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 2](#_Toc86174997)

[2. Tiền xử lý, mã hóa, biến đổi dữ liệu 2](#_Toc86174998)

[a. TF-IDF 2](#_Toc86174999)

[b. Word Embedding tự build 2](#_Toc86175000)

[c. Word Embedding tạo từ hàm có sẵn trong Keras 3](#_Toc86175001)

[d. Word Embedding pre-trained 3](#_Toc86175002)

[3. Huấn luyện mô hình 3](#_Toc86175003)

[4. Kết quả thu được 5](#_Toc86175004)

[5. Phân tích lỗi 6](#_Toc86175005)

# 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.

# Tiền xử lý, mã hóa, biến đổi dữ liệu

## 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

## 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.

## 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.

## 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.

# 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

Ảnh có chứa văn bản

Mô tả được tạo tự động

* + Mô hình neural với topic-base

Ảnh có chứa văn bản, màn hình, đen, ảnh chụp màn hình

Mô tả được tạo tự động

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.

# 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.

Ảnh có chứa văn bản, công cụ viết, văn phòng phẩm, ảnh chụp màn hình

Mô tả được tạo tự động

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.

# 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.