# Similaridade Semântica
## *Semantic Textual Similarity (STS)*

**Prof. Raimundo Moura**
**DC - UFPI**

1

## PLN: Conceitos

- Tarefas que tratam pares de textos
  - Inferência Textual
    - Determinar se o significado de um trecho implica o outro
  - Similaridade Semântica
    - Atribuir uma pontuação de similaridade semântica ao par
    - Problema difícil devido as nuances da língua natural.
    - Ex. Dois textos podem ser similares apesar de não ter uma única palavra em comum.

2

# Similaridade Textual

- Verificar o quão próximos são dois fragmentos de texto a partir do significado e estrutura
  - Frase 1: 'Os' 'gatos' 'comem' 'os' 'ratos'
  - Frase 2: 'Os' 'gatos' 'comem' 'os' 'insetos'

  - Similaridade Semântica: Carro / Automóvel
  - Similaridade Léxica: Carro / Barro

3

# Similaridade Textual: baseada em *Strings*

- *Baseada em termos*
  - *Block distance, Cosine distance, Dice's coefficient, Euclidean distance, Jaccard similarity, Matching coefficient, Overlap coefficient*

- *Baseada em caracteres*
  - *LCS, Damerau-Levenshtein, Jaro, Jaro-Winkler, Needleman-Wunsch, Smith-Waterman, N-gram*

4

## Similaridade Textual: baseada em *Termos*

- Distância entre $s$ e $t$ baseada no conjunto de palavras que aparecem em $s$ e $t$
- A ordem das palavras não é relevante
  - *Ex. "Raimundo Moura" = "Moura Raimundo"*
- *Normalmente as palavras são ponderadas e as mais comuns contam menos*
  - *Ex. "Silva" conta menos que "Digiampietri"*

5

## *Jaccard similarity*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $s$ | | William | Cohen | CM | Univ | | Pgh |
| $t$ | Dr. | William | Cohen | CM | | University | |
| $s \cup t$ | Dr. | William | Cohen | CM | Univ | University | Pgh |
| $s \cap t$ | | William | Cohen | CM | | | |

$$Jaccard\ Score = \frac{|s \cap t|}{|s \cup t|} = \frac{3}{7} = 0{,}428$$

6

## *Jaccard similarity*

- Vantagens:
  - Explora informações de frequência
  - Eficiente: encontrar *{t : sim(t,s)>k}* é sublinear!
  - Ordem das palavras é ignorada: *William Cohen = Coren William*
- Desvantagens:
  - Sensível a erros de digitação: *William ≠ Wiliam ≠ Willian*
  - Sensível a abreviaturas: *Univ vs University*
  - Ordem das palavras é ignorada: *William Cohen = Coren William*

7

## Similaridade Textual: baseada em *Caracteres*

MINIMUM EDIT DISTANCE

- Número mínimo de operações de edição (inserção, exclusão e substituição) necessárias para transformar uma string em outra
- Útil para tarefas como:
  - *Correção ortográfica*
  - *Resolução de correferência*
  - *Identificação de variantes linguísticas ou ortográficas*
  - *Identificação de cognatos*

8

# Similaridade Textual: Levenshtein

- Custo das operações: 1 (Alternativa: alteração = 2)
- Palavra 1: "ABDAC"
- *Palavra 2: "CADA"*
  - *Passo 1: Exclusão da primeira letra 'A', gerando "BDAC"*
  - *Passo 2: Substituição do 'B' por 'C', gerando "CDAC"*
  - *Passo 3: Inserção do 'A' após o 'C', gerando "CADAC"*
  - *Passo 4: Exclusão do 'C' no final da palavra, gerando "CADA"*

*4 operações: distância = 4*

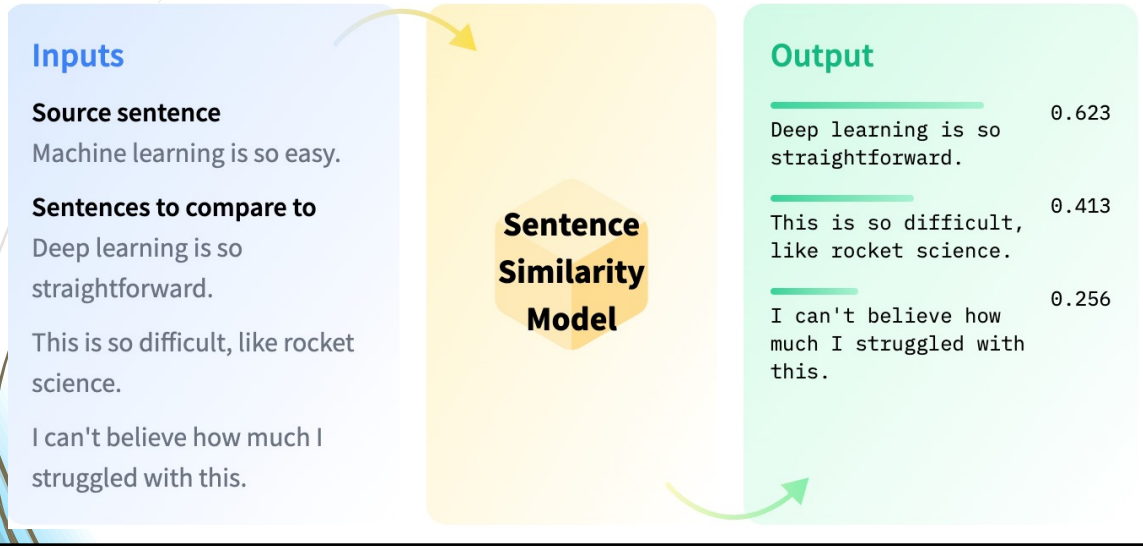
9

# Similaridade Semântica: métodos

- *Dados dois textos de entrada, fornece diretamente um escore de quão similares os dois textos são.*
- *Converter cada texto em um vetor numérico (embedding vector) e calcular matematicamente a distância entre eles*

10

# Similaridade Semântica: exemplo




11

# Similaridade Semântica: aplicação

- Recuperação de Informação
  - Extrair informações de documentos usando modelos de similaridade de sentenças.
    - Ranquear os documentos usando modelos de *Passage Ranking*
    - Obter os *top x* documentos ranqueados e pesquisá-los com modelos de *Sentence Similarity* para selecionar qual deles é o mais similar com a busca de entrada

12

# Biblioteca: SentenceTransformers

- Usada para calcular *embeddings* de sentenças, parágrafos e documentos completos

- **Tarefa:**
  - Ranquear documentos baseado na relevância a um documento base
  - *Input: doc-base e N documentos*
  - *Output: Documentos ranqueados de acordo com a relevância ao doc-base*

**Fonte**: https://www.sbert.net/docs/pretrained_models.html

13

# Biblioteca: SentenceTransformers

```
!pip install -U sentence-transformers

#Transformando as sentenças em embeddings
from sentence_transformers import SentenceTransformer
sentences = ["Universidade Federal do Piauí",
             "Processamento de Língua Natural"]

model = SentenceTransformer(
        'sentence-transformers/all-MiniLM-L6-v2')
embeddings = model.encode(sentences)
print(embeddings.shape) #(2, 384)
```

14

# Biblioteca: SentenceTransformers

```
(2, 384)
[[-1.51431533e-02  2.64892634e-03  8.84302612e-03 -7.22391065e-04
  -7.46961534e-02 -5.17140441e-02 -2.27254722e-02  2.43724827e-02
   4.41656820e-02  6.43426776e-02  4.81258556e-02 -2.99504641e-02
  -4.00524065e-02  6.35213032e-03 -2.59150323e-02 -8.18493813e-02
  -4.66059335e-02 -5.89818247e-02  8.92242938e-02 -4.05034795e-02
   2.79473234e-02 -5.13073169e-02  1.14993555e-02 -1.38607519e-02
  -9.39668193e-02  5.37259541e-02  4.55198996e-02 -5.49434721e-02
   1.04009816e-02 -8.94424170e-02 -2.32860111e-02  1.02503791e-01
   6.15699217e-02 -2.82766670e-02  3.35066654e-02  8.17140285e-03
   4.63760607e-02 -6.58199564e-02  2.26234421e-02  6.39546961e-02
  -4.05692868e-02 -4.10283022e-02  5.00348881e-02 -1.40527366e-02
   2.89945267e-02 -5.20879999e-02  1.00073945e-02 -3.77898626e-02
   4.24707308e-02 -4.04376984e-02 -5.55423200e-02 -2.86444463e-02
   1.54417651e-02 -2.50618532e-02  5.84021071e-03  5.99962138e-02
```

...

15

# Biblioteca: SentenceTransformers

- As sentenças (textos) são mapeadas de forma que sentenças com significados semelhantes fiquem próximas no *espaço vetorial*.
- Métricas de comparação:
  - Similaridade do cosseno (util.cos_sim)
  - Produto vetorial (util.dot_score)
  - Distância enclidiana

16

## *Sentence Embeddings*

- São inerentemente *compressões* de informações em uma sequência de texto e as *compressões* são inerentemente com *perdas.*
- São representações com um nível mais baixo de granularidade.

17

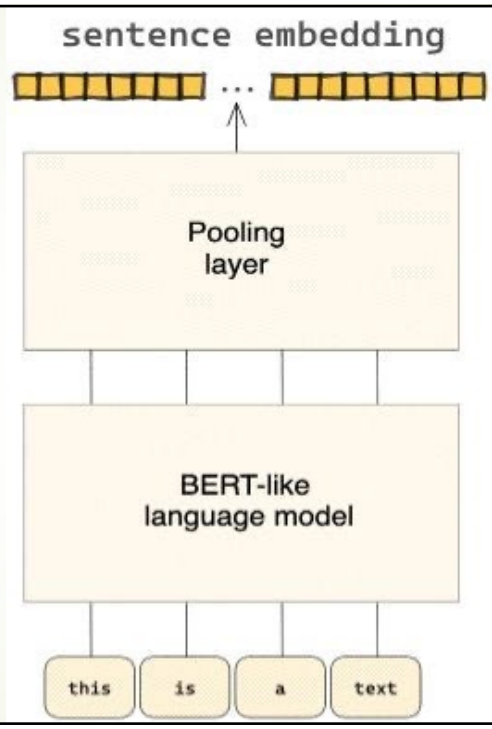
## *Sentence Embeddings: exemplo*

- Um texto que consiste de 200 *tokens*.
- Modelo de Língua: Usa *embeddings* 512 dimensões
- *Embeddings* resultantes: 200 *arrays* distintos de 512 dimensões
- *Sentence embeddings*: Representa a informação na sequência em um único *array* de 512 dimensões.
- *Pooling*: é o processo de converter uma sequência de *tokens embeddings* em uma *sentence embedding*

18

## *Pooling: Funções de agregação*

- *CLS*: saída do primeiro token
- *Mean*: média aritmética elemento a elemento das *embeddings* dos tokens
- Max: valor máximo elemento a elemento das *embeddings*
- *Mean_sqrt_len*: média elemento a elemento dividida pela raiz quadrada do número de tokens na sequência




19

## *Sentence Embeddings: discussão*

- As *Tokens Embeddings* de uma sequência representam muito mais informações.
  1. A *Sentence Embedding* captura as mesmas informações que as *Tokens Embeddings*? ou
  2. Captura informações relacionadas à sequência como um todo e não aos constituintes individuais?
- Alerta de *spoiler*: é o último.

20

## *Sentence Embeddings: discussão*

- São treinadas para tarefas que requerem conhecimento do significado da sentença como um todo e não aos *tokens* individuais.
- Exemplos:
  - Análise de sentimentos
  - Similaridade semântica
  - NSP (*Next Sentence Prediction*)

21

## E sem usar *SentenceTransformers?*

1. Passar a entrada através de um *modelo de transformer*
2. Aplicar uma operação de *pooling* no top das WEs contextualizadas

- *Ver exemplo04_similaridadeSemantica_Wes*
- *Ver exemplo04_Similaridade-Spacy*

22