Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 4

Text Classification

Nicola Capuano and Antonio Greco

DIEM – University of Salerno

Outline

- Text Classification
- Topic Labelling
  Example
- Sentiment Analysis
  Exercise

Text Classification

Text Classification

The process of **assigning one or more classes to a text document** for various purposes:

- **Topic labeling**, **Intent detection**, **Sentiment analysis**, …

The classification relies **only on the text content**:

- **Other attributes** of the document (metadata) are **disregarded**
  - … different from **document classification**
- The classes are **predefined**
  - … different from **document clustering**

Definition

Given:

- a set of documents $\boldsymbol{D} = \{\boldsymbol{d_1}, \cdots, \boldsymbol{d_n}\}$
- a set of predefined classes $\boldsymbol{C} = \{\boldsymbol{c_1}, \cdots, \boldsymbol{c_m}\}$

Text classification finds a **classifier function**:

$$\boldsymbol{\Phi}: \mathbf{D} \times \boldsymbol{C} \rightarrow \{\boldsymbol{True}, \boldsymbol{False}\}$$

that assigns a Boolean value in $\{\boldsymbol{True}, \boldsymbol{False}\}$ to each pair $(\boldsymbol{d_i}, \boldsymbol{c_j}) \in \boldsymbol{D} \times \boldsymbol{C}$

Types of Classification

Single-label:

- Assigns each document in D to **only one class in C**

Binary:

- Like Single-label but **C has only two classes**
- Classification is a decision between a class and its **complement**

Multi-label:

- Assigns each document to a **variable number of classes in C**
- Can be reduced to a series of **binary decisions**

ML-Based Classification

- A **machine learning model** is trained on a set of **annotated text documents**
- Each document in the training set is associated with **one or more class labels**
- After training, the model can **predict the category** (or categories) **for a new document**
- The classifier may provide a **confidence measure**
- A **vector representation** of documents, such as TF-IDF, must be used

Topic Labelling

Example

Classifying Reuters News

The Reuters 21578 dataset is **multi-class** and **multi-label**

- **90** distinct **classes**
- **7,769 training** documents, **3,019 test** documents
- The **number of words per document** ranges from **93** to **1,263**
- **Skewness:**
  - Some classes have over **1,000** documents
  - Other classes have fewer than 5 documents
- Most documents are assigned either **one or two labels**, some documents are labeled with **up to 15 categories**

More statistics on https://martin-thoma.com/nlp-reuters/

Corpus Management

```python
import nltk
from nltk.corpus import reuters
nltk.download('reuters')

ids = reuters.fileids()
training_ids = [id for id in ids if id.startswith("training")]
test_ids = [id for id in ids if id.startswith("test")]
categories = reuters.categories()

print("{} training items:".format(len(training_ids)), training_ids)
print("{} test items:".format(len(test_ids)), test_ids)
print("{} categories:".format(len(categories)), categories)
print("\nCategories of '{}':".format(training_ids[0]), reuters.categories(training_ids[0]))
print("Categories of '{}':".format(test_ids[2]), reuters.categories(test_ids[2]))
print("Items within the category 'trade'", reuters.fileids('trade'))
```

```
[nltk_data] Downloading package reuters to /root/nltk_data...
7769 training items: ['training/1', 'training/10', 'training/100', 'training/1000', 'training/10
3019 test items: ['test/14826', 'test/14828', 'test/14829', 'test/14832', 'test/14833', 'test/14
90 categories: ['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconut', 'co

Categories of 'training/1': ['cocoa']
Categories of 'test/14829': ['crude', 'nat-gas']
Items within the category 'trade' ['test/14826', 'test/14832', 'test/14858', 'test/14862', 'test
```

Process

- Extract **training** and **test samples** and related **labels**
- Create the **TF-IDF matrices** for training and test set
- Transform label lists in a **binary matrix** for training and test set (hot encoding)

	"a"	"b"	"c"	"d"	"e"	labels
0	1	1	0	0	1	["a", "b", "e"]
1	0	0	1	0	1	["c", "e"]
2	1	0	0	0	0	["a"]
3	0	1	1	0	1	["b", "c", "e"]

- **Train a classifier**
  - We can use an **MLP**
  - A sample is the **TF-IDF vector** of a text with its **binary label**
- **Test the classifier**

Pre-Processing

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MultiLabelBinarizer

# Generate training and test sets
training_corpus = [reuters.raw(id) for id in training_ids]
training_labels = [reuters.categories(id) for id in training_ids]
test_corpus = [reuters.raw(id) for id in test_ids]
test_labels = [reuters.categories(id) for id in test_ids]

# Create TF-IDF matrices
vectorizer = TfidfVectorizer(min_df = 3) # a word must appear in at least 3 documents
training_vectors = vectorizer.fit_transform(training_corpus)
test_vectors = vectorizer.transform(test_corpus)

# Transform a list of label lists in binary matrix
mlb = MultiLabelBinarizer()
training_mlb = mlb.fit_transform(training_labels)
test_mlb = mlb.transform(test_labels)

len(vectorizer.vocabulary_)
```

```
11361
```

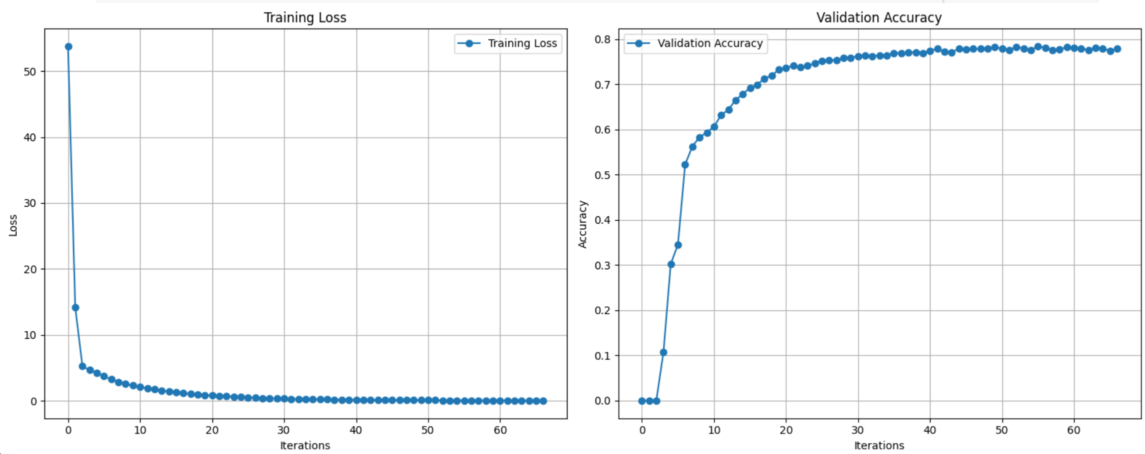
fit_transform combines two sequential steps, first applying the **fit** function and then the **transform** one

MLP Classifier Training

```python
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt

# Train an MLP classifier
classifier = MLPClassifier(hidden_layer_sizes = (128, 64), activation = 'relu', solver='adam',
                           max_iter = 100, early_stopping = True, verbose = True)
classifier.fit(training_vectors, training_mlb)

# Plot loss and validation curves
def plot (ax, data, title, xlabel, ylabel):
  ax.plot(data, label = title, marker='o')
  ax.set_xlabel(xlabel)
  ax.set_ylabel(ylabel)
  ax.set_title(title)
  ax.legend()
  ax.grid()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6)) # create subplots
plot(ax1, classifier.loss_curve_, 'Training Loss', 'Iterations', 'Loss')
plot(ax2, classifier.validation_scores_, 'Validation Accuracy', 'Iterations', 'Accuracy')
plt.tight_layout() # Adjust layout to prevent overlapping
plt.show()
```

MLP Classifier Training

```python
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt

# Train an MLP classifier
classifier = MLPClassifier(hidden_layer_sizes = (128, 64), activation = 'relu', solver='adam',
                           max_iter = 100, early_stopping = True, verbose = True)
classifier.fit(training_vectors, training_mlb)
```

Testing Metrics

- **Micro Average**: the average metric across all classes by considering the total number of true positives, false negatives, and false positives
- **Macro Average**: calculates the metric independently for each class and then takes the average (unweighted mean) of these values
- **Weighted Average**: computes the average of the metric, weighted by the support (the number of true instances) of each class
- **Samples Average**: computes the average of the metrics for each sample (instance) rather than for each class
  - Used in **multi-label classification** problems where each instance can belong to multiple classes

Testing Results

```
from sklearn.metrics import classification_report

# Predict the categories of the test set
predictions = classifier.predict(test_vectors)

# Print classification report
print(classification_report(test_mlb, predictions, target_names = mlb.classes_, zero_division = 0))
```

```
              precision    recall  f1-score   support

         acq       0.98      0.91      0.94       719
        alum       1.00      0.30      0.47        23
      barley       0.89      0.57      0.70        14
         bop       1.00      0.43      0.60        30
     carcass       0.67      0.22      0.33        18
  castor-oil       0.00      0.00      0.00         1
       cocoa       1.00      0.56      0.71        18
     coconut       0.00      0.00      0.00         2
 coconut-oil       0.00      0.00      0.00         3

          ...

   micro avg       0.93      0.72      0.81      3744
   macro avg       0.58      0.27      0.35      3744
weighted avg       0.90      0.72      0.78      3744
 samples avg       0.80      0.79      0.79      3744
```

Sentiment Analysis
Exercise

Sentiment Analysis

The process of **identifying and categorizing opinions** expressed in a piece of text

Applications:

- **Business:** Analyzing customer feedback and product reviews to understand customer satisfaction and brand perception
- **Finance:** Predicting market trends based on investor sentiment extracted from news articles and social media
- **Politics:** Analyzing public opinion during elections or policy changes

Can be seen as a **text classification** problem:

- Given a text, classifying it as **positive**, **negative**, or **neutral**.

IMDB Dataset

A set of **50,000** highly **polarized reviews** from the **Internet Movie Database**

- The set consists of:
  - **50% negative reviews**
  - **50% positive reviews**
- Download **CSV** from **Kaggle**

  https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

kaggle

review	sentiment
49582 unique values	**2** unique values
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive

Exercise

Build a **classifier for movies review**

- Given a review, the classifier will determine its **polarity (positive or negative)**
- Train the classifier on the IMDB Dataset
- Use **80% for training** and **20% for testing**
- Show and plot **metrics** and the **confusion matrix**

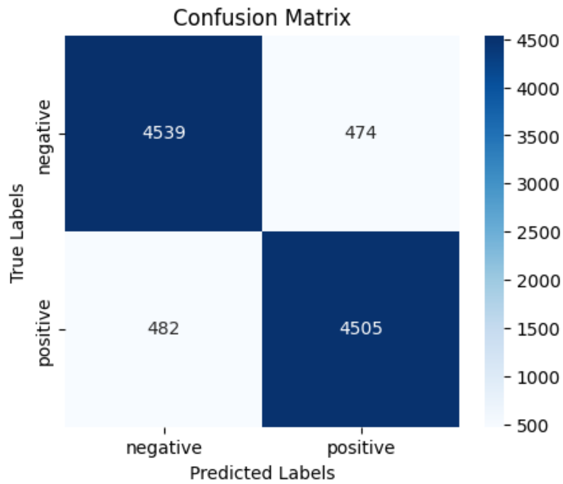
Suggestions

- **One-hot encode the labels**: **(1,0)** = negative, **(0, 1)** = positive using the **ScikitLearn** OneHotEncoder class
- To reduce the **TF-IDF matrix** consider only words that appear at least in **5 documents**

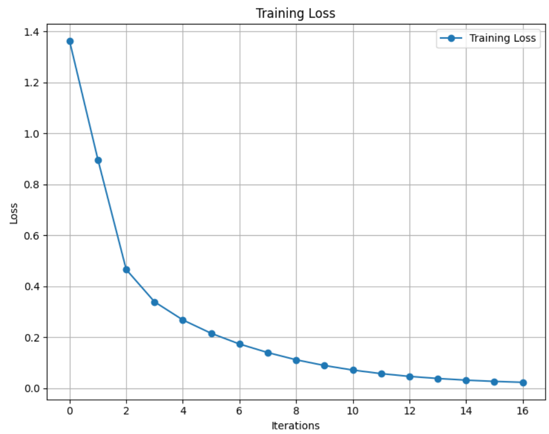
Exercise

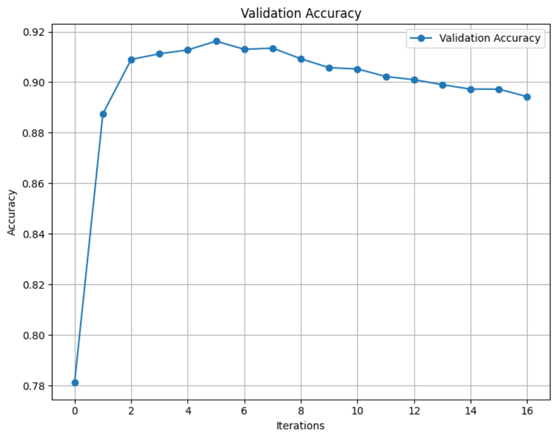
Suggestions (continue)

- Use the **ScikitLearn** confusion_matrix function to build the confusion matrix
- You can use the **Seaborn** heatmap to plot the confusion matrix
  - **pip install seaborn**
  - https://seaborn.pydata.org/generated/seaborn.heatmap.html



Exercise





	precision	recall	f1-score	support
negative	0.90	0.91	0.90	5013
positive	0.90	0.90	0.90	4987
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Text Classification Applications

- Topic Labelling
- Sentiment Analysis
- Spam Filtering
- Intent Detection
- Language Detection
- Content Moderation
- Products Categorization
- Author Attribution
- Content Recommendation
- Ad Click Prediction
- Job matching
- Legal case classification

Further Readings

- **Pandas Docs**
  https://pandas.pydata.org/docs/user_guide/
- **Scikit-Learn Docs**
  https://scikit-learn.org/stable/user_guide.html
- **Seaborn Docs**
  https://seaborn.pydata.org/api.html

Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 4

Text Classification

Nicola Capuano and Antonio Greco

DIEM – University of Salerno