Applied Generative AI & NLP - Bert Hollnick

- vs Code & Python venv & install reqs

Intro to NLP

- Tokenization: Process of breaking down text into smaller Units called tokens. Tokens can be words, subwords or even chars.
- Document: Any single piece of text
- Corpus: Collection of Documents

word Embeddings

- Converts words to numbers.
- Relationships to other words are captured
- Ideally Similar words are close

Tensor

- Multidimensional array of numbers
- 0th order Tensors: Scalars: Single numbers, esc: mass
- 1st order Tensors: Vectors: Ordered list of, esc: Velocity numbers
- 2nd order Tensors: Matrices: 2D arrays
- Higher order Tensors: Multidimensional

Word embedding approaches

① One hot encoding

It	was	a	bright	cold	day
0	1	2	3	4	5

	0	1	2	3	4	5
It	1	0	0	0	0	0
was	0	1	0	0	0	0
a	0	0	1	0	0	0
bright	0	0	0	1	0	0
cold	0	0	0	0	1	0
day	0	0	0	0	0	1

problems

→ No of words = No of columns, Curse of dimensionality
→ Memory issues, Sparse matrix
→ Words are isolated from each other, all words have the same distance to each other

---

② Frequency Based

→ Count : Similar to OHE
Gets count of words in doc

→ TF-IDF : Term frequency * Inverse doc frequency

More freq words in corpus ⇒ ↓ tf-idf score ⇒ less inform

Date: / /

TF: How frequently a term appears in a document

IDF: How important a term is by considering how many docs contain the term.

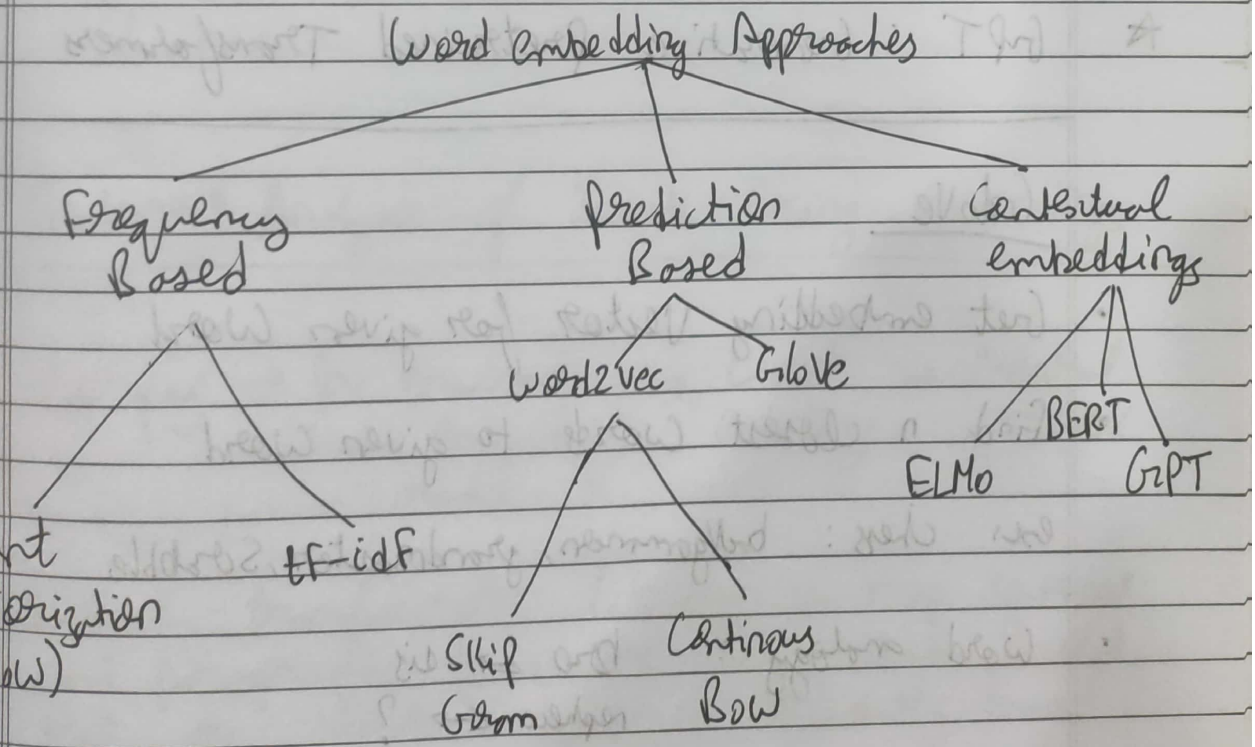
Common words (ex: the) get low TF-IDF score because their high freq across many docs indicate they are not unique or informative

→ Co-occurence : Gets Similarity of words.

→ Sentiment analysis of Tweets

- CSV with tweets → map labels to numbers
- Convert text data into a matrix of token counts
- NN to predict Sentiment using pytorch and Sklearn
- Model evaluated on test dataset

CountVectorizer used to Convert text to token matrix (BoW) ↳ like One hot encoding



Word embeddings w/ NN

- Represent words as low dimensional vectors.
- Captures their meaning.

Goal:

- Capture context/meaning and similarity to other words
- Reduce dimension, avoid memory issues

★ Word2Vec : NLP technique to convert words → Vectors

CBOW : predicts word based on its surrounding words
Skip gram : predicts surrounding words based on word

★ GloVe : Global Vectors for word Representations

★ BERT : Bidirectional Encoder Representations from Transformers

★ GPT : Generative pretrained Transformers

GloVe

- Get embedding vector for given word
- Find n closest words to given word

ex: chess : backgammon, grandmaster, Scrabble

- Word analogy : bro ↔ sis
  nephew ↔ ?
  (niece)

Date: / /

GloVe Wordcluster

- get_embedding_vector: word → word_emb vector
- get_closest_words_from_word: Finds closest words to a given word based on distance b/w their embedding vectors.
- uses this to find 20 closest words for each category (like: tech, Science, algebra etc)
- Gets word emb for each word using GloVe which has 100 dim
- uses t-SNE to reduce dim of word emb from 100 to 2, to visualize in 2D plot
- Scatter plot created using plotnine.

This shows that the embeddings in GloVe also capture semantic info. i.e from the graph we see that closest related words form clusters.

Sentiment Analysis of Tweets Using Embeddings

→ Corpus → Pre trained word Embedding model → Word Embedding → Neural Network

→ Sentence-transformers: Library to map sentences and paragraphs to vectors such that similar text are closer and can be efficiently found using cosine-similarity.

Date: / /

→ Sentence Transformer converts sentences to embeddings

→ NN trained on training data for n epochs

→ Model evaluated on test data

→ Mem Usage

OHE: 27400 tweets, so 27400 * 27400
ST: 27400 * 768 (768 is no of dims in model)

ST uses the pretrained model, so its mem usage should also be considered

---

Transformers

- Recurrent Neural Networks (RNN) works sequentially.
- words are processed one by one, so training is hard to parallelize
- Order of words are important
- Problems with longer Seqs, where past info was forgotten
- Hard to train (vanishing gradients)

- Transformer, Created in 2017, by Google, with focus on translation
- Benefits: Keeps track of Word order
  No Vanishing/exploding gradients
  Training can be parallelized
  Allows for huge models.
- Paper: Attention is all you need
- Ex: BERT, GPT-3, LaMDA
- Can be used in NLP, CV, timeSeries etc

Section 8

Apply Huggingface for pre-trained models.

- ML platform: Build, deploy and train ML models
- Provides infra to run & deploy AI
- Models, datasets, Spaces

① Text-Classification

- Assign pre-defined labels to given text
- relevant for Spam detection, topic categorization, Sentiment analysis etc
- pipeline takes in task & model. If no model is provided, it choses the default for that task
- A single string or list[str] can be passed to the pipeline

Date: / /

② Named Entity Recognition (NER)

- Find entities (such as persons, locations or Orgs) in a sentence. pipeline("ner")

③ Question Answering

- Understand and respond to questions

pipeline("question-answering")

pipe(context='', question='')

④ Text Summarization

- Distill essential info from source text
- preserve meaning & key details
- Ex: Doc Summarization, news & Social media post Summarization

⑤ Translation

pipe = pipeline("translate_en_to_fr")

⑥ Fill-Mask

Predict masked word within given sentence

pipe = pipeline("<mask> is the Capital of India")

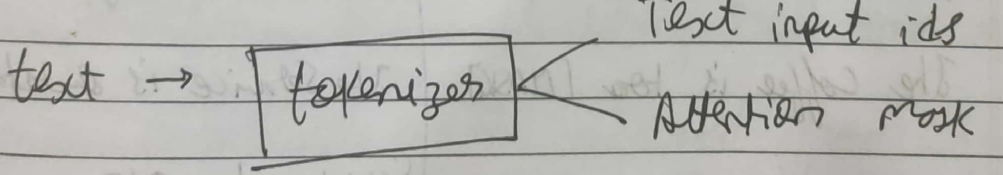
① Zero Shot Classification

- Apply model to task it was not trained on
- Generalize trained model to new and unseen classes
- Avoid retraining
- Useful if: Large no of classes is available
  Classes change frequently
- Types: Attribute based
  word embeddings
  Transfer learning

pipeline("zero-shot-classification")

Model Finetuning

- Why? ← Our data is different to data used to train model
  Our task is different to the model's task



- As text is of different length, text input ids is padded with zeros on the right
- Attention mask has 1 where text is present and 0 otherwise

Date: / /

Features.

- Last hidden layer used with simple ML model and change it.

Simple model

→ 1000 row sample of yelp reviews dataset used

→ DistilBERT : small, fast, cheap, light } model trained by distilling BERT base

→ Last hidden state extracted

Finetuning

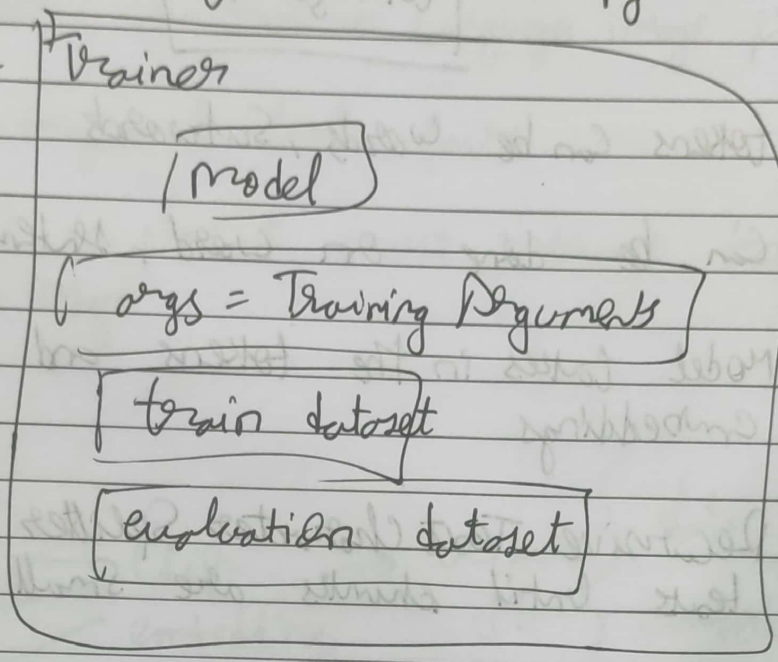
→ HF trainer being used

Pretrained Model	Finetuned Model
• distilbert-base-uncased Fill mask model	• distilbert-base-uncased-yelp Sentiment Analysis model
ex: The coffee is too [MASK]	The service is outstanding Sentiment 5/5.

→ This is called Feature extraction

HF Trainer

→ API for training a model in pytorch



trainer.train()

→ Finetuned model can be uploaded to HF for further use

Vector databases

- Store embeddings for faster querying and similarity analysis.
- Uses: Similarity search, clustering, real time analysis
- Image embeddings can be generated from CNN by discarding the output layer and considering last hidden state
- If data is small store it in np.array, but for large data, it gets slow, so use vector db

Tokenization

- text → [tokenization] → tokens
- tokens can be words, subwords
- Can be done on word, sentence, subword level
- Model takes in the tokens and creates embeddings
- Recursive Text Character Splitter : Splits texts until chunks are small enough
- OpenAI tokenizer page, tiktoken.

Encoding.encode: text → token id
str list[int]

Encoding.decode:
token id → text
list[int] → str

Movies Vector db

→ find title by entering some plot detail

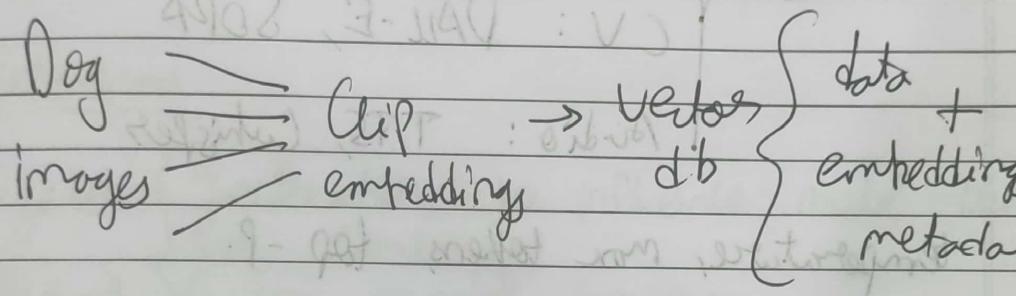
→ dataset from Kaggle, duplicates removed, subset considered

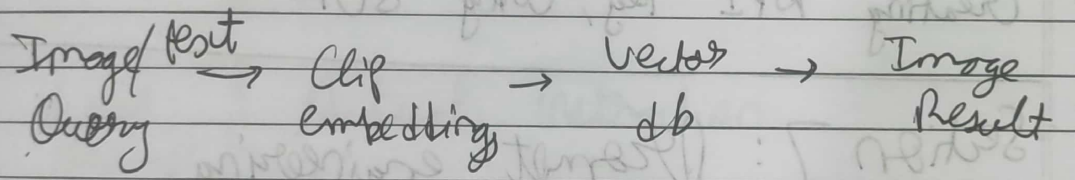
→ id, overview added w/ title as metadata

→ chroma_collection.query done to extract movie titles from movies

Multimodal RAG

- CLIP model (Contrastive Language image pretraining)
- trained on images & text pairs
- Can be used to predict most relevant text snippet given an image





→ text or image can be used to query images.

Section 6: OpenAI API

- GPT: Generative Pretrained Transformer
- Based on GPT-3.5, 175B parameters
- fine-tuned w/ Supervision & Reinforcement learning techniques
- Stateful: Remembers previous chat history & context

- OpenAI docs: prompt examples, models, Playground

Usecases
- Explaining Algos/concepts
- Debugging errors
- creating code
- Refactoring code
- code docs

- Open AI Models
  - NLP: GPT (3.5, 3.5 turbo, 4, 4 turbo, 4o), Embeddings
  - CV: DALL-E, SORA
  - Audio: TTS, Whisper

- temperature, max tokens, top-P
- Creating API key, using SDK

Section 7: Prompt engineering

Narrow NLP: Task Specific model
Trained and used Separately

LLM: Covers functionalities of many independent ma

LLM pros
- Generalization
- Versatility
- No need to maintain, in case of just using

LLM cons
- cost
- cannot be used offline (for OpenAI)
- privacy concerns
- Interpretability

- prompt is the input to the model
- prompt engineering: Improve quality & relevance of model results

① clear instructions

Include relevant details like:
- Output format
- Answer length & complexity
- Expert-level or high level

② Personas

System { Instructions to influence model behavior
Set Context
Front { Guide model response

User prompt { Actual instruction

ex: Behave like a linux Shell

③ Delimiters

Help model seperate sections in user prompt

④ Divide User input to subtasks: ex Do this, then this

⑤ provide prompt with example

⑥ Control output: format: Brief, detailed
length: one word, sentence, max 50 words
style: Shakespeare, JSON, ini, XML
tone: mood/sentiment: Casual, formal, ELI5
Context &

Section 8: Advanced prompt Engineering

Why?

- Simple prompts might not yield good results
- Model might need specific instructions to understand the problem.
- model jumps to conclusions

Few shot prompting

- provide few examples of task along with expected output
- model learns from examples and tries to generalize the pattern

Pros	Cons
① Adapts quickly to new tasks	Inconsistent performance when examples not fully captured
② effective when examples demonstrate behaviour	Depends on quality & diversity of examples
③ Custom output for specific usecases	Risk of overfitting and poor generalization

Q1 : A1 } User provides examples
Q2 : A2 }
:
Q_n : ? ← Answered by model

(R) Chain of Thought

To help the model avoid jumping to Conclusions.

→ Zero shot CoT : {Question}. Please think step-by-step

→ Few Shot CoT : • User prompt involves Question, Steps to Solve it & Answer.
• There can be one or more examples.

Pros :

- Encourage LLM to multi step reasoning, can improve performance on complex tasks
- provides transparency into thought process, increasing interpretability

Cons :

- ↑ Output length ⇒ ↑ Cost
- Reasoning Quality depends on ability of LLM to break down the problem into sub steps
- Relies on quality of intermediate steps, so there is a risk of compounding errors

Ex: Math Puzzle

(R) Self Consistency Chain of Thought

- Applies idea of "ensemble learning"
- performs multiple CoT experiments and then uses voting to pick most consistent answer Majority vote wins.

(R) Prompt Chaining

- Complex task broken down into Sub tasks
- Each Subtask has its own prompt + last Output
- Can incrementally Solve a task

(R) Reflection

- Encourage LLM to Critically evaluate their own responses and identify area of improvement

(R) Tree of thoughts

- Solve Complex problems that require strategic thinking and planning.
- Explore different Solutions and evaluate quality before committing to a valid path
- LLM Combined with Search Algo

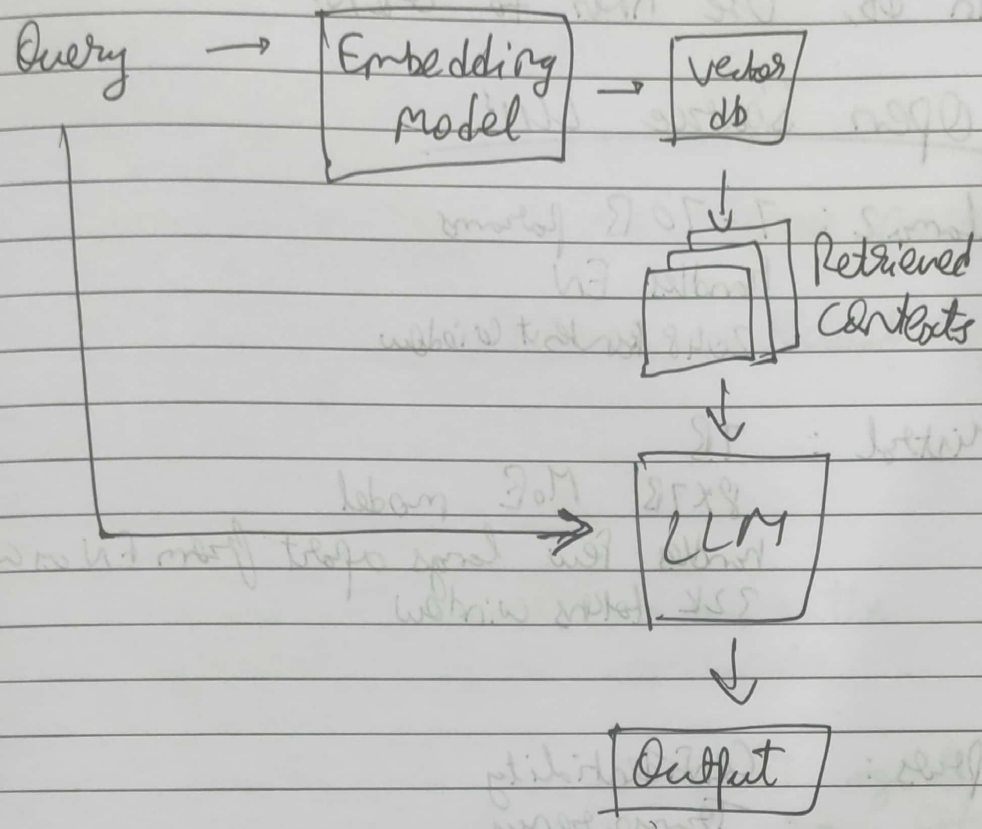
(R) Self-Feedback

- Iteratively improves quality of response
- Stop Criteria reached, Final response returned

(A) Self Critique

- Define number of outputs/ ideas
- LLM critiques all ideas and picks most appropriate idea
- LLM improves best idea from last step

RAG : Retrieval Augmented Generation



- Pass the query to the embedding model to semantically represent it as an embedded query vector
- Pass the embedded query vector to our vector db
- Retrieve top-K relevant contexts - measured by distance b/w query embedding and all embedded chunks in our knowledge base
- Pass the query text & retrieved context to our LLM
- LLM generates a response using the provided content

Ex: Vector db contains data on movie titles & their plots. User asks a query, relevant context from db is pulled and passed to LLM to answer it

Date: / /

Capstone : RAG Chatbot

Ingest from PDF, create embeddings, store in db, Use RAG to Q&A.

Open Source LLM's

Llama 2 : 7-70 B params
handles EN
2048 tokens context window

Mistral : 7B
8X7B MoE model
handles few langs apart from EN as well
32K tokens window

Pros : Confidentiality
Transparency
control

Cons : Performance
Maintenance & operations.

- Cost may vary.
- Prompt templates differ across models. If not proper, output will be incorrect/gibberish

Data Augmentation

- Making changes to dataset so as to create more data
- ex: Colour Scale, rotation & Cropping images in CV
- In NLP : Adapted sentences should have same meaning as original sentence

- Back translation: Translate lang1 → lang2 → lang1

This might exchange certain words in original sentence, with similar words.

- Replace with Synonyms
- Word embedding Augmentation: choose nearest neighbours
- Random cropping, Fill mask, Contextual Augmentation

Agents

- System that uses an LLM to decide the control flow of an application
- CrewAI is designed to enable AI agents to assume roles, share goals and operate in a cohesive unit

Claude

- Family of models developed by Anthropic
- Anthropic founded in 2021, Claude in 2023, Claude3 in 2024
- Opus > Sonnet > Haiku in model size & perf
- chatbot Arena can be used to test multiple models
- It is double blind study, to avoid bias
- LMSYS leaderboard

LLM functions

- Define tools with names, desc & schema
- Claude decides to use a tool
- Extract tool input, run code, return results