Vector Models and Text processing

10.03.2023

Munich

Definitions

Vocabulary = set of all words

Corpus = Collection of writings (dataset)

N-Gram = Sequence of N consecutive items

1 = Unigram 2 = Bigram 3 = Trigram

Bag of words

- Text is sequential
- Order doesn't matter used in Vector Models

Dog Toy = Toy Dog in Bag of words

Count Vectorizer

- instance of Bag of words, order doesn't matter

Document – file, sentence, full book

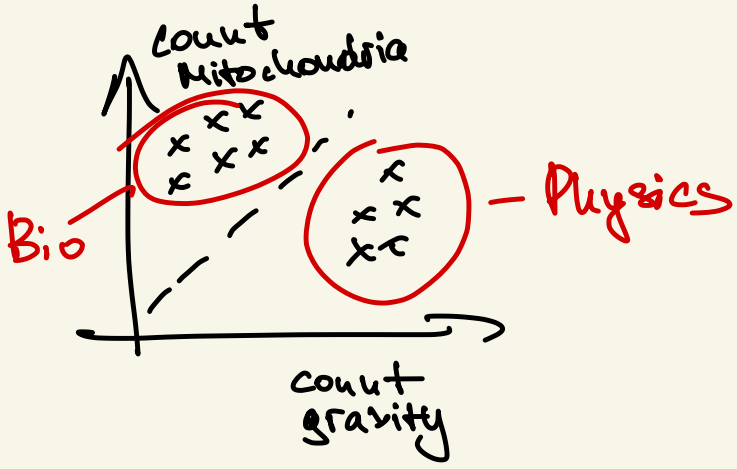
→ convert sentences into numerical representation

Vocabulary size → every word has a number

I like eggs
I hate cats
I like eggs and I like cats

→

and	cats	eggs	hate	I
0	0	1	0	1
0	1	0	1	1
1	1	1	0	2



Practical issues

Tokenization - convert string containing words into array of numbers

Normalization

long vs short documents $\hat{x} = \frac{x}{\|x\|_2}$ $\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$

sparse matrices to be used

Tokenization

- old approach -> .split()
- String split works well for RNN classification
- Punktuation I hate cats. vs I hate cats?
  CountVectorizer ignores punct. string split
- Casing Cat name vs cat animal
- Accents naive vs naïve
  count vectorizer strip accents
- Character Based
  1M words = 1M embeddings = 1M last layer
  not much information, but 26 letters + punctuation
  Subword - Based Tokenization
  walking -> walk + ing -> modifier
  walking <-> walk -> tree = equal weights
  CountVectorizer has word/char based tokenization

Stopwords

- very common words with less meaning (the, it, and, is)
- increasing vector dimensionality
- distance will be diminished (between vectors)

- Count Vectorizer has english stopword library
- NLTK has stopwords from other languages

Stemming and Lemmatization

- Basic tokenization: walk, walking, walks
  ↳ very high dimensions
  all different tokens
- Search engine will treat all words independently
- Stemming → chop the end of the word (crude)
- Lemmatization → true root of the word (sophisticated)

Reduces vocabulary size

Stemming

sses → remove es: Bosses → Boss
Replacement → Replac

Porter stemmer in NLTK

Lemmatization

- Look up table
  Better → Good
  Was → Be
  Is → Be
  Mice → Mice
- Wordnet DB in NLTK
- pos (parts of speech) signifies wether word is noun, adj, verb
  ↳ default is noun

  Trump has devoted following (noun)
  The cat is following me. (verb)

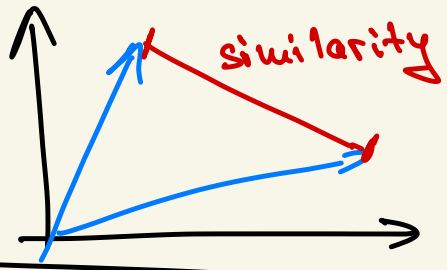
- NLTK can do POS tagging
- Application: search engine, doc. retrieval, social media tags, ads

- Search By converting terms in root form
- Online ads keyword → match ads to keywords
  Google Pixel → show Apple iPhone

Vector Similarity

$a = [0.5, 0.3 \ldots]$
$B = [-0.1, 0.2, \ldots]$
→ Similarity Function $s(a,B)$ → score = 0,12

- Find similar documents on specific topics
- If words were vectors: queen ↔ king
  car ↔ automobile } high similarity
  used in Article Spinning
- Euclidean Distance
  similarity = distance



$$\|x-y\|_2 = \sqrt{(x_1-y_1)^2+(x_2+y_2)^2+\ldots+(x_D-y_D)^2}$$

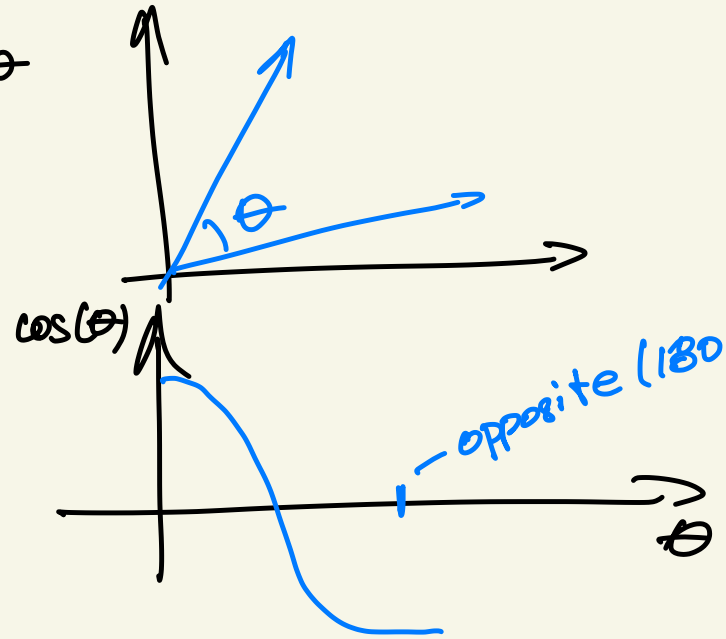
monotonically increasing

- Angle Between 2 Vectors (cosine similarity)

$$x^T y = \|x\|_2 \|y\|_2 \cos\theta$$

$$\cos\theta = \frac{x^T y}{\|x\|_2 \|y\|_2}$$

$\|x\|_2$ = magnitude

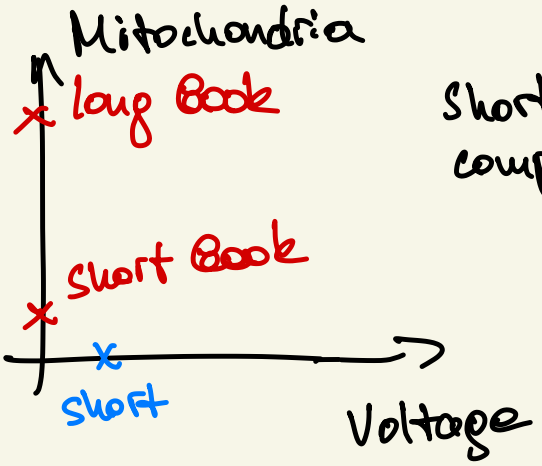


Cosine Distance = 1 − Cosine similarity

dist = 1 − (−1) = 2 for 180°

dist = 1 − 1 = 0 for 0°

Which to use?



Short Books have smaller eucl. distance compared to Books on Biology topics

- If vectors are Normalized in L2 form, distance can be used

$$\|x-y\|_2^2 = (x-y)^T(x-y)$$

$$= x^Tx - 2x^Ty + y^Ty \quad \text{But } x^Tx = y^Ty = 1 \text{ for L2}$$

$$= 2 - 2x^Ty \qquad x^Ty = \frac{x^Ty}{\|x\|_2 \|y\|_2} = \cos\theta$$

$$= 2 - 2\cos\theta$$

$$= 2 \cdot \text{cos distance}$$

Word to Index Mapping

	word 1	word 2	words	
Doc 1	0	1	1	....
Doc 2	2	2	0	...
Doc 3	1	1	1	....

Vocabulary size V = 5

0 → I 1 → like 2 → cats ...

```
current_idx = 0
word2idx = {}
for doc in docs:
    tokens = word_tokenize(doc)
    for token in tokens:
        if token not in word2idx:
            word2idx[token] = current_index
            current_idx += 1
```

→ map token to index value is index

Word2index → Count Vectorizer → TF IDF

- Words not appearing in test set?

if frequency in train set is low → ignore or assign <unk> token

- Reverse Mapping

input # 345 → word

Neural Word embeddings

embedding = vector

convert words into vectors, doc = sequence of vectors

CNN / RNN / Transformers are built for sequences

Word2vec

neural network, weights are the embeddings

NN goal is to predict if a word is in the context of another word

... the fox **jumped** over the lazy dog ...

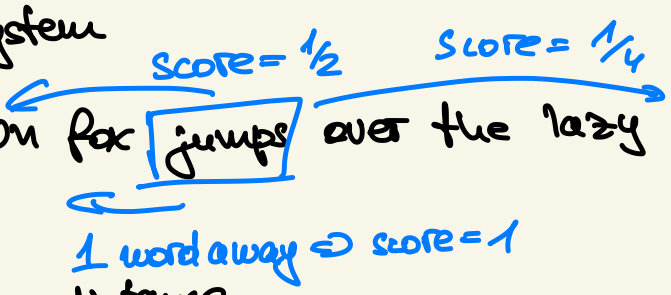
context window

Jumps → NN → Fox ✓
→ Brown ✗
→ Dog ✗

→ weights are the neural embeddings vector

GloVe

like recommender system



Ratings are based on distance
then: how high a word rates another word

Word Vectors

- Embeddings are low-dimentional and dense
- Doc → Tokenize → word vectors → Average(vectors)
  ↳ 1 vector of the same size for a Doc
- Word Analogies
  arithmetic: V(king) − V(queen) = V(man) − V(woman)
  x = V(king) − V(man) + V(woman)
  France : Paris Italy : Rome countries
  Japan : Japanese languages
  Miami : Florida cities
  Dec : Nov months
- Learns meaning from word pos. in vector space
  weights are close to optimal, no need for training