Word Encoding, Word Embedding and Sentence Embedding

Encoding and embedding are both techniques that transform data into different formats, but they serve different purposes and have different applications:

Encoding
Transforms raw data into a structured format that computers can process. Encoding is used in data compression, character encoding, and data serialization. For example, JPEG is used for image compression, MP3 is used for audio compression, and ASCII and UTF-8 are used for text character encoding.

Embedding
Maps data points into a lower-dimensional space to capture semantic relationships and similarities between data points. Embedding is used in natural language processing, recommendation systems, and deep learning models. For example, Word2Vec and GloVe are used for word embeddings, and item embeddings are used in recommendation systems.

Introduction

In Natural Language Processing (NLP), "word encoding techniques" primarily refer to methods like one-hot encoding and word embeddings which convert words into numerical representations that computers can understand, allowing them to process and analyze text data by capturing semantic relationships between words within a corpus; popular word embedding techniques include Word2Vec, GloVe, and FastText, each with different strengths in capturing contextual meaning.

Sentence Embeddings: BERT (Bidirectional Encoder Representations from Transformers), Universal Sentence Encoder (USE), Sentence-BERT (SBERT).

Sentence Embeddings:

 Capture the semantic meaning of a whole sentence.
 Consider word order and context within the sentence.
 Often used for tasks like text classification, sentiment analysis, and information retrieval.

Word Encoding

TF-IDF

TF-IDF stands for term frequency-inverse document frequency, and it's a measure of how important a word is to a document in a collection. It's used in information retrieval and machine learning to quantify the relevance of words, phrases, and lemmas in a document.

About

TF-IDF is calculated by multiplying two metrics:

Term frequency (TF): The number of times a term appears in a document

Inverse document frequency (IDF): Weighs down frequent terms and increases the weight of rare terms

TF-IDF is useful for:

Text classification

Helping machine learning models read words

Finding relevant documents in digital libraries, databases, and archives

Here's an example of how to calculate TF-IDF:

1. Calculate term frequency

If a blog post has 100 words and the word "JavaScript" appears 5 times, the term frequency is 5/100 = 0.05.

2. Calculate inverse document frequency

If the blog post collection has 10,000 documents and "JavaScript" appears in at least 100 of them, the inverse document frequency is log(10,000/100) = 2.

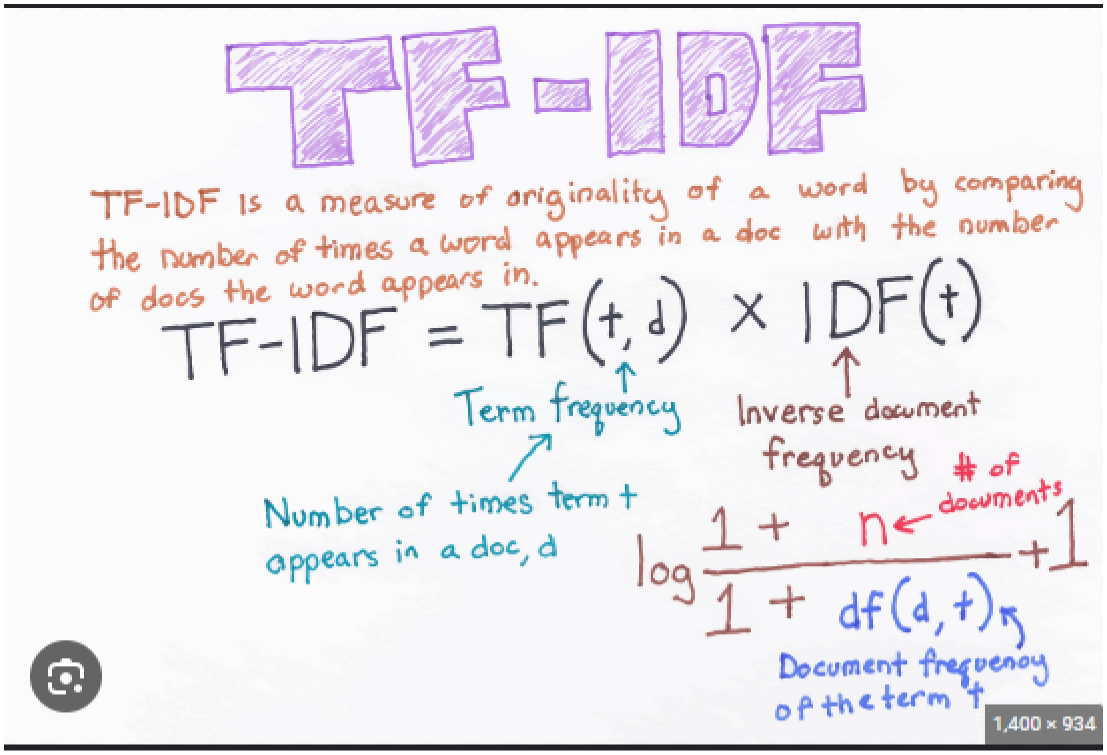
3. Calculate TF-IDF

Multiply the term frequency by the inverse document frequency to get the TF-IDF score: 0.05 * 2 = 0.1.

Mathematical expression

$$TF = \frac{Number\ of\ times\ a\ word\ "X"\ appears\ in\ a\ Document}{Number\ of\ words\ present\ in\ a\ Document}$$

$$IDF = log\left(\frac{Number\ of\ Documents\ present\ in\ a\ Corpus}{Number\ of\ Documents\ where\ word\ "X"\ has\ appeared}\right)$$


TF-IDF
Term frequency
Number of times term t appears in a doc, d
Inverse document frequency
# of documents
Document frequency of the term t
1,400 × 934

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

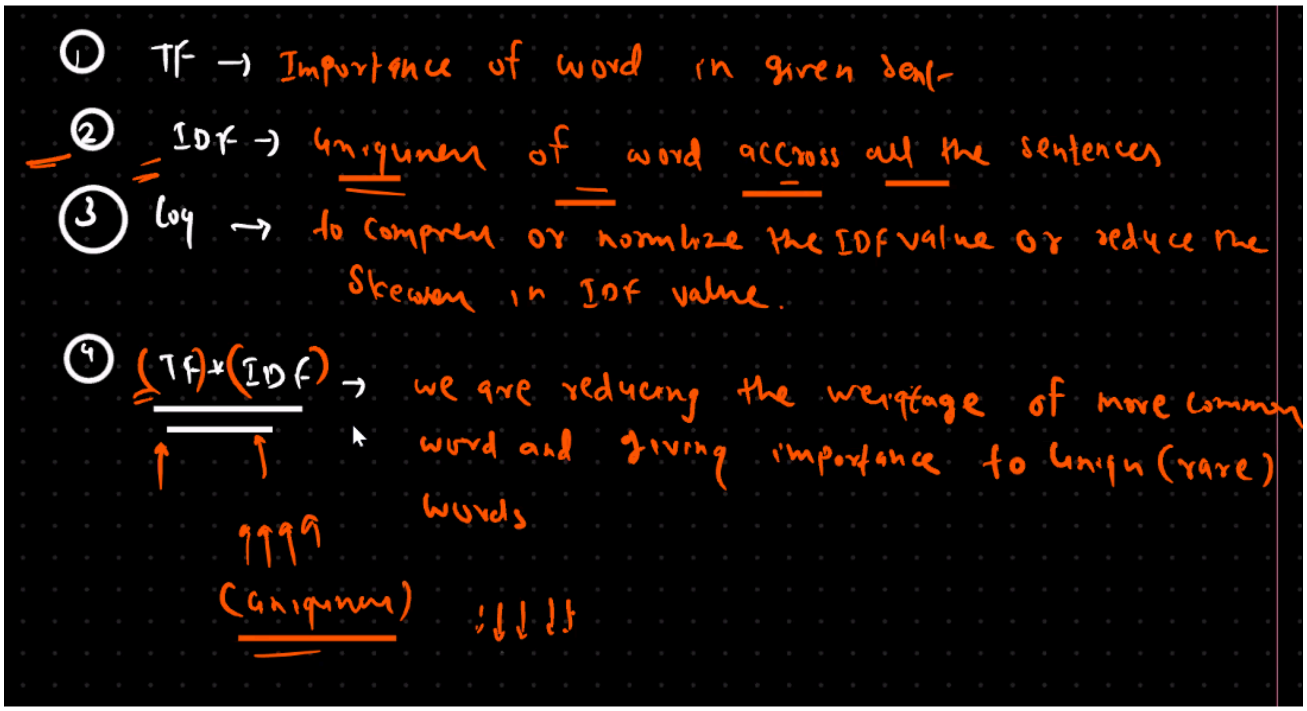
$$\text{TF-IDF} = \text{TF}(t,d) \times \text{IDF}(t)$$

$$\log \frac{1+n}{1+df(d,t)} + 1$$

essence

The essence of Inverse Document Frequency (IDF) is to measure how rare a word is across a collection of documents, assigning higher weights to terms that appear in fewer documents and lower weights to common words, essentially indicating how much information a particular word provides within a corpus by considering its distribution across all documents; the rarer the word, the more informative it is considered to be.

Screenshots



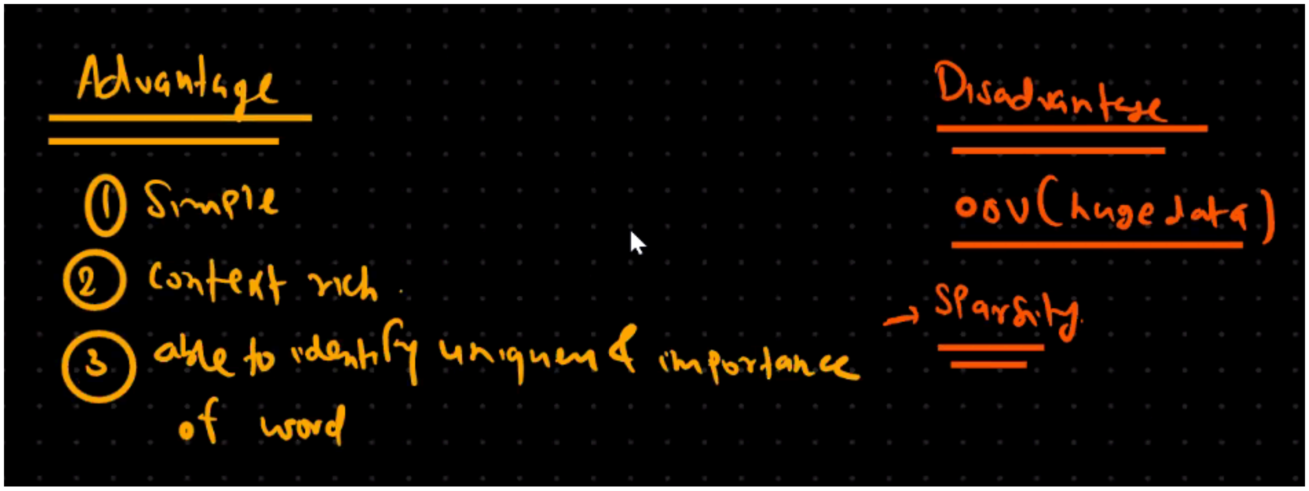
① TF → Importance of word in given sent-

② IDF → Uniqueness of word accross all the sentences

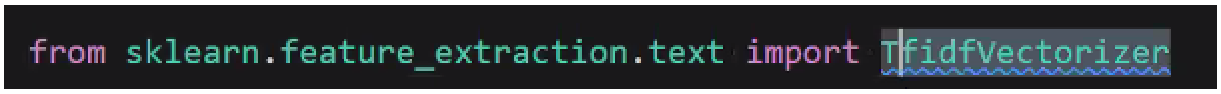
③ log → to compress or normalize the IDF value or reduce the Skewness in IDF value.

④ (TF)*(IDF) → we are reducing the weightage of more common word and giving importance to Uniqu (rare) words

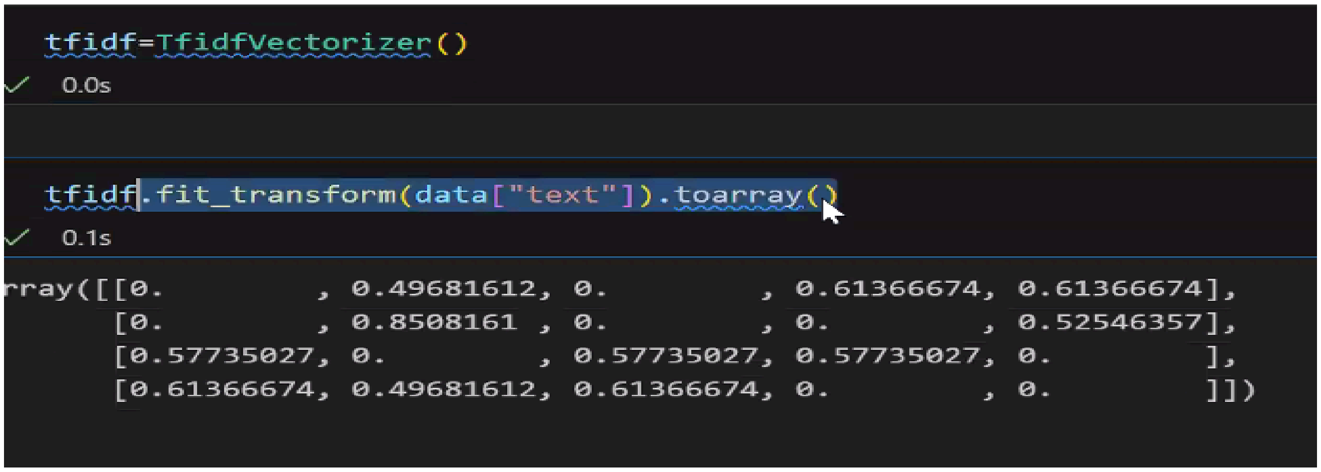
↑↑↑↑ (Uniqueness) ↓↓↓↓


Advantage
① Simple
② Context rich
③ able to identify unique and importance of word
Disadvantage
OOV (huge data)
→ Sparsity

Implementation



```
from sklearn.feature_extraction.text import TfidfVectorizer
```



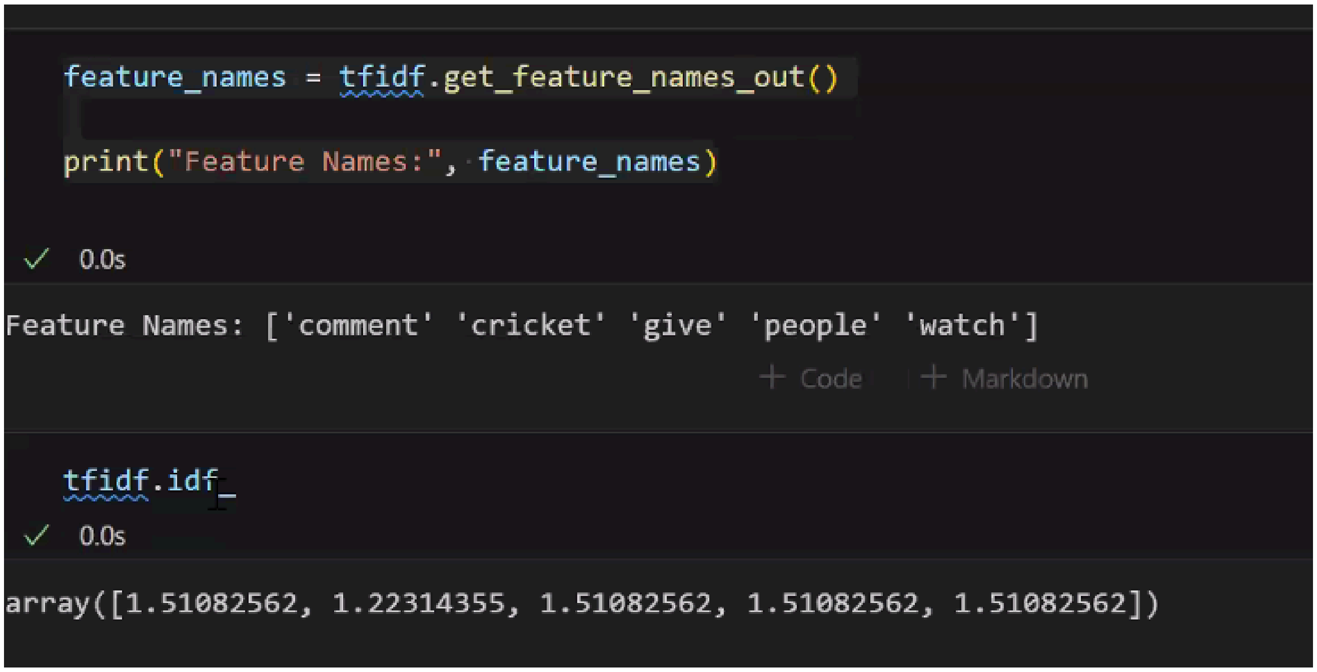
```
tfidf=TfidfVectorizer()
```

✓ 0.0s

```
tfidf.fit_transform(data["text"]).toarray()
```

✓ 0.1s

```
rray([[0.        , 0.49681612, 0.        , 0.61366674, 0.61366674],
      [0.        , 0.8508161 , 0.        , 0.        , 0.52546357],
      [0.57735027, 0.        , 0.57735027, 0.57735027, 0.        ],
      [0.61366674, 0.49681612, 0.61366674, 0.        , 0.        ]])
```



```
feature_names = tfidf.get_feature_names_out()

print("Feature Names:", feature_names)
```

✓ 0.0s

```
Feature Names: ['comment' 'cricket' 'give' 'people' 'watch']
```

+ Code + Markdown

```
tfidf.idf_
```

✓ 0.0s

```
array([1.51082562, 1.22314355, 1.51082562, 1.51082562, 1.51082562])
```