École national Supérieure d'Informatique
Master (2023-2024)
NLP: Natural Language Processing

Lab 01: Sentiment analysis

Abdelkrime Aries

This is a guided lab. Its purpose of workshops is to understand how to use some high level API's to solve a problem based on some demos about these API's. The problem we want to solve in this lab is a well known one: text-based sentiment analysis. Our aim is to understand how to manipulate different representations of words/sentences.

General information

Data

Financial sentiment analysis:

- URL: `https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis`
- Description: Some texts about finance with their sentiment orientation (positive, negative, neutral). It contains 860 negative samples, 1852 positive ones, and 3130 neutral ones.

Tools

NLTK, Scikit-learn, numpy, pandas, tensorflow, Gensim

Demos

- NLTK: `https://github.com/projeduc/ESI_TALN/blob/master/tuto/CH06/encoding_python_NLTK.ipynb`
- Gensim: `https://github.com/projeduc/ESI_TALN/blob/master/tuto/CH06/encoding_python_gensim.ipynb`
- Scikit-learn: `https://github.com/projeduc/ESI_TALN/blob/master/tuto/CH06/encoding_python_sklearn.ipynb`
- Tensorflow Auto-encoder: `https://github.com/projeduc/ESI_ML/blob/main/demos/NN/TF_Autoencoder.ipynb` (can be used for MLP)
- Tensorflow CNN: `https://github.com/projeduc/ESI_ML/blob/main/demos/NN/TF_CNN.ipynb` (CNN for images are similar to those for text)

1 Data preparation

1.1 Data lecture

- Import the financial sentiment analysis dataset.
- Split this last dataset into train and test using **train_test_split**; the test size is 30% using random_state=0 to get a stable split over executions.
- Train a **LabelBinarizer** on training labels. Then, transform both training labels and test labels into OneHot representation.

1.2 Words' encoding

- Using the previous function **tokenizer** which splits a sentence into words, create a list of list of words. This one will be used to train the next representations.
- Train a **Word2Vec** model of **Gensim** using a window of **3** and minimum count of **1**. The vector must have a size of **5**.

- Train a **Fasttext** model of **Gensim** using the same parameters. Fasttext is a character-based word embedding.

1.3 Supporting functions

- Define a function **tokenstem**
  - It takes a sentence and tokenize it using **tokenizer** (defined in the starter project).
  - We test if each token is a stop word; if it belong to **nltk.corpus.stopwords.words('english')**.
  - If so, we do not add it to the output list.
  - Otherwise, we stem it using **nltk.stem.porter.PorterStemmer** before adding it to the output list.
- Define a function **sentvectEncode(wv, sentWords, mx=20)**
  - It takes a word-vector (wv) of an embedding model (Word2Vec or Fasttext), a list of lists of words, and a maximum size. Then it generates sentences representations (3 types of representations): centroids, concats and matrix.
  - for each word of the sentence, we try to encode it using (wv). If the word exists in the vocabulary, we will have a code. Otherwise, the encoding function will raise an error **KeyError**. In this case, we will consider the code as a list of zeros (5 to be exact).
  - Encode each sentence using the centroid of its words (in this case **size = 5**).
  - Encode each sentence using the consecutive codes of its words. The maximum sentence size is **mx** so their concatenation will have **size = 20 * 5 = 100**. If the sentence is longer, it must be truncated. If it is less, you must add some padding (a vector of zeros).
  - Encode each sentence as a matrix of the codes of its words. The maximum sentence size is **mx** so the matrix will have **size = 20X5**. If the sentence is longer, it must be truncated. If it is less, you must add some padding (a vector of zeros).
  - the function returns a tuple of encodings: centroids, concats and matrix.

1.4 Sentence representation

- **TF**: train a **CountVectorizer** using **tokenstem** as **tokenizer** and limiting the vector's size **max_features = 3000**.
- Use it to generate TF representations of the training dataset and test dataset.
- Use **sentvectEncode** to generate 6 different encodings for the training dataset and the test one: word2vec-centroids, word2vec-concats, word2vec-matrix, fasttext-centroids, fasttext-concats and fasttext-matrix.

2 Training

2.1 F-based/MultinomialNB

- Train a **Multinomial Naive Bayes** model on the TF representation.

2.2 Vector-based/MLP

- Train 4 **MLP** models based on word2vec-centroids, word2vec-concats, fasttext-centroids and fasttext-concats.

2.3 Matrix-based/CNN

- Train 2 **CNN** models based on word2vec-centroids and fasttext-centroids.

3 Testing

- Predict all classes of test dataset based on all the trained models.
- Calculate the prediction time using **timeit**
- For each one, show the classification report.