Feature Extraction From Text

Renato R. Maaliw III, *DIT*
College of Engineering
Southern Luzon State University
Lucban, Quezon, Philippines

- Most classic machine learning algorithms cannot take in raw text

- Instead we need to perform a feature "extraction" from the raw text in order to pass numerical features to the machine learning algorithm

- For example, we could count the occurrence of each word to map text to a number

- Counter Vectorization, Term-Frequency and Inverse Document Frequency

Count Vectorization

```
messages = ["Hey, lets go to the game today!",
            "Call your sister.",
            "Want to go walk your dogs?"]
```

Document Term Matrix (DTM)

```
messages = ["Hey, lets go to the game today!",
            "Call your sister.",
            "Want to go walk your dogs?"]
```

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0	0	1	1	1	1	0	1	1	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	0	1	0	1	1	1

- An alternative to CountVectorizer is a TfidfVectorizer. It also creates a document term matrix from our messages.

- However, instead of filling the DTM with token counts it calculates term frequency-inverse document frequency value for each word (TF-IDF)

- Term frequency **tf(t,d)**: is the raw count of a term in a document, i.e the number of times that term **t** occurs in a document **d**

- However, instead of filling the DTM with token counts it calculates term frequency-inverse document frequency value for each word (TF-IDF)

- However, Term Frequency alone isn't enough for a thorough feature analysis of the text.

- Let's imagine very common terms, like "a" or "the"

- Because the term **“the”** is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word **“the”** more frequently, without giving enough weight to the more meaningful terms **“red”** and **“dogs”**

- An inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

- It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

- TF-IDF = term frequency * (1 / document frequency)
- TF-IDF = term frequency * inverse document freq

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- Fortunately Scikit-learn can calculate all these terms for us through the use of its API

- Notice how similar the syntax is to our previous use of ML models in Scikit-learn

```
from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer()
dtm = vect.fit_transform(messages)
```

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0.000	0.00	0.403	0.307	0.403	0.403	0.000	0.403	0.307	0.403	0.00	0.00	0.000
0.623	0.00	0.000	0.000	0.000	0.000	0.623	0.000	0.000	0.000	0.00	0.00	0.474
0.000	0.46	0.000	0.349	0.000	0.000	0.000	0.000	0.349	0.000	0.46	0.46	0.349

- TF-IDF allows us to understand the context of words across an entire corpus of document, instead of just its relative importance in a single document

[Code Demo]

Thank you very much for listening.