Text Representation

Text Representation

Often we want to be able to quantify what a document is about so we can perform analysis, such as comparing documents to determine which documents are similar. In order to do this, we have to convert the text to a numerical representation.

There are several ways this can be done, though we are going to focus on a commonly-used simple method called TF-IDF.

Bag of Words

Bag of Words (BoW) is a straightforward way to encode text (similar to one-hot encoding). It converts text into numerical vectors of 0s and 1s.

```
sentence_1 = 'Jen is a good student.'
sentence_2 = 'Jen is also a great guitarist.'
```

	good	great	guitarist	jen	student
0	1	0	0	1	1
1	0	1	1	1	0

Limitations of BoW

BoW is easy to implement and understand. However, there are some limitations:

1. It is called a “bag” because it does not account for any structure in the text.
2. It does not account for semantics of the word. For example, it cannot recognize that “buy used clothes” is very similar to “buy old outfits”.
3. It can produce a very large sparse matrix. Therefore, you may need to do some preprocessing to help reduce the number of words in your text (eliminate stop words, use stemming/lemmatization).

TF-IDF Transformation

TF-IDF is a way to measure how meaningful a word is to a document in a corpus of documents. The idea is if a word occurs multiple times in a document, it may be more meaningful than other words that appear fewer times (TF). However, if a word occurs many times in many other documents, it could be the word is just a frequent word and it is not particularly meaningful (IDF).

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Term frequency: number of times word occurs in document divided by total words in document.

Inverse Document Frequency: Log of total number of documents divided by number of documents the word occurs in.

TF-IDF Example

```
sentence_1 = 'Jen is a good student.'
sentence_2 = 'Jen is also a great guitarist.'
```

	good	great	guitarist	jen	student
0	1	0	0	1	1
1	0	1	1	1	0

Term frequency of “good” in [0] = 1/5

Inverse Document Frequency of “good” = log(2/1) = log(2)

Note: the result you get in Python won’t match the formula exactly because it normalizes and smooths the TF-IDF.