THỰC HÀNH 1: PHÂN LỚP VĂN BẢN SỬ DỤNG PRE-TRAINED WORD EMBEDDING

Bộ dữ liệu: **UIT-VSFC**

Công bố khoa học: K. V. Nguyen, V. D. Nguyen, P. X. V. Nguyen, T. T. H. Truong and N. L. Nguyen, "***UIT-VSFC: Vietnamese Students' Feedback Corpus for Sentiment Analysis***", KSE 2018.

Số lượng dữ liệu: khoảng hơn 16,000 câu.

Số lượng nhãn: 3 nhãn gồm tích cực (*positive*), tiêu cực (*negative*) và trung tính (*neural*).

Sử dụng cho tác vụ **sentiment-based** và **topic-based**.

Đọc dữ liệu:

```
import pandas as pd

X_train = pd.read_csv(UIT-VSFC/train/sents.txt',
sep='\n', header=None, index_col=None)
y_train = pd.read_csv('UIT-VSFC/train/sentiments.txt',
sep='\n', header=None, index_col=None)

X_dev = pd.read_csv('UIT-VSFC/dev/sents.txt',
sep='\n', header=None, index_col=None)
y_dev = pd.read_csv('UIT-VSFC/dev/sentiments.txt',
sep='\n', header=None, index_col=None)

X_test = pd.read_csv('UIT-VSFC/test/sents.txt',
sep='\n', header=None, index_col=None)
y_test = pd.read_csv('UIT-VSFC/test/sentiments.txt',
sep='\n', header=None, index_col=None)

y_train = y_train.values.flatten()
y_dev = y_dev.values.flatten()
y_test = y_test.values.flatten()
```

1. Tiền xử lý dữ liệu

Các bước thực hiện:

+ Bước 1: Xây dựng tập từ vựng.

+ Bước 2: Xây dựng từ điển từ và index tương ứng (word_to_index).

+ Bước 3: Chuyển dữ liệu văn bản ban đầu thành dạng vector theo index trong từ điển (encode). Để độ dài các vector là như nhau, ta sử dụng kỹ thuật padding (được hỗ trợ sẵn trong thư viện keras).

+ Bước 4: Xây dựng từ điển index ánh xạ vào từ (index_to_word). Từ điển này sẽ dùng để chuyển index ban đầu lại thành văn bản (decode).

Để thực hiện 3 bước tiền xử lý dữ liệu trên, trong bài thực hành này giới thiệu 2 cách xử lý khác nhau với thư viện keras như sau:

Cách 1: Tự xây dựng bộ từ vựng và từ điển tương ứng

Xây dựng tập từ vựng:

```
# make vocabulary
from pyvi import ViTokenizer

V=[]
for t in X_train:
    tokenized_sentence = ViTokenizer.tokenize(t)
    V = V + tokenized_sentence.split()

V = list(set(V))
```

Xây dựng word_to_index và index_to_word

```
# Building dictionary
word_to_index = {w : (i+2) for i, w in enumerate(V)}

word_to_index['UNK'] = 1
word_to_index['PAD'] = 0

# Build index2w
index_to_word = {i: w for w, i in word_to_index.items()}
```

Tiến hành encode dữ liệu văn bản:

```
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from pyvi import ViTokenizer

max_len = 100
```

```
def encoding(X):
    sentences = []

    for t in X:
        tokenized_sentence = ViTokenizer.tokenize(t)
        sentences.append(tokenized_sentence)

    X=[]
    for s in sentences:
        sent = []
        for w in s.split():
            try:
                w = w.lower()
                sent.append(word_to_index [w])
            except:
                sent.append(word_to_index ["UNK"])
        X.append(sent)

    # Padding du lieu theo do dai cau (maxlen)
X = pad_sequences(maxlen = max_len, sequences = X,
padding = "post", value = word_to_index ["PAD"])
    return X
```

Cách 2: Dùng thư viện có sẵn trong keras. Sử dụng thư viện **Tokenizer**.

```
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from pyvi import ViTokenizer
from keras.preprocessing.text import Tokenizer

max_len = 100

word_tokenizer = Tokenizer(oov_token=-1)
word_tokenizer.fit_on_texts(X_train)

word_to_index = word_tokenizer.word_index
word_to_index['pad'] = 0
word_to_index['unk'] = -1

index_to_word = {i: w for w, i in word_to_index.items()}
```

```
def encoding(X):
    sentences = []

    for t in X:
        tokenized_sentence = ViTokenizer.tokenize(t)
        sentences.append(tokenized_sentence)

    X = embedded_sentences =
word_tokenizer.texts_to_sequences(sentences)
X = pad_sequences(maxlen = max_len, sequences = X,
padding = "post", value = word_to_index ["PAD"])

    return X
```

Tiến hành encode dữ liệu:

```
X_train_encoded = encoding(X_train)
X_dev_encoded = encoding(X_dev)
X_test_encoded = encoding(X_test)
```

2. Huấn luyện mô hình

Sử dụng 1 lớp Embedding trong mạng neural kết hợp với 1 lớp đầu ra.

```
from keras.layers import Dense, Embedding, Flatten
from keras.models import Model, Input
from keras.initializers import Constant

num_words = len(word_to_index)
input = Input(shape = (max_len, ))
emb = Embedding(input_dim=num_words,
                    output_dim=300,
                    input_length=max_len)(input)
flat = Flatten()(emb)
output = Dense(3, activation="sigmoid")(flat)

model = Model(input, output)
model.compile(optimizer="adam",
loss='binary_crossentropy', metrics=['binary_accuracy'])

model.summary()
```

Huấn luyện mô hình

```
from tensorflow.keras.utils import to_categorical

model.fit(X_train_encoded,
          to_categorical(y_train, num_classes=3),
          validation_data=(X_dev_encoded,
          to_categorical(y_dev, num_classes=3)),
           batch_size=128, epochs=10
     )
```

Dự đoán và kiểm thử mô hình: Các bạn tự thực hiện nhé. Đối với bài toán phân tích cảm xúc, các bạn nên dùng độ đo macro F1 score để đánh giá mô hình.

3. Sử dụng pre-trained embedding

Pre-trained embedding là một bộ word embedding được huấn luyện sẵn trước đó trên dữ liệu rất nhiều và có kích thước lớn. Mục tiêu của các bộ pre-trained word embedding là sử dụng cho nhiều tác vụ (tasks) khác nhau chứ không chỉ sử dụng đơn lẻ cho duy nhất một tác vụ.

Các bộ pre-trained word embedding các bạn có thể tham khảo tại đây: https://github.com/vietnlp/etnlp

Trong bài thực hành này sẽ sử dụng bộ **W2V_ner** để thực hiện.

Ghi chú: khi sử dụng pre-trained word embedding, phải chú ý chiều (dimension) của các bộ pre-trained word embedding.

Đọc dữ liệu từ bộ word embedding:

```
word_dict = []
embeddings_index = {}
embedding_dim = 300
max_feature = len(embeddings_index) + 1

f = open('W2V_ner.vec')
for line in f:
    values = line.split(' ')
    word = values[0]
    word_dict.append(word)
    try:
```

```
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
    except Exception as e:
        pass
f.close()

print('Embedding data loaded')
```

Xây dựng embedding matrix cho pre-trained embedding

```
# first create a matrix of zeros, this is our embedding matrix
embedding_matrix = np.zeros((num_words, embedding_dim))

# for each word in out tokenizer lets try to find that work in
our w2v model
for word, i in word_to_index.items():
    if i > max_feature:
        continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # we found the word - add that words vector to the
matrix
        embedding_matrix[i] = embedding_vector
    else:
        # doesn't exist, assign a random vector
        embedding_matrix[i] = np.random.randn(embedding_dim)
```

Các bước tiền xử lý dữ liệu tương tự như mục 1 nhé (các bạn dùng thư viện Tokenizer) Để sử dụng pre-trained embedding, ta sẽ đưa giá trị embedding matrix vào lớp embedding của mô hình.

```
from keras.layers import Dense, Embedding, Flatten
from keras.models import Model, Input
from keras.initializers import Constant

num_words = len(word_to_index)
input = Input(shape = (max_len, ))
emb = Embedding(input_dim=num_words,
                    output_dim=embedding_dim,
embeddings_initializer=Constant(embedding_matrix),
                    input_length=max_len,
                    trainable=True)(input)
```

```
flat = Flatten()(emb)
output = Dense(3, activation="sigmoid")(flat)

model = Model(input, output)
model.compile(optimizer="adam",
loss='binary_crossentropy', metrics=['binary_accuracy'])

model.summary()
```

Các bước huấn luyện mô hình, dự đoán và kiểm thử tương tự như đã hướng dẫn trong mục 3. Các bạn tự làm tiếp theo nhé.

Ghi chú: Có 2 phương pháp sử dụng pre-trained embedding:
+ **Static**: sử dụng toàn bộ trọng số đã huấn luyện trước đó của pre-trained word embedding. Không cập nhật trọng số trong quá trình học. Để sử dụng chế độ này, ta đặt tham số **trainable = False** trong lớp Embedding.
+ **Jointly**: Sử dụng trọng số đã huấn luyện trước đó của pre-trained word embedding như là trọng số khởi tạo, sau đó tiến hành cập nhật trọng số trong quá trình huấn luyện mô hình trên 1 tác vụ cụ thể. Để sử dụng chế độ này, ta đặt tham số **trainable = True** trong lớp Embedding.

4. Bài tập

Bài 1: Thực hiện xây dựng mô hình dự đoán cho bài toán phân tích cảm xúc (sentiment-based) dựa trên bộ dữ liệu UIT-VSFC. Có thể chọn 1 bộ pre-trained embedding khác. (Tham khảo tại đây: https://github.com/vietnlp/etnlp)
Bài 2: Thực hiện xây dựng mô hình dự đoán cho bài toán phân loại chủ đề (topic-based) dựa trên bộ dữ liệu UIT-VSFC. Lưu ý: pre-trained embedding ở 2 bài phải trùng nhau.

Chú ý nộp bài: Gồm tất cả các file dưới đây trong 1 thứ mục nén lại và nộp (Họ tên_mssv_lab1.zip)

- 02 File jupyter notebook của hai bài tập trên.
- 01 file word trình bày kết quả thí nghiệm của hai câu bài tập trên (Độ đo F1 và Accuracy), nhớ ghi chú tên bộ dữ pre-trained embedding và link tải.