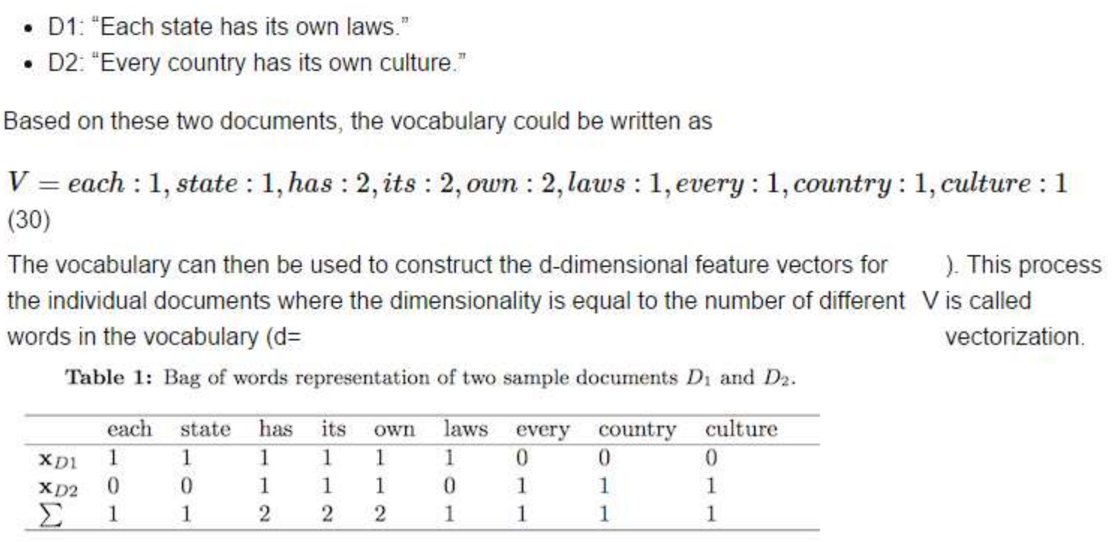
The Bag of Words Model

One of the most important sub-tasks in pattern classification are feature extraction and selection; the three main criteria of good features are listed below:

- Salient. The features are important and meaningful with respect to the problem domain.
- Invariant. Invariance is often described in context of image classification: The features are insusceptible to distortion, scaling, orientation, etc. A nice example is given by C. Yao et al. in Rotation-Invariant Features for Multi-Oriented Text Detection in Natural Images.
- Discriminatory. The selected features bear enough information to distinguish well between patterns when used to train the classifier.

Prior to fitting the model and using machine learning algorithms for training, we need to think about how to best represent a text document as a feature vector. A commonly used model in Natural Language Processing is the so-called bag of words model. The idea behind this model really is as simple as it sounds. First comes the creation of the vocabulary — the collection of all different words that occur in the training set and each word is associated with a count of how it occurs. This vocabulary can be understood as a set of non-redundant items where the order doesn’t matter. Let D١ and D٢ be two documents in a training dataset:



- D1: “Each state has its own laws.”
- D2: “Every country has its own culture.”

Based on these two documents, the vocabulary could be written as

$$V = each: 1, state: 1, has: 2, its: 2, own: 2, laws: 1, every: 1, country: 1, culture: 1$$

(30)

The vocabulary can then be used to construct the d-dimensional feature vectors for). This process
the individual documents where the dimensionality is equal to the number of different V is called
words in the vocabulary (d= vectorization.

Table 1: Bag of words representation of two sample documents D_1 and D_2.

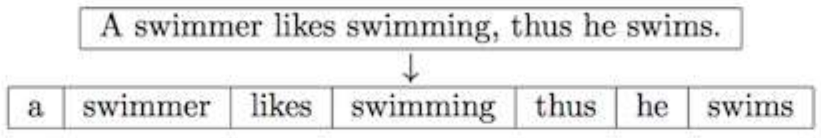
	each	state	has	its	own	laws	every	country	culture
$\mathbf{x}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{x}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Given the example in Table ١ one question is whether the ١s and ٠s of the feature vectors are binary counts (١ if the word occurs in a particular document, ٠ otherwise) or absolute counts (how often the word occurs in each document).

Tokenization

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. Usually, tokenization is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construction of n-grams. Below is an example of a simple but typical tokenization step that splits a sentence into individual words, removes punctuation, and converts all letters to lowercase.

Table 2: Example of tokenization.



A swimmer likes swimming, thus he swims.

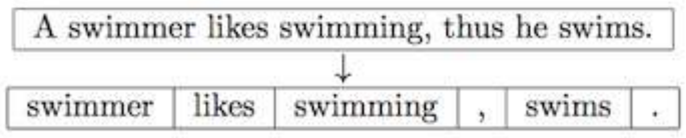
$\downarrow$

a	swimmer	likes	swimming	thus	he	swims

Stop Words

Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., words such as so, and, or, the, …”). One approach to stop word removal is to search against a language-specific stop word dictionary. An alternative approach is to create a stop list by sorting all words in the entire text corpus by frequency. The stop list — after conversion into a set of non-redundant words — is then used to remove all those words from the input documents that are ranked among the top n words in this stop list.

Table 3: Example of stop word removal.



A swimmer likes swimming, thus he swims.

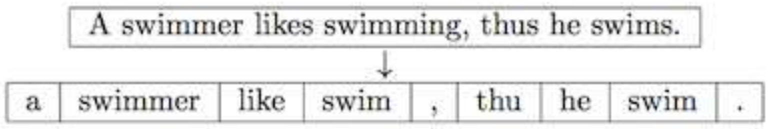
$\downarrow$

swimmer	likes	swimming	,	swims	.

Stemming and Lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed my Martin F. Porter in ١٩٧٩ and is hence known as Porter stemmer

Table 4: Example of Porter stemming.



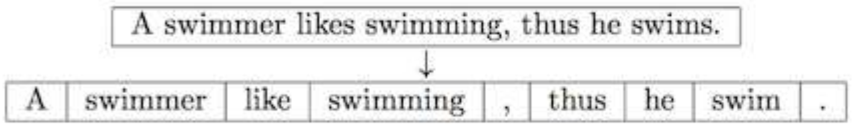
A swimmer likes swimming, thus he swims.

↓

a	swimmer	like	swim	,	thu	he	swim	.

Stemming can create non-real words, such as “thu” in the example above. In contrast to stemming, lemmatization aims to obtain the canonical (grammatically correct) forms of the words, the so-called lemmas. Lemmatization is computationally more difficult and expensive than stemming, and in practice, both stemming and lemmatization have little impact on the performance of text classification

Table 5: Example of lemmatization.



A swimmer likes swimming, thus he swims.

↓

A	swimmer	like	swimming	,	thus	he	swim	.