In [1]:
```
import os
import numpy as np
import pandas as pd
from tqdm import tqdm
```

Work with data

Read data

Emodji and labels look like:

Read train data

In [2]:
```
with open('./data/train/spanish_train.text', 'r') as f:
    texts = [l.strip() for l in f]
```

In [3]:
```
with open('./data/train/spanish_train.labels', 'r') as f:
    labels = [int(l.strip()) for l in f]
```

In [4]:
```
# with open('./data/data_us/tweet_by_ID_20_11_2017__03_39_34.txt.ids', 'r')
#     ids = [l.strip() for l in f]
```

Read trial data (only texts)

In [5]:
```
with open('./data/test/spanish_test.text', 'r') as f:
    texts_trial = [l.strip() for l in f]
```

Read mapping for emodji and labels

In [6]:
```
with open('./data/mapping/english_mapping.txt', 'r') as f:
    maps = [l.strip().split() for l in f]
```

In [7]:
```
emodji = [maps[l][1] for l in labels]
```

Look at the data

In [8]:
```
texts[:5]
```

Out[8]:
```
['Plaza de Oriente , Madrid .......#madrid #city #plazadeoriente #puertad
esol #tour…',
 'Por ser la columna de mi templo, por ser lo mejor que tengo. @ Parquesu
r',
 'Me gustan las motos! #cheste2016 #nicoabad #elañoquevienemás @user',
 'Sevilla tiene un color especial, Sevilla tiene un color diferente. #Sev
illa #SemanaSanta…',
 'Que (la) Chipi no se caiga .Cuánto os quiero chavales!! @ Valdemoro Cen
tro']
```

In [9]:
```
labels[:5]
```

Out[9]:
```
[9, 0, 2, 16, 1]
```

In [10]:
```
emodji[:5]
```

Out[10]:
```
['😘 ', '❤', '😂 ', '😁 ', '😍 ']
```

Preprocessing data

We clean our data from:

- URLS
- Punctiation
- Symbols '#' and '@'
- Stop-words

We also transform it into lowercase and use stemming.

In [11]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from string import punctuation
from nltk.stem.snowball import EnglishStemmer

import re

import preprocessor as p
from preprocessor.api import clean, tokenize, parse
```

In [12]:
```
# import nltk
# nltk.download('stopwords')
```

In [13]:
```
# p.set_options(p.OPT.URL)
```

In [14]:
```
translator = str.maketrans("", "", punctuation)
stemmer = EnglishStemmer()
```

In [15]:
```
def preproc_eng(texts):
    clear_texts = []
    count = 0
    for text in texts:
        # TODO: hack
        text = re.sub('\s[\@]\s', '', text)

        text = ' '.join([word for word in text.split() if word not in (stop
        # delete punctuation
        text = word_tokenize(text.translate(translator))

        # stemming
        text = [stemmer.stem(w) for w in text]
        # preprocessing as tweet
        text = clean(' '.join(text))
        clear_texts.append(text)

        # Increment
        count += 1
        if count % 5000 == 0:
            print(str(count) + "/" + str(len(texts)))
    return clear_texts
```

In [16]:
```
texts_clear = preproc_eng(texts)
texts_trial_clear = preproc_eng(texts_trial)
```

```
5000/19000
10000/19000
15000/19000
```

In [17]:
```
texts_clear[:5]
```

Out[17]:
```
['plaza de orient madrid madrid citi plazadeorient puertadesol tour…',
 'por ser la columna de mi templo por ser lo mejor que tengoparquesur',
 'me gustan las moto cheste2016 nicoabad elañoquevienemá user',
 'sevilla tien un color especi sevilla tien un color diferent sevilla sem
anasanta…',
 'que la chipi se caiga cuánto os quiero chavalesvaldemoro centro']
```

Build model

Baseline 1

Firstly, build the simplest model with TF_IDF as feautures and LogitRegression Classifier

Best score: 45.256

In [18]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
```

In [19]:
```
tf = TfidfVectorizer()
```

Split our data to train and to validation, get scores

In [20]:
```
def get_scores_valid(X, y, C=1.0, ratio=0.9, seed=14):
    '''
    X, y – выборка
    ratio – в каком отношении поделить выборку
    C, seed – коэф-т регуляризации и random_state
              логистической регрессии
    '''

    idx_split = int(ratio * len(X))
    X_train = X[:idx_split]
    X_valid = X[idx_split:]
    y_train = y[:idx_split]
    y_valid = y[idx_split:]

    X_train_tf = tf.fit_transform(X_train)
    X_valid_tf = tf.transform(X_valid)

    logit = LogisticRegression(C=C, n_jobs=-1, random_state=seed) # removed

    logit.fit(X_train_tf, y_train)

    valid_pred = logit.predict(X_valid_tf)

    valid_pred.dtype = np.int
    np.savetxt('res.txt', valid_pred, fmt='%d')
    np.savetxt('goldres.txt', np.array(y_valid), fmt='%d')
```

Select parameters

In [21]:
```
Cs = np.logspace(-3, 1, 10)
scores =[]
count = 0
for C in Cs:
    print("C value: ", C)
    get_scores_valid(texts_clear, labels, C=C)
    %run ./tools/evaluationscript/scorer_semeval18.py goldres.txt res.txt
    count += 1
    print("{}/{}\n".format(count, len(Cs)))
```

```
Recall: 22.158
1/10

C value:  0.0027825594022071257
Macro F-Score (official): 1.909
-----
Micro F-Score: 22.158
Precision: 22.158
Recall: 22.158
2/10

C value:  0.007742636826811269
Macro F-Score (official): 2.211
-----
Micro F-Score: 22.579
Precision: 22.579
Recall: 22.579
3/10

C value:  0.021544346900318832
```

In [22]:
```
C_best = 10.0
```

Check best model on trial data

In [23]:
```
logit = LogisticRegression(n_jobs=-1, random_state=14, C=C_best)
```

In [24]:
```
X_train_tf = tf.fit_transform(texts_clear)
X_test_tf = tf.transform(texts_trial_clear)
```

In [25]:
```
logit.fit(X_train_tf, labels)
```

Out[25]:
```
LogisticRegression(C=10.0, class_weight=None, dual=False, fit_intercept=T
rue,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=-1, penalty='l2', random_st
ate=14,
                   solver='lbfgs', tol=0.0001, verbose=0, warm_start=Fals
e)
```

In [26]:
```
res = logit.predict(X_test_tf)
```

In [27]:
```
res.dtype = np.int
```

In [28]:
```
np.savetxt('res.txt', res, fmt='%d')
```

In [29]:
```
%run ./tools/evaluationscript/scorer_semeval18.py ./data/test/spanish_test.
```

```
Macro F-Score (official): 12.384
-----
Micro F-Score: 27.1
Precision: 27.1
Recall: 27.1
```

In [ ]:
```

```