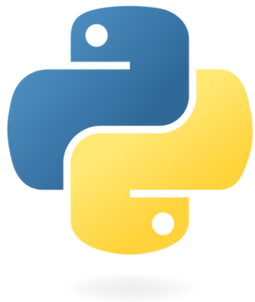
UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

AEI FACULTÉ D'ADMINISTRATION ET ÉCHANGES INTERNATIONAUX

Big Data Analytics

Chapter 3 : Applications for Comment Classification

Python Artificial Intelligence projects for beginners



python™

Anis Hellaf
Fourat Dridi
Sonia Hamadache
Ewelina Hejmo
Roqya Abdul
Xavier Jean-Louis

1. **Introduction**

The importance of monitoring and analyzing comments made by users or customers:

In this chapter, we will explain differents techniques :

- bag of words
- doc2vec
- word2vec

Demonstration of :

- spam detector
- sentiment analysis
- sentiment new

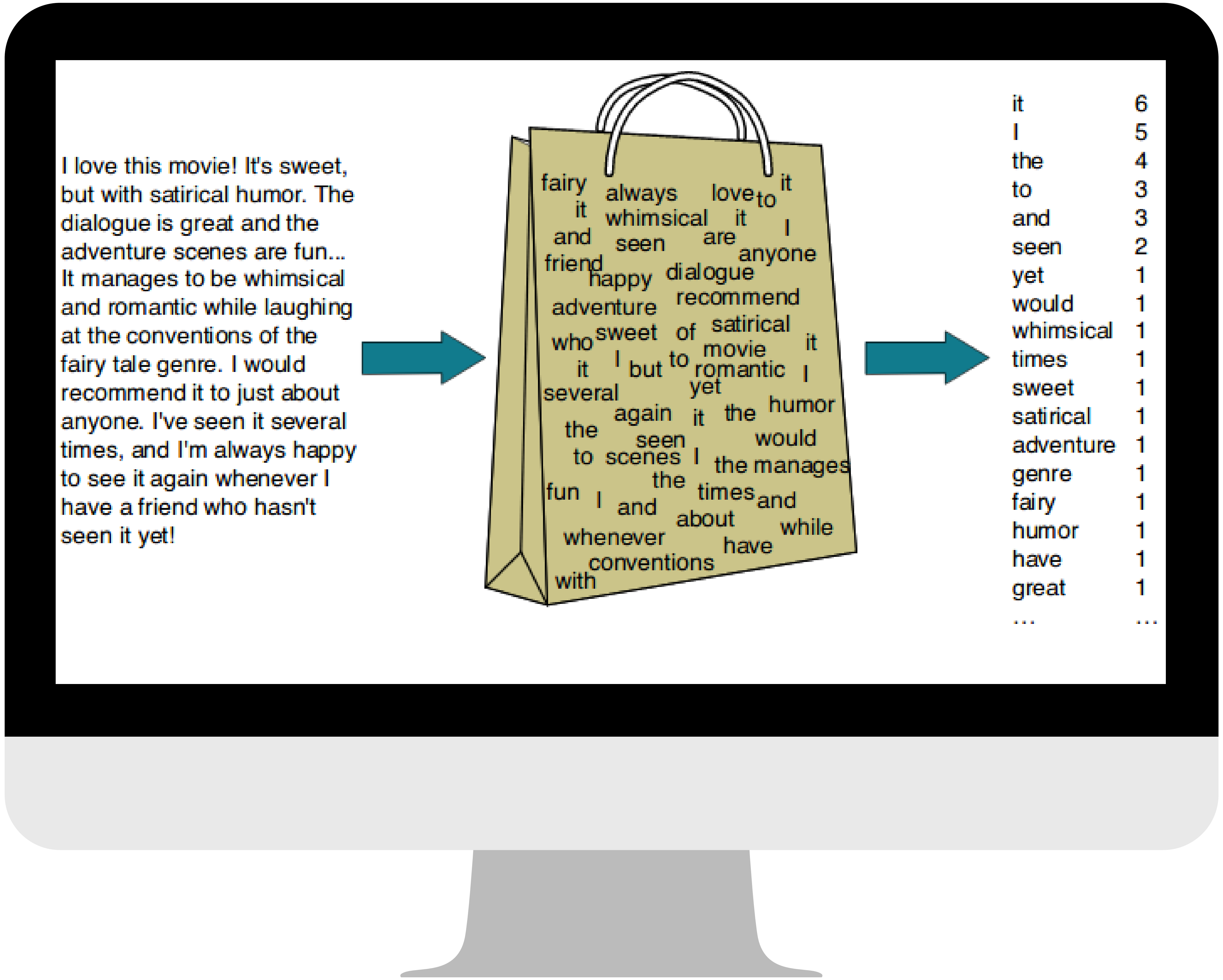
2. Use case presentation & problem definition

- The importance of artificial intelligence and code to facilitate the analysis of the desired data.
- What makes text classification an interesting problem?
- the importance of the analysis and the different methods.

3. Data Analytics approach definition and explanation

- ***Bag of words*** is a simplifying representation used in natural language processing and information retrieval.
- ***Word2Vec***: used for learning vector representations of words from large amounts of textual data. It's based on distributionality hypothesis.
- ***Doc2vec*** is a modified version of Word2vec. While Word2vec focuses on the representation of words as vectors, Doc2vec goes much further by allowing the representation of sentences, paragraphs or documents as vectors.

Bag of words


I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
fairy always love to it
it whimsical it I
and seen are anyone
friend happy dialogue
adventure recommend
who sweet of satirical
it I but to movie romantic I it
several yet
the again it the humor
seen would
to scenes I the manages
fun I and the times and
about while
whenever have
conventions
with
it 6
I 5
the 4
to 3
and 3
seen 2
yet 1
would 1
whimsical 1
times 1
sweet 1
satirical 1
adventure 1
genre 1
fairy 1
humor 1
have 1
great 1
... ...

Bag of words

- The bag-of-words model has also been used for computer vision. It is commonly used in document classification methods in which the (frequency) occurrence of each word is used as a feature for training a classifier.
- This model can be used in different situations such as text classification, sentiment analysis or information retrieval.

- Example one –

and	back	channel	i	my	plz	subscribe	to	xx
1	1	1	1	1	1	2	1	1

- Example two –

channel	grow	guys	help	me	my	please	subscribe	to
1	1	2	1	1	1	2	1	1

	and	back	channel	grow	guys	help	i	me	my	please	plz	subscribe	to	xx
Example one	1	1	1	0	0	0	1	0	1	0	1	2	1	1
Example two	0	0	1	1	2	1	0	1	1	1	0	1	1	0

Bag of words

- TF - IDF : Term Frequency Inverse Document Frequency

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Bag of words

Processing :

- Normalization
- Lemmatization
- Lowercase every word
- Drop punctuation
- Drop very common words (stop words)
- Remove plurals (for example, bunnies => bunny)
- Perform lemmatization (for example, reader => read, reading = read) Use n-grams, such as bigrams (two-word pairs) or trigrams
- Keep only frequent words (for example, must appear in >10 examples) Keep only the most frequent M words (for example, keep only 1,000) Record binary counts (1 = present, 0 = absent) rather than true counts

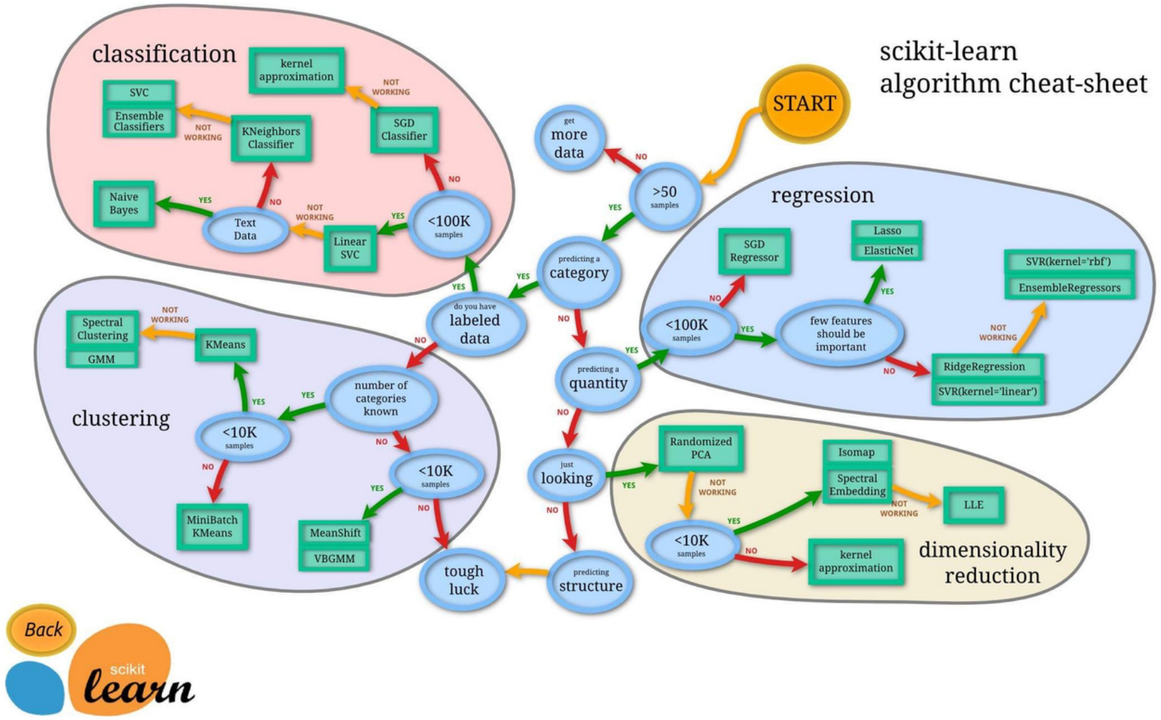
```
def lemmatize(sent):
    tokens = word_tokenize(sent.lower())
    tokens = [lemma.lemmatize(lemma.lemmatize(lemma.lemmatize(w, 'v'),'n'),'a') for w in tokens]
    return ' '.join(tokens)

df['CONTENT'] = df.CONTENT.apply(lambda sent: lemmatize(sent))
```

There are many other combinations for best practice, and finding the best that suit

Bag of words

Processing method that allows you to represent text as a bag of words. In Python, a Bag of words model can be created using the scikit-learn library.



Word2Vec

- ***Word2Vec*** learns from a corpus of texts how words are used in context and generates numerical vectors for each word

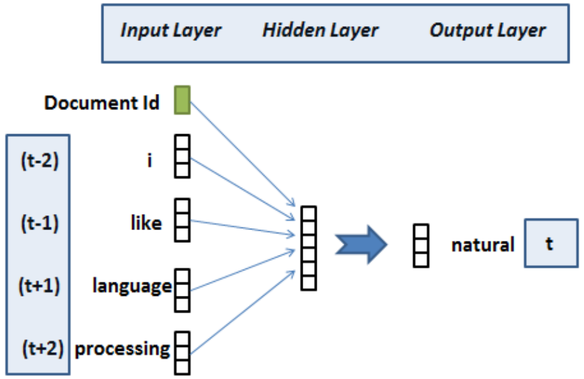
- ***Word2Vec***, we can measure the similarity between words

Doc2Vec

An extension of Word2vec that enables to classify documentents !

2 variants :

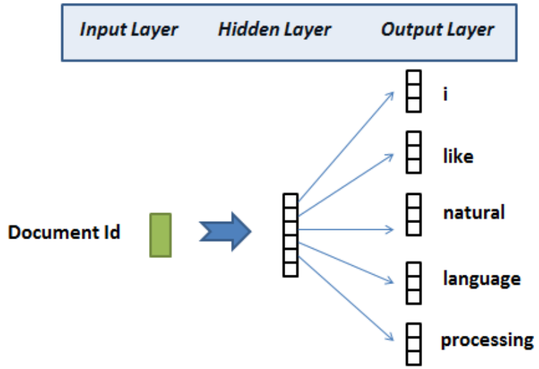
(PV-DM)



Distributed Memory Model

A model which attempts to guess the output (target word) from its neighboring words (context words) with the addition of a paragraph ID.

(DBOW)



Distributed Bag of Words Model

A model which guesses the context words from a target word.

Doc2Vec

An extension of Word2vec that enables to classify documentents !

When should we use this model ?

- Find the most relevant results between several documents (information search)
- Performing document classification
- Analyze negative or positive opinions between several documents

4. Solution development and illustration

We will demonstrate classification by using Python:

- ***Spam Detector***
- ***Sentiment Analysis***
- ***Sentiment Analysis New***

Spam Detector

Entrée [1]:
```
import pandas as pd
d = pd.read_csv("C:/Users/dridi/Downloads/Youtube01-Psy.csv")
```

Entrée [2]:
```
d.tail(10)
```

Out[2]:

The second line uses the Pandas function "read_csv()" to read the CSV file "Youtube01-Psy.csv" located in the directory "C:/Users/dridi/Downloads/". The data is stored in an object called "d", which is a Pandas DataFrame."

Entrée [21]:
```
# load all datasets and combine them
d = pd.concat([pd.read_csv("C:/Users/dridi/Downloads/Youtube01-Psy.csv"),
               pd.read_csv("C:/Users/dridi/Downloads/Youtube02-KatyPerry.c
               pd.read_csv("C:/Users/dridi/Downloads/Youtube03-LMFAO.csv")
               pd.read_csv("C:/Users/dridi/Downloads/Youtube04-Eminem.csv"
               pd.read_csv("C:/Users/dridi/Downloads/Youtube05-Shakira.csv
```

Entrée [22]:
```
len(d)
```

The code loads several CSV files containing data on YouTube comments for different videos and combines them into a single DataFrame using the Pandas "concat()" function. The files are specified with their full path in the arguments of the "read_csv()" method. The data of the resulting DataFrame is stored in an object called "d".

Sentiment analysis

```
Entrée [4]: with open("C:/Users/dridi/Downloads/yelp_labelled.txt") as f:
                for item_no, line in enumerate(f):
                    print(item_no, line)
```

```
0 Wow... Loved this place.      1

1 Crust is not good.    0
```

This code opens a text file "yelp_labelled.txt" located in the directory "C:/Users/dridi/Downloads/" using the "open()" function in read mode ("with open()" is used to automatically handle file closing after use).
Then, it uses a "for" loop to iterate through each line of the file. The variable "item_no" contains the number of the line being processed, while the variable "line" contains the content of the line itself.
At each iteration of the loop, the code displays the item number and the content of the line using the "print()" function.

```
Entrée [5]: sentences = []
            sentiments = []
            with open ("C:/Users/dridi/Downloads/yelp_labelled.txt") as f:
                for item_no, line in enumerate(f):
                    line_split = line.strip().split('\t')
                    sentences.append((line_split[0], "yelp_%d" % item_no))
                    sentiments.append(int(line_split[1]))
```

The code creates two empty lists, "sentences" and "sentiments". It opens the text file "yelp_labelled.txt" located in the directory "C:/Users/dridi/Downloads/" and uses a "for" loop to iterate through each line of the file. For each line, it splits the elements using the tab ("\t") character and adds the first element to the "sentences" list and the second element to the "sentiments" list as an integer.

Entrée [8]:

```
from gensim.models.doc2vec import TaggedDocument
import re

sentences = []
sentiments = []
for fname in ["yelp", "amazon_cells", "imdb"]:
    with open (f"C:/Users/dridi/Downloads/{fname}_labelled.txt") as f:
        for item_no, line in enumerate(f):
            line_split = line.strip().split('\t')
            sent = line_split[0].lower()
            sent = re.sub(r'\'', '', sent)
            sent = re.sub(r'\W', ' ', sent)
            sent = re.sub(r'\s+', ' ', sent).strip()
            sentences.append(TaggedDocument(words=sent.split(), tags=["%s_%d" % (fname, item_no)]))
            sentiments.append(int(line_split[1]))
```

This code prepares a dataset for sentiment analysis using the doc2vec model from the gensim library. It reads three text files (yelp_labelled.txt, amazon_cells_labelled.txt, and imdb_labelled.txt) from a specific directory, and preprocesses the textual data by cleaning it up using regular expressions. The cleaned text is then stored as TaggedDocument objects in the sentences list, along with a unique identifier based on the file name and line number. Additionally, the code extracts the sentiment value for each line and stores it as an integer in the sentiments list. Overall, this code is used to create a dataset of labeled sentences and their corresponding sentiment values for training a doc2vec model in gensim

SentimentAnalysisnew

Entrée [11]:
```
# unsupervised training data
import re
import os
unsup_sentences = []

# source: http://ai.stanford.edu/~amaas/data/sentiment/, data from IMDB
for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
    for fname in sorted(os.listdir(f"C:/Users/dridi/Downloads/aclImdb")):
        if fname[-4:] == '.txt':
            with open(f"C:/Users/dridi/Downloads/aclImdb" + dirname + "/" + fname, encoding='UTF-8') as f:
                sent = f.read()
                words = extract_words(sent)
                unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

# source: http://www.cs.cornell.edu/people/pabo/movie-review-data/
for dirname in ["review_polarity/txt_sentoken/pos", "review_polarity/txt_sentoken/neg"]:
    for fname in sorted(os.listdir(f"C:/Users/dridi/Downloads/review_polarity")):
        if fname[-4:] == '.txt':
            with open(f"C:/Users/dridi/Downloads/review_polarity" + dirname + "/" + fname, encoding='UTF-8') as f:
                for i, sent in enumerate(f):
                    words = extract_words(sent)
                    unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))

# source: https://nlp.stanford.edu/sentiment/, data from Rotten Tomatoes
with open (f"C:/Users/dridi/Downloads/stanfordSentimentTreebank/original_rt_snippets.txt" , encoding='UTF-8') as f:
    for i, line in enumerate(f):
        words = extract_words(line)
        unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Entrée [12]:
```
len(unsup_sentences)
```

Out[12]:
```
175325
```

Entrée [13]:
```
unsup_sentences[0:10]
```

Out[13]:
```
[TaggedDocument(words=['bromwell', 'high', 'is', 'a', 'cartoon', 'comedy', 'it', 'ran', 'at', 'the', 'same', 'time', 'as', 's
```

The code collects unsupervised data from various text files stored on the disk downloaded from aclImdb /review_polarit and stanfordSentimentTreebank. The data is collected using loops that traverse the different directories where the textual data is stored. For each text file, the content is read and functions are applied to extract the words from the text.

The obtained sentences are stored as TaggedDocument objects. The total number of collected unsupervised sentences is 175,325.

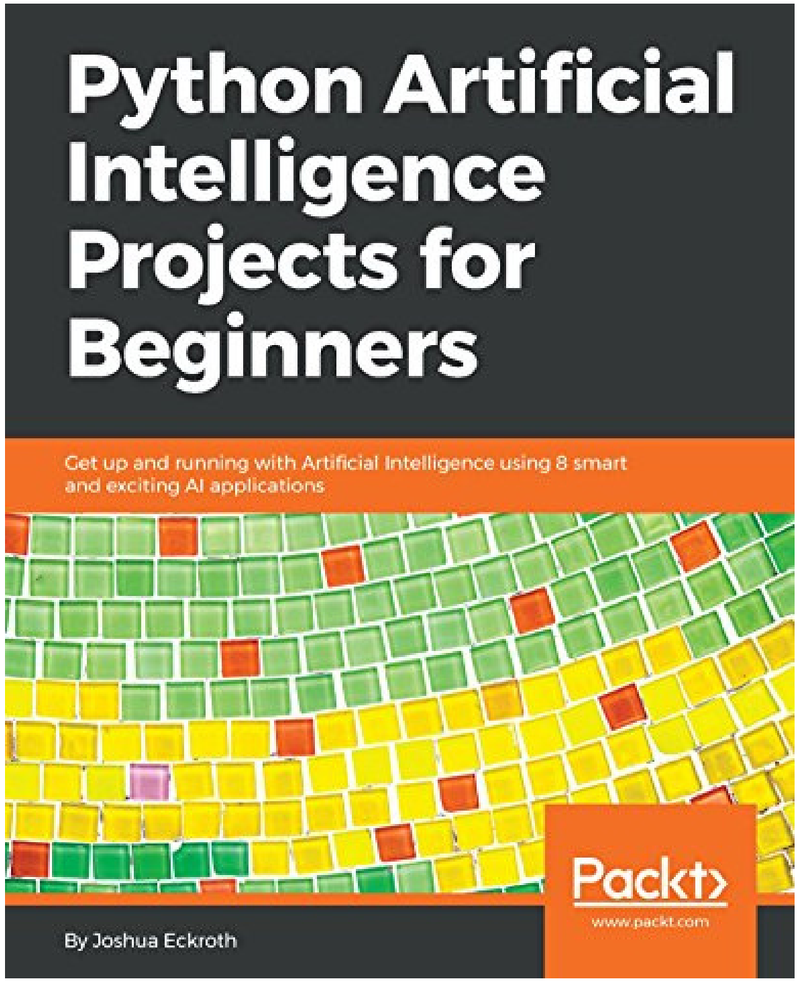
These data can be used to train natural language processing models such as word vector models or unsupervised language models.

5. Evaluation and feedback

- The exercise was not really simple, due to our lack of knowledge in terms of Jupyter and Python
- The most difficult part was to use Jupyter and to code
- At the opposate it was interesting thanks to the new tools and models of classification of text that we learnt during this exercise and also during all the courses

6. Conclusion

7. Appendix: software and datasets



python™

jupyter

kaggle

Packt

7ZIP


Thank
You