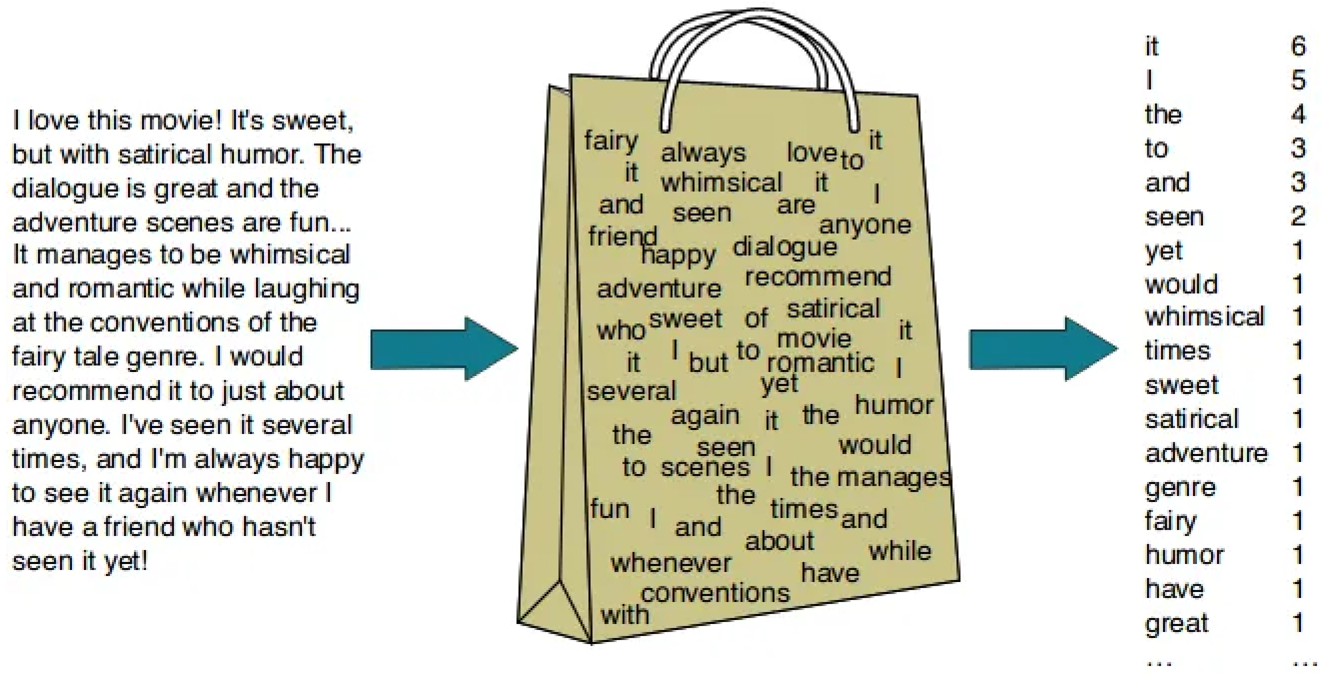
NLP: Bag of words and TF-IDF explained!

Koushik kumar · Follow

5 min read · Jan 24, 2021



Source: Bag of words!

In the previous article, we have been through tokenization, use of stop words, stemming and lemmatization. Basically, processing the text while it is still readable. To give this data as input to any model, we'd need to transform them to some numerical format — 'The vectors'. Let us go through a few ways this can be done.

1. Bag of words
2. TF-IDF
3. Word2Vec

We do know how a document or paragraph text data can be tokenized, breaking it down into a list of sentences or words. We will now be taking these tokenized units, map them to a vector and send as input to the model.

Bag of words

In bag of words, we take all unique words from the corpus, note the frequency of occurrence and sort them in descending order. All the word — vector mapping we have will be used to represent each sentence. Let us take it in steps and examples to have a clear picture.

Paragraph: "The news mentioned here is fake. Audience do not encourage fake news. Fake news is false or misleading"

Step 1: Tokenize the data, remove stop words and perform stemming or lemmatization.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import re

paragraph = """The news mentioned here is fake. Audience do not
encourage fake news. Fake news is false or misleading"""


sentences = nltk.sent_tokenize(paragraph)
lemmatizer = WordNetLemmatizer()

corpus = []

for i in range(len(sentences)):
    sent = re.sub('[^a-zA-Z]', ' ', sentences[i])
    sent = sent.lower()
    sent= sent.split()
    sent = [lemmatizer.lemmatize(word) for word in sent if not word
in set(stopwords.words('english'))]
    sent = ' '.join(sent)
    corpus.append(sent)

print(corpus)
```

Output:

```
['news mentioned fake', 'audience encourage fake news', 'fake news
false misleading']
```

Step 2: List all unique words

Unique words: ['news', 'mentioned', 'fake', 'audience', 'encourage', 'false', 'misleading']

Step 3: Create a dictionary with mapping of words to a number. This should now be sorted on frequency of occurrence in descending order.

Vector (Number)	Words	Frequency
0	fake	3
1	news	3
2	audience	1
3	encourage	1
4	false	1
5	mentioned	1
6	misleading	1

Step 4: Now, create a table in each sentence, for the presence of each word in the dictionary, assign '1' else assign '0'

Sentence	0 (fake)	1 (news)	2 (audience)	3 (encourage)	4 (false)	5 (mentioned)	6 (misleading)
news mentioned fake	1	1	0	0	0	1	0
audience encourage fake news	1	1	1	1	0	0	0
fake news false misleading	1	1	0	0	1	0	1

The data we had is now transformed as:

- news mentioned fake — [1 1 0 0 0 1 0]
- audience encourage fake news — [1 1 1 1 0 0 0]
- fake news false misleading — [1 1 0 0 1 0 1]

And these vectors are sent as input to the model as independent features. We can use 'CountVectorizer' from sci-kit learn to perform steps 2, 3 and 4

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
independentFeatures = cv.fit_transform(corpus).toarray()
```

Output:

independentFeatures - NumPy object array

	0	1	2	3	4	5	6
0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	1
2	0	0	1	1	0	1	1

Bag of words will really be helpful in prediction problems like language modeling and documentation classification. Bag of words do have few shortcomings. Mentioning a few of them below:

1. Vocabulary: The vocabulary requires careful design to manage the size, which in-turn impacts the sparsity of the document representations.
2. Meaning: The values here are represented either as 1's or 0's. Consider the words 'news' and 'fake' — both the words are having same representation, and semantics of the words are same. This causes the difficulty to identify the importance of words.

To overcome these shortcomings, TF-IDF can be used.

Term Frequency — Inverse Document Frequency (TF-IDF)

We calculate the term frequency and Inverse document frequency for every word in the corpus, and multiplication of TF and IDF gives the document vectors. So, how do we calculate TF-IDF

Term Frequency (TF) — (No. of repeated words in sentence) / (No. of words in sentence)

Inverse Document Frequency (IDF) — log[(No. of sentences) / (No. of sentences containing word)]

Let us take the same example — "The news mentioned here is fake. Audience do not encourage fake news. Fake news is false or misleading"

Step 1: Passing the data through stemming or lemmatization. Take all the unique words, and sort based on frequency of occurrence. These are steps 1,2,3 we have observed in Bag of words (BOW)

# Vector (Number)	Aa Words	# Frequency
0	fake	3
1	news	3
2	audience	1
3	encourage	1
4	false	1
5	mentioned	1
6	misleading	1

Step 2: Calculate Term Frequency

Let us calculate TF for sentence — 1

- news — 1 / 3 = 0.33 {News is repeated once in the sentence, and total words are 3 — giving 1/3}
- mentioned — 1 / 3 = 0.33
- fake — 1 / 3 = 0.33
- audience , encourage, false, mentioned, misleading — 0 / 3 = 0 {These words did not occur in the sentence — there is no repetition, hence zero}

Let us calculate TF for sentence — 2

- audience — 1 / 4 = 0.25 (Audience word is repeated once in the sentence, and total words in the sentence are 4 — giving 1/4)
- encourage — 1 / 4 = 0.25
- fake — 1 / 4 = 0.25
- news — 1 / 4 = 0.25
- false, mentioned, misleading — 0 / 4 = 0

Similarly — we calculate Term Frequency for rest of sentences.

Aa Sentence	0 (fake)	1 (news)	2 (audie...	3 (enco...	4 (false)	5 (menti...	6 (misle...
news mentioned fake	0.33	0.33	0	0	0	0.33	0
audience encourage fake news	0.25	0.25	0.25	0.25	0	0	0

Step 3: Calculate IDF

Let us calculate IDF for all the words:

- news — log_e(3/3) = 0 {we have 3 sentences, and news word is present in all three sentences, hence log(3/3)}
- mentioned — log_e(3/1) = 1.0986
- fake — log_e(3/3) = 0
- audience — log_e(3/1) = 1.0986
- encourage — log_e(3/1) = 1.0986
- false — log_e(3/1) = 1.0986
- misleading — log_e(3/1) = 1.0986

Step 4: Calculate document vectors multiplying TF and IDF values. Steps 2, 3, 4 can be achieved through TF-IDF vectorizer from sci-kit learn.

```
# Creating the TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
independentFeatures_tfIDF = tfidf.fit_transform(corpus).toarray()
```

Output:

independentFeatures_tfIDF - NumPy object array

	0	1	2	3	4	5	6
0	0	0	0.453295	0	0.767495	0	0.453295
1	0.608845	0.608845	0.359594	0	0	0	0.359594
2	0	0	0.359594	0.608845	0	0.608845	0.359594

You can access the code snippet on GitHub, give it a try. Contrary to bag of words, the vectors here have different values, giving importance to a set of words. Though the models solves the issues observed on BOW, there are shortcomings even here, such as

- TF-IDF does not capture position in text, semantics, co-occurrences

- TF-IDF computes document similarity directly in the word-count space, making it slow for large documents

Bag of words or TF-IDF features can be used as inputs for Naive bayes model to classify spam and ham. The upcoming blogs will be on classification of Spam and Ham, and word2vec. Happy learning :)

Bag Of Words NLP Data Science Machine Learning Python

K

Follow

Written by Koushik kumar

14 Followers

Data-driven analyst and Machine learning professional with an ability to apply ML, DL and NLP techniques and leverage algorithms to solve real-world problems.

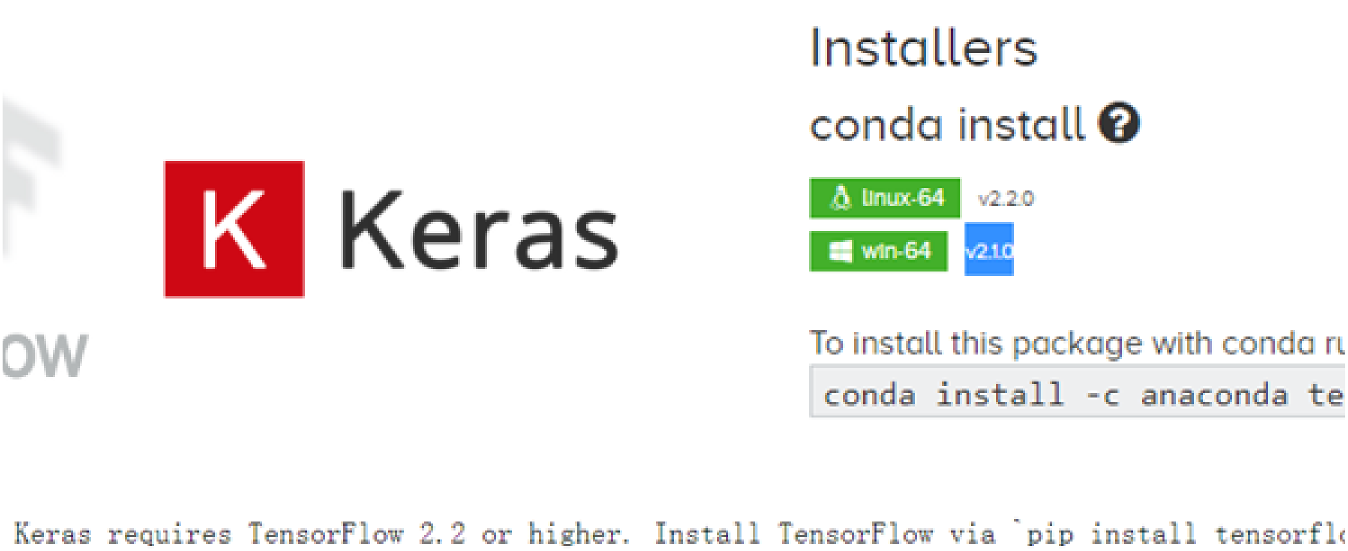
More from Koushik kumar

Koushik kumar

Transfer learning with VGG16 and VGG19, the simpler way!

Find VGG16 and VGG19 implementation template!

5 min read · Jan 20, 2021



Koushik kumar in Analytics Vidhya

TensorFlow 2.3.0 with GPU support on Windows 10

A few days back, I had to make a decision of keep aside the dual core I5 laptop I'd been using for the past 7 years and purchase a new...

7 min read · Oct 5, 2020

135 2

Koushik kumar

Speedup NumPy through the new TF NumPy API

TensorFlow team in the recent Google I/O 2021 event has announced NumPy API. Kemal El Moujahid, the product director for TensorFlow and...

4 min read · May 24, 2021

62

K Koushik kumar

Google I/O — 2021: Glimpses and event video links

Google took things to really next level through their product lineup and innovations for the year 2021!!! Google I/O Key note, Android12 ...

1 min read · May 22, 2021

1

See all from Koushik kumar

Recommended from Medium

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

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

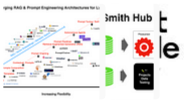
Lists

Predictive Modeling w/ Python

20 stories · 501 saves

Practical Guides to Machine Learning

10 stories · 568 saves



Natural Language Processing

721 stories · 319 saves

Coding & Development

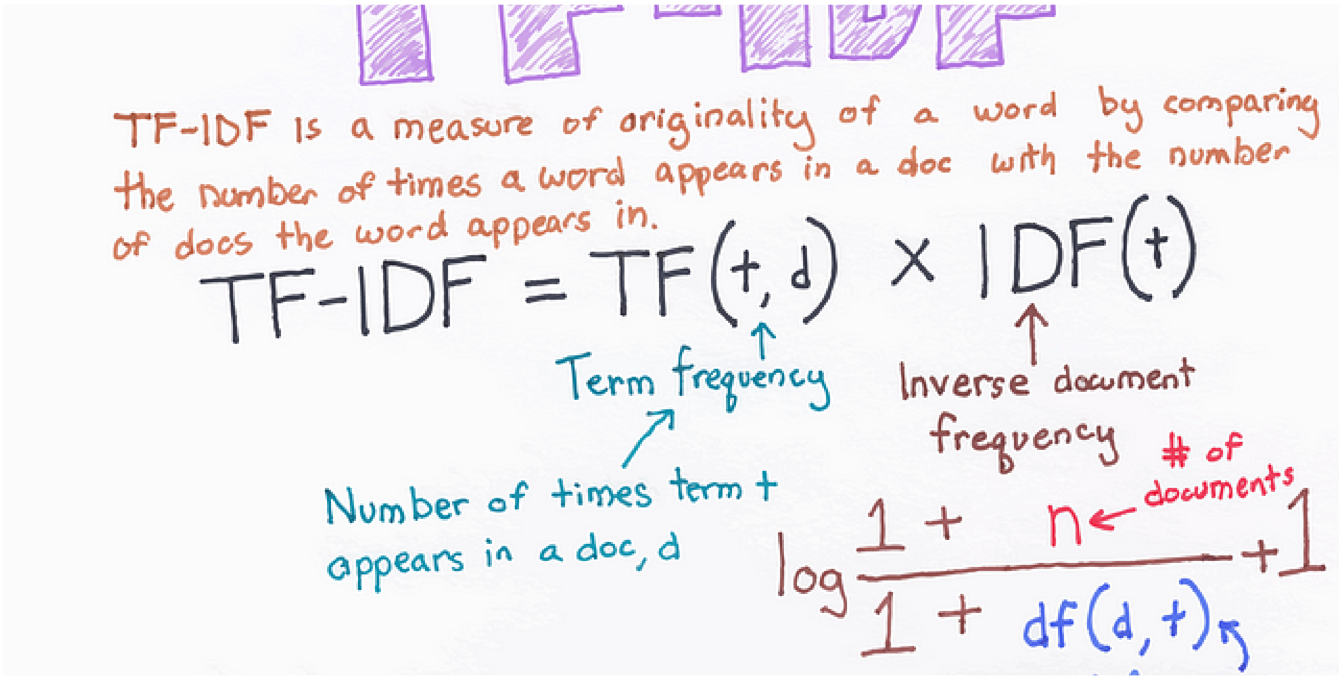
11 stories · 219 saves

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



Rahul S

NLP: TF-IDF (Term Frequency-Inverse Document Frequency)

Convert words into numbers

· 3 min read · Jun 8

5

Mayur Ghadge in Python in Plain English

Mastering Multilingualism: Top 5 Python Language Detection Techniques Explained

Accurate language detection is a fundamental aspect of many NLP applications. In this comprehensive guide, we'll delve into the top 5...

6 min read · Sep 1

1

Shivamshinde in Towards AI

Unlocking the Potential of Text: A Closer Look at Pre-Embedding Text Cleaning Methods

This article will discuss different cleaning techniques that are essential to obtain maximum performance from textual data.

5 min read · Jun 29