18521655 – Nguyễn Xuân Vinh

Bài tập Thực hành 1 - Phân lớp văn bản

Mô tả sơ lược về các bước thử nghiệm

Nội dung

1. Đọc dữ liệu

- Đọc dữ liệu từ bộ dữ liệu UIT-VSFC theo các folder train, dev và test.
- Sử dụng framework Pandas, gọi pd.read_csv để đọc dữ liệu cho từng file sents, sentiment và topics.
- Nội dung ở các file:
    - Sents: Mỗi dòng là câu , document gốc.
    - Sentiment: Mỗi dòng là một nhãn về cảm xúc của câu: negative, neutral, positive.
    - Topics: Mỗi dòng là nhãn của một chủ đề: Lecturer, Curriculum, Facility và others.
- Chúng ta sẽ có:
    - x_train, x_dev và x_test: Được đọc từ các file sents.
    - y_train, y_dev và y_test: Được đọc từ các file sentiment.
    - y_train_topic, y_dev_topic và y_test_topic: Được đọc từ các file topics.

2. Tiền xử lý, mã hóa, biến đổi dữ liệu

a. TF-IDF

- Sử dụng TF-IDF để biến đổi text trong sents thành các vector (tạo ma trận đồng hiện từ các từ trong tập x_train).
- TF-IDF sẽ được cài đặt với tham số: analyzer='word', ngram_range=(2,2)
- TF-IDF sẽ được fit với tập x_train.
- Sau đó TF-IDF sẽ transform các tập x_train, x_dev, x_test thành các tập X_train_encoded, X_dev_encoded, X_test_encoded.
- X_train_encoded sẽ có dạng ma trận với kích thước 11426x31384

b. Word Embedding tự build

- Xây dựng tập từ vựng (V) từ tập x_train bằng cách tách từ dùng ViTokenizer.tokenize và tokenized_sentence.split() để đưa vào tập từ vựng
- Xây dựng từ điển từ và index tương ứng (word_to_index). Để mã hóa các text thành số.
- Word_to_index sẽ có key (từ) và value (số).
- Tiếp đến chuyển dữ liệu văn bản ban đầu x_train 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). Kỹ thuật padding sẽ thêm các số 0 để độ dài của vector = maxlen được tạo từ ban đầu.
- 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).

- Cuối cung tạo hàm encoding để chuyển đổi x_train, x_dev, x_test thành X_train_encoded, X_dev_encoded, X_test_encoded bằng word_to_index.

c. Word Embedding tạo từ hàm có sẵn trong Keras

- Sử dụng thư viện from keras.preprocessing.text import Tokenizer.
- Tạo word_tokenizer = Tokenizer(oov_token=-1) từ thư viện trên.
- Sau đó đưa word_tokenizer vào fit với dữ liệu x_train.
- Word_2_index sẽ được tạo bằng cách gọi word_tokenizer.word_index, cùng với đó word_2_index['pad'] = 0 và word_2_index['unk'] = -1.
- Tiếp đến tạo index_to_word để chuyển index ban đầu lại thành văn bản.
- Cuối cung tạo hàm encoding để chuyển đổi x_train, x_dev, x_test thành X_train_encoded, X_dev_encoded, X_test_encoded bằng word_2_index.

d. Word Embedding pre-trained

- Mô hình word embedding pre-trained được sử dụng là **W2V_ner**.
- Đọc dữ liệu từ word embedding. Xây dựng tập từ vựng.
- Xây dựng embedding matrix cho pre-trained embedding
- Xây dựng word_to_index và index_to_word (như ở mục b,c)
- Cuối cung tạo hàm encoding để chuyển đổi x_train, x_dev, x_test thành X_train_encoded, X_dev_encoded, X_test_encoded từ word_to_index được tạo ở trên.

3. Huấn luyện mô hình

- Các mô hình được huấn luyện gồm: Navie Bayes, Logistic Regression, SVM, Neural.
- Các mô hình học máy cơ bản: : Navie Bayes, Logistic Regression, SVM sẽ được cài đặt với các tham số mặc định và được huấn luyện với dữ liệu được biến đổi qua TF-IDF.

- Mô hình Neural sẽ được cài đặt như sau:
    - Mô hình neural với sentiment-base

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding (Embedding)        (None, 100, 300)          1112100
_________________________________________________________________
flatten (Flatten)            (None, 30000)             0
_________________________________________________________________
dense (Dense)                (None, 3)                 90003
=================================================================
Total params: 1,202,103
Trainable params: 1,202,103
Non-trainable params: 0
```

    - Mô hình neural với topic-base

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 100, 300)          1112100
_________________________________________________________________
flatten_1 (Flatten)          (None, 30000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 120004
=================================================================
Total params: 1,232,104
Trainable params: 1,232,104
Non-trainable params: 0
```

- Hai mô hình trên đều được cài đặt cùng số lượng lớp, và sử dụng chung lớp Input và Embedding. Sự khác nhau giữa 2 mô hình là lớp output với **sentiment** sẽ là 3 và ở **topic** sẽ là 4, tương ứng với các nhãn.
- Embedding được sử dụng ở đây gồm **word embedding tự build**, **word embedding tạo từ hàm có sẵn trong Keras** và **embedding pre-trained**.
- Riêng đối với embedding pre-trained thì ở tham số **trainable = True** để cập nhật trọng số trong quá trình huấn luyện mô hình trên tác vụ phân loại văn cảm xúc và chủ đề.

4. Kết quả thu được

- Dưới đây là bảng kết quả thu được của tác vụ **sentiment-base** và **topic-base** sau khi thực nghiệm với độ đo đánh giá là **F1 socre** và **Accuracy** ở cột cuối cùng

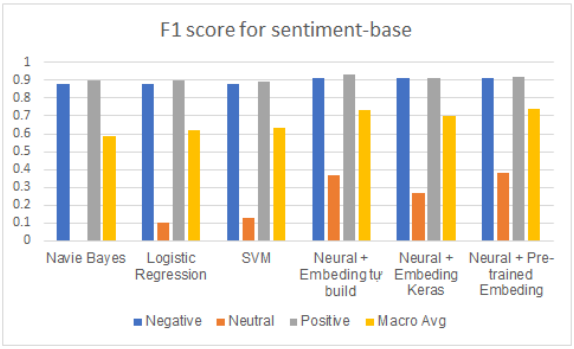
Mô hình	Negative	Neutral	Positive	Macro Avg	Accuracy
Navie Bayes	0.88	0	0.9	0.59	0.86
Logistic Regression	0.88	0.1	0.9	0.62	0.86
SVM	0.88	0.13	0.89	0.63	0.86
Neural + Embeding tự build	0.91	0.37	0.93	0.73	0.89
Neural + Embeding Keras	0.91	0.27	0.91	0.7	0.89
Neural + Pre-trained Embeding	0.91	0.38	0.92	0.74	0.89

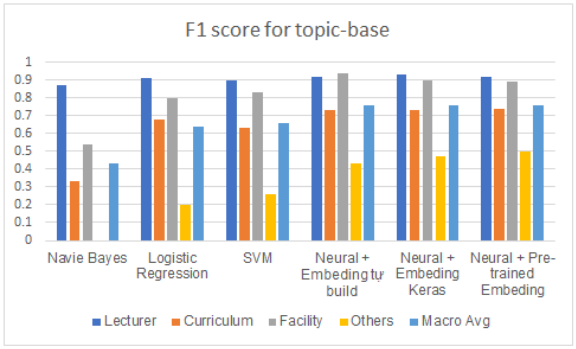
Bảng 1. Bảng kết quả thực nghiệm F1 socre và accuracy cho sentiment-base

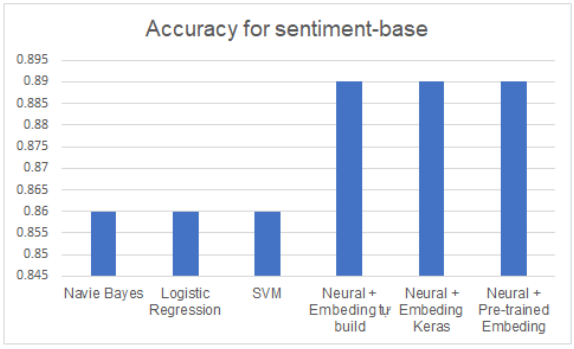
Mô hình	Lecturer	Curriculum	Facility	Others	Macro Avg	Accuracy
Navie Bayes	0.87	0.33	0.54	0	0.43	0.78
Logistic Regression	0.91	0.68	0.8	0.2	0.64	0.84
SVM	0.9	0.63	0.83	0.26	0.66	0.84
Neural + Embeding tự build	0.92	0.73	0.94	0.43	0.76	0.87
Neural + Embeding Keras	0.93	0.73	0.9	0.47	0.76	0.87
Neural + Pre-trained Embeding	0.92	0.74	0.89	0.5	0.76	0.87

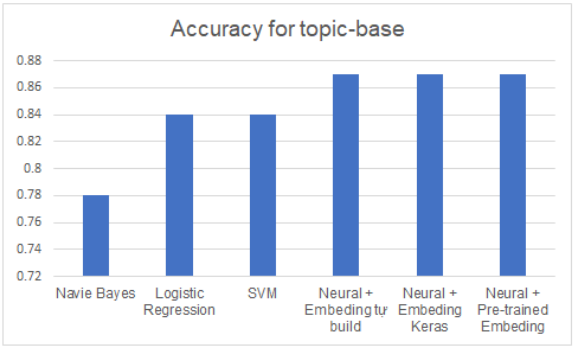
Bảng 2. Bảng kết quả thực nghiệm F1 socre và accuracy cho topic-base

- Đồ thị dưới đây sẽ cho cái nhìn trực quan hơn về kết quả thực nghiệm thu được.









- Qua 2 bảng và đồ thị kết quả trên ta có thể nhận thấy các mô hình tốt nhất cho 2 tác vụ:
    - Sentiment-base là: **Neural + Pre-trained Embeding** có macro avg F1_socre = 0.74 và accuracy = 0.89.
    - Topic-base là: **Neural + Embedding (tất cả)**, tất cả các mô hình neural kết hợp với embedding đều cho kết quả macro avg F1_score= 0.76 và accracy = 0.87.
- Tiếp đến ta nhìn nhận sự khác biệt giữa học máy cơ bản với mạng neural, các kết quả ở 2 bảng trên đều cho kết quả về sự chênh lệch rõ rệt giữa cả 2. Đặc biệt ở F1_Score thì sự chênh lệch đó càng rõ ràng. Với tác vụ sentiment thì ở học máy macro avg F1_score cao nhất chỉ là 0.63 trong khi ở mạng neural avg F1_score thấp nhất =0.7, cao hơn nhiều so với học máy. Tương tự ở tác vụ topic cũng có sự chênh lệch giữa 0.66 ở học máy và 0.76 ở mạng neural.

5. Phân tích lỗi

- Mô hình chưa thật sự có kết quả tốt vì dữ liệu học tập có chênh lệnh khá lớn giữa các nhãn. Ở bộ dữ liệu ta có số lượng các nhãn sentiment-base tỉ lệ như sau negative:neutral:positive = 17:1:19. Tương tự các nhãn của topic-base có tỉ lệ Lecturer: Curriculum:Facility:Others = 15:4:1:1.
- Sự chênh lệch giữa các quá lớn đã gây nhiễu trong quá trình huấn luyện như ta thấy ở bảng 1 của sentiment-base với nhãn neutral các kết quả F1_score cao nhất chỉ 0.38, thấp nhất là về 0. Nó giống với nhãn Others trong bảng 2 của topic-base
- Còn về topic nhờ ở nhãn Facility có các đặc trưng khá tốt nên không có kết quả thấp như vậy, giá trị thấp nhất chỉ đạt 0.54 và tốt nhất lên đến 0.94.