Bag of Words vs TF-IDF — Penjelasan dan Perbedaannya

Affandy Fahrizain · Follow

Published in Data Folks Indonesia

7 min read · Nov 1, 2021

Photo by Alfons Morales on Unsplash

> *Ketika kita berhubungan dengan data teks seperti klasifikasi teks misalnya, kita tentunya harus melakukan transformasi data teks menjadi sekumpulan angka (vektor) terlebih dahulu sebelum melakukan modelling. Nah, 2 metode yang cukup populer diantaranya*

> *adalah Bag of Words dan TF-IDF. Mari kita bahas bagaimana mereka bekerja serta apa perbedaannya!*

The Story

Bayangkan saja kita adalah pemilik restoran. Setiap pengunjung selesai makan, kita meminta mereka untuk menuliskan review dari segi apapun sebagai bahan evaluasi restoran. Dan setiap akhir bulan kita melakukan evaluasi berdasarkan review pengunjung. Kebetulan, bulan ini kita mendapat 3 review yang isinya seperti berikut:

Review 1: *Makanan disini gurih dan enak!*

Review 2: *Makanan disini biasa saja.*

Review 3: *Makanan disini hambar dan tidak enak!*

Sebagai pemilik restoran yang melek IT, kita ingin seluruh review nantinya diproses menggunakan komputer. Sayangnya oh sayangnya, komputer tidak mengerti bahasa manusia. Mereka hanya memahami angka. Oleh karena itu, kita perlu melakukan transformasi terhadap data kita dari teks menjadi sekumpulan angka yang biasa disebut vektor. Yuk, mari kita lakukan!

Bag of Words

Bag of Words (BoW) merupakan salah satu metode paling sederhana dalam mengubah data teks menjadi vektor yang dapat dipahami oleh komputer. Metode ini sejatinya hanya menghitung frekuensi kemunculan kata pada seluruh dokumen.

Mari kita ingat kembali contoh yang sudah kita baca sebelumnya.

Review 1: *Makanan disini gurih dan enak!*

Review 2: *Makanan disini biasa saja.*

Review 3: *Makanan disini hambar dan tidak enak!*

Pertama, kita abaikan tanda baca serta huruf kapital dari ketiga review tersebut. Kemudian kita bisa membentuk sebuah korpus / kamus kata seperti berikut.

“makanan”

“disini”

“gurih”

“dan”

“enak”

“biasa”

“saja”

“hambar”

“tidak”

Perlu diperhatikan sebelumnya, bahwa dalam membentuk korpus, kita hanya menghitung kata secara unik. Artinya, setiap kata yang berulang hanya akan ditulis sekali.

Berikutnya, mari kita hitung frekuensi kemunculan kata di korpus tersebut kepada ketiga review sebelumnya. Kita beri nilai **1** jika kata tersebut muncul pada sebuah review dan **0** jika tidak muncul.

Agar lebih mudah dalam memahminya, mari kita perhatikan tabel berikut.

	makanan	disini	gurih	dan	enak	biasa	saja	hambar	tidak
Review 1	1	1	1	1	1	0	0	0	0
Review 2	1	1	0	0	0	1	1	0	0
Review 3	1	1	0	1	1	0	0	1	1

Perhitungan Bag of Words (BoW)

Dari tabel tersebut, akhirnya kita dapatkan vektor dari setiap review seperti berikut.

Vektor Review 1 = [1, 1, 1, 1, 1, 0, 0, 0, 0]

Vektor Review 2 = [1, 1, 0, 0, 0, 1, 1, 0, 0]

Vektor Review 3 = [1, 1, 0, 1, 1, 0, 0, 1, 1]

Itulah konsep dari Bag of Words, cukup mudah bukan? Namun, meski demikian metode ini ternyata memiliki beberapa kekurangan. Yuk mari kita ulas.

Kekurangan Bag of Words

1. Ukuran korpus Bag of Words mengikuti jumlah kata unik dari seluruh dokumen. Artinya, jika nantinya terdapat berbagai kata unik baru maka ukuran korpus juga akan semakin membesar. Tentunya hal ini akan berpengaruh pada komputasi yang dibutuhkan pada saat kita melatih model machine learning.

2. Seperti yang kita lihat pada tabel diatas, ada banyak angka 0 dalam vektor kita. Kondisi ini biasa juga disebut dengan *sparse matrix*. Hal tersebut harusnya kita hindari karena model harus menemukan informasi yang sedikit dalam ukuran data yang besar, yang tentunya juga akan membutuhkan proses komputasi lebih tinggi.

3. Bag of Words menghilangkan konteks kalimat akibat tidak memperhatikan urutan kata.

TF-IDF

TF-IDF merupakan singkatan dari *Term Frequency — Inverse Document Frequency*. Sejatinya, TF-IDF merupakan gabungan dari 2 proses yaitu ***Term Frequency*** **(TF)** dan ***Inverse Document Frequency*** **(IDF)**.

TF-IDF biasa digunakan ketika kita ingin mengubah data teks menjadi vektor namun dengan memperhatikan apakah sebuah kata tersebut cukup informatif atau tidak. Mudahnya, TF-IDF membuat kata yang sering muncul memiliki nilai yang cenderung kecil, sedangkan untuk kata yang semakin jarang muncul akan memiliki nilai yang cenderung besar. Kata yang sering muncul disebut juga ***Stopwords*** biasanya dianggap kurang penting, salah satu contohnya adalah kata hubung (yang, di, akan, dengan, dll).

Sekarang, mari kita coba aplikasikan TF-IDF terhadap 3 review yang telah kita miliki sebelumnya.

Term Frequency (TF)

Term Frequency (TF) menghitung frekuensi jumlah kemunculan kata pada sebuah dokumen. Karena panjang dari setiap dokumen bisa berbeda-beda, maka umumnya nilai TF ini dibagi dengan panjang dokumen (jumlah seluruh kata pada dokumen).

$$tf_{t,d} = \frac{n_{t,d}}{Total\ number\ of\ terms\ in\ document}$$

Rumus Term Frequency (TF)

Keterangan

tf = frekuensi kemunculan kata pada sebuah dokumen

Mari kita ambil contoh kalimat Review 1 untuk dihitung nilai TF nya.

Review 1: *Makanan disini gurih dan enak!*

- Korpus = [“makanan”, “disini”, “gurih”, “dan”, “enak”]
- Panjang kalimat = 5

Sehingga perhitungan untuk nilai TF nya menjadi:

- TF(“**makanan**”) = 1/5 ≈ 0.2
- TF(“**disini**”) = 1/5 ≈ 0.2
- TF(“**gurih**”) = 1/5 ≈ 0.2
- TF(“**dan**”) = 1/5 ≈ 0.2
- TF(“**enak**”) = 1/5 ≈ 0.2

Berikutnya, mari kita coba terapkan pada seluruh review dan kita formulasikan ke dalam bentuk tabel seperti berikut.

	R1	R2	R3	TF1	TF2	TF3
makan	1	1	1	0.2	0.25	0.167
disini	1	1	1	0.2	0.25	0.167
gurih	1	0	0	0.2	0	0.000
dan	1	0	1	0.2	0	0.167
enak	1	0	1	0.2	0	0.167
biasa	0	1	0	0	0.25	0.000
saja	0	1	0	0	0.25	0.000
hambar	0	0	1	0	0	0.167
tidak	0	0	1	0	0	0.167

Perhitungan Term Frequency (TF)

R1, R2, R3 merupakan notasi untuk setiap *Review 1, Review 2,* dan *Review 3*. Sedangkan **TF1, TF2, TF3** merupakan notasi untuk nilai *Term Frequency* setiap Review.

Inverse Document Frequency (IDF)

Setelah kita berhasil menghitung nilai Term Frequency, selanjutnya kita hitung nilai **Inverse Document Frequency** (IDF), yang merupakan nilai untuk mengukur seberapa penting sebuah kata. IDF akan menilai kata yang sering muncul sebagai kata yang kurang penting berdasarkan kemunculan kata tersebut pada seluruh dokumen. Semakin kecil nilai IDF maka akan dianggap semakin tidak penting kata tersebut, begitu pula sebaliknya.

$$idf_d = log(\frac{Number\ of\ document}{Number\ of\ document\ with\ term\ t'})$$

Rumus Inverse Document Frequency (IDF)

Setiap review yang diberikan oleh pelanggan merupakan sebuah dokumen. Karena pada tulisan ini kita mempunyai 3 review, maka artinya kita mempunyai 3 dokumen.

Mari kita coba hitung nilai IDF untuk masing-masing kata pada Review 1.

Review 1: *Makanan disini gurih dan enak!*

- Korpus = [“makanan”, “disini”, “gurih”, “dan”, “enak”]
- Jumlah dokumen = 3

Sehingga perhitungan untuk nilai IDF nya menjadi:

- IDF(“**makanan**”) = $log(\frac{3} {3})$ ≈ 0
- IDF(“**disini**”) = $log(\frac{3} {3})$ ≈ 0
- IDF(“**gurih**”) = $log(\frac{3} {1})$ ≈ 0.48
- IDF(“**dan**”) = $log(\frac{3} {2})$ ≈ 0.18
- IDF(“**enak**”) = $log(\frac{3} {2})$ ≈ 0.18

Sekarang, mari kita coba terapkan pada seluruh kata dan kita lengkapi tabel TF sebelumnya seperti berikut.

	R1	R2	R3	TF1	TF2	TF3	IDF
makan	1	1	1	0.2	0.25	0.167	0
disini	1	1	1	0.2	0.25	0.167	0
gurih	1	0	0	0.2	0	0.000	0.48
dan	1	0	1	0.2	0	0.167	0.18
enak	1	0	1	0.2	0	0.167	0.18
biasa	0	1	0	0	0.25	0.000	0.48
saja	0	1	0	0	0.25	0.000	0.48
hambar	0	0	1	0	0	0.167	0.48
tidak	0	0	1	0	0	0.167	0.48

Perhitungan Inverse Document Frequency (IDF)

Term Frequency — Inverse Document Frequency (TF-IDF)

Setelah kita punya TF dan IDF, berikutnya kita bisa menghitung nilai TF-IDF yang merupakan hasil perkalian dari TF dan IDF.

$$tfidf_{t,d} = tf_{t,d} \times idf_d$$

Rumus TF-IDF

Karena kita sudah memiliki nilai TF dan IDF untuk setiap kata, maka mari kita coba hitung nilai TF-IDF untuk setiap kata pada Review 1.

Review 1: *Makanan disini gurih dan enak!*

makanan

- TF(“**makanan**”) = 1/5 ≈ 0.2
- IDF(“**makanan**”) = $log(\frac{3} {3})$ ≈ 0
- TFIDF(“**makanan**”) = $0.2 \times0=0$

disini

- TF(“**disini**”) = 1/5 ≈ 0.2
- IDF(“**disini**”) = $log(\frac{3} {3})$ ≈ 0
- TFIDF(“**disini**”) = $0.2 \times0=0$

gurih

- TF(“**gurih**”) = 1/5 ≈ 0.2
- IDF(“**gurih**”) = $log(\frac{3} {1})$ ≈ 0.48
- TFIDF(“**gurih**”) = $0.2 \times0.48=0.095$

dan

- TF(“**dan**”) = 1/5 ≈ 0.2
- IDF(“**dan**”) = $log(\frac{3} {2})$ ≈ 0.18
- TFIDF(“**makanan**”) = $0.2 \times0.18=0.035$

enak

- TF(“**enak**”) = 1/5 ≈ 0.2

- IDF(“**enak**”) = $log(\frac{3} {2})$ ≈ 0.18
- TFIDF(“**makanan**”) = $0.2 \times0.18=0.035$

Sekarang, mari kita coba lengkapi tabel sebelumnya dengan nilai TF-IDF pada seluruh kata seperti berikut.

	R1	R2	R3	TF1	TF2	TF3	IDF	TFIDF1	TFIDF2	TFIDF3
makan	1	1	1	0.2	0.25	0.167	0	0	0	0
disini	1	1	1	0.2	0.25	0.167	0	0	0	0
gurih	1	0	0	0.2	0	0.000	0.48	0.095	0	0
dan	1	0	1	0.2	0	0.167	0.18	0.035	0	0.0293
enak	1	0	1	0.2	0	0.167	0.18	0.035	0	0.0293
biasa	0	1	0	0	0.25	0.000	0.48	0	0.119	0
saja	0	1	0	0	0.25	0.000	0.48	0	0.119	0
hambar	0	0	1	0	0	0.167	0.48	0	0	0.080
tidak	0	0	1	0	0	0.167	0.48	0	0	0.080

Perhitungan Term Frequency — Inverse Document Frequency (TF-IDF)

> *Note: Mungkin untuk sebagian perhitungan, angkanya tidak presisi dikarenakan tools yang saya gunakan. Semoga bisa dimaklumi dan tetap bisa diambil konsepnya* 🙂

Dari tabel tersebut, akhirnya kita dapatkan vektor dari setiap review yang dinotasikan oleh ***TFIDF1, TFIDF2,*** dan ***TFIDF3*** seperti berikut.

Vektor Review 1 = [0, 0, 0.095, 0.035, 0.035, 0, 0, 0, 0]

Vektor Review 2 = [0, 0, 0, 0, 0, 0.119, 0.119, 0, 0]

Vektor Review 3 = [0, 0, 0, 0.0293, 0.0293, 0, 0, 0.080, 0.080]

Kekurangan TF-IDF

1. TF-IDF sejatinya berdasar pada Bag of Words (BoW), sehingga TF-IDF pun tidak bisa menangkap posisi teks dan semantiknya.
2. TF-IDF hanya berguna sebagai fitur di level leksikal.

So, itulah perbedaan antara Bag of Words (BoW) dan TF-IDF sebagai metode untuk transformasi teks menjadi vektor. Jika ada pertanyaan, diskusi, sanggahan, kritik, maupun saran jangan pernah ragu untuk menuliskannya di kolom komentar 🙂

Sekian tulisan saya kali ini, mohon maaf apabila ada kekurangan dan salah kata, semoga bermanfaat. Terima kasih!

Yuk, belajar dan diskusi lebih lanjut tentang seputar Data Science, Artificial Intelligence, dan Machine Learning dengan gabung di discord Jakarta AI Research. Dan jangan lupa follow medium Data Folks Indonesia biar nggak ketinggalan update terbaru dari kami.

Referensi

1. https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/
2. https://machinelearningmastery.com/gentle-introduction-bag-words-model/
3. http://www.tfidf.com/
4. https://www.quora.com/What-are-the-advantages-and-disadvantages-of-TF-IDF

NLP | Tf Idf | Bag Of Words | Text Mining

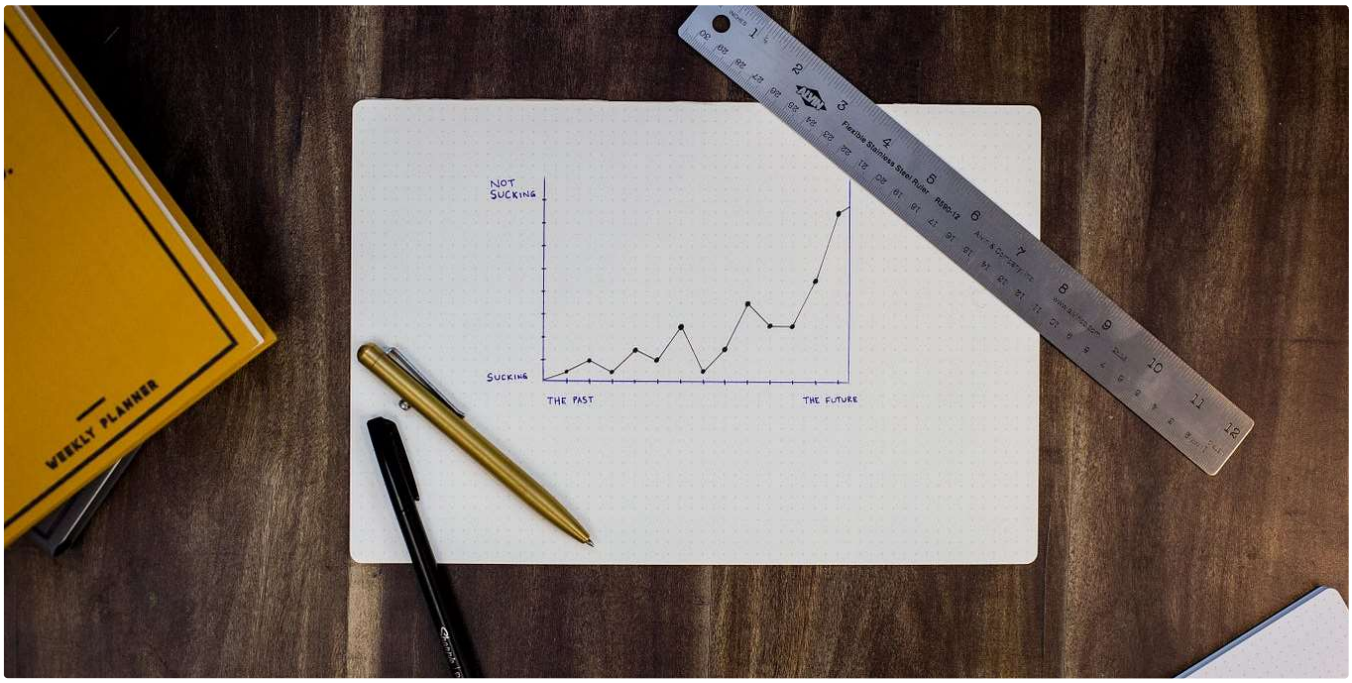
Follow

Written by Affandy Fahrizain

52 Followers · Writer for Data Folks Indonesia

Machine Learning / NLP Enthusiast | Student @ITMO University, Russia

More from Affandy Fahrizain and Data Folks Indonesia



Affandy Fahrizain in Data Folks Indonesia

Quick Export your Jupyter Notebook to PDF

What if I tell you there are some way to simply export your beloved notebook to PDFs?

4 min read · Aug 7, 2022

84 1

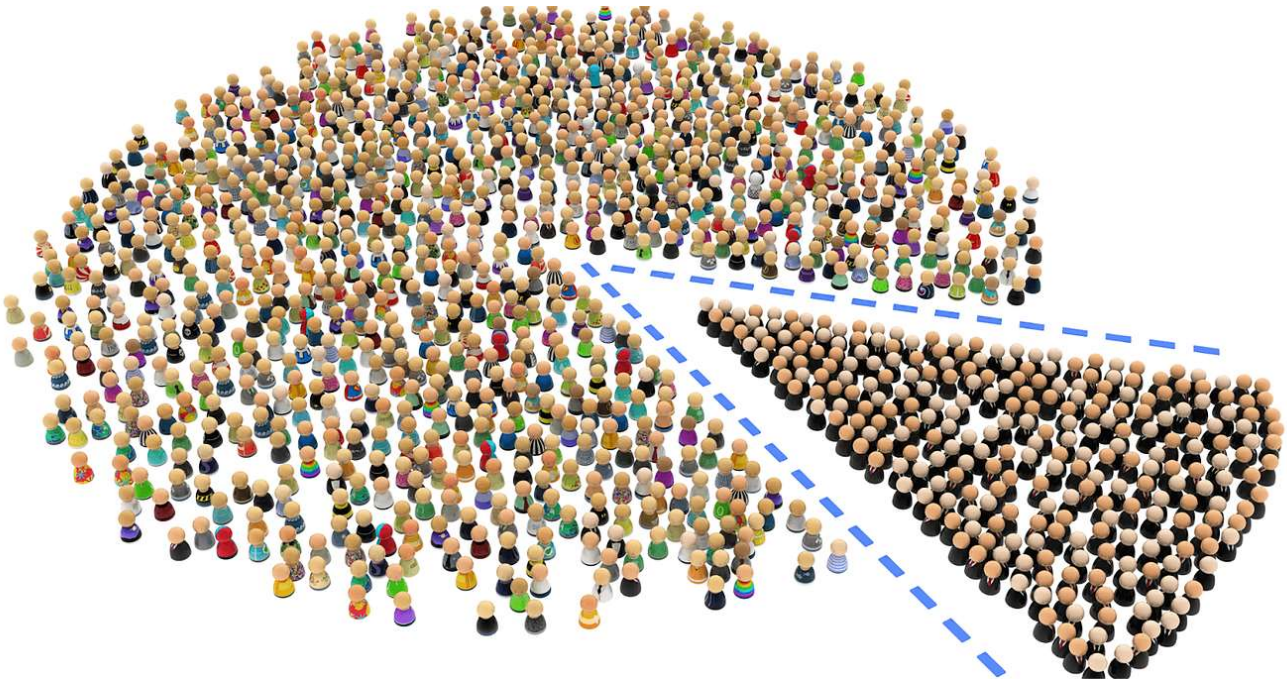
Andhika S Pratama in Data Folks Indonesia

5 Best Free Image Annotation Tools

Image annotation is one of the techniques of labeling data for supervised machine learning. To do image annotation, one must need a...

6 min read · Jan 12, 2021

212 2



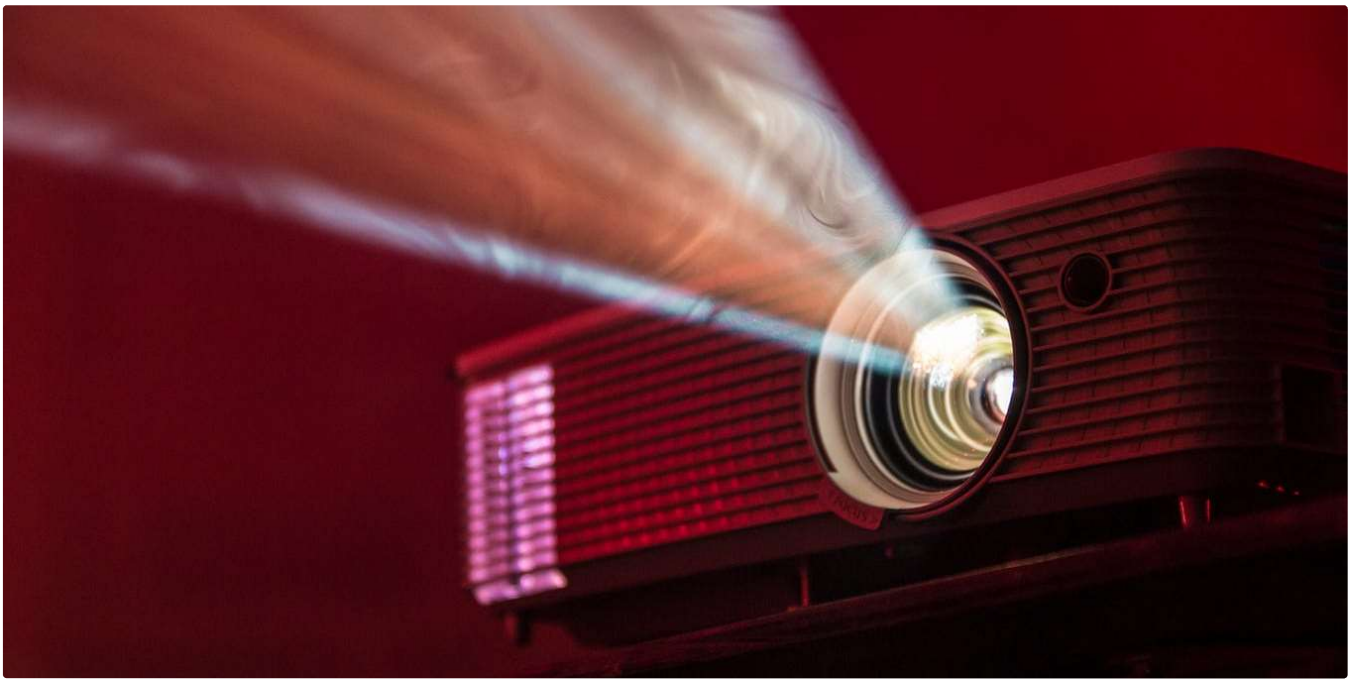
Arif R in Data Folks Indonesia

Step by Step to Understanding K-means Clustering and Implementation with sklearn

Simple explanation regarding K-means Clustering in Unsupervised Learning and simple practice with sklearn in python

6 min read · Oct 4, 2020

91 1



Affandy Fahrizain in Data Folks Indonesia

Exploring Visual Transformers (ViT) with 🤗 Huggingface

An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale (Dosovitskiy et al., 2021)

8 min read · Oct 14, 2022

Rob W in Artificially Intelligent

The Best Way to Determine Word Relevance: Understand TF-IDF in a Hurry

How does the internet know which terms are most representative of a document's contents? Hint: it's not just volume

5 min read · Apr 28

4

```
or([[-1.8824e-01, -7.6512e-04,  1.0336e-01,  ..., -2.0809e
     -3.9280e-01,  7.9072e-01],
    [-4.1441e-01, -1.7607e-01,  5.4728e-02,  ..., -1.0659e
     -3.8406e-01,  7.9451e-01],
    [-5.5160e-01,  1.7655e-01,  2.4592e-01,  ...,  1.5593e
     -5.1121e-01,  1.3524e+00],
    [-2.6705e-01,  2.0308e-01, -3.6436e-02,  ..., -9.6218e
     -5.8836e-01,  6.6819e-01],
    [-1.7557e-01,  1.8462e-01,  3.5970e-02,  ..., -1.4965e
     -3.1363e-01,  6.9866e-01],
    [-1.6752e-01, -3.2122e-01,  7.4659e-02,  ..., -2.1811e
     -3.7288e-01,  7.0560e-01]])
```

Abhijat Sarari in Python in Plain English

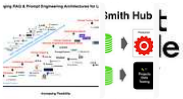
How to Generate Word Embedding Using BERT?

Introduction

9 min read · Aug 28

2

Lists



Natural Language Processing

721 stories · 319 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 149 saves

New_Reading_List

174 stories · 151 saves

Staff Picks

476 stories · 356 saves

Stefan Neefischer

Semantic keyword clustering in Python

We already shared some clustering approaches using TF-IDF Vectorizer for grouping keywords together. This works great for grouping keywords...

3 min read · May 16

29

Data Products

How To Visualize Word Embeddings with Google Embedding Projector in Tensorboard

Learn how to use Google Embedding Projector to visualize word embeddings. Visualize words in three dimensions, see similarities and contrast

· 9 min read · Sep 12

2

3 And this is the third one.
4 Is this the first document?

DF VALUES

and	document	first	is	one	second	the	third	this
1	3	2	4	1	1	4	1	4

IDF VALUES

and	document	first	is	one	second	the	third	this
1.91629073	1.22314355	1.51082562	1	1.91629073	1.91629073	1	1.91629073	1

TF VALUES

	and	document	first	is	one	second	the	third	this
1	0	1	1	1	0	0	1	0	1
2	0	2	0	1	0	1	1	0	1
3	1	0	0	1	1	0	1	1	1
4	0	1	1	1	0	0	1	0	1

TFIDF VALUES

	and	document	first	is	one	second	the	third	this
1	0	0.46979139	0.58028582	0.38408524	0	0	0.38408524	0	0.38408524

Mohamad Mahmood in Dev Genius

TFIDF Calculation Using SKLearn's TfidfVectorizer

accompanied by the step-by-step manual TFIDF calculation

8 min read · Jul 20

6

Farhan Sarguroh

03: Basic Text Preprocessing (NLP)

Text preprocessing is an essential step in natural language processing (NLP) tasks that involves cleaning and transforming raw text data...

6 min read · Jul 11