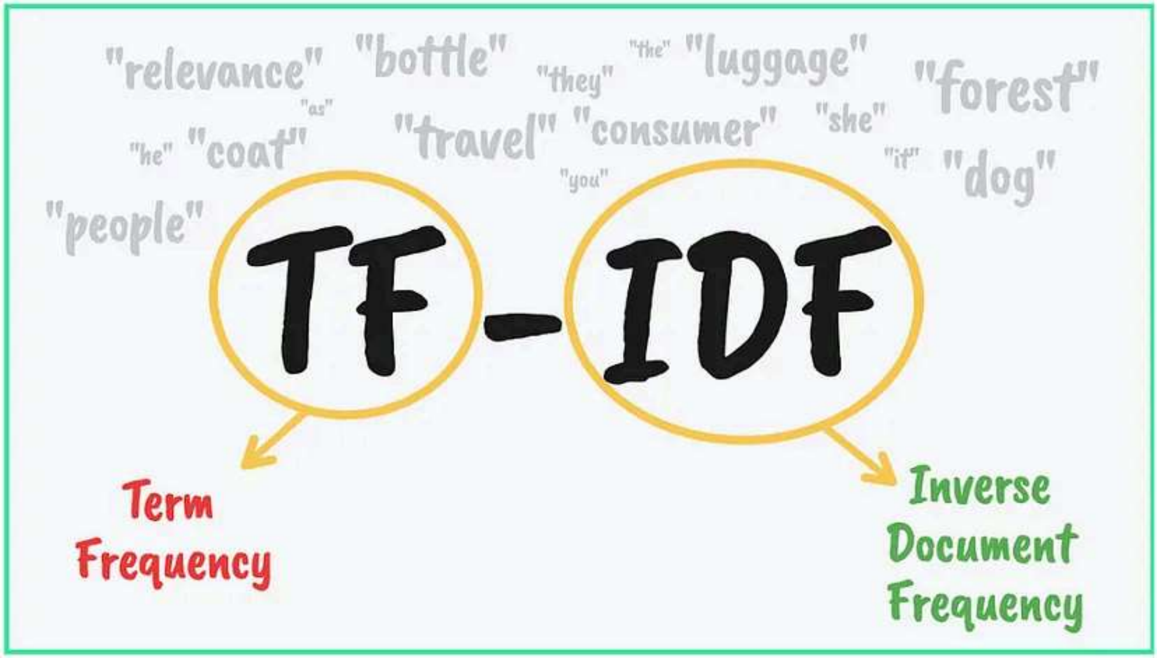
5 — TF-IDF: A Traditional Approach to Feature Extraction in NLP using Python

Aysel Aydin · Follow

4 min read · Oct 29, 2023



In the last article, we covered the topic of Bag of Words, a Natural Language Processing strategy used to convert a text document into numbers that can be used by ML.

The BoW method is simple and works well, but it creates a problem because it treats all words equally. As a result, we can't distinguish very common words or rare words when BoW is used. TF-IDF comes into play at this stage, to solve this problem.

Unlike the bag of words model, the TF-IDF representation takes into account the importance of each word in a document. The term **TF** stands for **term frequency**, and the term **IDF** stands for **inverse document frequency**.

To understand TF-IDF, firstly we should cover the two terms separately:

- Term frequency (TF)
- Inverse document frequency (IDF)

Term Frequency (TF)

Term frequency refers to the frequency of a word in a document. For a specified word, it is defined as the ratio of the number of times a word appears in a document to the total number of words in the document.

$$TF(t, d) = \frac{\text{number of times t appears in d}}{\text{total number of words in d}}$$

— t is the word or token.

— d is the document.

Inverse document frequency (IDF)

Inverse document frequency measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus.

$$\mathrm{IDF}(t) = \log \frac{\text{Total number of documents}}{\text{number of documents that contain t}}$$

TF-IDF Score

The TF-IDF score for a term in a document is calculated by multiplying its TF and IDF values. This score reflects how important the term is within the context of the document and across the entire corpus. Terms with higher TF-IDF scores are considered more significant.

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

Let’s take an example to understand this concept in depth.

Let’s examine the example we gave for bag of words in our previous article with TF-IDF.

Imagine a social media platform that aims to analyze customer reviews and understand the popularity of services among users. This platform decides to employ the **TF-IDF** method for processing customer reviews.

Step 1: Data Preprocessing

```
from nltk.stem import WordNetLemmatizer
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords

def preprocessing_text(text):
    lemmatizer = WordNetLemmatizer()
    emoji_pattern = r'^(?:[\u2700-\u27bf]|(?:\ud83c[\udde6-\uddff]){1,2}|(?:\ud8

    text= text.lower()
    text = text.split()
    text = [lemmatizer.lemmatize(word) for word in text if not word in set(stopw
    text = ' '.join(text)
    text = re.sub(r'[0-9]+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(emoji_pattern, '', text)
    text= re.sub(r'\s+', ' ', text)

    return text

comments = """I am really disappointed this product.
I would not use it again. It has really bad feature.
I love this product! It has some good features"""

sentences_list = nltk.sent_tokenize(comments)

corpus = [preprocessing_text(sentence) for sentence in sentences_list]

print(corpus)
```

Output:

```
[
'really disappointed product',
'would use again',
'really bad feature',
'love product',
'good feature'
]

Unique Word List:
['again' 'bad' 'disappointed' 'feature' 'good' 'love' 'product' 'really'
 'use' 'would']
```

Step 2: Calculating Product of Term Frequency & Inverse Document Frequency

```python
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

corpus = [
    'really disappointed product',
    'would use again',
    'really bad feature',
    'love product',
    'good feature'
]

tfidf_vectorizer = TfidfVectorizer()

tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)

terms = tfidf_vectorizer.get_feature_names_out()

df = pd.DataFrame(tfidf_matrix.toarray(), columns=terms)

print(df)
```

Output:

Comments	again	bad	disappointed	feature	good	love	product	really	use	would
really disappointed product	0	0	0.659118	0	0	0	0.531772	0.531772	0	0
would use again	0.57735	0	0	0	0	0	0	0	0.57735	0.57735
really bad feature	0	0.659118	0	0.531772	0	0	0	0.531772	0	0
love product	0	0	0	0	0	0.778283	0.627914	0	0	0
good feature	0	0	0	0.627914	0.778283	0	0	0	0	0

> *The TF-IDF calculated in Scikit-learn's TfidfTransformer and TfidfVectorizer is slightly different from the standard calculation. A constant 1 is added to the numerator and denominator of the IDF as if an extra document was seen containing every term in the collection exactly once, which prevents zero divisions. The standard calculation present doesn't have the constant 1.*
>
> *(Github: Scikit-Learn)*

Scikit Learn

$$\bullet\ IDF(t) = \log \frac{1 + n}{1 + df(t)} + 1$$

Standard calculation

$$\bullet\ IDF(t) = \log \frac{n}{df(t)}$$

Conclusion

Through this article, we have explained Term Frequency — Inverse Document Frequency (TF-IDF) and how to use with Python and NLP techniques.

> *In the next article, we will create a word cloud using NLP and TF-IDF in Python.*

I hope it will be a useful article for you. If you stayed with me until the end, thank you for reading! Happy coding 🤞

Contact Accounts: Twitter, LinkedIn

Tf Idf NLP Python Scikit Learn Sklearn

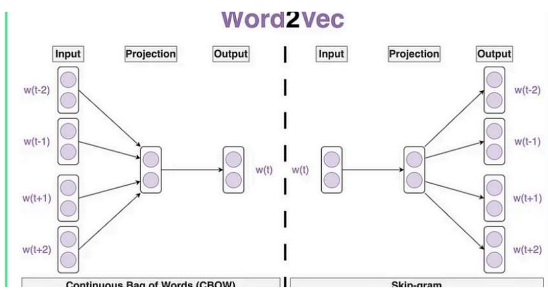
Written by Aysel Aydin

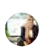
920 Followers · 72 Following

Follow

Master Expert AI & ML Engineer @Turkcell

More from Aysel Aydin

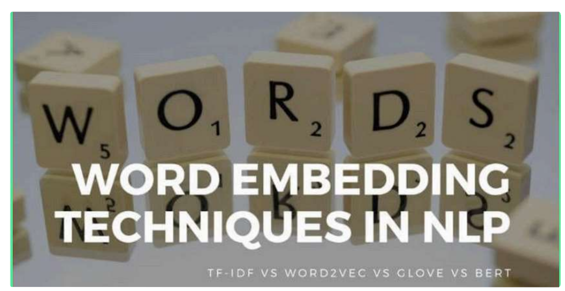




Aysel Aydin

11—Word2Vec Approaches: Word Embedding in NLP

In this article, we will talk about Continuous Bag of Words (CBOW) and Skip-Gram, whic...



Aysel Aydin

9—Understanding Word Embeddings in NLP

In this article, we will talk about word embedding and techniques, their usage...

Jul 19 101

Jun 30 164 1

In Mobile Growth Istanbul by Aysel Aydin

App Store Optimization (ASO) Nedir ve Nasıl Yapılır?

Merhaba arkadaşlar, bugün sizlerle Android uygulamalarımızı geliştirdikten sonra…

Mar 8, 2021 715 2

Aysel Aydin

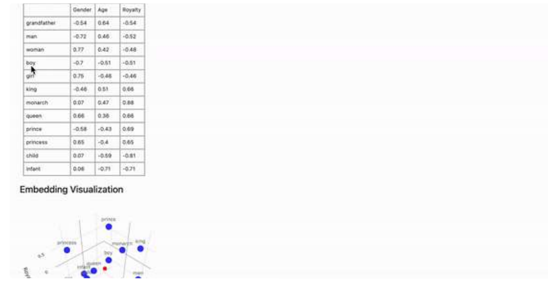
13—Understanding FastText: Efficient Word Representations fo…

In this article, we will talk about FastText one of the word embedding techniques. Until no…

Aug 11 79

See all from Aysel Aydin

Recommended from Medium



Rahul Kumar

NLP Hands-On with Text Classification

This post is a part of the NLP Hands-on series and consists of the following tasks: 1. Text...

Oct 28

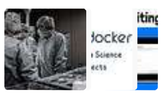
Muneeb S. Ahmad

Mastering NLP with GloVe Embeddings: Word Similarity,...

Introduction

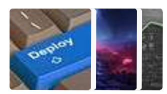
Oct 22

Lists



Coding & Development

11 stories · 927 saves



Predictive Modeling w/ Python

20 stories · 1700 saves

Practical Guides to Machine Learning

10 stories · 2068 saves

Natural Language Processing

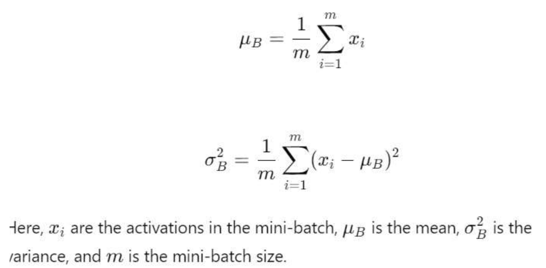
1842 stories · 1466 saves

Mdabdullahalhasib

A Complete Guide of Output Parser with LangChain Implementation

Explore how we can get output from the LLM model into any structural format like CSV,...

Oct 5 · 71



$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

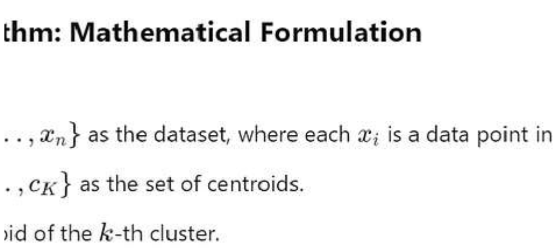
Here, x_i are the activations in the mini-batch, μ_B is the mean, σ_B^2 is the variance, and m is the mini-batch size.

Jo Wang

Deep Learning Part 5 -How to prevent overfitting

Techniques used to prevent overfitting in deep learning models:

Jun 29 · 1 · 1



thm: Mathematical Formulation

. . , $x_n\}$ as the dataset, where each x_i is a data point in

. , $c_K\}$ as the set of centroids.

oid of the k-th cluster.

Sujatha Mudadla

K-Means Clustering Algorithm

K-Means Clustering Algorithm:

Nov 18 · 2

Mayurkumar Surani

End-to-End ETL Process with PySpark and Scala: From MySQL ...

In the world of big data, ETL (Extract, Transform, Load) processes are crucial for...

Aug 31 · 55 · 1