UNIVERSITÀ
DI PARMA

DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

LAB: Learning from sequences- RNN

Gianfranco Lombardo, Ph.D
gianfranco.lombardo@unipr.it

Ex. 1: Sentiment analysis without Neural networks

- Classify the review in “corpus.csv” (Sentiment Analysis)
  - The structure is class#SEP#document
    - class: Category to predict, can be positive or negative
    - document: Content of reviews
    - sep =”#!#”

- Read the csv with pandas and create a vector y and a matrix with documents X

- Divide in train and test
  - X_train, X_test, y_train, y_test = train_test_split(X, y)
- Encode categorical y_train and y_test as in the code:

```
# label encode the target variable Y
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.fit_transform(y_test)
```

- Follow the example for BoW
- Design a classification system choosing an algorithm among Logistic regression, Decision tree, Adaboost, Gradient boosting or XGBoost
- Compute the accuracy

Ex. 1: Sentiment analysis without Neural networks

- Visualize (matplotlib) how the accuracy change by changing the max_features (step = 1000)
    - CountVectorizer(max_features=1000) … CountVectorizer(max_features=10000)

- Find out which is the best ngram_range to consider:
    - CountVectorizer(ngram_range=(1,1)) … CountVectorizer(ngram_range=(1,3))

- Try The tf-Idf Vectorizer
    - TfidfVectorizer()

Ex. 2 - Sentiment Analysis with LSTM

- Repeat the exercise of sentiment analysis starting from corpus.csv

- Design your own LSTM selecting all the parameters (try embedding dimension equal to 100 as a starting point)

- Plot the loss or the accuracy with “history object” to understand if you are overfitting or not

Ex. 2 - Sentiment Analysis with LSTM

- Classify your own sentences !!

```
# replace with the data you want to classify

newtexts = ["i love you", "I Hate you", "yes, this movie is amazing" ,"New
Zealand is beautiful"]

# note that we shouldn't call "fit" on the tokenizer again
sequences = tokenizer.texts to sequences(newtexts)
data = pad sequences(sequences, maxlen=150)
# get predictions for each of your new texts
predictions = model.predict_classes(data)

print(predictions)
print(encoder.inverse_transform(predictions))
```