Term Paper

Performance Review Research

Language model optimized with least amount of data
(child language acquisition)

Bert, Roberta — in traditional sense, they are not LLMs
(at least a billion parameter)

Chatgpt - General task solvers

Benchmark of chatgpt in summarization datasets

General task solvers are better than specific task solvers.

Performance of LLM in low research language.

only good at previously seen data

Task contamination

SOTA

Class Notes

Edit distance : way to quantify string similarly

spell mistake: graffe

giraffe graf grail | spell checking

Harvard President Claudine

Harvard university President Claudine | coreference

↓

if two strings contain same entity

Define cost — LEVENSHTEIN

The apple was red

1. The apple was ~~big~~ ~~and~~ red. → 4$ (2$ 2$)
2. The ~~fish~~ was ~~blue~~. → 8$ (2$ 2$ 2$ 2$)

task	cost
i insertion	2$
d deletion	2$
s substitution	

1st was more similar to the string.

↳ minimum Edit distance (check least similarity)

INTENTION → EXECUTION

d s s - i s (cost=8)

INTENTION

EXECUTION

min Edit Distance ?

↓ can start from anywhere

i=1$
d=1$
s=d+i

use DP (Dynamic Programming)

- search problem

Find most cost efficient approach

Edit Distance Algo

✓1. Levenshtein
✓2. Hamming Distance
3. Jaro-winkler

Token-based

✓1. Cosine-similarity (similarity in text/sent)
✓2. Jaccard index
3. Tversky Index

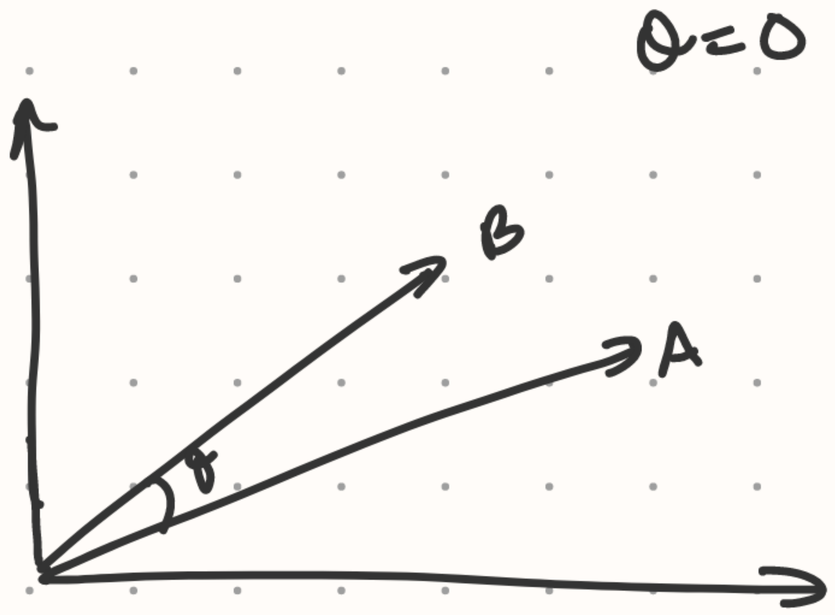
Sequence-based (programming)

1. LCS (Longest common substring)
2. Ratcliff-Obershelp similarity

Cosine-similarity

converts the whole corpus into same dim. vectors and projects it.

similar sentence → similar vectors
more similar → less distance
2 same sentence → project on each other



cosine similarity (doc1, doc2)

Document 1: The cat in the hat.

Document 2: A black cat in a black hat.

DOC → vectors : Find all unique tokens from the docs.

	The	cat	in	hat	a	black
Doc1	2	1	1	1	0	0
Doc2	0	1	1	1	2	2

Frequency

$$A = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

cosine similarity (doc1, doc2)

$$= \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{2.0 + 1.1 + 1.1 + 1.1 + 0.2 + 0.2}{\sqrt{7} \cdot \sqrt{11}}$$

$$= 0.342$$

similar → close to 1
diff → close to -1
⊥ vectors → close to 0
(no similarity / dissimilarity)

— completely diff set of words (0 frequency)

There should be a (-ve) value theoretically.

Insight

1) The patient died before he doc came.
2) the doc came after the patient died.

cosine similarity cannot detect semantic meaning of sentence.

(cons)

computationally strong (Pros)

- sensitive to sentence long. long vector (not necessary) | cons

sparsity — having a lot of 0s in vectors/matrix

cosine similarity create sparse vectors that will dominate the entire value

Jaccard Index is more sensitive to context in finding text similarity.

$$J.I = \frac{A \cap B}{A \cup B}$$

Jaccard similarity (doc1, doc2) = $\frac{doc1 \cap doc2}{doc1 \cup doc2}$

if 2 docs talking about same context

docs → tokens

J.S = $\frac{\{\text{'The'}, \text{'cat'}, \text{'in'}, \text{'the'}, \text{'hat'}\} \cap \{\text{'A'}, \text{'black'}, \text{'cat'}, \text{'in'}, \text{'a'}, \text{'black'}, \text{'hat'}\}}{\cup}$

$= \frac{3}{\{\text{'the'}, \text{'cat'}, \text{'in'}, \text{'hat'}, \text{'a'}, \text{'black'}\}}$

$= \frac{3}{6} \approx 50\%$ similarity.

Imp of tokenization/ lemmatization.

throw, throws, throwing → W/O lemmatization, consider them diff.
(verb terms)