Progetto di INTELLIGENZA ARTIFICIALE

SPACY SENTIMENT ANALYSIS

1. Riferimenti esterni

- *https://spacy.io/usage*;
- *https://scikit-learn.org/*.

2. Descrizione

Lo scopo del programma è quello di creare un modello per l'analisi e la classificazione del testo utilizzando spacy e sklearn.

Il programma prende in input da file delle recensioni del Castel dell'Ovo, che possono essere negative o positive, utilizza spacy per lemmizzarle e ricavarne token, termine usato nella terminologia spacy per indicare le unità atomiche della semantica del periodo; dopodiché, il programma genererà un modello logit utilizzando sklearn.

3. Documentazione

Innanzitutto, il programma importerà le librerie necessarie per creare un tokenizer italiano,

```
import spacy
from spacy.lang.it import Italian

nlp = spacy.load('it')
```

la punteggiatura,

```
import string
punctuations = string.punctuation
```

e gli stopwords:

```
from stop_words import get_stop_words
stop_words = get_stop_words('it')
```

Dopodiché, il programma carica da file una lista di recensioni del Castel dell'Ovo nel formato:

Reviews (text)	Feedback (bool)

```
df_castello = pd.read_csv("reviews.csv", sep="\t")
```

Su queste basi la funzione spacy_tokenizer, partendo da sentence, definisce un tokenizer personalizzato che genera una lista di tokens priva di stopwords mytokens:

```
def spacy_tokenizer(sentence):

    mytokens = parser(sentence)

    mytokens = [ word.lemma_.lower().strip() if word.lemma_ !
    = "-  PRON-" else word.lower_ for word in mytokens ]

   mytokens  =  [  word  for  word  in  mytokens  if  word  not  in
    stop_words and word not in punctuations ]

  return mytokens
```

Le righe seguenti generano un tokenizer italiano ed inizializzano una bag of words in cui ogni parola, filtrata dal tokenizer precedentemente creato, rappresenta un token:

```
    parser = Italian()

    bow_vector = CountVectorizer(tokenizer = spacy_tokenizer,
    ngram_range=(1,1))
```

Utilizziamo sklearn per costruire una pipeline e su di essa un modello logit che utilizza come classificatore Logistic Regression:

```
    from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression()

    pipe = Pipeline([("cleaner", predictors()),
                     ('vectorizer', bow_vector),
```

```
                ('classifier', classifier)])

    pipe.fit(X_train,y_train)
```

dove predictors è così definita:

```
    class predictors(TransformerMixin):

        def transform(self, X, **transform_params):
        return [clean_text(text) for text in X]

        def fit(self, X, y=None, **fit_params):
            return self

        def get_params(self, deep=True):
            return {}

    def clean_text(text):
        return text.strip().lower()
```

Per conoscere come si comporta il nostro modello, dividiamo a metà il dataset in training set e test set:

```
    from sklearn.model_selection import train_test_split

    X = df_castello['reviews']
    ylabels = df_castello['feedback']
```

```
X_train, X_test, y_train, y_test = train_test_split(X,
ylabels, test_size=0.3)
```

Utilizzando il modulo metrics di sklearn, l'accuratezza del modello sarà data da tre valori:

```
from sklearn import metrics
predicted = pipe.predict(X_test)

metrics.accuracy_score(y_test, predicted)
metrics.precision_score(y_test, predicted)
metrics.recall_score(y_test, predicted)
```

Dove per accuratezza si intende la percentuale di predizioni corrette, precisione è la percentuale di veri positivi sul totale delle predizioni positive e recall è il rapporto tra il numero di veri positivi e la somma tra il numero di veri positivi e falsi negativi, ossia non è altro che la capacità del classificatore di trovare tutti i campioni positivi.

Yuri Spaziani

N46003377