The Problem with Text

A problem with modeling text is that it is messy, and techniques like machine learning algorithms prefer well dened xed-length inputs and outputs.

Machine learning algorithms cannot work with raw text directly; the text must be converted into numbers. Specically, vectors of numbers.

This is called feature extraction or feature encoding.

Bag of Words *A popular and simple method of feature extraction with text data is called the* ***bag- of-words*** *model of text.*

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. vocabulary of known words
2. measure of the presence of known words

The model is only concerned with whether known words occur in the document, not where in the document. Any information about the order or structure of words in the document is discarded.

Data

Below is a snippet of the rst few lines of text from the book "A Tale of Two Cities" by Charles Dickens, taken from Project Gutenberg.

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,

Vocabulary

We can make a list of all of the words in our model vocabulary. The unique words here (ignoring case and punctuation) are:

"it" "was" "the" "best" "of" "times" "worst" "age" "wisdom" "foolishness"

Document Vectors

The next step is to score the words in each document

Because we know the vocabulary has 10 words, we can use a xed-length document representation of 10, with one position in the vector to score each word.

The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.

Vector for Document 1 : *It was the best of times*
“it” = 1 “was” = 1 “the” = 1 “best” = 1 “of” = 1 “times” = 1 “worst” = 0 “age” = 0 “wisdom” = 0

“foolishness” = 0
[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

Samples in vectorized form

	it	was	the	best	of	times	worst	age	wisdom	foolishness
Sample 1	1	1	1	1	1	1	0	0	0	0
Sample 2	1	1	1	0	1	1	1	0	0	0
Sample 3	1	1	1	0	1	0	0	1	1	0
Sample 4	1	1	1	0	1	0	0	1	0	1

Features Data Instance Feature Values

Some additional simple scoring methods include:

Counts: Count the number of times each word appears in a document.

Frequencies: Calculate the frequency that each word appears in a document out of all the words in the document.

Managing Data

As the vocabulary size increases, so does the vector representation of documents.

In the previous example, the length of the document vector is equal to the number of known words.

You can imagine that for a very large corpus, such as thousands of

books, that the length of the vector might be thousands or millions of positions. Further, each document may contain very few of the known words in the vocabulary.

This results in a vector with lots of zero scores, called a sparse vector or sparse representation.

Sparse vectors require more memory and computational resources when modeling and the vast number of positions or dimensions can make the modeling process very challenging for traditional algorithms.

As such, there is pressure to decrease the size of the vocabulary when using a bag-of-words model.

There are simple text cleaning techniques that can be used as a rst step, such as:

Ignoring case
Ignoring punctuation
Ignoring frequent words that don't contain much information, called stop words, like "a," "of," etc.
Reducing words to their stem (e.g. "play" from "playing") using stemming algorithms.
Fixing misspelled words.

Handling the steps can be too much work Splitting the words in the corpus (dataset) Counting the frequency of words

Sklearn provides us with a function called **CountVectorizer**

CountVectorizer's ***t transform*** method will create a matrix, the rows in the matrix are data samples and the columns are the feature words, each cell in the matrix will denote the number of times a word appeared in that sample.

```
from sklearn.feature_extraction.text import CountVectorizer
matrix = CountVectorizer()
X = matrix.fit_transform(data).toarray()
```

	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	1	0	1

```
y = /*dataset labels*/
```

Feature Generation

lowercase = False
ngram_range(min_n, max_n)
(1, 1) means only unigrams , default value (1, 2) means unigrams and bigrams
(2, 2) means only bigrams

Curse of Dimensionality

```
print(vectorizer.get_feature_names())
```

Large number of features.

Limiting Features:

lowercase = True
stop_words CountVectorizer(max_features=1000) min_df & max_df

```
# Split train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```