# Mining Unstructured Data

Language Detection

General Structure

Detailed Structure

Core task

Deliverables

# Outline

1 Language Detection

2 General Structure

3 Detailed Structure

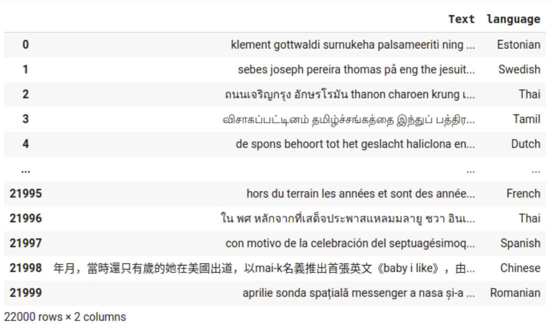
4 Core task

5 Deliverables

# Session 1 - Language Detection

## Assignment

Study the impact of different preprocessing techniques on NLP task. To do so, we will perform **Language Detection** over the provided csv file *data.csv* and will detect in which language each sentence is written.



| | Text | language |
|---|---|---|
| **0** | klement gottwaldi surnukeha palsameeriti ning ... | Estonian |
| **1** | sebes joseph pereira thomas på eng the jesuit... | Swedish |
| **2** | ถนนเจริญกรุง อักษรโรมัน thanon charoen krung เ... | Thai |
| **3** | விசாகப்பட்டினம் தமிழ்ச்சங்கத்தை இந்துப் பத்திர... | Tamil |
| **4** | de spons behoort tot het geslacht haliclona en... | Dutch |
| **...** | ... | ... |
| **21995** | hors du terrain les années et sont des année... | French |
| **21996** | ใน พศ หลักจากที่เสด็จประพาสแหลมมลายู ชวา อินเ... | Thai |
| **21997** | con motivo de la celebración del septuagésimoq... | Spanish |
| **21998** | 年月，當時還只有歲的她在美國出道，以mai-k名義推出首張英文《baby i like》，由... | Chinese |
| **21999** | aprilie sonda spațială messenger a nasa și-a ... | Romanian |

22000 rows × 2 columns

# Outline

1. Language Detection
2. General Structure
3. Detailed Structure
4. Core task
5. Deliverables

# General Structure - Main function

The program (langdetect.py) expects three arguments: Path to the input data, vocabulary size and analyzer granularity (words or chars)

```
def get_parser():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--input",
                         help="Input data in csv format", type=str)
    parser.add_argument("-v", "--voc_size",
                         help="Vocabulary size", type=int)
    parser.add_argument("-a", "--analyzer",
                          help="Tokenization level: {word, char}",
                         type=str, choices=[’word’,’char’])
    return parser
```

# General Structure - Main function II

Then, it reads all files in the given directory, and splits the data in train and test splits

```
raw = pd.read_csv(args.input)

# Languages
languages = set(raw[’language’])
print(’========’)
print(’Languages’, languages)
print(’========’)

# Split Train and Test sets
X=raw[’Text’]
y=raw[’language’]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
  random_state=seed)

print(’========’)
print(’Split sizes:’)
print(’Train:’, len(X_train))
print(’Test:’, len(X_test))
print(’========’)
```

# General Structure - Main function III

Then, it preprocesses the data, computes its features and the coverage of the vocabulary over the test data

```
# Preprocess text (Word granularity only)
    if args.analyzer == 'word':
        X_train, y_train = preprocess(X_train,y_train)
        X_test, y_test = preprocess(X_test,y_test)

    #Compute text features
    features, X_train_raw, X_test_raw = compute_features(X_train, X_test,
     analyzer=args.analyzer,max_features=args.voc_size)

    print('========')
    print('Number of tokens in the vocabulary:', len(features))
    print('Coverage: ', compute_coverage(features, X_test.values, analyzer=
     args.analyzer))
    print('========')
```

# General Structure - Main function IV

Finally, it trains a classifier model, predicts over the test set, reports its performance and plots its PCA dimensionality reduction

```
#Apply Classifier
X_train, X_test = normalizeData(X_train_raw, X_test_raw)
y_predict = applyNaiveBayes(X_train, y_train, X_test)

print(’========’)
print(’Prediction Results:’)
plot_F_Scores(y_test, y_predict)
print(’========’)

plot_Confusion_Matrix(y_test, y_predict, "Greens")


#Plot PCA
print(’========’)
print(’PCA and Explained Variance:’)
plotPCA(X_train, X_test,y_test, languages)
print(’========’)
```

# Outline

1 Language Detection

2 General Structure

3 Detailed Structure

4 Core task

5 Deliverables

# Functions - Tokenize text

This function is currently empty, you can apply all preprocessing steps. Resources you may use: NLTK, Spacy

```
#Tokenizer function. You can add here different
    preprocesses.
def preprocess(sentence, labels):
    '''
    Task: Given a sentence apply all the required
    preprocessing steps
    to compute train our classifier, such as sentence
    splitting,
    tokenization or lemmatization.

    Input: Sentence in string format
    Output: Preprocessed sentence either as a list or a
    string
    '''
    # Place your code here
    # Keep in mind that sentence splitting affectes the
    number of sentences
    # and therefore, you should replicate labels to match
    .
    return sentence,labels
```

# Functions - Classifier models

```
# You may add more classifier methods replicating this
    function
def applyNaiveBayes(X_train, y_train, X_test):
    '''
    Task: Given some features train a Naive Bayes
    classifier
          and return its predictions over a test set
    Input; X_train -> Train features
           y_train -> Train_labels
           X_test -> Test features
    Output: y_predict -> Predictions over the test set
    '''
    trainArray = toNumpyArray(X_train)
    testArray = toNumpyArray(X_test)

    clf = MultinomialNB()
    clf.fit(trainArray, y_train)
    y_predict = clf.predict(testArray)
    return y_predict
```

# Outline

1 Language Detection

2 General Structure

3 Detailed Structure

4 Core task

5 Deliverables

# Language Detection - First baseline

Without modifying the code run the following configurations and compare their vocabulary coverage and performance. Explain why they show different error patterns.

- **Character level:**

  ```
  python langdetect.py -i dataset.csv -v 1000 -a char
  ```

- **Word level:**

  ```
  python langdetect.py -i dataset.csv -v 1000 -a word
  ```

# 1st Exercise - First Baseline

Focus on the following elements to explain the behavior:

- How well does the vocabulary cover the data?
- Which languages produce more errors? What do they have in common (family, script, etc)?
- How languages overlap on the PCA plot? What could that overlapping mean?

# 2nd Exercise - Document Structure

- Try different vocabulary sizes and preprocessing steps to analyze the behavior of this kind of data.
- Improving F1 score is **NOT** the objective of the task. Focus on understanding how different parameters affect the results.

Language Detection
General Structure
Detailed Structure
Core task
Deliverables

# Outline

1. Language Detection
2. General Structure
3. Detailed Structure
4. Core task
5. Deliverables

# Deliverables

Write a report describing the work carried out in this exercise.
The report must be a **single self-contained PDF document**, under 10 pages, containing:

- *Introduction:* What is this report about. What is the goal of the presented work.
- *Preprocess:* Describe the preprocessing steps tried and the rationale to employ them.
- *Code:* Include your preprocessing functions as well as classifiers in the document **Do not include any other code.**
- *Experiments and results:* Results obtained on the **test** datasets, for different rule combinations you deem relevant.
  **Keep result tables in the format produced by the program. You can just donwnload them and use them in your document.**
- *Conclusions:* Final remarks and insights gained in this task.