Feature Engineering for Texts

Part 1

Machine learning problems with texts

- Document classification
  - Goal: assign a category for each document.
  - Application: online advertisement placement.
- Spam filtering
  - Goal: identify which e-mails are spam.
  - Application: used by all e-mail providers.
- Sentiment analysis
  - Goal: identify an emotion of a document.
  - Application: social media monitoring.

Supervised Machine Learning

Given a set of n objects $Z=(z_1, \ldots, z_n)$ and their labels $(y_1, \ldots, y_n)$, $y_i \in Y$
the goal is to predict label y_i for object z_i based on a vector of features $\boldsymbol{x}_i \in X$

Find a function $f: X \rightarrow Y$

How to generate a p-dimensional vector of features $\boldsymbol{x}_i$ for object z_i?

Document: "This is an apple. An apple is a fruit, not a vegetable."

Step 1. **Conversion to lowercase**

"this is an apple. an apple is a fruit, not a vegetable."

Step 2. **Punctuation removal and tokenization.**

["this", "is", "an", "apple", "an", "apple", "is", "a", "fruit", "not", "a", "vegetable"]

Step 3. **Lemmatization or stemming** (optional).

Lemmatization: transform each word to its **dictionary form**.
Stemming: suffix-stripping

Example: sits -> sit, swimming -> swim, etc.

Important in *fusional* languages (Russian, French, Italian, German etc.)

The English language is not very fusional.

Not supported in Spark ML 2.1.0, try NLTK python package

Step 4. **Stop words removal**

Stop word - an extremely common word ("is", "a", "an", "not")
["this", "apple", "apple", "fruit", "vegetable"]

Step 5. **Dictionary creation.**

Dict – a set of distinct words in all documents D
Typical size of the dictionary ≈ 10,000 - 100,000

Dict = {"art" : 1, "this": 2, …, "apple" : 105, "book" : 106, ….}

Step 6. **Calculating term frequencies and vectorization**

["this", "apple", "apple", "fruit", "vegetable"]

Term frequencies:
this – 1
apple – 2
fruit – 1
vegetable - 1

[0, **1**, 0, …, 0, **2**, 0, …., **1**, 0, **1**, 0]

this apple fruit vegetable

Size of the vector equals to the size of the dictionary

t – term (word)
d – document
D – all documents

$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$

$TF(t, d)$ – term frequency - count of a term *t* in a document *d*
$DF(t, D)$ – document frequency - number of the documents where a term *t* appears

$$IDF(t, D) = log \frac{|D| + 1}{DF(t, D) + 1}$$

$|D|$ - number of documents

Step 7. **Calculating TF*IDF vector**

IDF – inverse document frequency

IDF:

this – 0.01
apple – 0.1
fruit – 0.05
vegetable – 0.05

Size of the vector equals to the size of the dictionary

Summary

Steps of feature engineering for text:

- Step 1 Conversion to lowercase,
- Step 2 Punctuation removal and tokenization,
- Step 3 Lemmatization or stemming (optional),
- Step 4 Stop words removal,
- Step 5 Dictionary creation,
- Step 6 Calculating term frequencies and vectorization,
- Step 7 Calculating TF*IDF vector.

N-grams

Document: "This is an apple. An apple is a fruit, not a vegetable."

N-gram – n consecutive words
1-gram – unigram (word)
2-gram – bigram
3-gram – trigram
...

N-grams

Document: "This is an apple. An apple is a fruit, not a vegetable."

1-gram (unigram)
["this", "is", "an", "apple", "an", "apple", "is", "a", "fruit", "not", "a", "vegetable"]

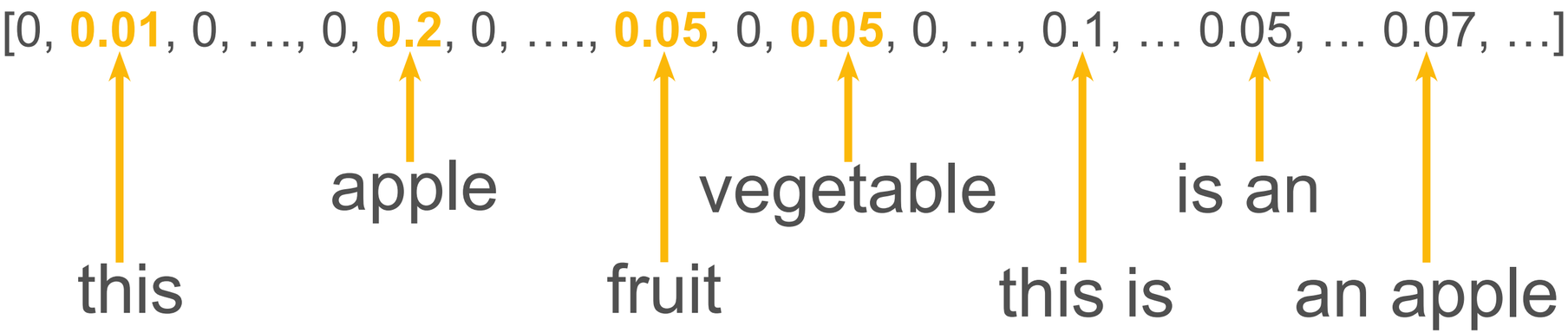
2-grams (bigrams):
["this is", "is an", "an apple", "an apple", "apple is", "is a", "a fruit", ...]

3-grams:
["this is an", "is an apple", "an apple is", "apple is a", ...]

Dict = {"art" : 1, "this": 2, …, "apple" : 105, "book" : 106, …., "this is" : 2340, "is an" : 2341, …, "apple is" : 56010, …}

Problem: a number of **bigrams** is much larger than a number of distinct **words**

Step 7. **Calculating TF*IDF vector with 2-grams (bigrams)**



Size of the vector equals to the size of the **dictionary**.
Dictionary is a set of all unique **words** and **bigrams**.

Summary

- N-gram is N consecutive words in a text
- While working with N-grams, all N-grams with a degree ≤N are generated
- Number of distinct N-grams with N > 1 is large