NLP Module: Feature Engineering & Text Representation
NLP Process

**Text Processing**
Clean up the text to make it easier to use and more consistent to increase prediction accuracy later on.

**Feature Engineering & Text Representation**
Learn how to extract information from text and represent it numerically.

**Learning Models**
Use learning models to identify parts of speech, entities, sentiment, and other aspects of the text.
### Feature Engineering & Text Representation

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-Hot-Encoding</strong></td>
<td>Transforms categorical feature into many binary features</td>
</tr>
<tr>
<td><strong>Bag of Words Model</strong></td>
<td>Generalization of one-hot-encoding for a string of text</td>
</tr>
<tr>
<td><strong>N-Gram Encoding</strong></td>
<td>Captures word order in a vector model</td>
</tr>
<tr>
<td><strong>TFIDF</strong></td>
<td>Converts a collection of raw documents to a matrix of TFIDF features</td>
</tr>
</tbody>
</table>
What is Feature Engineering & What is Text Representation?
Feature engineering is the process of transforming the raw data to improve the accuracy of models by creating new features from existing data.
Text Representation is numerically representing text to make it mathematically computable
Encodes categorical data as real numbers such that the magnitude of each dimension is meaningful

For each distinct possible value, a new feature is created

Exp:

<table>
<thead>
<tr>
<th>name</th>
<th>kind</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldy</td>
<td>Fish</td>
<td>0.5</td>
</tr>
<tr>
<td>Scooby</td>
<td>Dog</td>
<td>7.0</td>
</tr>
<tr>
<td>Brian</td>
<td>Dog</td>
<td>3.0</td>
</tr>
<tr>
<td>Francine</td>
<td>Cat</td>
<td>10.0</td>
</tr>
<tr>
<td>Goldy</td>
<td>Dog</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>age</th>
<th>name_Brian</th>
<th>name_Francine</th>
<th>name_Goldy</th>
<th>name_Scooby</th>
<th>kind_Cat</th>
<th>kind_Dog</th>
<th>kind_Fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3.0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10.0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Berkeley SCET
from sklearn.preprocessing import OneHotEncoder

oh_enc = OneHotEncoder()

oh_enc.fit(df[['name', 'kind']])

oh_enc.transform(df[['name', 'kind']]).todense()

One Hot Encoding in Pandas

pd.get_dummies(df[['name', 'kind']])
Bag of Words Model
Bag of Words Model

Extracts features from text

Stop words not included, word order is lost, sparse encoding

Exp:

“Learning about machine learning is fun.”

![Diagram showing the Bag of Words Model](image)
Bag of Words Model using Scikit-Learn

```python
sample_text = ['This is the first document.', 'This document is the second document.', 'This is the third document. ']

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(stop_words="english")

vectorizer.fit(sample_text)

# to see what words were kept

print("Words:", list(enumerate(vectorizer.get_feature_names())))
```

Berkeley SCET
N-gram Encoding
N-gram Encoding

Extracts features from text while capturing local word order by defining counts over sliding windows

Exp n = 2:

The book was well written but I did not enjoy it.

book well
well written
written not
not enjoy
N-gram Encoding using Scikit-Learn

```python
sample_text = ['This is the first document.', 'This document is the second document.', 'This is the third document. ']

from sklearn.feature_extraction.text import CountVectorizer

bigram = CountVectorizer(ngram_range=(1, 2))

bigram.fit(sample_text)

#to see what words were kept

print("Words:", list(zip(range(0,len(bigram.get_feature_names())), bigram.get_feature_names())))
```
TFIDF Vectorizer
TFIDF Vectorizer

Converts a collection of raw documents to a matrix of TFIDF features

What are TFIDF Features?

TFIDF stands for term frequency inverse document frequency and it represents text data by indicating the importance of the word relative to the other words in the text

2 Parts:

TF: (# of times term t appears in a document)/ (total # of terms in the document)

IDF: (log10 (total # of documents)/(# of documents with term t in it))
TFIDF Vectorizer con.

$$\text{TFIDF} = \text{TF} \times \text{IDF}$$

TF represents how frequently the word shows up in the document.

IDF represents how important the word is to the document (rare words).
from sklearn.feature_extraction.text import TfidfVectorizer

sample_text = ['This is the first document.', 'This document is the second document.', 'This is the third document. ']

vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(sample_text)

print(vectorizer.get_feature_names())