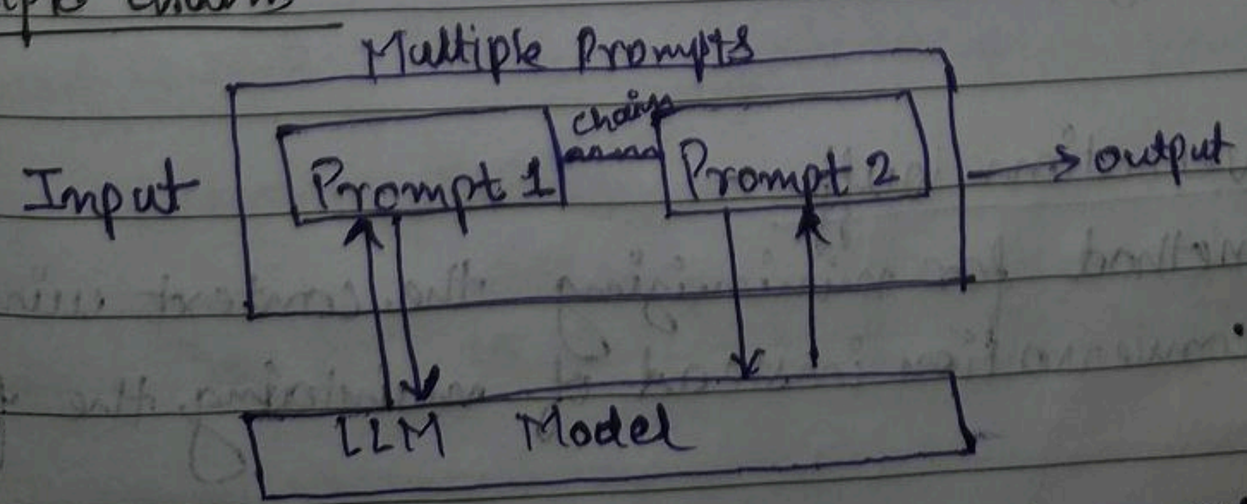




Single chain connect modular component like prompt template, or external memory

- To generate a single chain template we first need to create a prompt template that the model takes the input from the user and generate the text after the end of prompt.

Multiple chains -

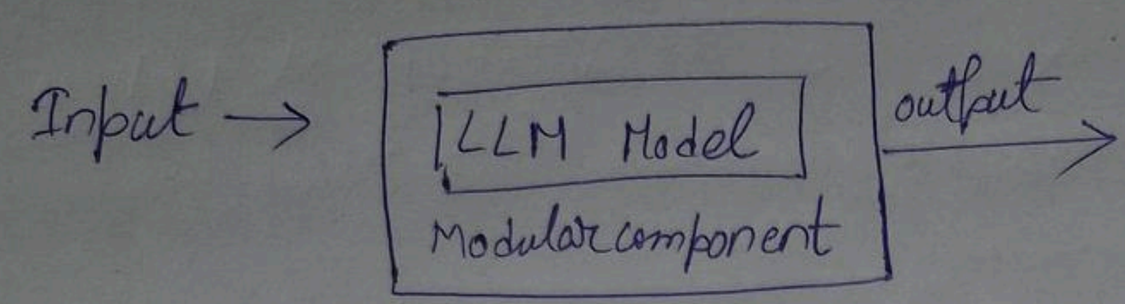


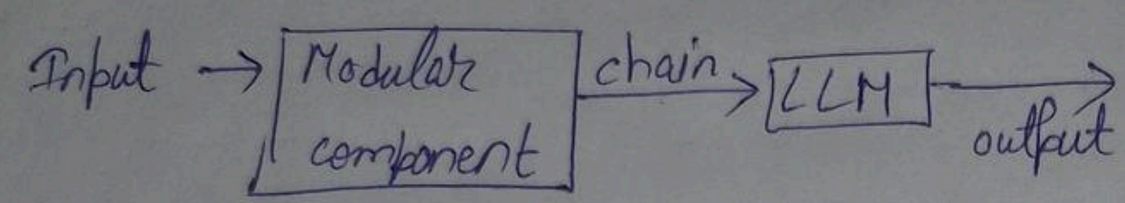
The process of using multiple prompts is an extension of single prompt used in single chain.

2/12

Chains - In LLM models the lang chain is used to take many forms of chains with having different complexity additional tools, prompts and features.

- There can be a single chain that contains

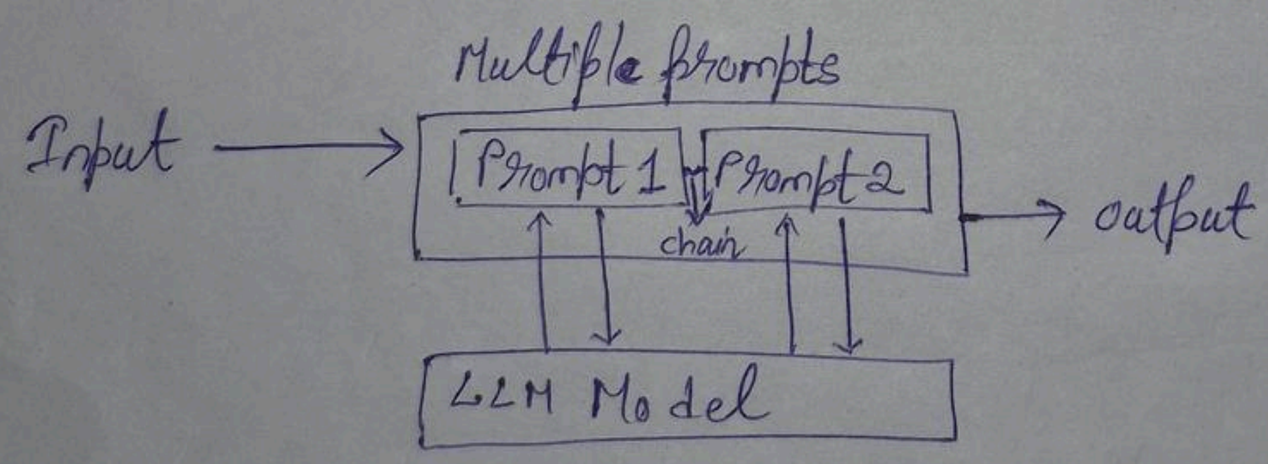




Single chain connect modular component like prompt template or external memory.

- To generate a single chain template we first need to create a prompt template that the model takes the input ~~th~~ from the user and generate the text after the end of prompt.

Multiple chains -



The process of using multiple prompts is a extension of single prompt used in single chain.

Eg: We want to generate a story that has 3 components like a title, a description of main character, ~~a~~ summary of story.

- Instead of giving in one output, here the single input can be used to chain with the further input and continues this process to generate the ~~answer.~~ output.

ii) <u>Memory</u> - The LLM model expect to remember the past conversations in the chain, for this reason the model requires memory that can improve the strength of the models.

- There are 2 common methods to remember conversations -
  i) Conversation buffer
  ii) Conversation summary

1) <u>Conversation buffer</u> - One of the most important LLM memory that used to remember the past data. The data can be stored as conversation history and passing it into the prompt.

Prompt

1. Hi! I am Ram
   What is 1+1?

2. Hello Ram
   The answer for 1+1 is 2
   What is my name?

2) <u>Windowed conversation buffer</u> -

Another method for minimizing the context window is to use the last K conversations instead of maintaining the full chart history.

3) <u>Conversation summary</u> -

The conversation summary contains by the conversation history and to create a consise summary by the LLM model. The advantage of ~~this~~ using an external LLM helps to create a meaningful summary in the prompt template.

- For this the K conversations can be retained inside the conversation buffer window memory to resolve the issues of the limitation of tokens.

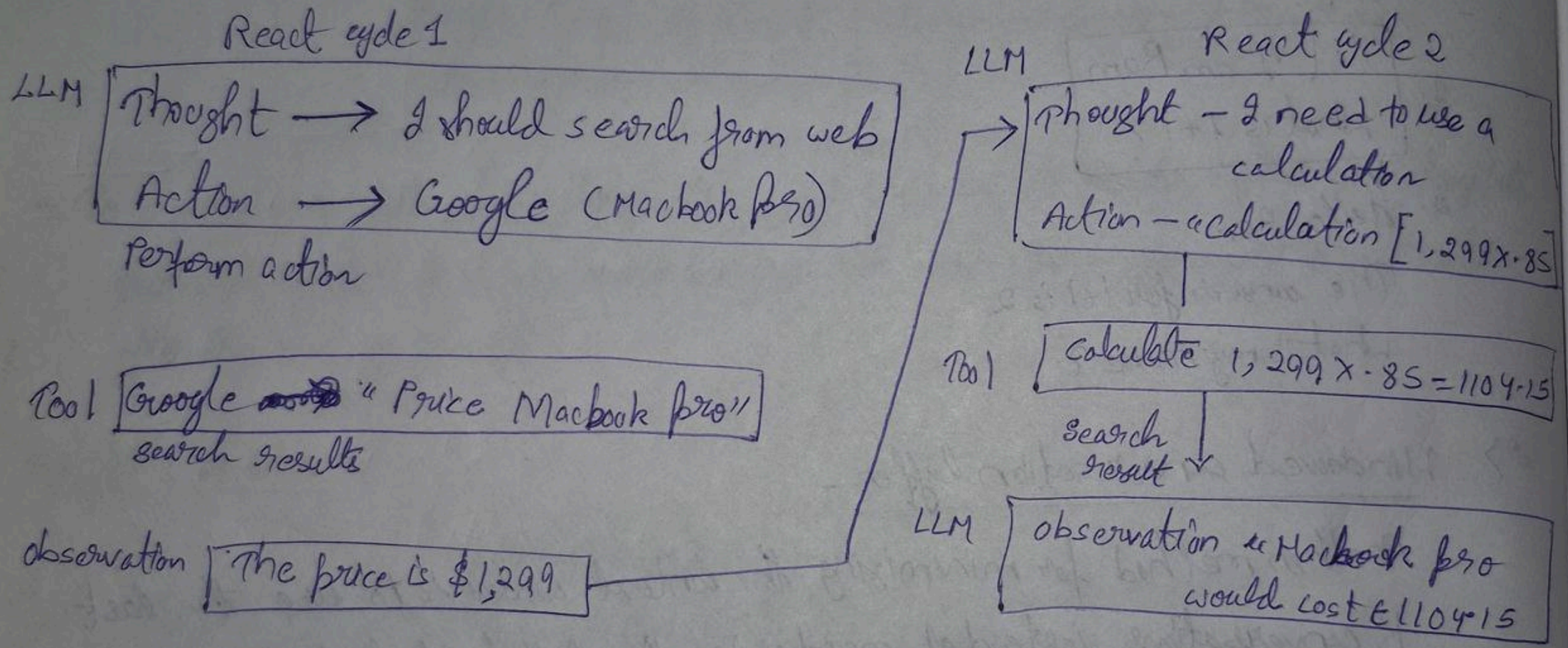
<u>Agents</u> - The agents in LLM can make use of everything like the model, I/O, chains and memory.

- Agents are generally determined to take actions to generate the output.

- The agents are classified in 2 vital components.
  i) Tools - that the agent can do things, it couldn't do itself.
  ii) The agent type which plans the actions to take or tools to use.

- Giving LLM the ability to choose which tools they use for a particular problem results in more complex & accurate behaviour.

- In general this process use 3 steps
  i) Thought
  ii) Action
  iii) Observations.



Q) What is the current price of a macbook pro in USD? How much would it cost in Euro if the exchange rate is 0.85 Euro for 1 USD?

3/12/25

Q) Why

Sol: In

- The

i) D

3/12/25

CH-8: Semantic Search and Retrieval augmented generation

Q> Why RAG is needed?

Sol: In general the models are able to answer fluently and confindently but their ~~answer~~ answers are ~~alway~~ not always correct and up to date.

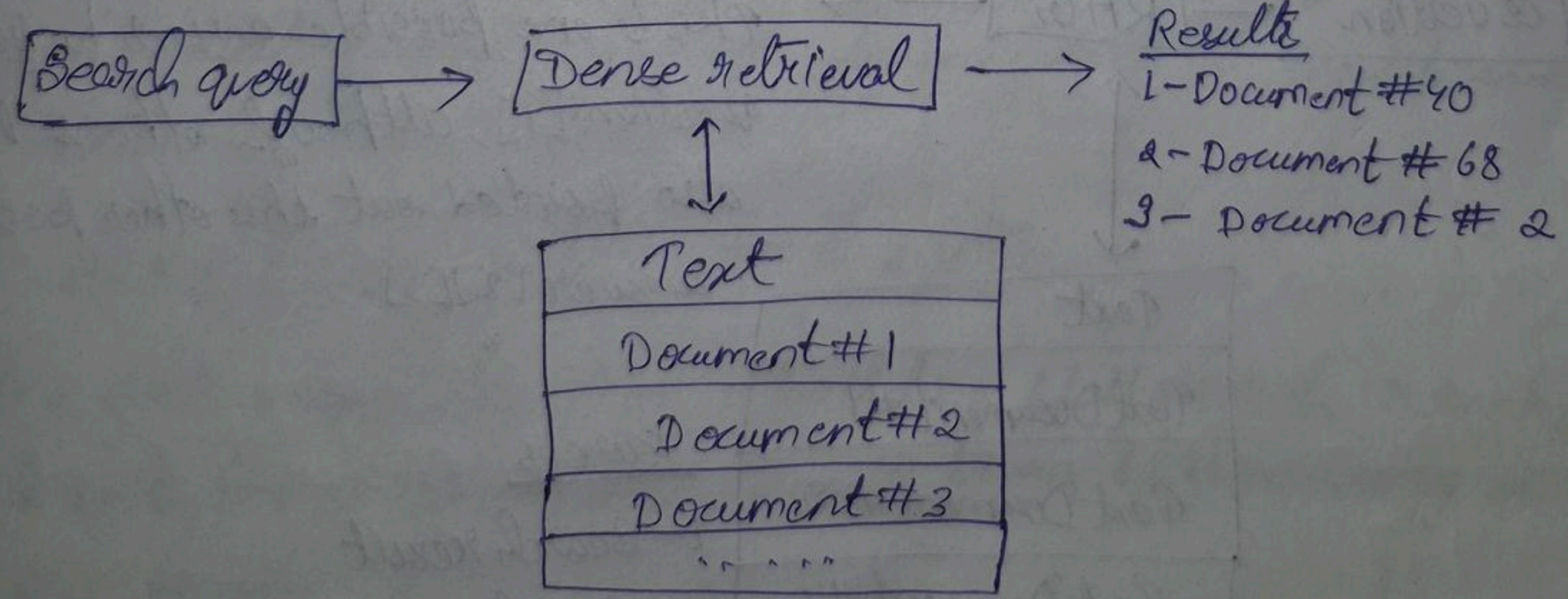
- This problem is known as 'hallucination' in model.

- To overcome ~~from~~ it to build system that can retreive the relavant information and provide it to the LLM to generate more actual answers. This is known as retreival augmented generation.

- The symentatic search in LLM classifies in 3 categories
  i> Dense retrieval
  ii> Reranking
  iii> RAG

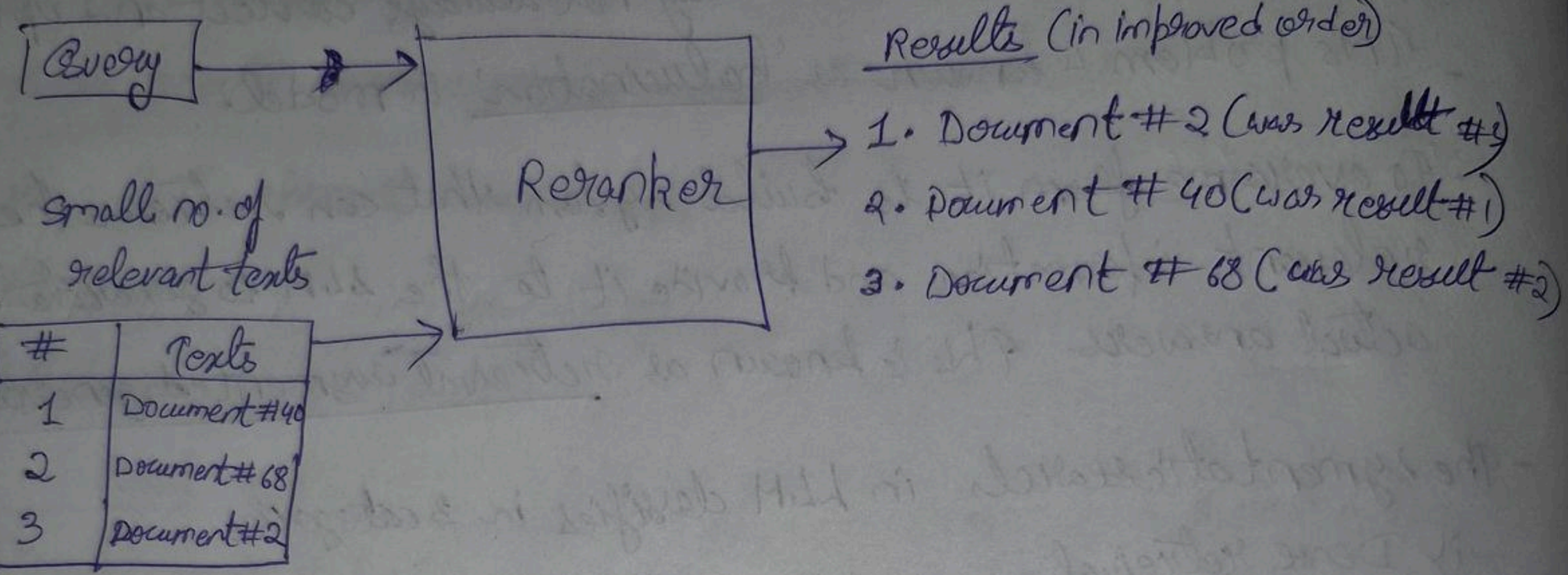
i> Dense retreival - This system relay on the concept of embeddings, means search the problem into retreiving the nearest neighbour of the search ~~fairi~~ query.

- The dense retreival takes a search query, then consult ~~to~~ from the datasets or texts and outputs is ~~some~~ set of relivant results.



- The dense retreival is one of the key concept of symentic search relaying on similarity of text embedding to retreive the relavant results.

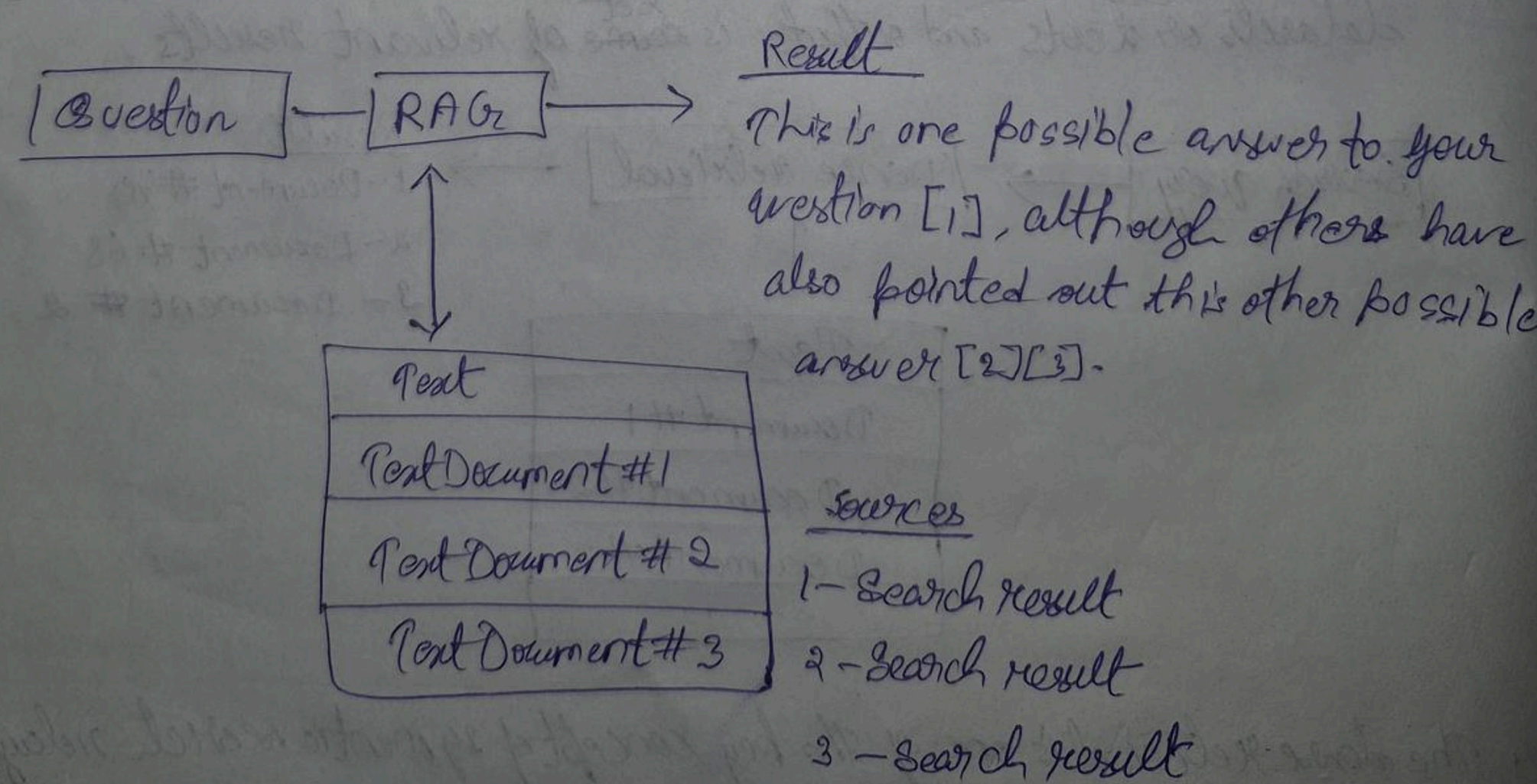
ii) Reranking - The search system often uses multiple steps for generating the result. The reranking language model is one of the ~~steps~~ tests or steps i.e used to calculate the score of the relevance subset of results against the query, then it puts the scores in improved ordered.



iii) RAG -

- To improve the LLM capability for the text generation led to a new type of search system that includes a model to generate an answer in response to a query.

- The generated search is subset of RAG system as it incorporates the search capabilities to reduce the hallucination increase the accuracy from the text.



Dense retrieval -

Symantic search with language models - The dense retrieval uses some steps for the numeric representations means when the query asked by the results then the semantic search finds which ~~text~~ text are more similar or nearer to that query.

- Acc. to that it will give the result.

Steps -

i) Get the text what we want to search.

ii) Embedding the sentences or text ~~chunk~~ chunks.

iii) Building the search index.

iv) Dense retrieval using semantic search. (Finding the ~~nu~~ numerical representations)

- Fine tunning the embedding model.

4/12

Reranking - A lot of organization have already been search system. The 1st step is to change the order of search result based on relevance to the search query. In further step the ~~new~~ initial results are reranked in 2nd stage to watching the improvement to the search results.

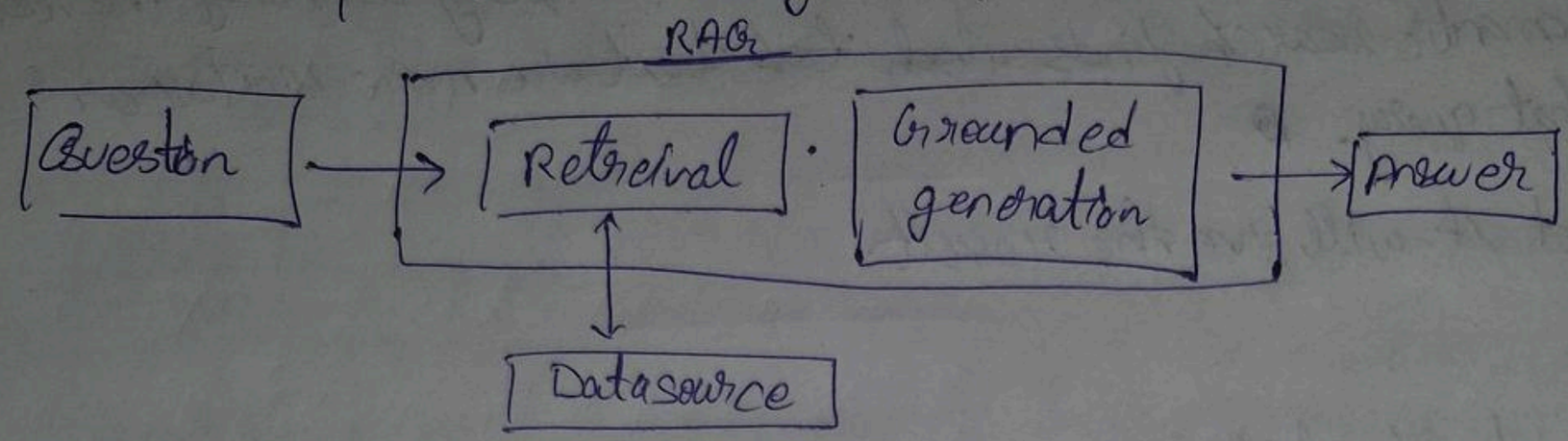
- The structure of rerank search system follows 2 stage search and the final result gives the highest relevance data at the top and lowest relevance is at the bottom.

- Eg: BERT follows 2 stage search system.

- One of the most common way of building LLM search rerankers is to present the query and each result to an LLM working as a cross encoder.

- The relevance score of each result gives which one should be present at the top.

RAG - It is a basic pipeline made of search steps followed by a grounded generation step where the LLM is prompted with the question and the information retrieve from the search result.



- The RAG system here incorporates the search capabilities in addition to generation capabilities ~~&~~ to they reduce the hallucination and ~~input~~ improve the accuracy.

- The grounded generation in LLM means the retrieve relevance information that fits the search result to give the accurate outputs.

- The generative search formulates answer and summarizes at the end of search pipeline while ~~all~~ citing its sources.

- The advanced RAG techniques use additional information such as
  i) Query re-writing
  ii) Multiquery RAG
  iii) Multihop RAG - A series of sequencial queries
  iv) Query routing
  v) Agentic RAG.

RAG evaluation - This is the major to evaluate the performance of RAG model. It generally follows 4 steps -

i) Fluency - Whether the generated text is fluent and cohesive.
ii) Perceived utility - Whether the generated answer is helpful and informative.

iii) Citation recall - The proportion of generated statement about the external world.

iv) Citation perception - The proportion of generated citation that supports the associated statement.

Also, the faithfulness and answer relevance are used to ~~major~~ measure the performance of RAG models.

CH-9 : Multimodal and LLMs

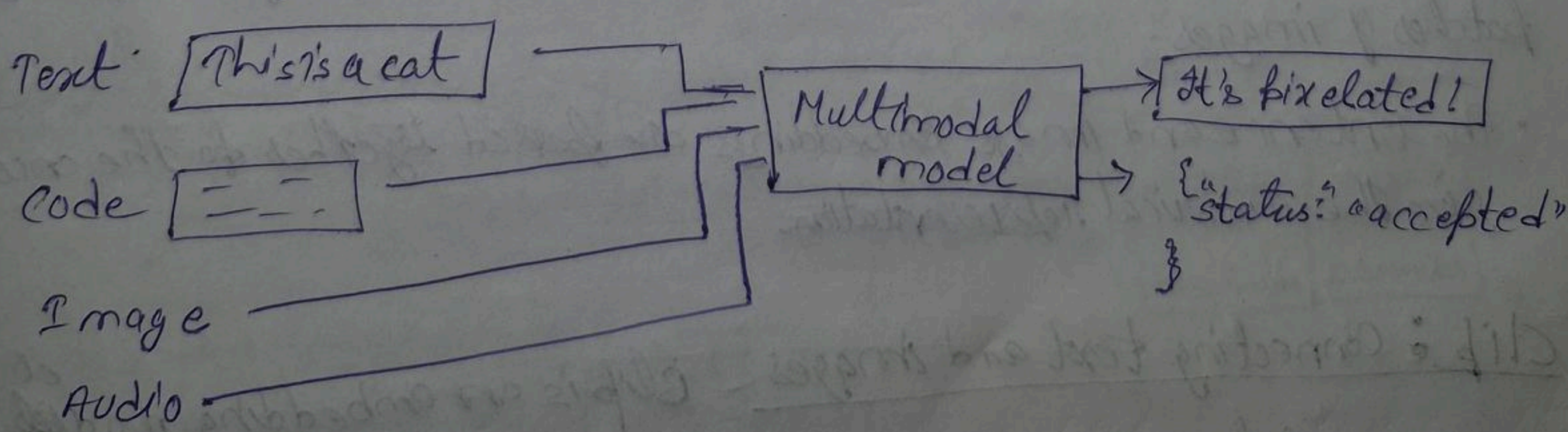
- There are various methods in LLM to breach the gap b/w textual and visual representations. In the mean time the multimodal helps to find ~~that gap~~ the solution for the gap in the transformers.
- The transformers in LLMs are used to convert the images into numerical representations.

- A model that is able to handle text, images, audios & videos is said to be multimodal.

9/12

A model that is able to handle ~~text~~ text, images, audios & videos is said to be multimodal.

Input modality



Transformer ~~for~~ for vision -

The transformer for vision are the models that converts images into numerical representation.

- This work achieve through the use of image encoders and batch embedding with allow the model to process the image as various scales.

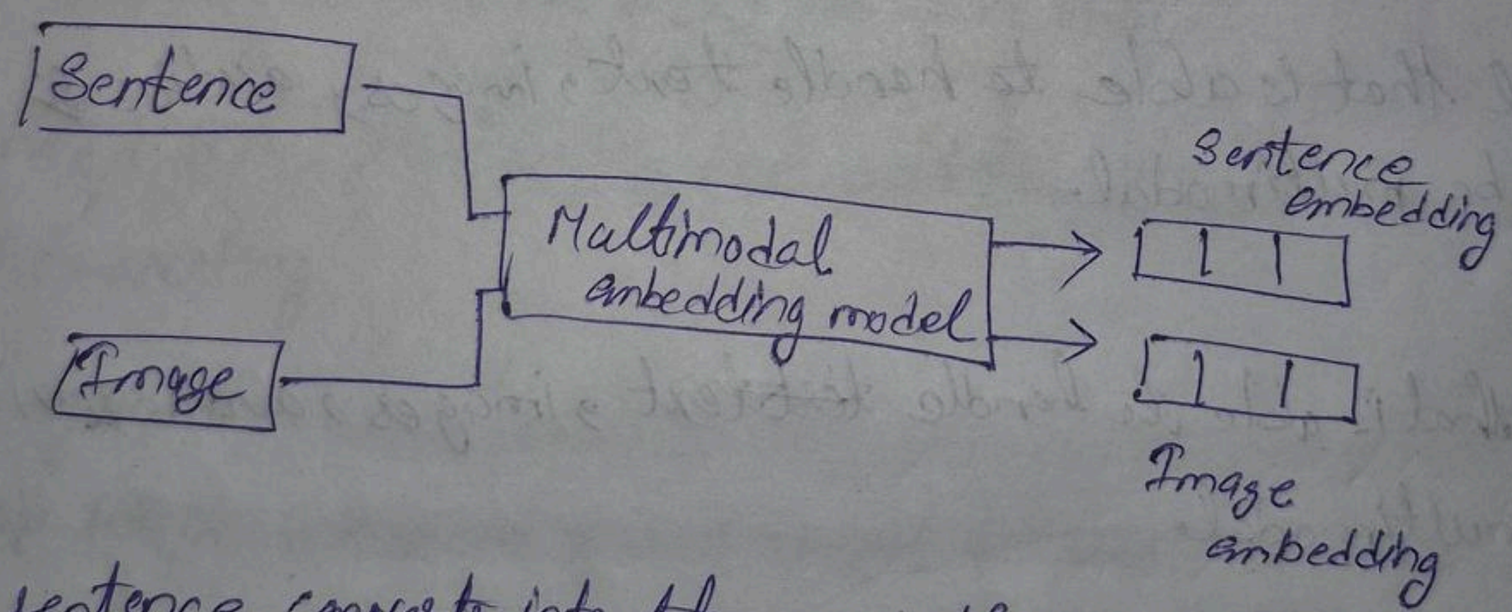
- The vision transformer (VIT) works in advancement of CNN for the image recognisation.

- The VIT used to transform unstructured data as image into numerical representations for variety of task like classification.

- Both the original and vision transformer takes the unstructured data, convert it into numerical representation and finally use that for classification task.

- The VIT converge the original image into small patches of images.

<u>Multimodal embeeding models</u> - In the case of multimodal embeeding model, multiple modalities create in the same vector space.



Here, the sentence converts into tokens and the image file convert into patches of images.

- The sentence and image embeddings are passed together to the encoder to give the numerical representation.

<u>Clip : Connecting text and images</u> - Clip is an embedding model that can compute embeddings of both images and texts.

- It follows 4 common tasks

i) <u>Zero shot classification</u> - We can compare the embeddings of an

class.

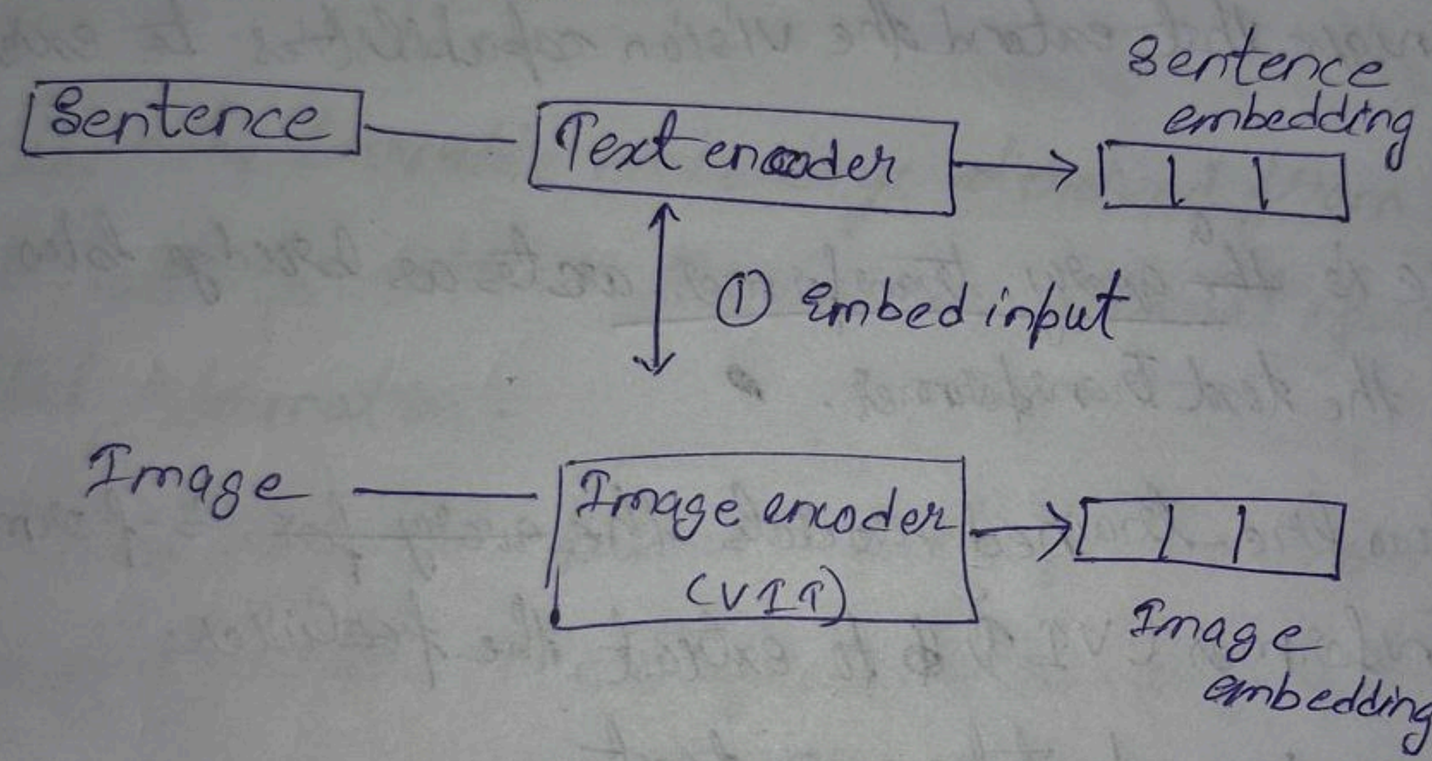
ii) Clustering - Cluster both images and a collection of keywords to find which keywords belongs to which sets of images.

iii) Search - across billions of images and text we can quickly find what relates to an input text or image.

iv) Generation - To drive the generation of images.

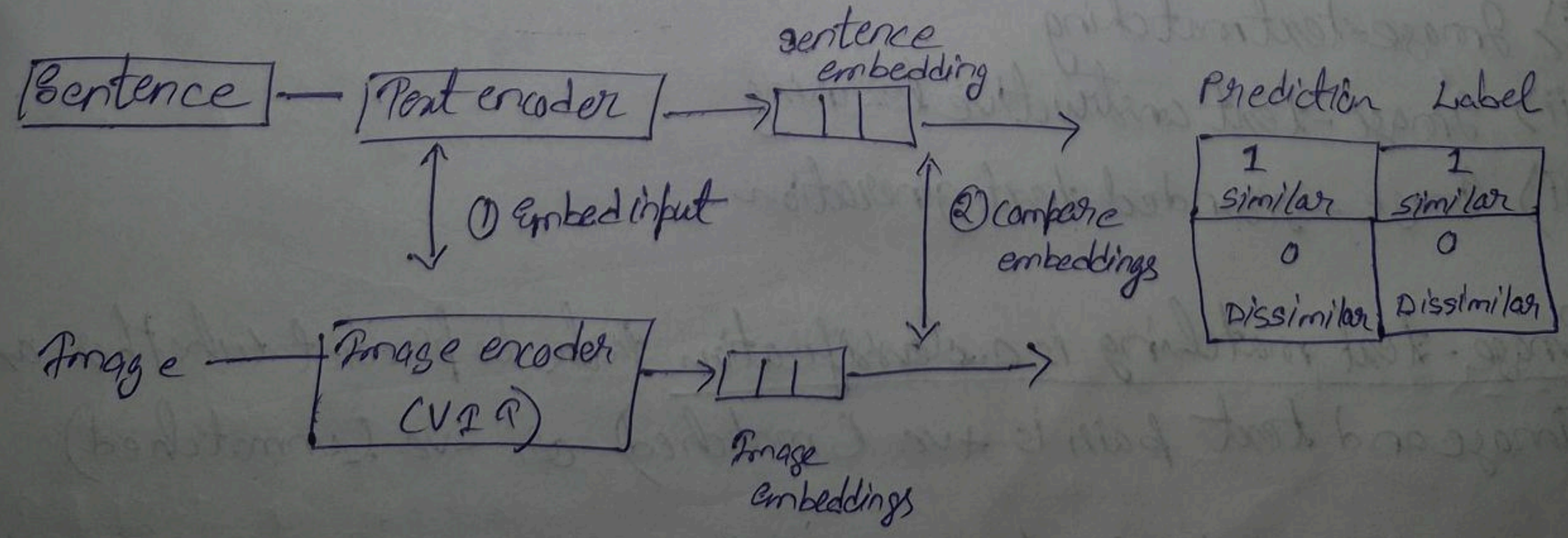
~~Step I~~

The 1st step of clip, both images and text are embeded using an image and text encoder.



In the 2nd step of training clip, used to find the similarity b/w the sentence and the image embedding is calculated using cosine similarity.

Cosine similarity - the angle b/w vectors (cosine angle) which is calculated through the dot product of the embeddings ÷ product of their length.



In the 3rd step of training clip, both text and image encoder are updated to match the ~~wor~~ what the intended similarity b/w them. These updates the embeddings such that they are closer in the vector space.

- For that the LLM model use an open source varient of clip known as <u>Openclip</u> for the experiments.

12/12

Blip - to :

Bootstrapping language image pre-training for unified vision language understanding and generation tool.

- To bridge the gap b/w two domains means (the text and image) here blip-to use a modular technique that extend the vision capabilities to existing language models.

- In the blip-to there is ~~the~~ a <u>query transformer</u> acts as bridge b/w vision transformer and the text transformer.

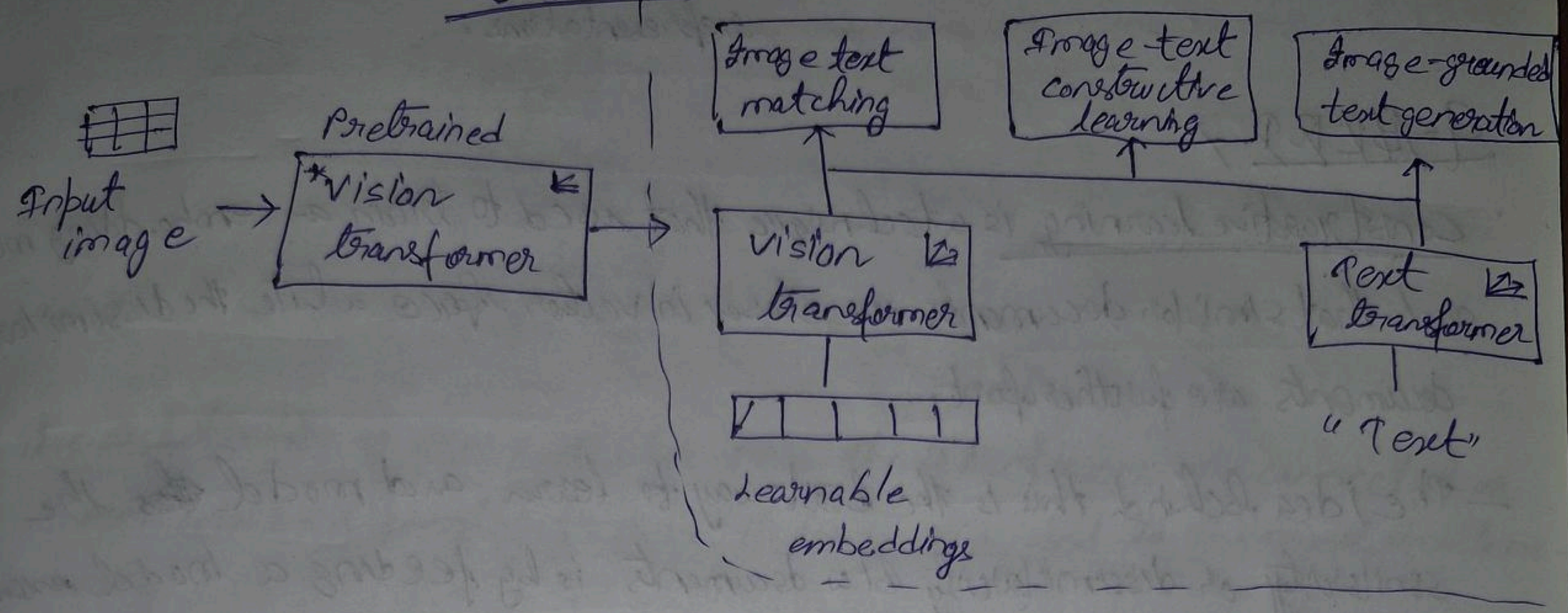
- To connect the two pre-trained models the ~~query for~~ Q-former uses
  i) An image transformer (VIT) to extract the features.
  ii) Text transformer to create tokens on text.

- The Q-former is used to represent both images and text in step 1. The output of frozen VIT is used together with its caption and trained on 3 ~~contractive~~ constructive ~~tab~~ task to learn visual and text representation.
  i) Image-text matching
  ii) Image-text constructive learning
  iii) Image-grounded text generation

- <u>Image-text matching is a classification task</u> to predict whether an image and text pair is +ve (matched) or -ve (unmatched)

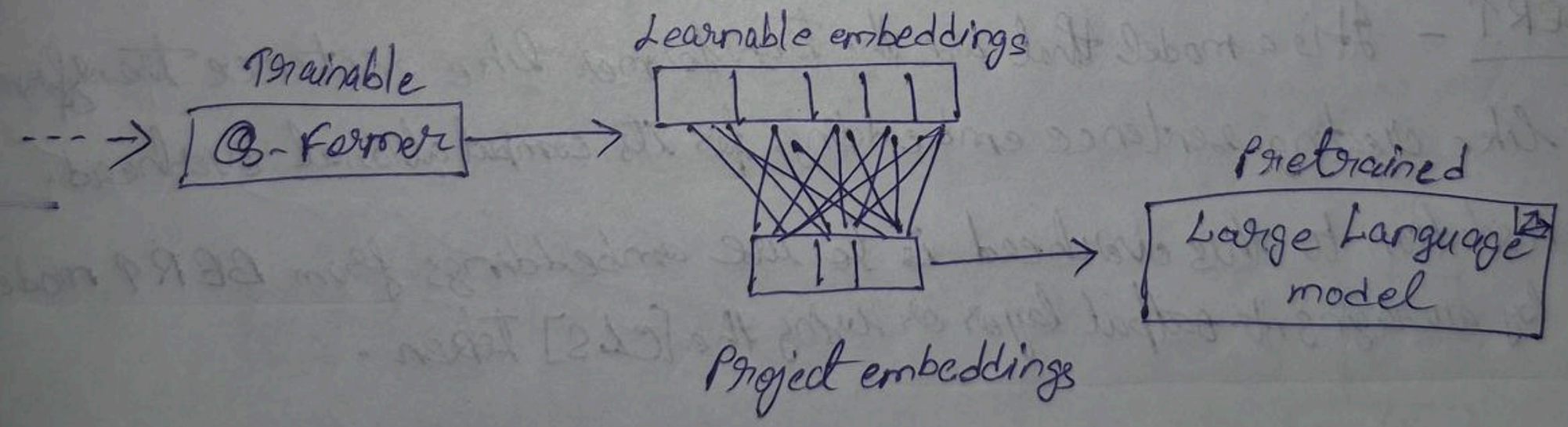
- <u>Image-text constructive learning</u> attends to align pairs of images and text embeddings such that they maximize their mutual ~~connection~~ information.

- Image-grounded text generation trains the model to generate text based on information extracted from the input image.



In step II - the learnable embeddings derived from step I contains the visual information in the same dimensional space as the corresponding textual information.

- The embeddings are passed in ~~forthe~~ further through a projection layer to the LLM model. The projected embeddings acts as ~~sub~~ soft visual prompts.



12/12

CH-10 Text Embedding Model

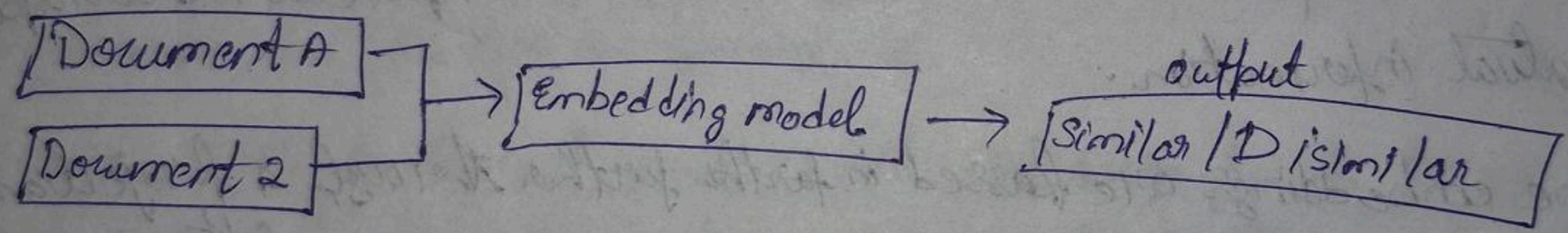
Creating text embedding models - It converts the textual data into numerical representations.

S-BERT :

Constrastive learning is a technique that used to train an embedding model such that similar documents are closer in vector space while the dissimilar documents are further apart.

- The idea behind this is the best way to learn and model the similarity or dissimilarity b/w documents is by feeding a model example of similar and dissimilar pairs.

Input



→ sentence-BERT

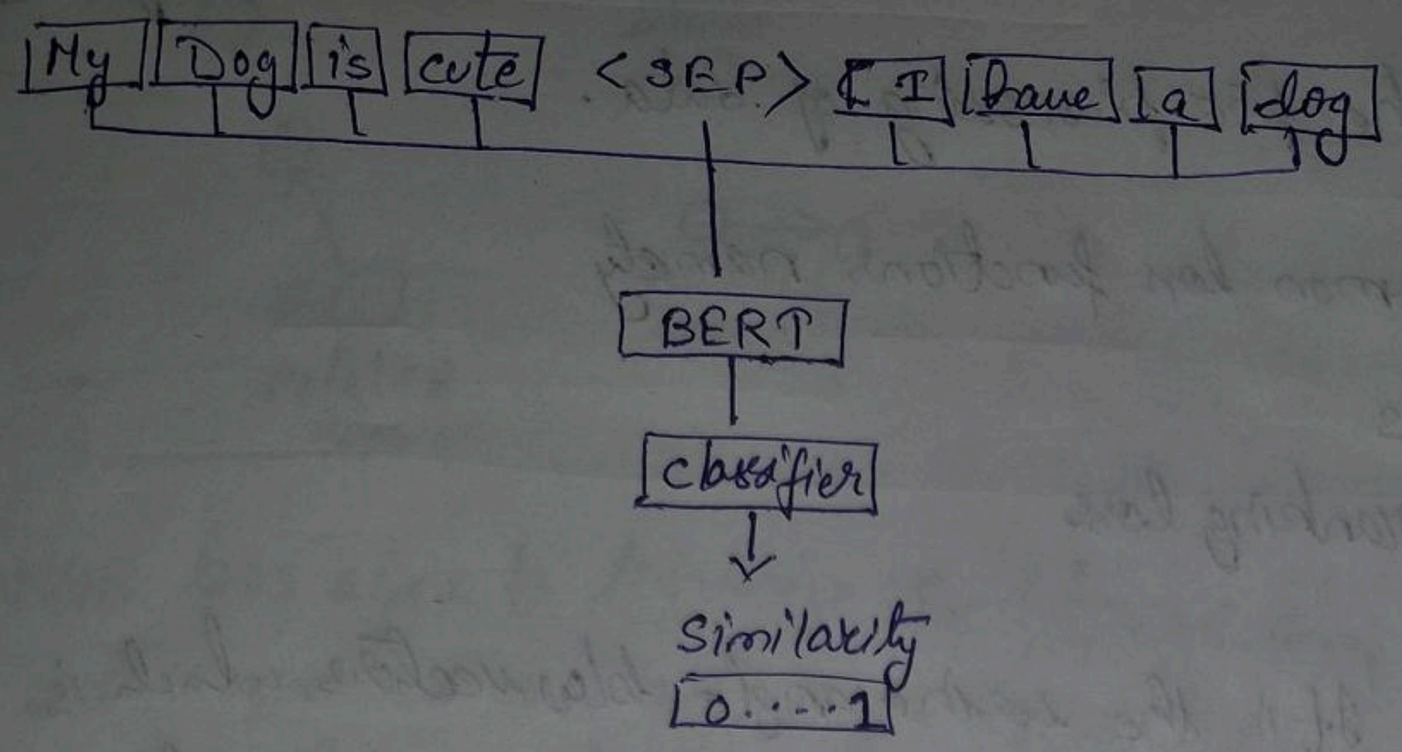
S-BERT - It's a model that use the transformer-like sentence transformer like creating sentence embeddings for its computational overhead.

- A solution to this overhead is to use embeddings from BERT model by averaging its output layer or using the [CLS] token.

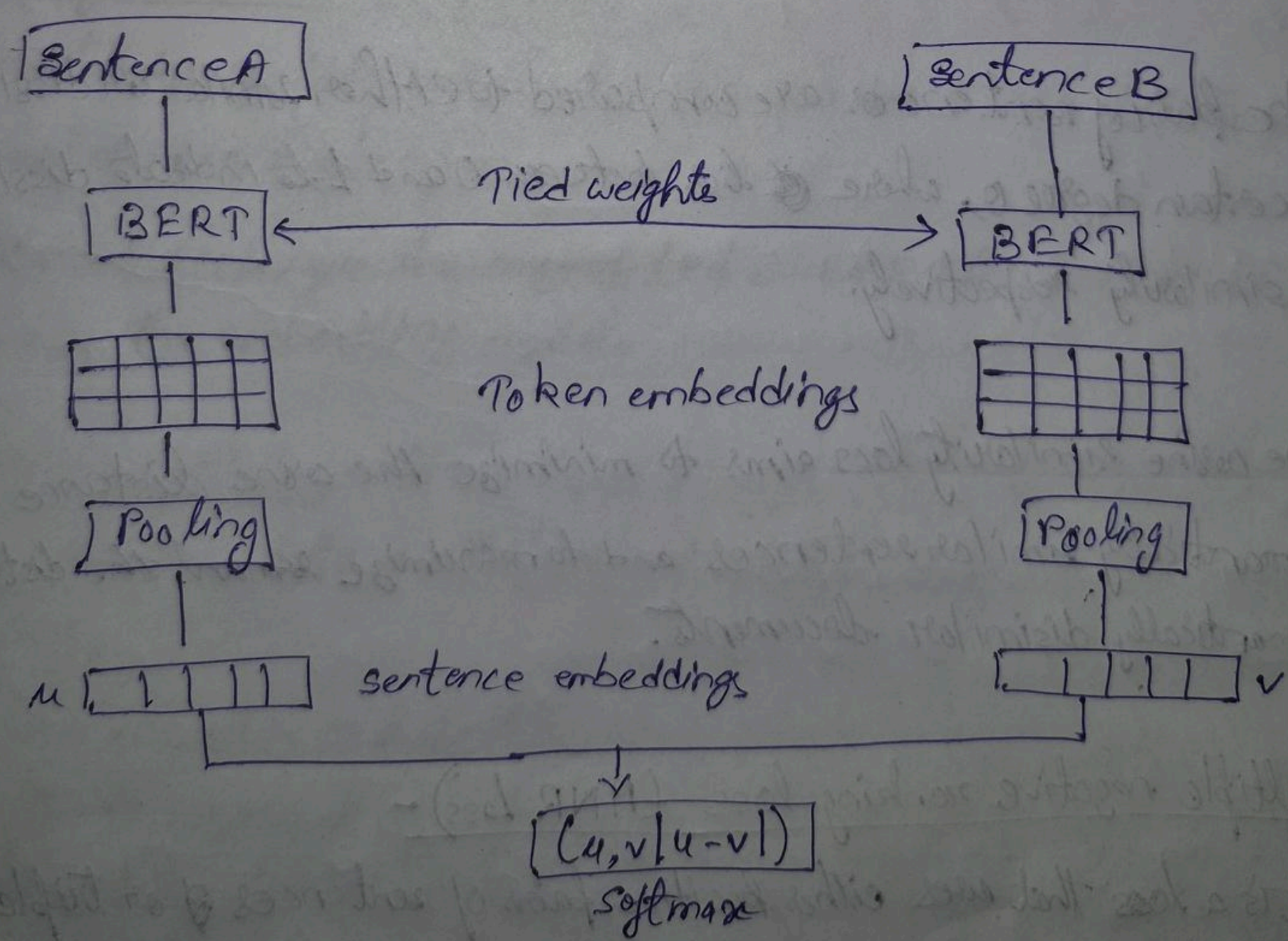
Cross encoder - is used to identify the similarity of sentences when fed to the model simultaneously.

16/12

S-BERT - is an embedding model for creating sentence embedding. Before sentence transformers the sentence embedding in S-BERT is used an architectural structure known as cross encoder.



- The architecture of cross encoder shows the 2 sentences are connected or concatenated separated by a <SEP> token then fed to the model simultaneously.
- The training after cross encoder the sentence transformer has been used ~~further~~ for the S-BERT model called as <u>bi-encoder</u>.
- The bi-encoder like a cross-encoder gives or follows contrastive learning.

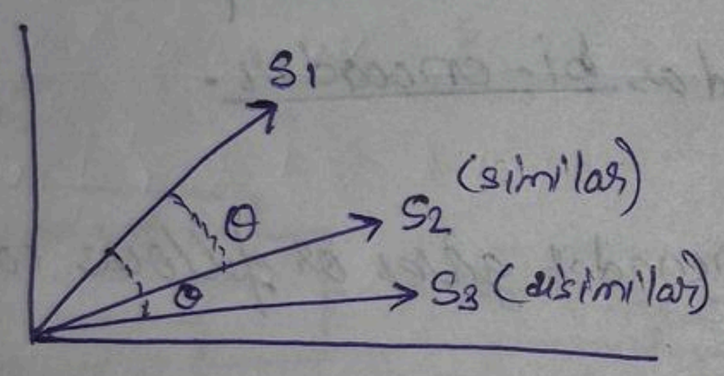


Loss function - When to train a model there might a chance of loss due to noise during the pre processing of data.

- There are 2 most common loss functions namely
  i) Cosine similarity loss
  ii) Multiple negatives ranking loss

Cosine similarity loss - It is the cosine angle b/w vectors which is calculated through the dot product of the embeddings divided by the product of their lengths.

$$\text{Cosine similarity}(s_1, s_2) = \frac{\|s_1\| \|s_2\| \times \cos\theta}{\|s_1\| \|s_2\|}$$



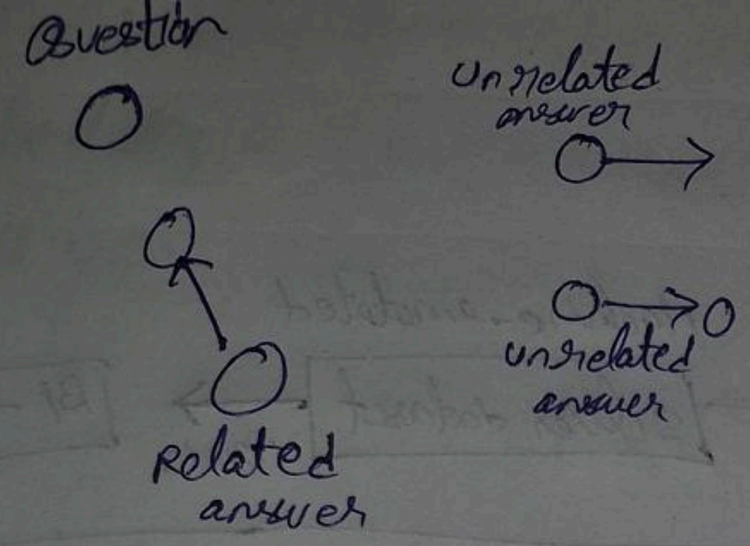
- The pairing sentences are compared to either similar or dissimilar to a certain degree θ, where lies between 0 and 1 to indicate dissimilarity & similarity respectively.

- The cosine similarity loss aims to minimize the cosine distance between semantically similar sentences and to maximize ~~semant~~ the distance between semantically disimilar documents.

Multiple negative ranking loss (MNR loss) -

It is a loss that uses either positive pairs of sentences or triplets that contain a positive pairs of sentences and an unrelated sentence.

- This unrelated sentence is called a negative and represents the dissimilarity b/w the positive sentences.



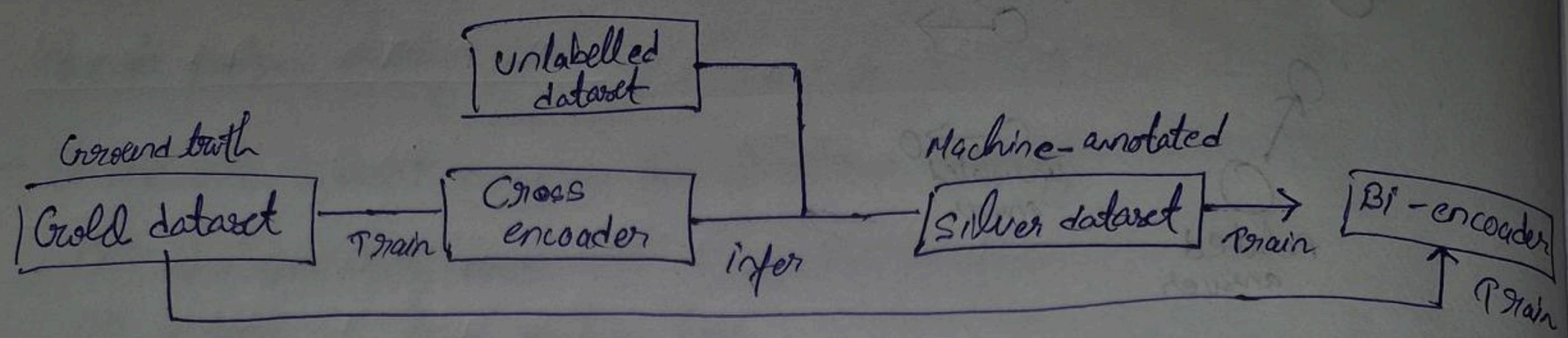
- MNR loss aims to minimize the distance b/w the related pairs of texts such as question and answer and maximize the distance b/w the ~~unrelated~~ unrelated pairs of texts i.e questions and unrelated answers.

- An easy negative is typically unrelated to both question and answer.
- A semi-hard negative has some similarities to the topic of the question and answer but is somewhat unrelated.
- A hard negative is very similar to the question but is generally the wrong answer.

Fine tunning an embedding model -

- The fine tunning of LLM model can be acheived using both supervised and unsupervised techniques.
- In supervised technique the augmented S-BERT has been used for fine tunning the embedding model.

- Augmented S-BERT - It is a method to fine tune the dataset when there is few label data available this procedure referred to as augmented S-BERT.
- The cross encoder architecture in S-BERT is slow but gives more accurate to augment and label a larger set of input pairs.
- These new label pairs are then used for fine tunning a bi-encoder in SBERT.



- Here, the augmented S-BERT works through training a cross encoder on small gold dataset then use unlabelled dataset to generate a large silver dataset finally both gold and silver dataset to train the ~~dataset~~. bi-encoder.

17/12

Fine tunning using unsupervised learning -

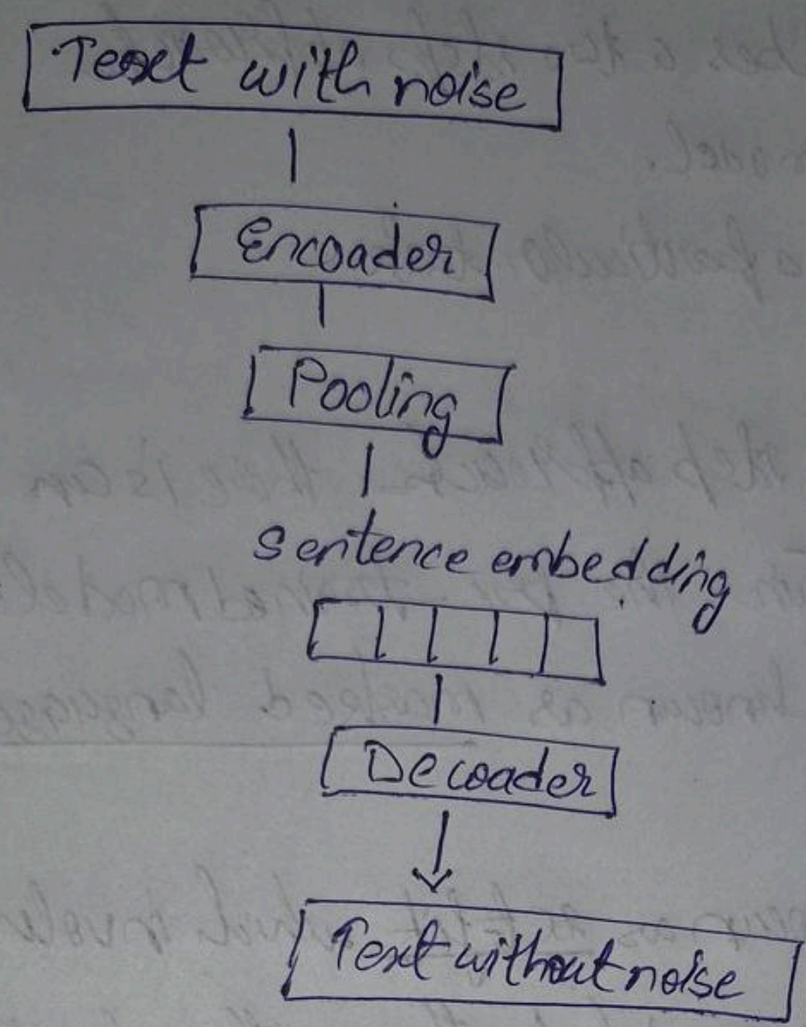
- The transformer based LLM model supports unsupervised learning means the data with no label and in the form of unstructure (class is unknown)
- To handle this there is a method known as ~~tsdae~~ TSDAE (Transformer based sequencial Denoising Auto Encoder).

- The TSDAE model is to create an embedding model with unsupervised learning. This method assumes no label data and require to artificially create the labels.

- The main idea behind the TSDAE method is that we add noise to the input sequence by removing a certain percentage of words from it.

- This damaged sentence is put through the encoder with a pooling layer on top of it to map it to a sentence embedding.
- From the sentence embedding a decoder tries to reconstruct the original sentence from the damaged sentence but without artificial noise.

- The main idea of TSDAE method is that the more accurate the sentence embedding is the more accurate the ~~reconstructed~~ reconstructed sentence will be.



<u>Fine tunning representation models for classification</u> -

- The fine tunning in representation model for the supervised learning use 2 methods
  i) Task specific model to perform the classification.
  ii) Embedding model to create the embeddings.
- The fine tunning for the BERT model uses label data in case of supervised learning and the classification heads is feed forward neural network.

<u>Few shot classification</u> - It's a technique in supervised classification where we have a classifier that learn target labels based on only a few label examples.

- We only use limited no. of labelled data points to learn the ~~though~~ term means the target class.

~~Masked language~~

Masked language modelling - In general a pre trained model is fine tunned from scratch on target task to perform the classification.

- This process describes a two step approach.
  i) Pre trainning a model.
  ii) Fine tune it for a particular task.

- Instead of these 2 step approach there is an another step that continues to pre-train the pre-trained model before fine tunning it on the target task i.e known as masked language model.

Setfit -

- There is a method known as setfit which involves fine-tunning a pre-trained embedding model together with a classification head using limited labelled data.

- This is a few shot classification technique.

Fine tunning the generative models -