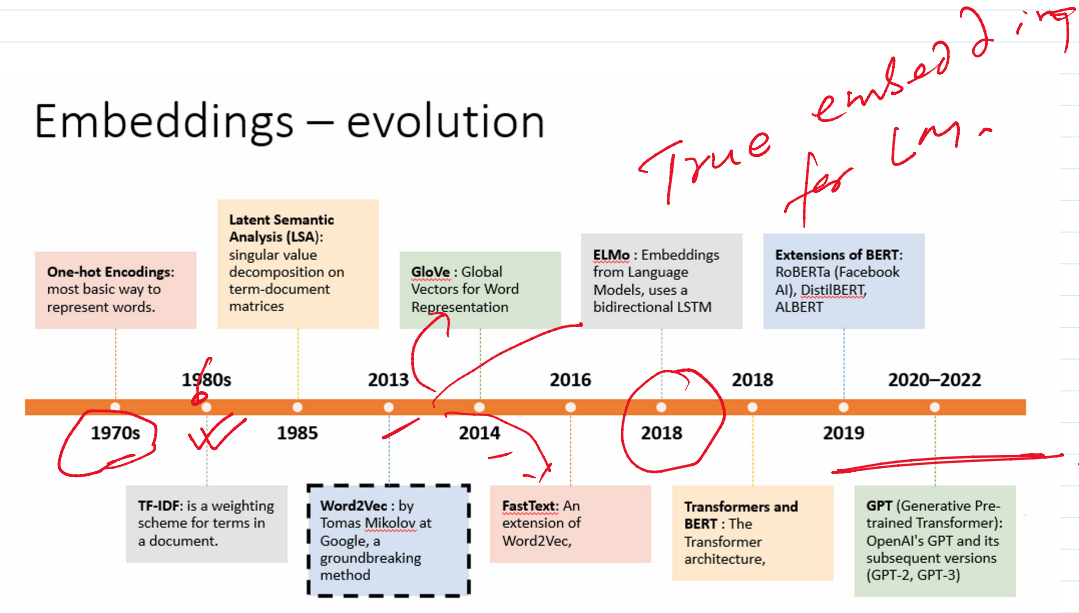
Day - 5 (Sat, 19-OCT)

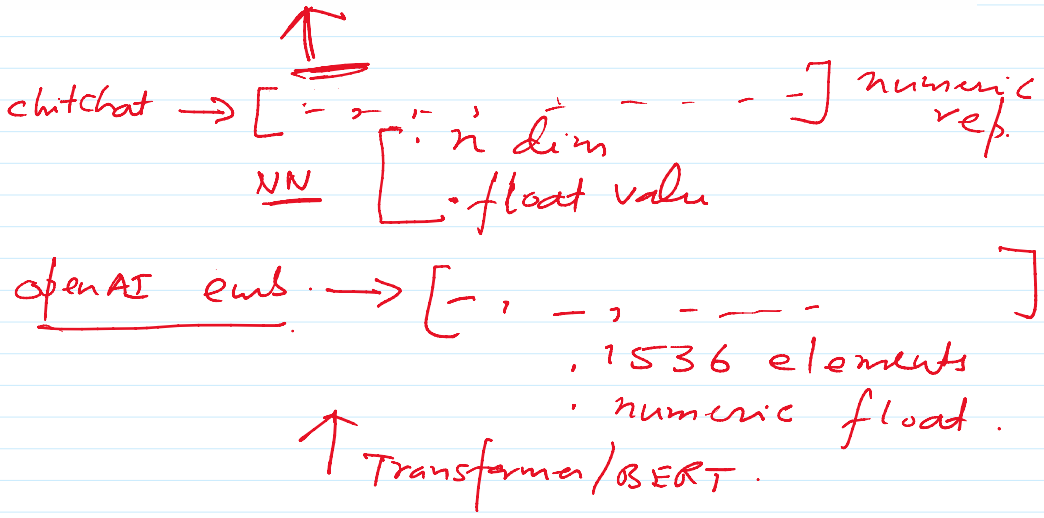
15 October 2024 22:21

Topics for today

- Recap (vectors)
- Overview on embedding models
- DEMO: Embedding layer in TF/Keras
- DEMO : classification IMDB data using default embedding layer of keras
- Detailed understanding of word2vec (first true semantic embeddings - 2013/14)
- DEMO : Generate word vectors using Gensim
- DEMO : Example use cases with Gensim word vectors using simpsons dataset
- Limitations with word vectors
- Vector Databases
  - Detailed walk thru of vector DB (ChromaDB)

- Detailed walk thru on OpenAI Embeddings
- Overview on Transformer Architecture
- Tokenization in Transformers





What are Embeddings?

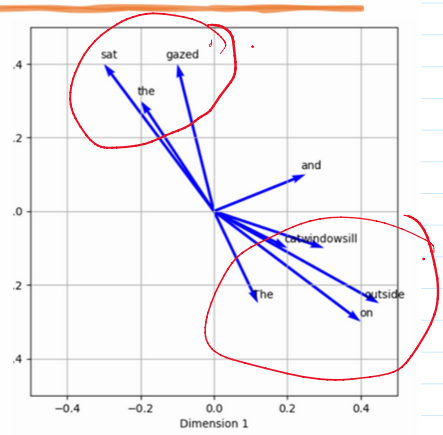
- embeddings are representations of data in a continuous vector space.

- Example
  - Sentence 1: "Natural language processing is fascinating."
  - Sentence 2: "Machine learning algorithms are powerful."

  - Sentence 1 embedding: [0.2, 1.5, 0.8, -0.3, 2.1, -1.2, 0.9, -0.5]
  - Sentence 2 embedding : [1.0, -0.7, 2.3, 0.1, -1.8, 0.6, 1.2, -2.0]

Illustration

- representations capture the semantics and relationships inherent in the original data.
- Example : "The cat sat on the windowsill and gazed outside."
- each word is represented by a corresponding vector in an 8-dimensional float array.
  - "The": [0.12, -0.45, 0.67, 0.89, -0.23, 0.56, -0.78, 0.91]
  - "cat": [0.34, -0.12, 0.56, 0.78, -0.45, 0.67, -0.23, 0.89]
  - "sat": [-0.56, 0.67, -0.23, 0.91, 0.12, -0.45, 0.78, 0.56]
  - "on": [0.78, -0.56, 0.12, -0.45, 0.91, 0.67, -0.23, 0.56]
  - "the": [-0.45, 0.56, 0.89, 0.12, 0.67, -0.23, 0.78, 0.91]
  - "windowsill": [0.67, -0.23, 0.56, 0.78, -0.45, 0.12, -0.91, 0.34]
  - "and": [0.56, 0.23, -0.67, 0.12, -0.45, 0.78, -0.91, 0.34]
  - "gazed": [-0.23, 0.78, 0.12, 0.91, -0.45, 0.56, -0.67, 0.34]
  - "outside": [0.91, -0.45, 0.78, 0.56, -0.23, 0.67, -0.12, 0.34]
- Each word's representation captures its semantic meaning and its relationships with other words in the sentence.



Similar vectors (words)

dog	sheep	november	jerusalem
• cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig	• cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock	• october, december, april, june, february, july, september, january, aur march	• tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ram safed

Corpus (wiki / bio - related)

Lots... (millions)

bio
months
animal name

⇒ 1) tokenization (global)

⇒ Paracetamol ↔ 4 spoon
ER

word2vec

1) apple (fruits) X
apple (US ---) X

2) position X
(limited)

Bert embedding

✓ multi head attention.
✓

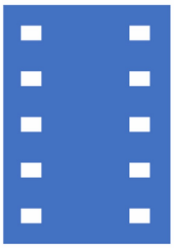
✓ (positional enc)

Why are embeddings Important in AI?

- **Semantic Relationships:**
- Example:
- if you perform the vector arithmetic:
  - vector("king") - vector("man") + vector("woman"),
  - resulting vector is often close to vector("queen").

queen.

Dimensionality Reduction



- Example:
- Consider a movie recommendation system with over 10,000 movies.
- One-hot encoding each movie would lead to 10,000-dimensional vectors.
- By using embeddings, we can represent each movie in, say, a 50-dimensional space, massively reducing the data's dimensionality while preserving the relationships between different movies.

token: [- - - - - -]
50
100
1536

TF-IDF; 10,000
20000 -
[- - - - -]
20000
• o/f
• Computational complexity

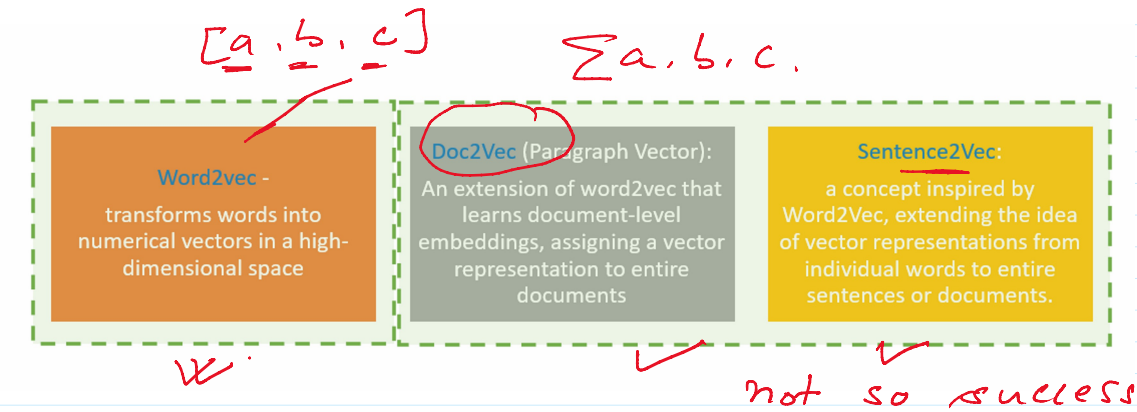
Handling Categorical Data

- Example:
- In e-commerce, you might have categorical data like product IDs or user IDs.
- Directly feeding such IDs to a model isn't meaningful.
- Instead, by representing each product or user as an embedding, the model can understand user preferences or product similarities, enhancing recommendation quality.

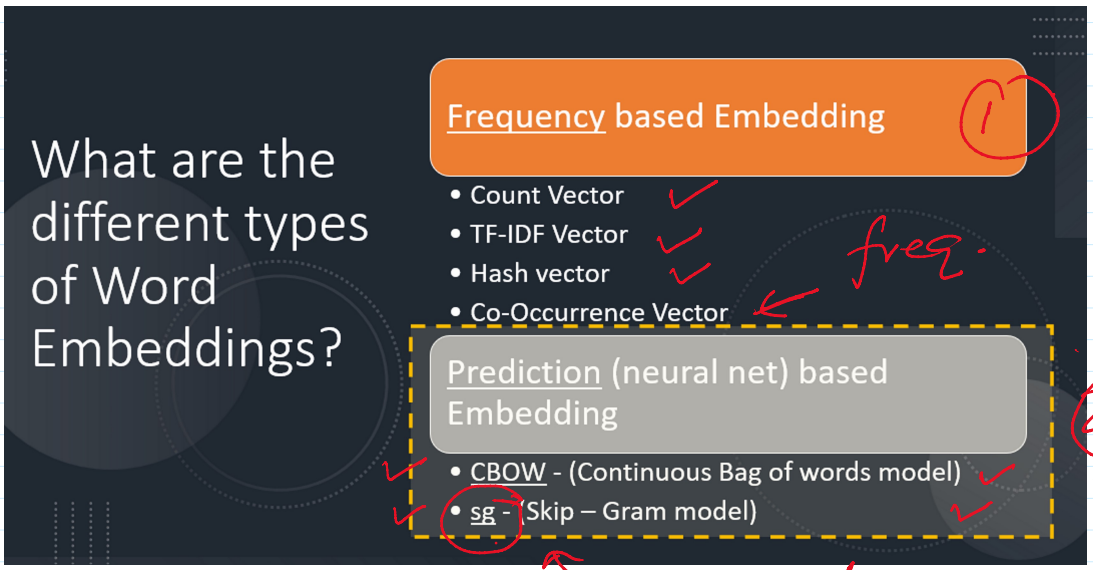
X5
A -1 — [- - - - -]
B -2 — [- - - - -]
C -3 — [- - -]

Word Embeddings

- Word embeddings represent words as dense, continuous-valued vectors that capture semantic relationships between words.
- Techniques like Word2Vec, GloVe, and FastText generate word embeddings based on word co-occurrence patterns.
- [above are examples of transfer learning]



not so successful.



vectors

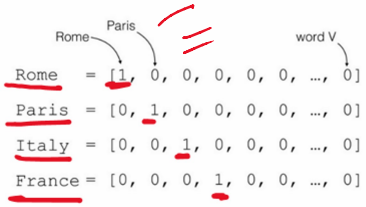
freq.

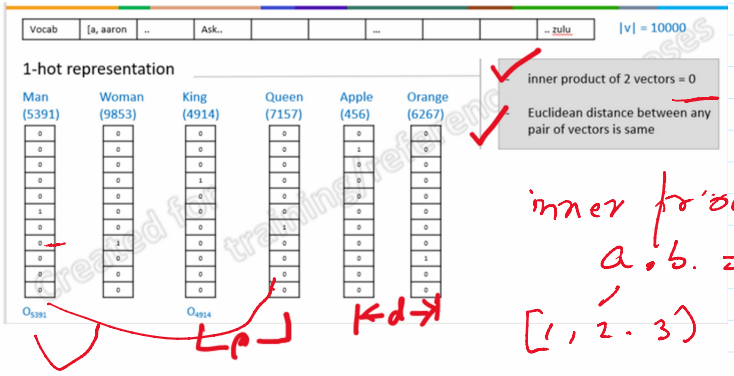
emb.

word2v.

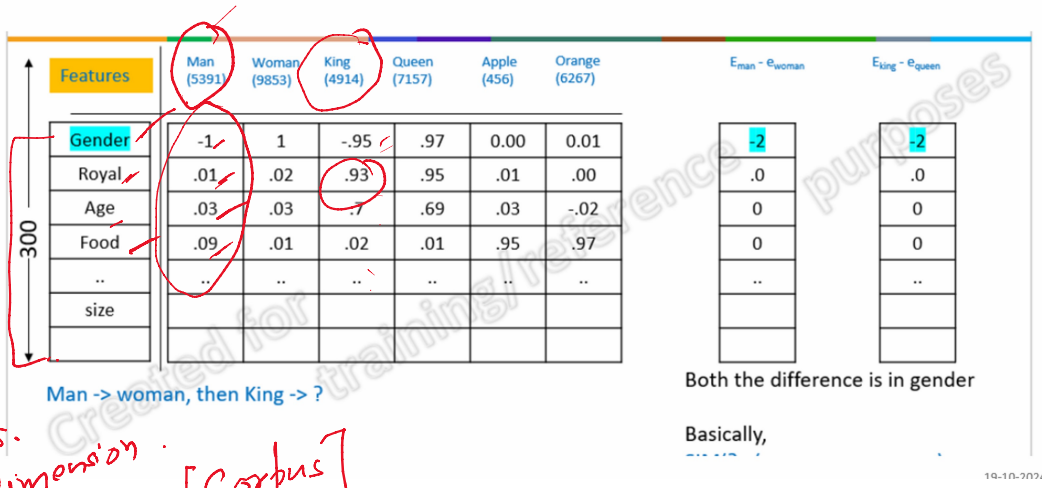
rare words

Representing text as numbers (one hot encoding - OHE)

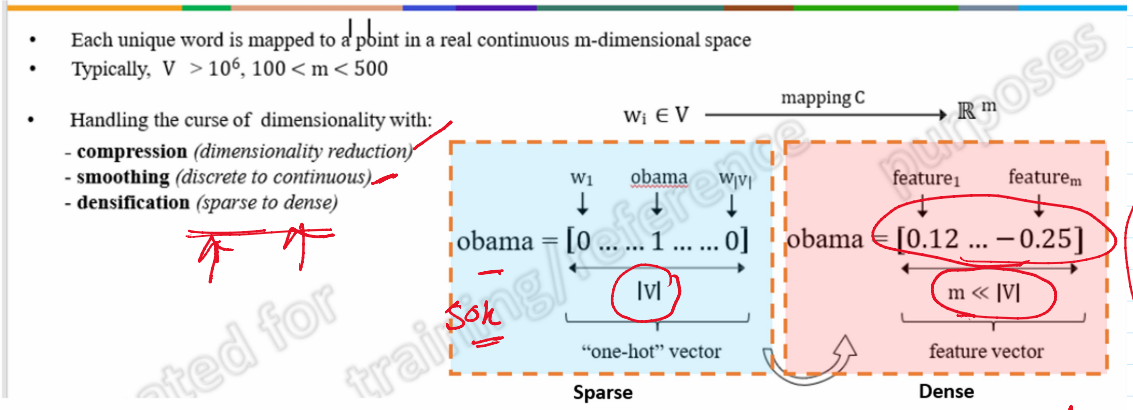


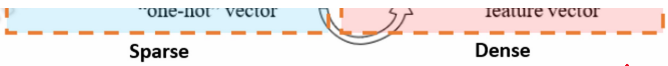


Featurized representation



word embedding - sparse vs dense





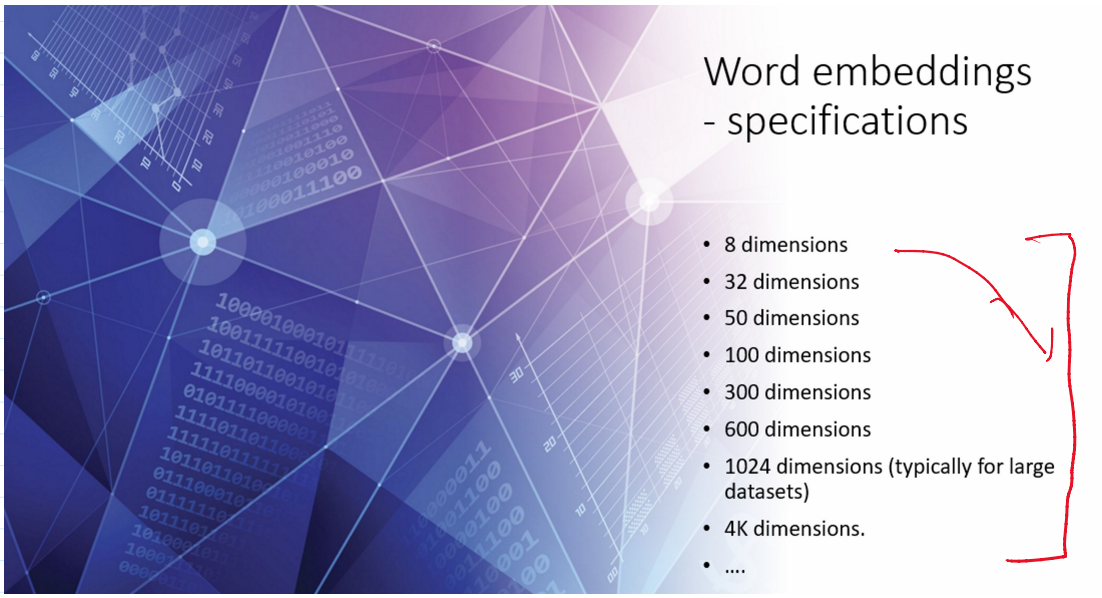
Intuitive view

sales = adv + brand pos + mrkt

$\Rightarrow y = m_1 x_1 + m_2 x_2 + m_3 x_3$

(1) forward prop. (2) back prop.

emb. [ ]



[. . . . .]

0
0
0
0
0
0
0

Ways to build word embeddings

training your own word embeddings

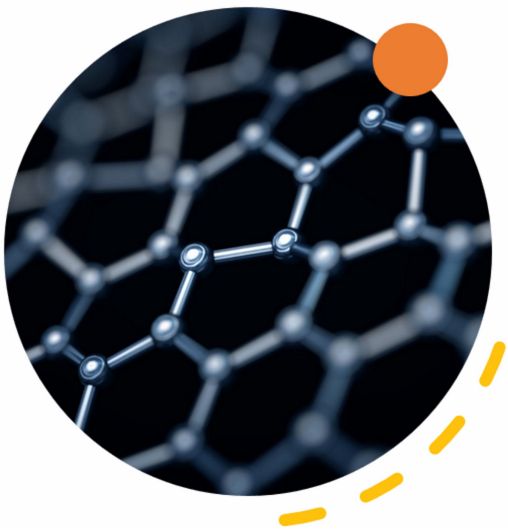
Using large amounts of unannotated plain text, learns relationships between words.

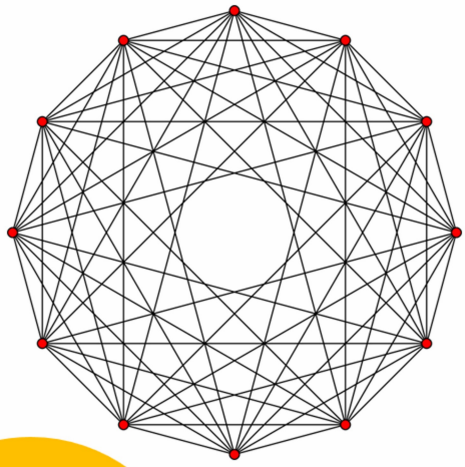
using a 3 layer shallow NN

pre-trained embeddings available such as GloVe, fastText and ELMo

Also called transfer learning

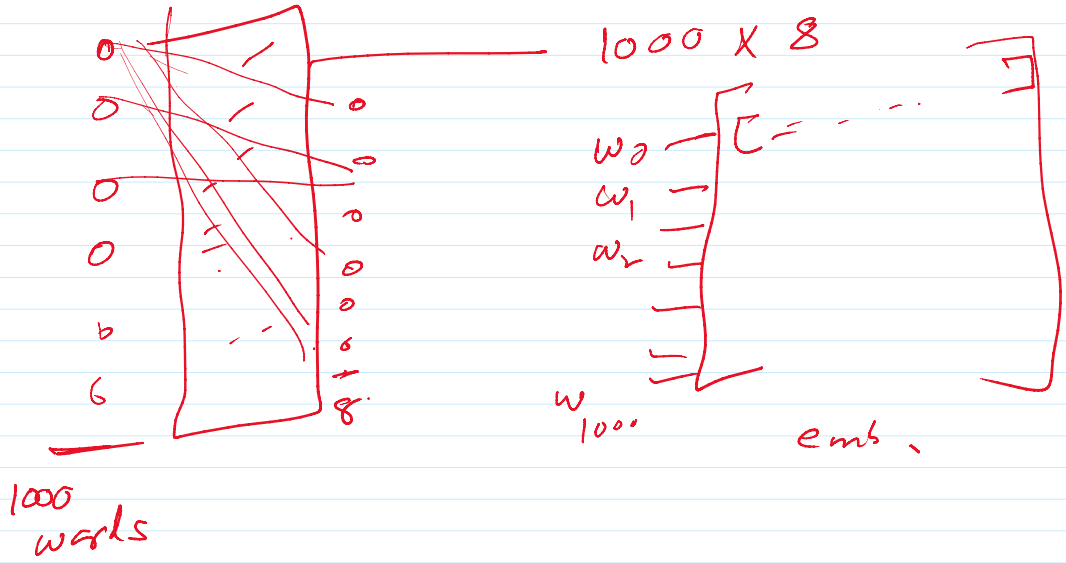
extensions of Word2Vec such as Doc2Vec and the most recent Code2Vec where documents and codes are turned into vectors.



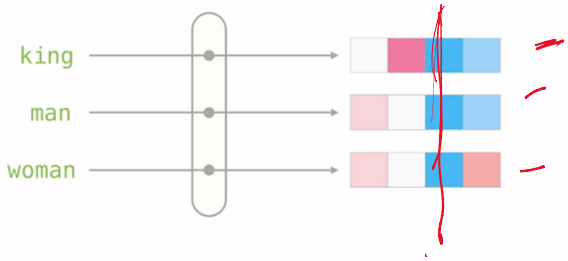


Word2vec

- is a group of related models (extensions etc)
- models are shallow, 3-layer neural networks
  - input layer
  - hidden layer
  - output layer
- takes as its input a large corpus of text and produces a vector space,
- typically, of several hundred dimensions, with each unique word in the corpus being assigned a corresponding entry in the vector space.
- Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space



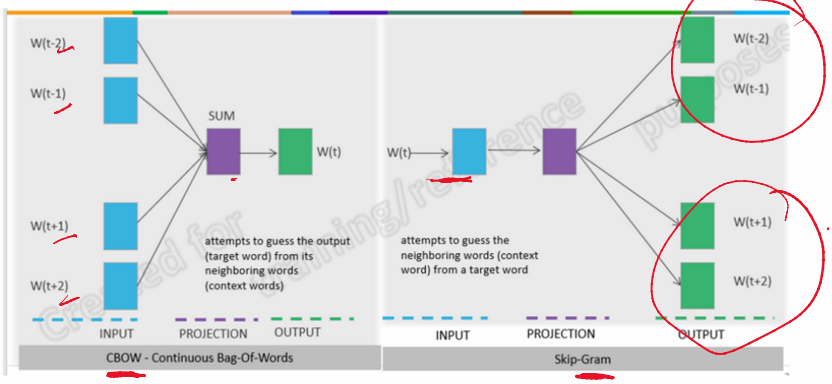
Example - how the embeddings look like



[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

word embedding for the word "king"

Two forms of NN for word2vec



anticipate a given word by considering its surrounding context words.

forecast context words by analyzing the present word.

Example (N=2)

The quick brown fox jumps over the lazy dog.

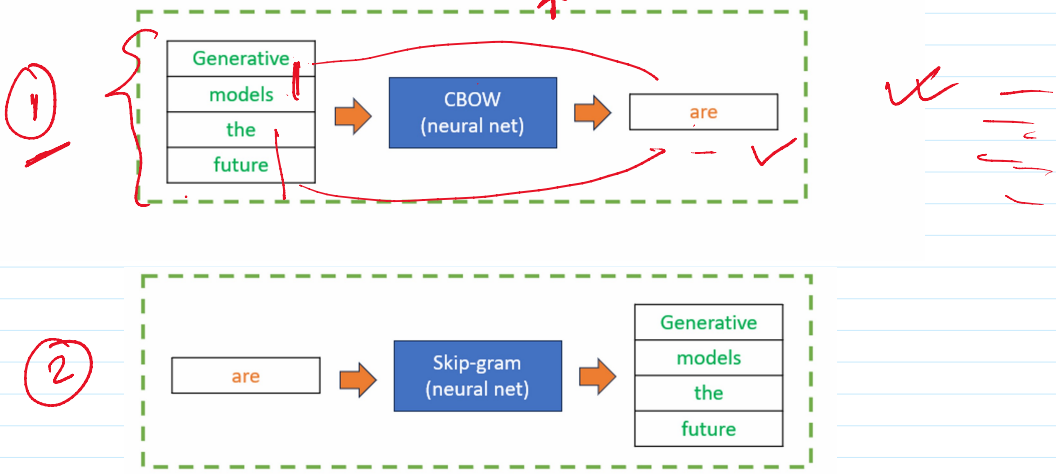
Left context words

Right context words

Target word

Example (data for the model)

sentence : "Generative models are the future of AI"



The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog

The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog
The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog
The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog
The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog

$y_{c==1}$ $y_{c==2}$ x_k $y_{c==3}$ $y_{c==4}$

The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog
The	Quick	Brown	Fox	Jumps	Over	The	Lazy	dog
The	Quick	Brown	Fox	Jumps	Over	The	Lazy	Dog

data preparation (sliding window, N=2)

- Suppose we have a sliding window of a fixed size moving along a sentence:
  - the word in the middle is the "target" and
  - those on its left and right within the sliding window are the context words.

Data samples for the NN of word2vec

- Given the sentence: "He says make India great again."
- Windows:
  - For each center word, the window would consist of 3 words to the left and 3 words to the right
    - (when available):

1.[-, -, -, **He**, says, make, India]
2.[-, -, He, **says**, make, India, great]
3.[-, He, says, **make**, India, great, again]
4.[He, says, make, **India**, great, again, -]
5.[says, make, India, **great**, again, -, -]
6.[make, India, great, **again**, -, -, -]

Let's generate training samples using the first window

Given the window: [-, -, -, **He**, says, make, India]

- Skip-Gram Training Samples:
- For the center word "He", the surrounding words in its window are used as context:

Input (center word)	Output (Context)
He	says
He	make
He	India

- CBOW Training Samples:
- Using the surrounding words in the window, we try to predict the center word "He":

Input (Context words)	Output (Center word)
says, make, India	He

Training samples

Skip-Gram Training Samples:

for each center word, we use it as the input to predict its context (surrounding words).

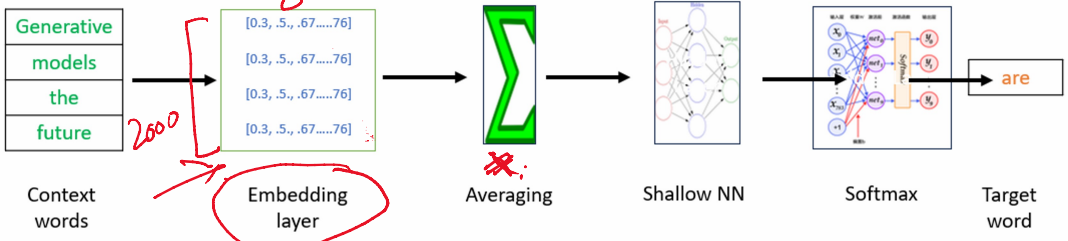
Input (Center Word)	Output (Context Word)
He	says
He	make
He	India

CBOW Training Samples:

In CBOW, we use the surrounding words as input to predict the center word.

Input (Context Words)	Output (Center Word)
says	He
make	He
India	He

Architecture



Context words

CBOW uses a fixed-size context window (N words before and N words after the target word).

Embedding layer

The context words are converted into their respective embeddings.

Averaging

embeddings of context words are aggregated (typically averaged) to form a single vector representation.

Shallow NN

hidden layer with nonlinear activation functions

loss functions include categorical cross-entropy

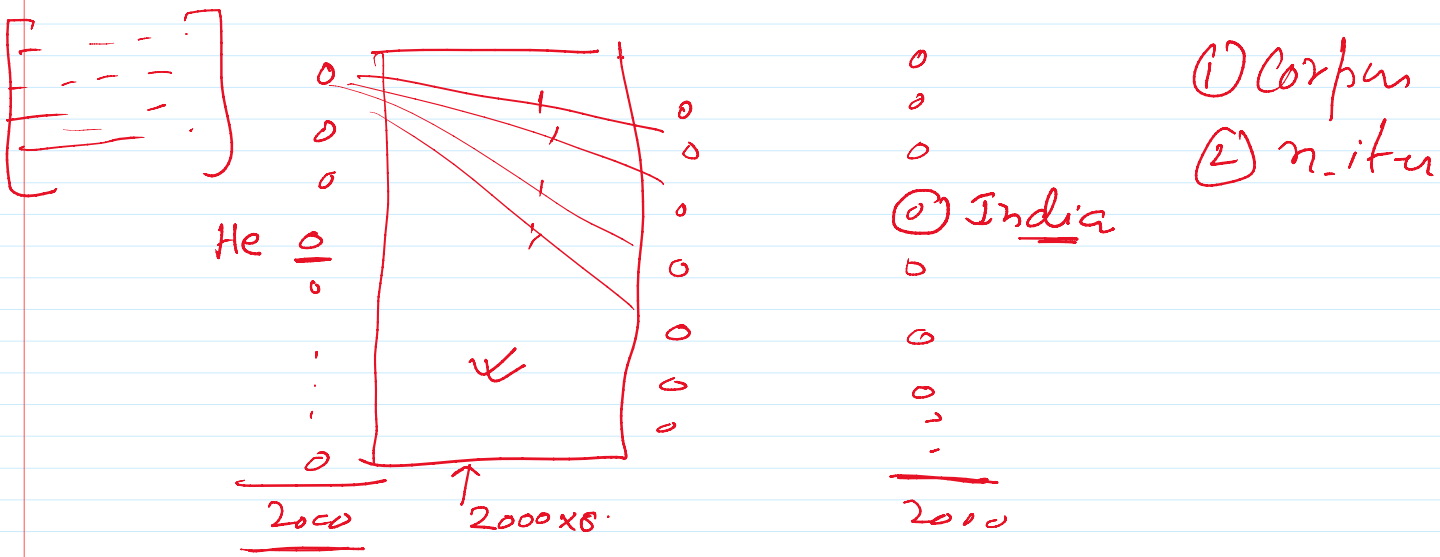
Softmax

The hidden layer's output is connected to the output layer, which produces a probability distribution over the vocabulary.

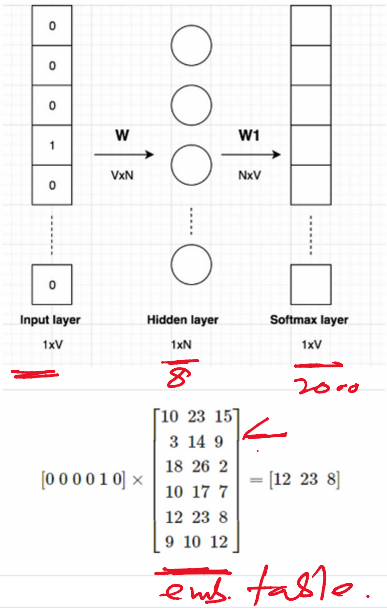
Softmax activation function is applied to convert the output into probabilities.

Target word

"adjusted" table → embs. table.



1) Corpus
2) n_items



Extraction of Word Vector:

- After training, the word vectors (or embeddings) for words can be extracted from the input weight matrix W.
- Specifically, the row corresponding to each word in this matrix represents its embedding.
- For example, to get the vector for the word "says", you would look at the 5th row (from our previous example) of matrix W.

Shortcomings of a typical w2v

Contextual information

Word2Vec creates fixed embeddings for each word, which means that the same word can have the same representation regardless of its meaning in different contexts.

Contextual embeddings, like those produced by models such as BERT or GPT or LLMs, can capture word meanings more accurately in context.

Cold start problem (OOV words)

Word2Vec does not provide a solution for new or unseen words.

cannot generate embeddings for words that were not in the training corpus, which can be a problem in dynamic and evolving language environments.

Lack of sub word information

- Word2Vec treats words as atomic units and does not consider sub word information.
- This means that it may struggle to handle out-of-vocabulary words, misspellings, or rare words that do not appear in the training corpus.
- Sub word models like FastText address this limitation

FB

Polysemy and homonymy

Word2Vec may struggle to disambiguate between words with multiple meanings (polysemy) or distinguish between different words that are spelled the same (homonymy).

Contextual embeddings can handle these cases better.

Examples ... next

Examples

Polysemy

(Words with Multiple Meanings):

"Bank": Word2Vec may represent "bank" (financial institution) and "bank" (side of a river) with the same vector

Sentence 1: "I need to deposit money at the bank."

Sentence 2: "I saw a beautiful sunset by the river bank."

Homonymy

(Same-Spelled Words with Different Meanings):

"Lead": Word2Vec may not distinguish between "lead" (to guide) and "lead" (the metal).

Sentence 1: "She will lead the team to victory."

Sentence 2: "The pencil is made of lead."

BOW/TF-IDF/ → variation → cold start ⇒ Keyword based search [Elastic search, Solr

Word2Vec → ✗ Sentiment classification ✗

bert/trf.

a, b, c, d

[- - - -]

[0 0 1 0] (f).

c f

ground work → openAI

- corpus ✓
- cleaning ✓
- various form of datasets ✓
- many trf/bert model ✓ emb.

LLM

- layers (NLP).

Vector Databases (chromaDB)

What is vector database

→ 250. (1) Pinecone
(2) ChromaDB
(3) Weaviate
(4) FAISS
⋮

Vector databases are designed to efficiently store and retrieve vector embeddings.

Each movie can be represented as a vector in a high-dimensional space,

(10)

Complex Data

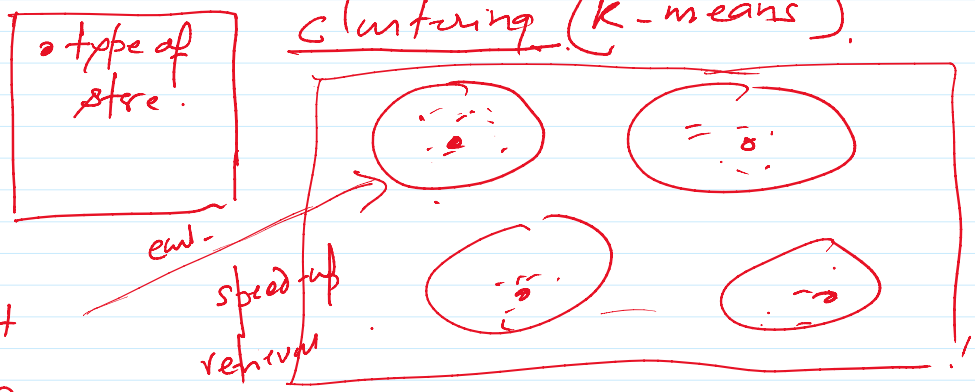
- Imagine you have a large collection of documents, each containing a diverse set of topics, themes, and language styles.
- Representing these documents in a structured way is challenging due to the complexity of the content.

① store the emb. ✓

② Find the sim. emb. ✓ | ML algo.

KNN → variations / advanced ✓

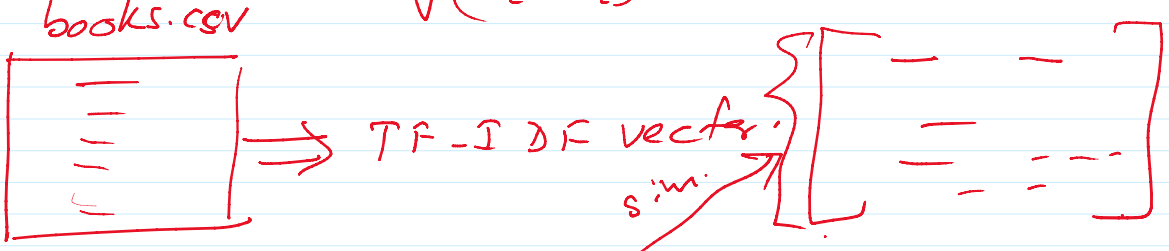
ChromaDB ( ).



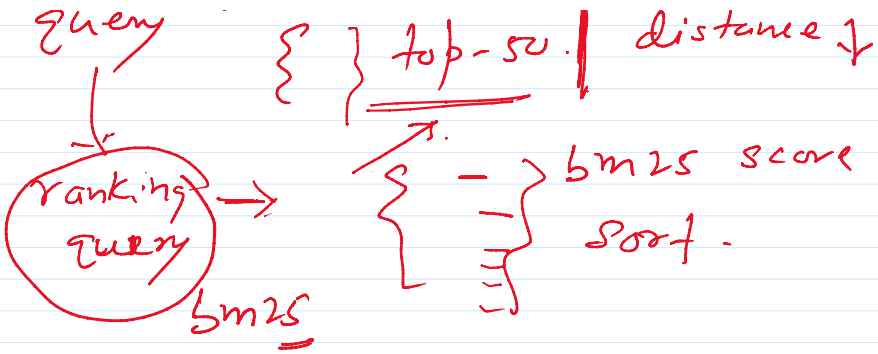
ChromaDB → default dist calc.

euclidean (L2 distance)

$\sqrt{(u_i - v_i)^2}$

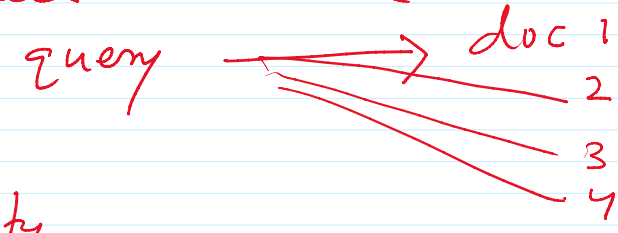


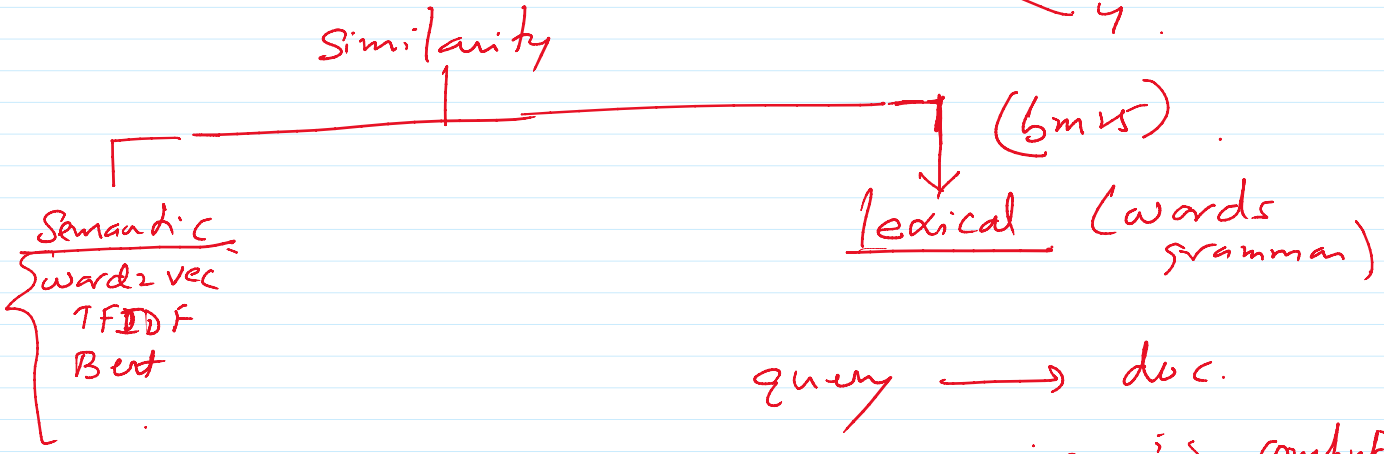
query → { } top-50 | distance ↓



bm25 (

- algorithm.
- similar to TF-IDF
- But this does not vectors
- produces a score (similarity).





query ⟶ doc.

. sim score is computed Lexically .. ✓

✓

"RAG"