Multimodal BI Applications

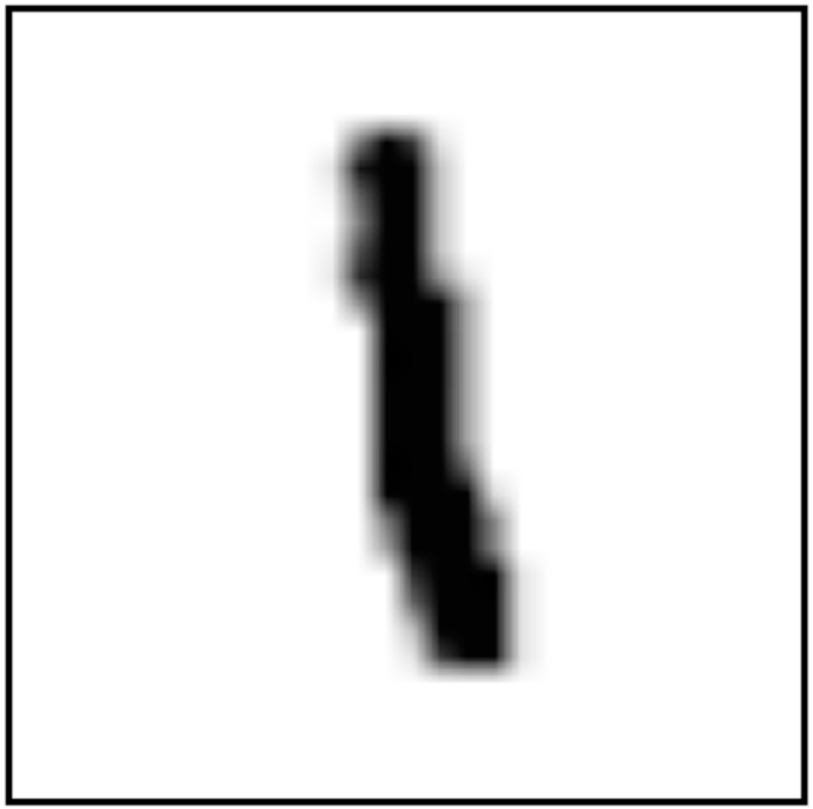
by tdi@ek.dk

Agenda

- What is multimodality?
- How BI applications process multimodal resources?
  - Images
  - Natural Language
    - NLP/NLU Terminology and Tasks
    - Text Embedding
    - Bag of Words
    - Vector Similarity
  - Audio
  - Video
- Programming Exercises

Image Processing

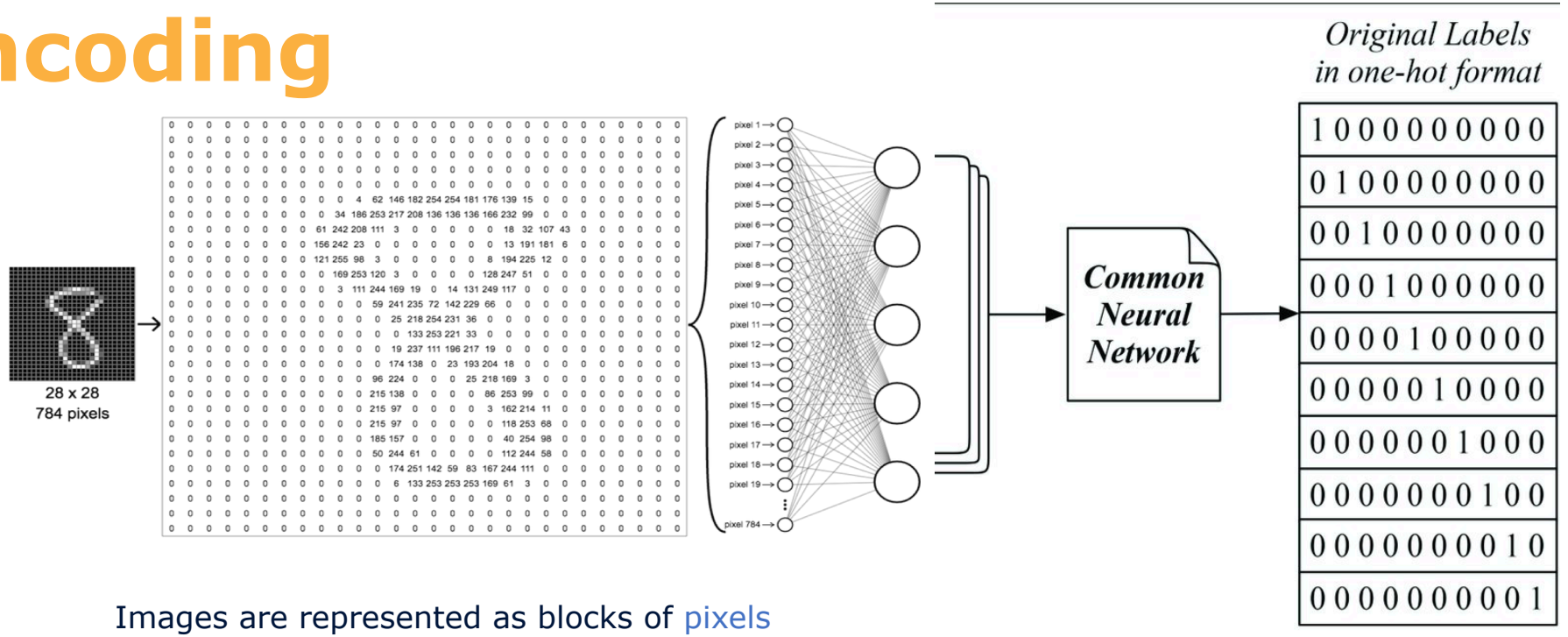
Image Representation



$\cong$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

One-Hot Encoding

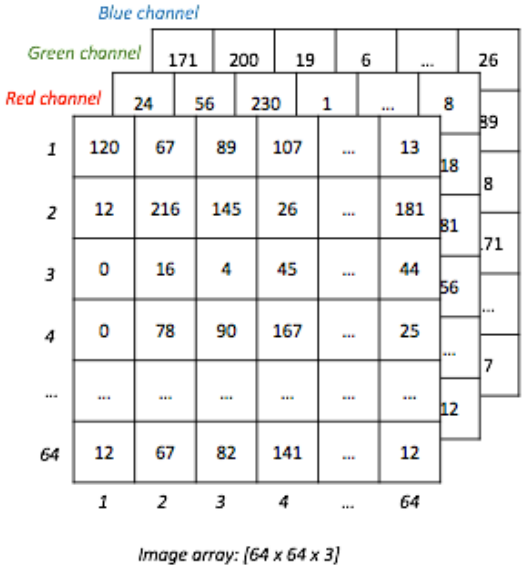


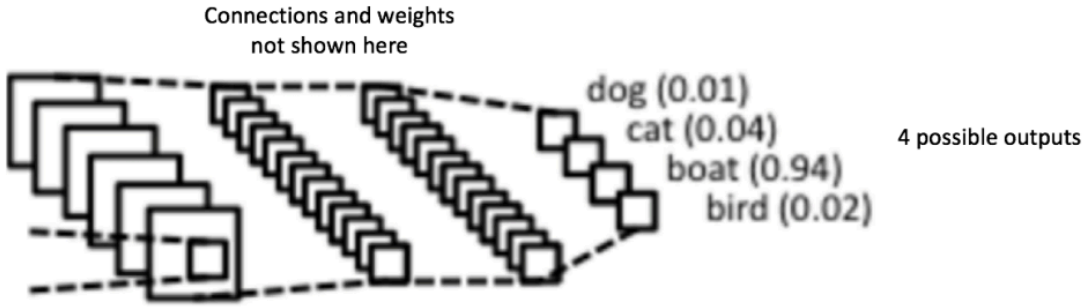
Images are represented as blocks of pixels

- located in the centre of mass
- for 28 x 28 block the are 784 pixels per digit/image needed

Training

- provide the values of the pixels as training data (e.g. column by column)
- create a model of the relationships between pixels for each digit
- test the model, implementing (supervised) ML algorithms
- apply it to predict new examples



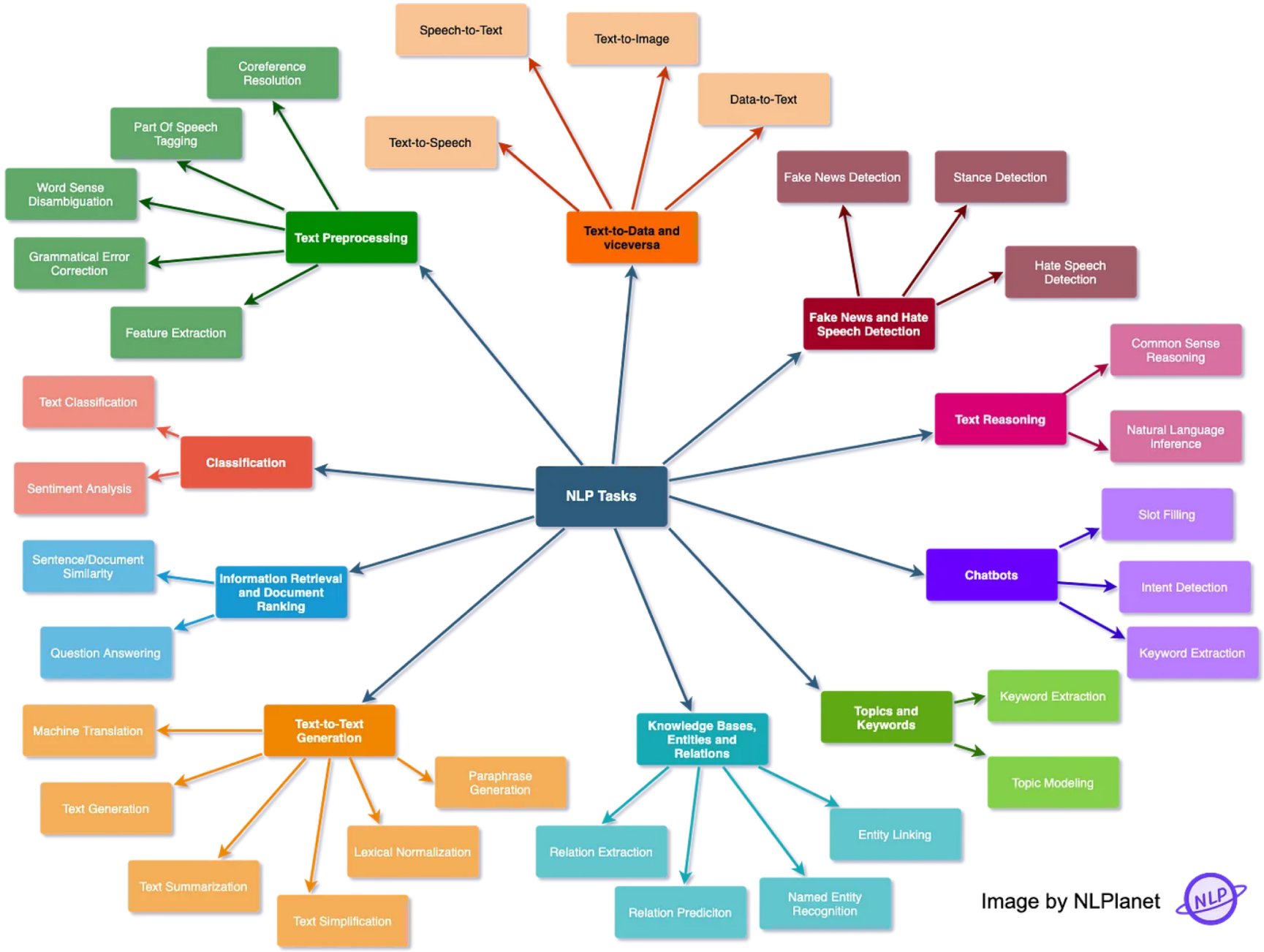


Coloured Images Need More Pixels

Natural Language Processing

and Understanding

Natural Language Processing Tasks



https://medium.com/nlplanet/two-minutes-nlp-33-important-nlp-tasks-explained-31e2caad2b1b

Documents

Pre-processing

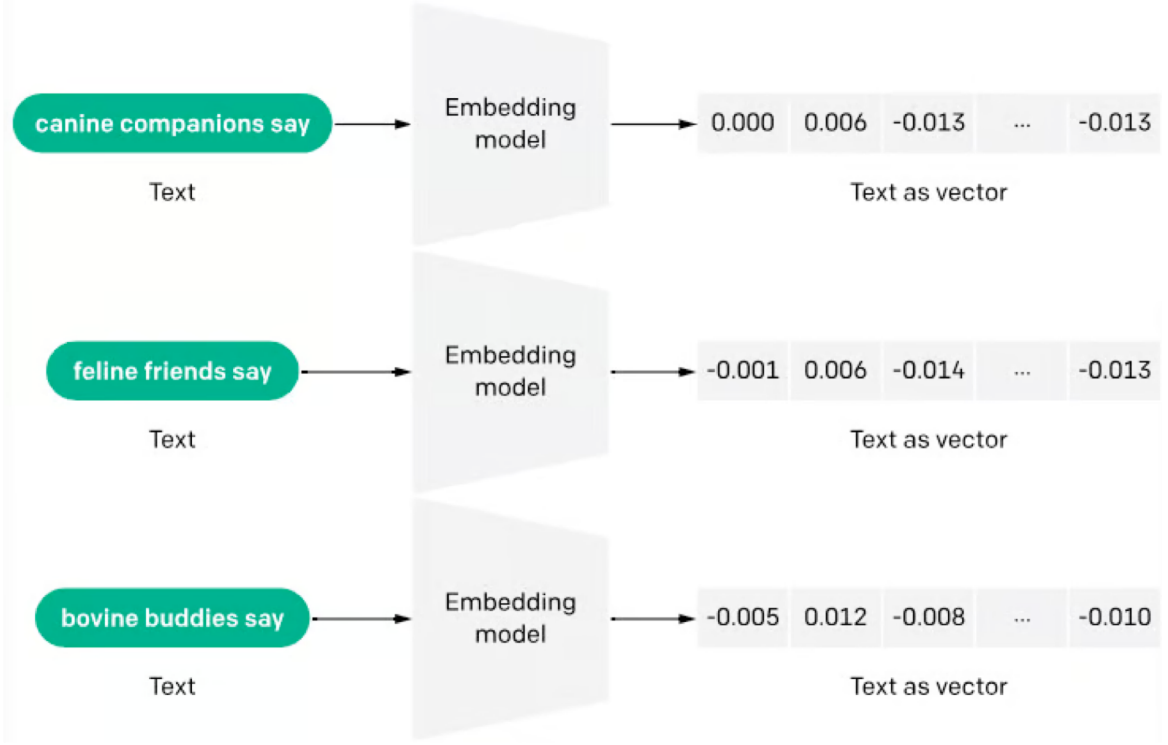
- Anonymization
- Cleaning, removing
  - stop words
  - punctuation
- Linguistic pre-processing
  - segmentation
  - tokenization
  - POS
  - NER
  - Coreference Resolution
- Test augmentation
  - for testing and other purposes

Processing

- Summarization
- Information Extraction
- Topic Extraction
- Keywords Extraction
- Questions Answering
- Text Classification and Clustering
- Emotion Extraction
- Language Translation
- Text-to-Speech Conversion
- Speech-to-Text
- Text Generation
- Gramma control

Text Embedding

- What is it?
  - converting text into a numerical representation called a vector
  - vectors must capture the meaning and context of the text
- Why to vectorise text?
  - to allow computers to understand and process language more effectively
  - to convert unstructured text into structured data
  - for enabling machine learning



Source

Text Embedding Evolution

The new advances based on achievements on the previous ones

- 1950s-2000s: Frequency-based embedding - *observing how often one word occurs in the text*
  - Bag of Words
  - TF-IDF
- 2000s-2010s: *modelling the relationships between words in a continuous text space*
  - Word2Vec – vectors capture semantic similarity between words
  - CBoW (Continuous Bag of Words)
  - Skip–Gram
- 2010s-2020s: Sentences, attention, and transformers - *focusing on specific parts of a sentence to understand the relationships between the words and how they contribute to the meaning*
  - pre-trained LLMs
  - text generation

Count Vectors

- Consider a Corpus C of D documents $\{d_1, d_2 \ldots . d_D\}$ and N unique tokens (terms) extracted out of the corpus C
- The N tokens will form our dictionary (vocabulary)
- One-hot encoding of a token in a document (1-occurs, 0-doesn't)
- Matrix M (D x N): Each row contains the frequency of token j in document i
- We can build a vocabulary of all terms and their frequency of their use in each document

Example: C => movie reviews

d_1 => Review 1: "This movie is very scary and long"

d_2 => Review 2: "This movie is not scary and is slow"

d_3 => Review 3: "This movie is spooky and good"

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Bag-of-Words BoW

- a simplified representation of text, where a document is treated as a collection of words, ignoring word order and grammar
- used to convert text data into a numerical format suitable for machine learning algorithms
- focuses on the frequency of words within a document, creating a "bag" of unique words and their counts, ignoring the context and order of words

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]
Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]
Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

Word Vector "This": [1 1 1]

Drawbacks of BoW Model

- If the new sentences contain new words, the vocabulary size and the length of the vectors would increase
- It is a sparse matrix with lots of meaningless zeros
- There is no information on the grammar and the semantic of the sentences

TF-IDF

Term Frequency – Invert Document Frequency

- Calculates the weight of each word not only by its frequency in a specific document but also by its commonness across all documents, assigning higher values to less common words
- Term Frequency – relative frequency of one term in one documents
- Inverse Document Frequency – relative frequency of occurring of one term in all documents – less is more
- TF-IDF – calculated weight based on the multiplication of the previous two

Term Frequency TF

- n is the number of times the term "t" appears in the document "d"
- all documents considered
- Example
  - Vocabulary**:** 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'
  - Number of words in Review 2 = 8
  - TF for the word 'this' :

tf $_{\text{'this', doc2}}$= (number of times 'this' appears in review 2)/(number of terms in review 2) = 1/8

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Inverse Document Frequency IDF

- Computing just the TF is not sufficient to understand the role of words in the sentences
- IDF is a measure of how important a term is

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0

IDF('movie',) = log(3/3) = 0

IDF('is') = log(3/3) = 0

IDF('not') = log(3/1) = log(3) = 0.48

IDF('scary') = log(3/2) = 0.18

IDF('and') = log(3/3) = 0

IDF('slow') = log(3/1) = 0.48

Words like 'is', 'and', 'movie' do not seem important

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Term Frequency-Inverse Document Frequency TF-IDF

- Assigns weights to each word based on its frequency across all documents
- The more frequent the word is across all documents, the less weight it carries

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

TF-IDF('this', Review 2) = TF('this', Review 2) * IDF('this') = 1/8 * 0 = 0

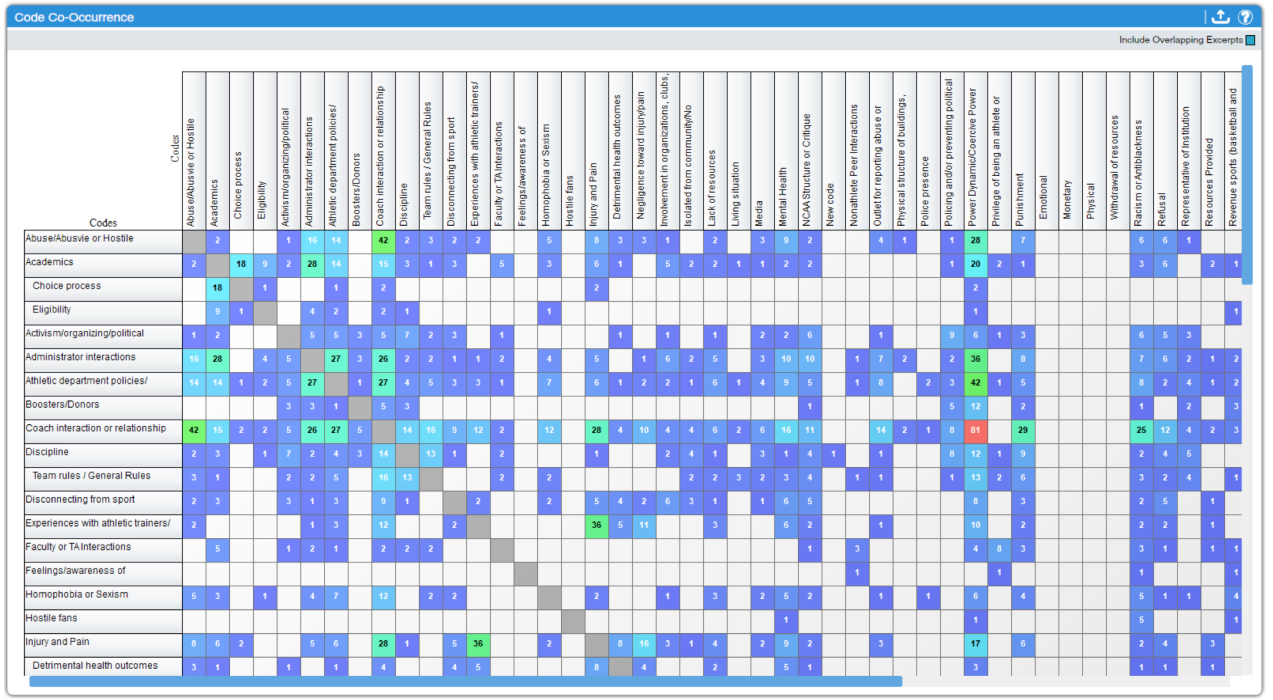
Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

Context Window and Co-occurrence

Quick brown **fox** jumps over the lazy dog

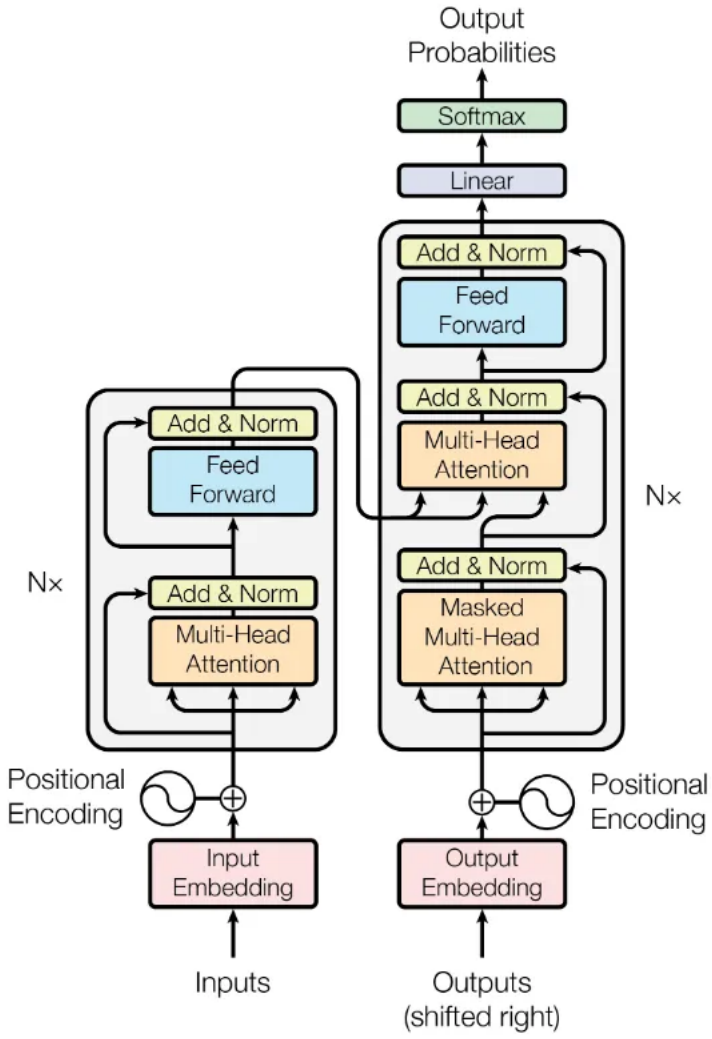
The green words form a context window for the word ‘Fox’

These words will be counted for calculating the co-occurrence.



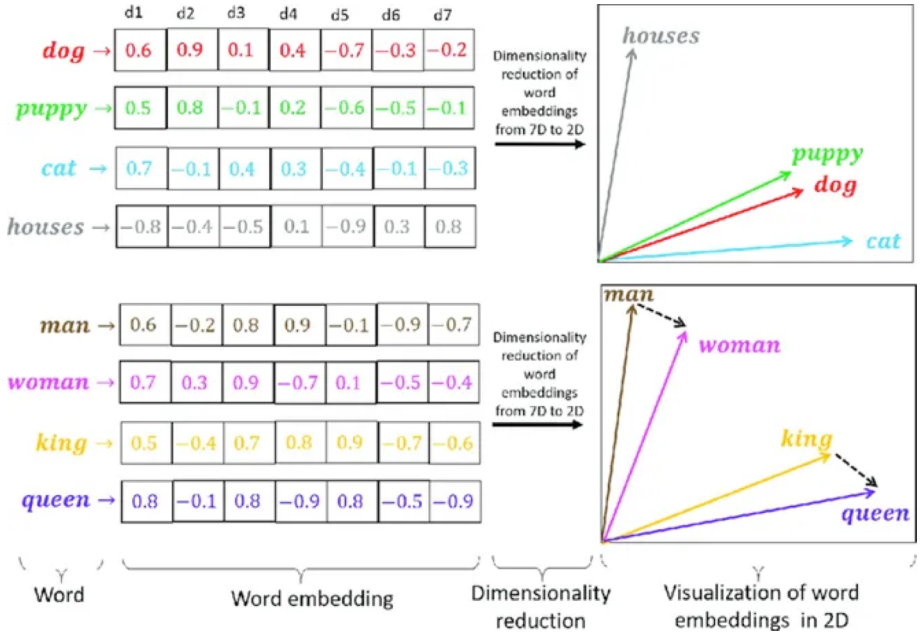
Attention

- Attention is a concept that allows the algorithm to focus on the most relevant parts of the input sequence
- An input of the attention layer is called a QUERY
- For a query, attention returns an output based on a set of KEY-VALUE pairs encoded in the attention layer
- If the QUERY is similar to a KEY, the VALUE of that key is unlocked
- Similarity is cosine similarity



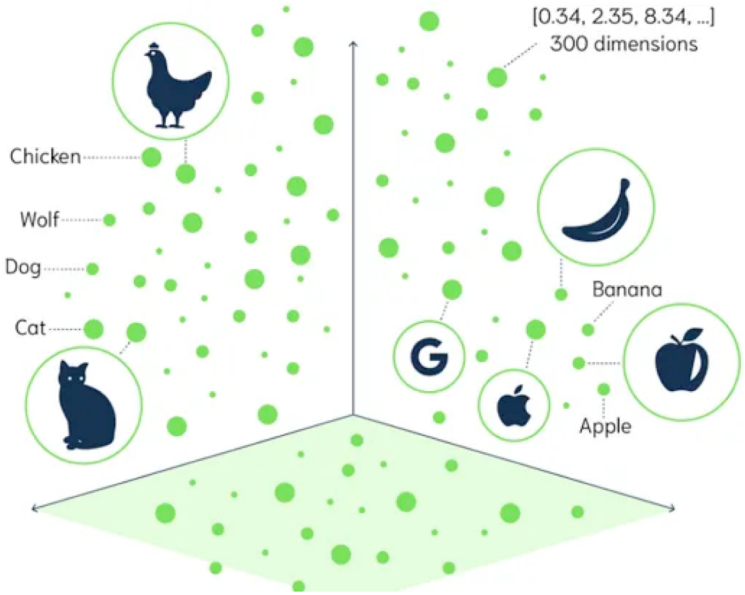
From "Attention is all you need" paper by Vaswani, et al., 2017

Word Embedding



Rozado, David (2020). Word embeddings map words in a corpus of text to vector space.. PLOS ONE. Figure.
https://doi.org/10.1371/journal.pone.0231189.g008

cat =
[3.73303443e-02 5.11619523e-02 -3.06102040e-04 6.02099150e-02
-1.17494389e-01 -1.42300949e-02 1.05776198e-01 2.67862827e-02
2.63378248e-02 -2.57008411e-02 -2.34903619e-02 -5.95552400e-02
-3.02139055e-02 1.63201671e-02 -2.90702265e-02 -2.16897614e-02
-6.62499443e-02 1.85664429e-03 -2.40063071e-02 -2.84625385e-02
-4.66316454e-02 4.97048348e-02 3.08292918e-03 1.76271854e-03
-6.77575320e-02 7.61016831e-02 -4.53299582e-02 -3.64345126e-02
-1.87947508e-02 -5.91583699e-02 -6.60743490e-02 -3.26584210e-04
-8.92037898e-03 5.34161404e-02 -5.47015965e-02 -5.12043238e-02
-9.80858505e-03 1.13824371e-03 5.64007498e-02 6.17820993e-02
-3.50218304e-02 -8.47024024e-02 -2.72386894e-02 -1.92160401e-02
-3.01410556e-02 4.26462246e-03 2.97398493e-02 -6.17982373e-02
4.49534170e-02 -3.92310089e-03 -6.46450147e-02 2.06935499e-02
-3.68908755e-02 -5.05909743e-03 -1.63175017e-02 1.64349127e-04
5.13037406e-02 -1.89798046e-02 -2.52253618e-02 -2.96597406e-02
4.61248681e-03 1.05470354e-02 -1.25657054e-04 7.81174228e-02
2.54771039e-02 -3.30794592e-06 2.75657745e-04 9.69488919e-03
4.97767329e-02 -1.11800507e-02 1.20957186e-02 3.86810675e-02
-2.93666199e-02 -2.84539303e-04 1.92649979e-02 -3.29860263e-02
1.31716192e-01 -7.33047491e-03 1.03678852e-01 1.57747865e-02
-6.97513251e-03 2.82031056e-02 -2.68339161e-02 3.72036286e-02
4.00002599e-02 6.58613965e-02 -1.76047208e-03 1.63966529e-02
-6.74865693e-02 2.06541773e-02 1.71644408e-02 -1.01331957e-02
6.46720454e-02 1.49434833e-02 -1.10849641e-01 3.11816279e-02
-7.72632984e-03 -7.74633735e-02 -4.20717746e-02 2.35207275e-01
2.54941061e-02 2.58553810e-02 -4.41658944e-02 5.68786561e-02
4.32346156e-03 -1.53301423e-02 1.12162950e-02 1.71305821e-03
6.89385645e-03 1.49210459e-02 -2.00052117e-03 -4.70863059e-02
-5.93807437e-02 5.45828007e-02 3.66447754e-02 2.11964045e-02
-2.26663183e-02 -3.33854854e-02 8.89618918e-02 -1.80630554e-02
3.75270024e-02 -9.95676126e-03 -4.76089418e-02 -1.10070100e-02
-5.20975925e-02 -8.54046121e-02 -4.77136187e-02 -4.54564657e-33
-4.84421710e-03 -9.76089910e-02 1.47302561e-02 -2.60839518e-02
4.91283946e-02 5.85032329e-02 2.89115310e-03 2.90651266e-02
-8.84265378e-02 1.61231942e-02 -8.86038244e-02 1.25194546e-02
-7.36623406e-02 -1.23630613e-02 3.29441689e-02 -9.77547839e-03
-1.29911872e-02
...]



https://www.truefoundry.com/blog/similarity-search

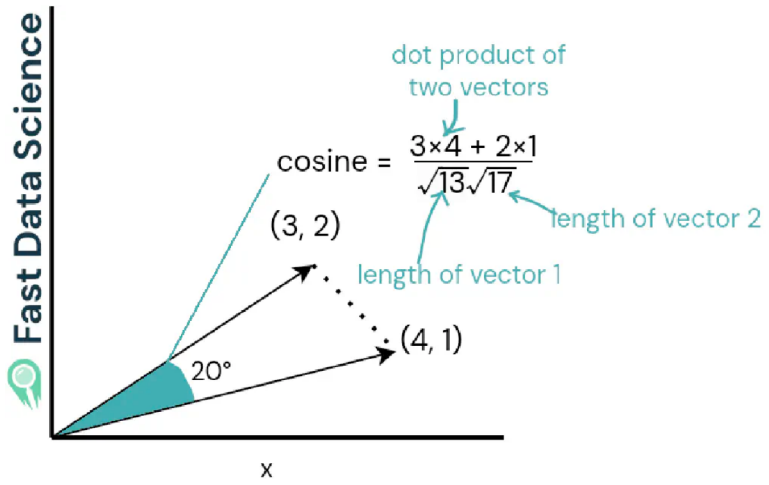
Semantic Similarity

- Similar words tend to occur together and have similar context
- Example
  - Apple is a fruit
  - Mango is a fruit

  Apple and mango tend to have a similar context i.e fruit
- Co-occurrence
  - for a given corpus, the co-occurrence of a pair of words w1 and w2 is the number of times they have appeared together in a context window
- Context Window
  - an interval, specified by a number and the direction
  - covers subset of words in the text

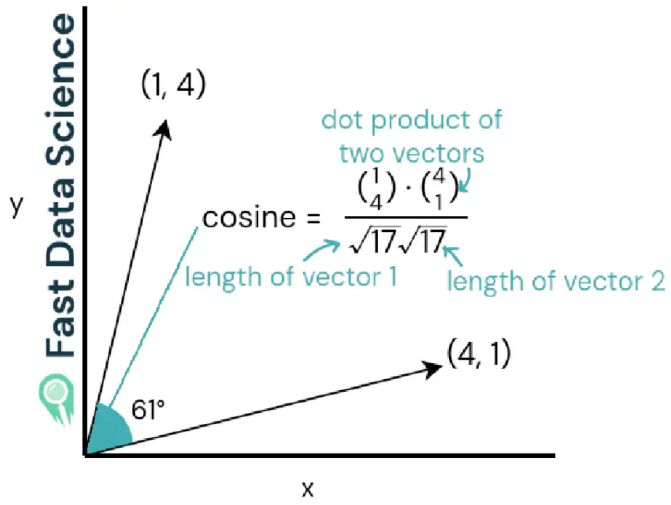
Cosine Similarity Measure

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



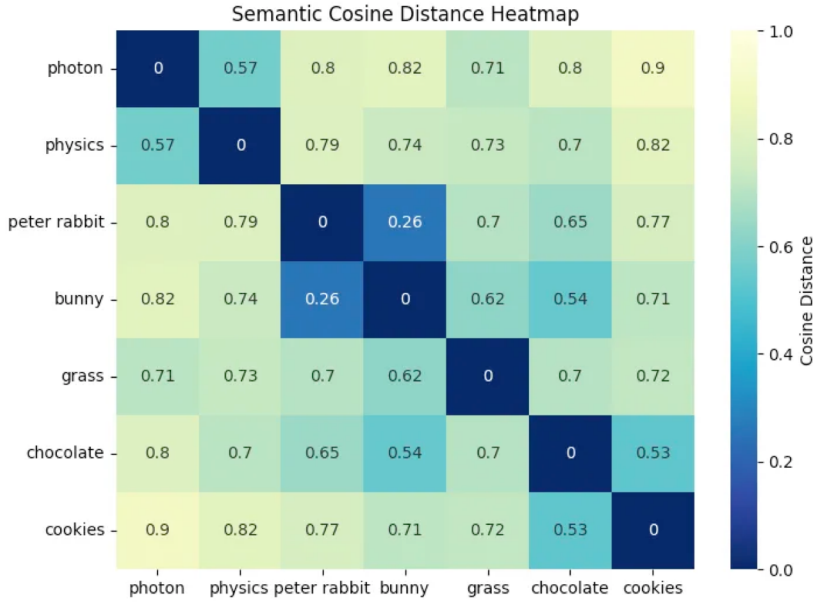
cosine similarity is 0.94
this corresponds to angle of 20°

these two sentences would be more similar



cosine similarity is 0.47
this corresponds to angle of 61°

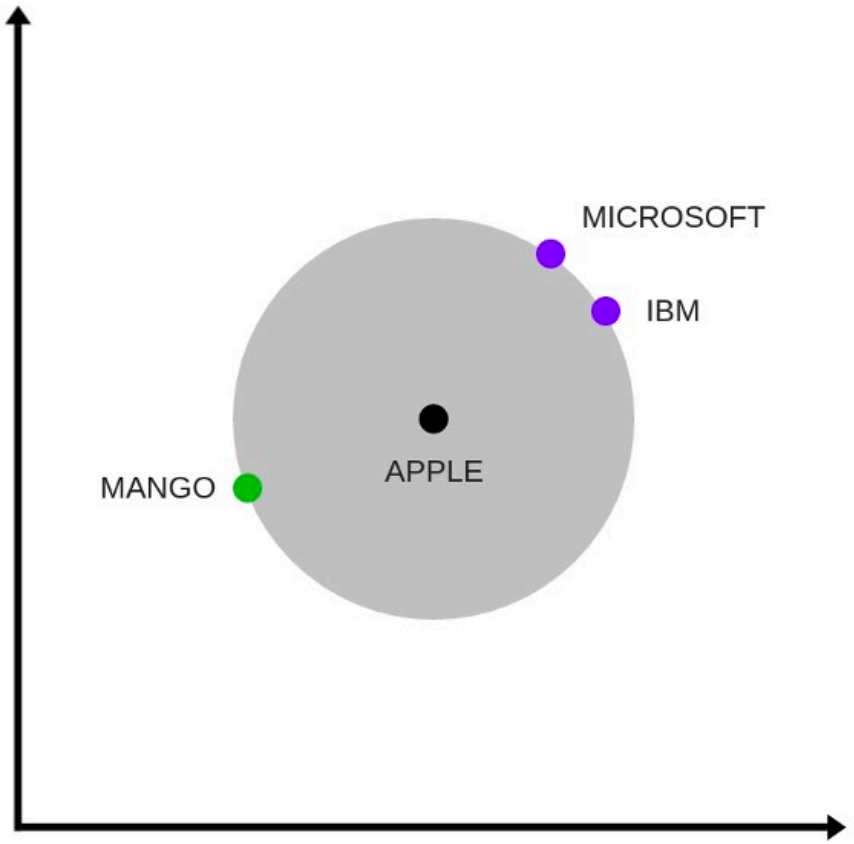
these two sentences would be far apart and not semantically similar



https://fastdatascience.com/natural-language-processing/semantic-similarity-with-sentence-embeddings/

https://ai.gopubby.com/what-is-semantic-similarity-an-explanation-in-the-context-of-retrieval-augmented-generation-rag-78d9f293a93b

Similarity Search Scenarios



Since word embeddings are numerical representations of contextual similarities between words, they can perform tasks like:

- Finding the degree of similarity between two words

```
model.similarity('woman','man')
0.73723527
```

- Finding the odd one out of many

```
model.doesnt_match('breakfast cereal dinner
lunch';.split())
'cereal'
```

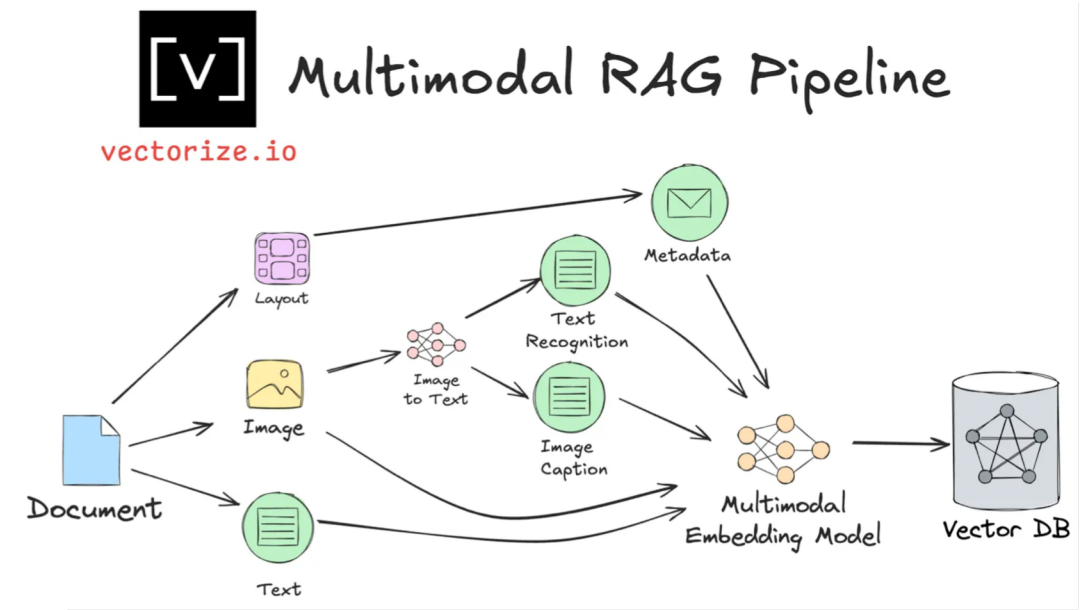
- Amazing things like woman+king-man = queen

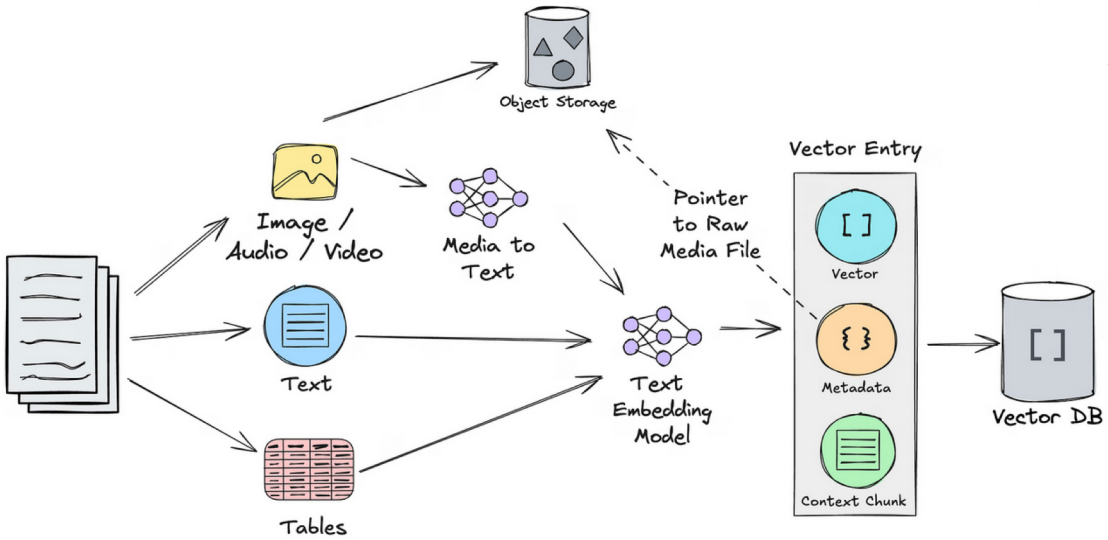
```
model.most_similar(positive=['woman','king'],negative=['ma
n'],topn=1)
queen: 0.508
```

RAG

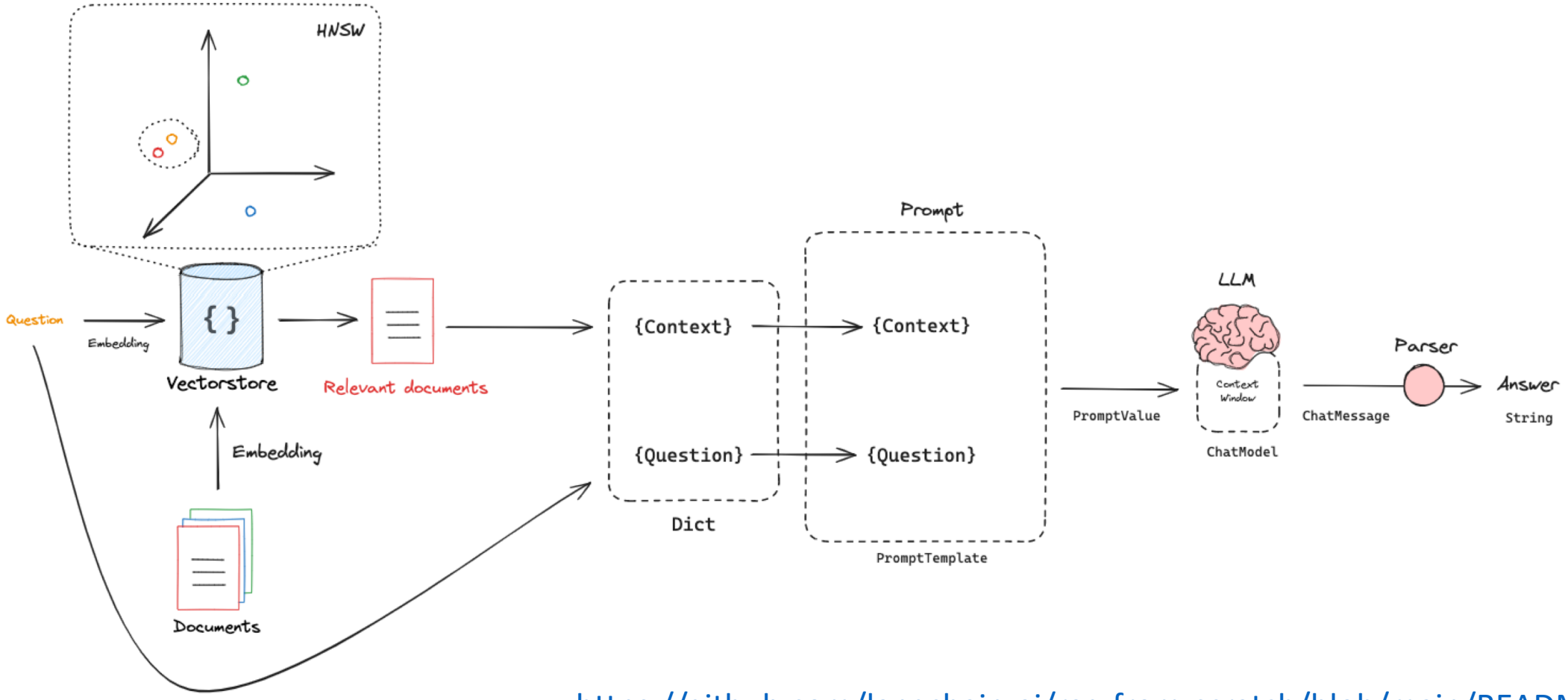
Retrieval-Augmented Generation

Vector DB


Multimodal RAG Pipeline
vectorize.io
Layout
Metadata
Text Recognition
Image to Text
Image
Image Caption
Document
Text
Multimodal Embedding Model
Vector DB

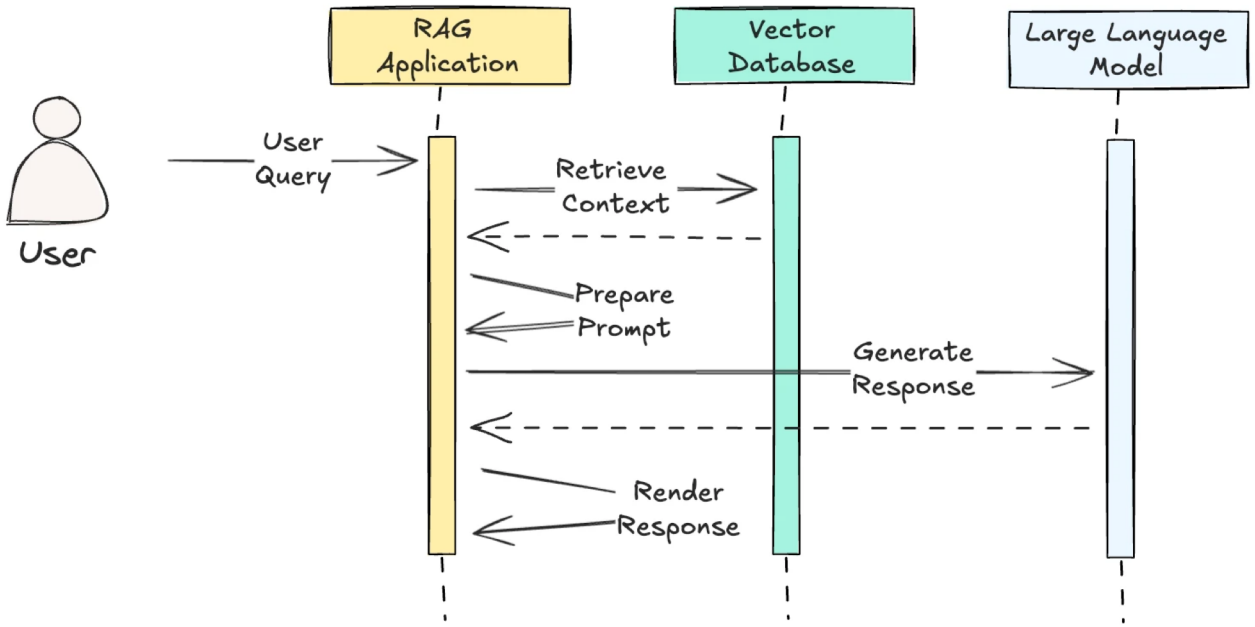

Object Storage
Image / Audio / Video
Media to Text
Pointer to Raw Media File
Vector Entry
Vector
Metadata
Context Chunk
Text
Text Embedding Model
Vector DB
Tables

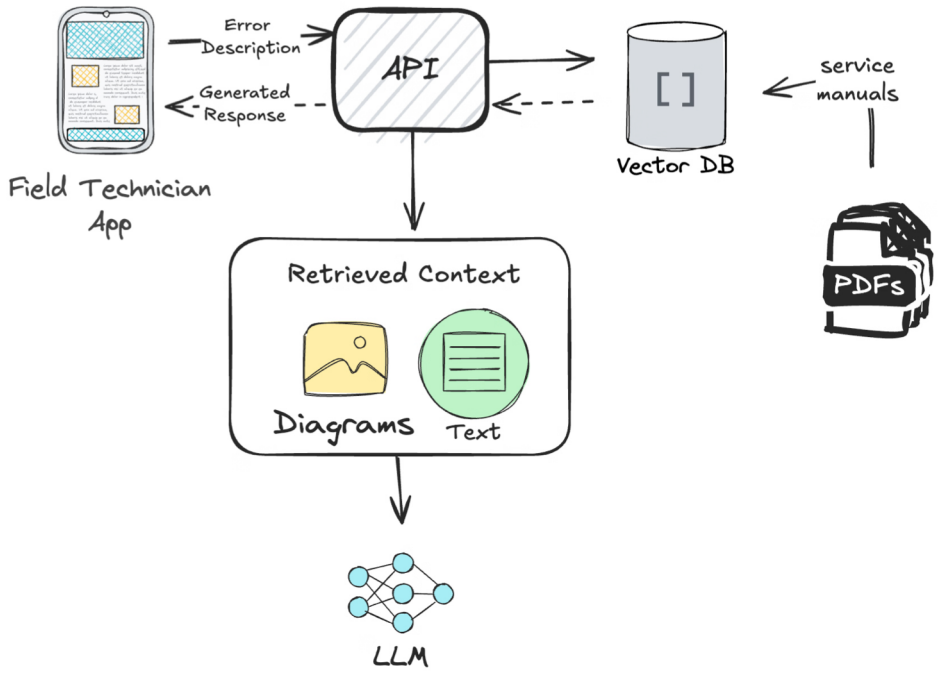
RAG Pipeline



https://github.com/langchain-ai/rag-from-scratch/blob/main/README.md

Example of Application





Programming

Interactive Multimodal Applications