Xử lý ngôn ngữ tự nhiên cho Khoa học dữ liệu
Bài tập thực hành 2 - DS310.M11.1

Nguyễn Lưu Thùy Ngân, Nguyễn Văn Kiệt, Huỳnh Văn Tín,
Lưu Thanh Sơn, Nguyễn Đức Vũ, Nguyễn Thành Luân
Sinh viên: Phạm Đức Thể - 19522253
Ngôn ngữ lập trình: Python

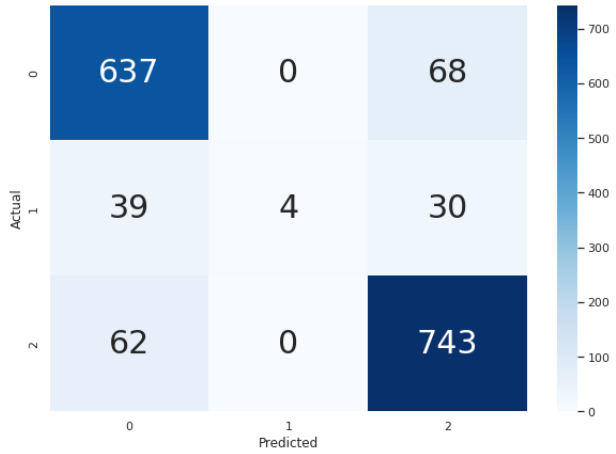
Thứ 5, ngày 16 tháng 10 năm 2021

Bài tập: PHÂN LỚP VĂN BẢN SỬ DỤNG PRE-TRAINED WORD EMBEDDING

- Đọc dữ liệu
  - Để đọc được dữ liệu từ folder nằm trong Google drive, đầu tiên ta phải đảm bảo rằng đã connect file Colab với tài khoản Google drive.
  - Đọc dữ liệu train: Sử dụng phương thức **read_csv()** của thư viện **pandas** để đọc dữ liệu train và lưu vào đối tượng X_train, y_train, y_train_topic.
  - Đọc dữ liệu dev: Sử dụng phương thức **read_csv()** của thư viện **pandas** để đọc dữ liệu dev và lưu vào đối tượng X_dev, y_dev, y_dev_topic.
  - Đọc dữ liệu test: Sử dụng phương thức **read_csv()** của thư viện **pandas** để đọc dữ liệu test và lưu vào đối tượng X_test, y_test., y_test_topic.

Bài 1: Tác vụ sentiment-based

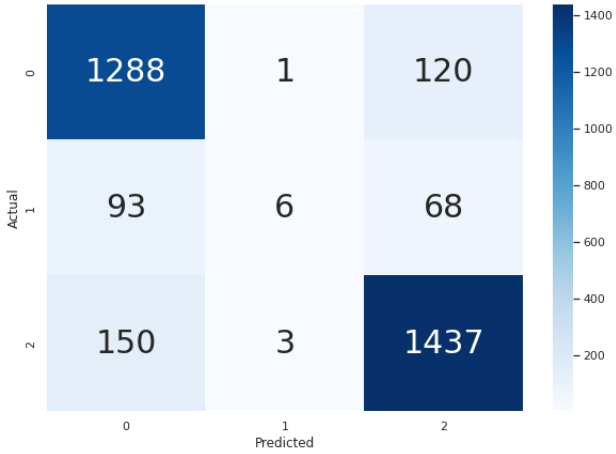
1. Thuật toán máy học cơ bản
   (a) Naive Bayes
      i. Tiền xử lý dữ liệu
         - Sử dụng phương thức **CountVectorizer()** với đối số **analyzer='word'** và **ngram_range=(2,2)**, tức là sử dụng mô hình 2-gram.
         - Tiếp theo chúng ta sẽ sử dụng phương thức **transform()** để tiến hành mã hóa văn bản train set, dev set và test set.
      ii. Huấn luyện mô hình
         - Sử dụng phương thức **MultinomialNB()** mặc định của **sklearn.naive_bayes** để khởi tạo model.
         - Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train**.
      iii. Đánh giá mô hình
         - Trên tập dev:
           – Confusion matrix (Hình 1)
             * Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
             * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 1: Confusion matrix of dev set

- F1-score: Mô hình đạt được độ chính xác F1-score macro = 62.99% và F1-score micro = 87.43%.

- Trên tập test:
  - Confusion matrix (Hình 2)
    - Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
    - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 2: Confusion matrix of test set

  - F1-score: Mô hình đạt được độ chính xác F1-score macro = 61.26% và F1-score micro = 86.26%.

(b) Logistic Regression

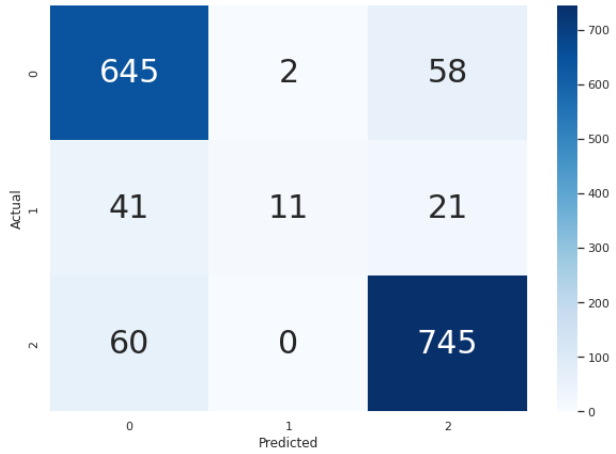
i. Tiền xử lý dữ liệu (tương tự phần 1a)

ii. Huấn luyện mô hình

- Sử dụng phương thức **LogisticRegression()** mặc định của **sklearn.linear_model** để khởi tạo model.
- Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train**.

iii. Đánh giá mô hình

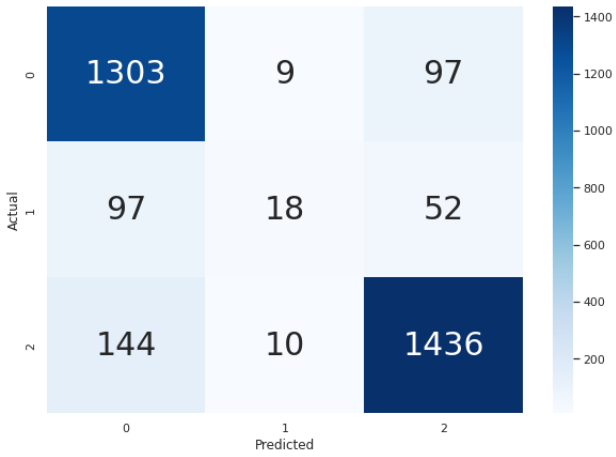
- Trên tập dev:
  - Confusion matrix (Hình 3)
    - Mô hình dự đoán tốt với nhãn có label = {0,2} và dự đoán không tốt với label = {1}.
    - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 3: Confusion matrix of dev set

- F1-score: Mô hình đạt được độ chính xác F1-score macro = 68.65% và F1-score micro = 88.5%.

- Trên tập test:
  - Confusion matrix (Hình 4)
    - Mô hình dự đoán tốt với nhãn có label = {0,2} và dự đoán không tốt với label = {1}.
    - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 4: Confusion matrix of test set

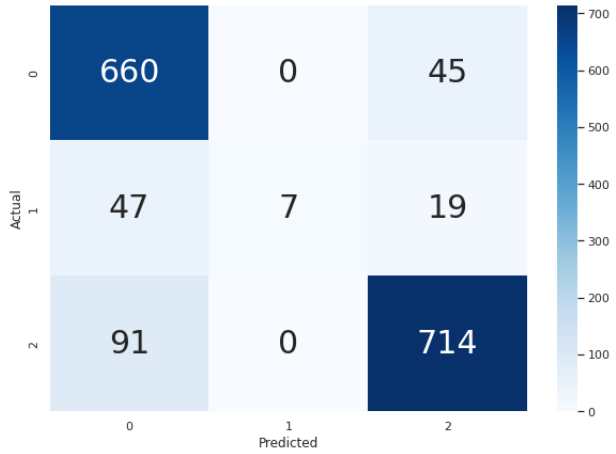
  - F1-score: Mô hình đạt được độ chính xác F1-score macro = 65.45% và F1-score micro = 87.08%.

(c) Support Vector Machine

i. Tiền xử lý dữ liệu (tương tự phần 1a)

ii. Huấn luyện mô hình
- Sử dụng phương thức **SVC()** mặc định của **sklearn.svm** để khởi tạo model.
- Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train**.

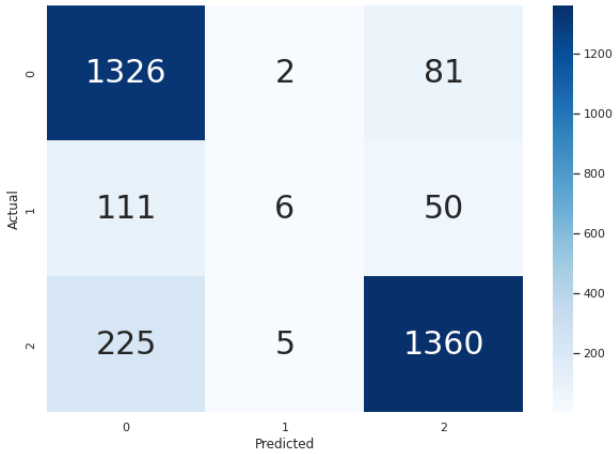
iii. Đánh giá mô hình
- Trên tập dev:
  - Confusion matrix (Hình 5)
    - Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
    - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 5: Confusion matrix of dev set

- F1-score: Mô hình đạt được độ chính xác F1-score macro = 65.18% và F1-score micro = 87.24%.

- Trên tập test:
  - Confusion matrix (Hình 6)
    - Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
    - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 6: Confusion matrix of test set

  - F1-score: Mô hình đạt được độ chính xác F1-score macro = 60.44% và F1-score micro = 85.03%.

- **Kết luận về 3 thuật toán máy học cơ bản (dựa trên test set)**:
  - Mô hình sử dụng thuật toán Logistic Regression có độ chính xác F1-score macro và F1-score micro cao nhất trong 3 mô hình (F1-score macro = 65.45%, F1-score micro = 87.08%).
  - Mô hình sử dụng thuật toán Support Vector Machine có độ chính xác F1-score macro và F1-score micro thấp nhất trong 3 mô hình (F1-score macro = 60.44%, F1-score micro = 85.03%).
  - Nhìn chung ta có thể đánh giá được là mô hình sử dụng thuật toán Logistic Regression là tốt nhất trong 3 mô hình khi mà độ chính xác F1-score khá cao và dựa vào confusion matrix có thể thấy label = {1} đã được dự đoán tốt hơn.

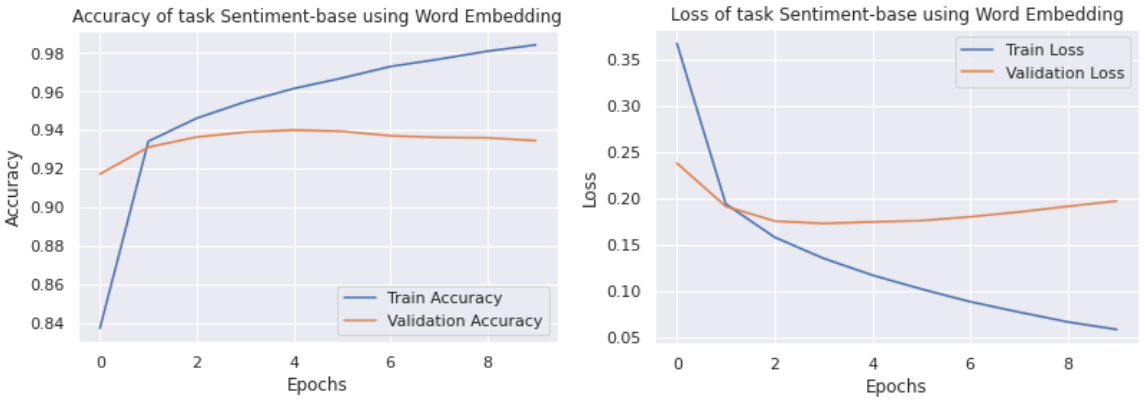
2. Mạng neural, sử dụng word embedding

   (a) Tiền xử lý dữ liệu

   - Cách 1: Tự xây dựng bộ từ vựng và từ điển tương ứng:
     - Xây dựng tập từ vựng **V**
     - Xây dựng word_to_index và index_to_word
     - Tiến hành encode dữ liệu văn bản
   - Cách 2: Dùng thư viện có sẵn trong keras. Sử dụng thư viện Tokenizer

   (b) Huấn luyện mô hình

   - Sử dụng 1 lớp Embedding trong mạng neural kết hợp với 1 lớp đầu ra.
   - Accuracy và loss của mô hình (Hình 7)



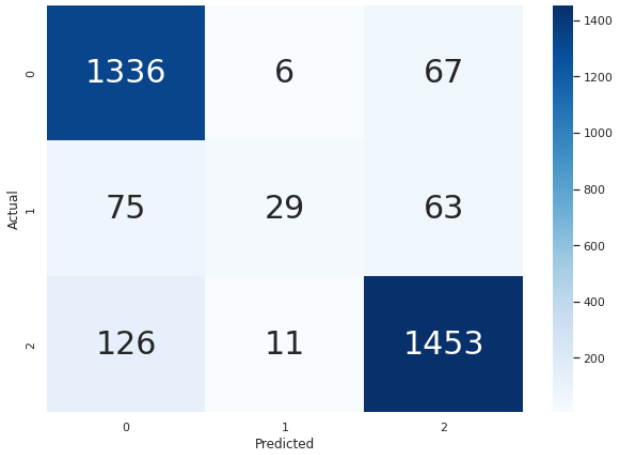
(a) Accuracy (b) Loss

Hình 7: Accuracy and Loss of task Sentiment-base using Word Embedding

   - **Nhận xét**: Mô hình bị overfitting. Accuracy trên tập train được cải thiện qua các epoch, ngược lại accuracy trên tập dev lại đi ngang và có xu hướng giảm ở các epoch sau. Loss giảm ở vài epoch đầu nhưng sau đó lại có xu hướng tăng cao.

   (c) Đánh giá mô hình

   - Confusion matrix (Hình 8)
     - Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
     - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 8: Confusion matrix of test set

   - F1-score: Mô hình đạt được độ chính xác F1-score macro = 69.84% và F1-score micro = 89.01%.

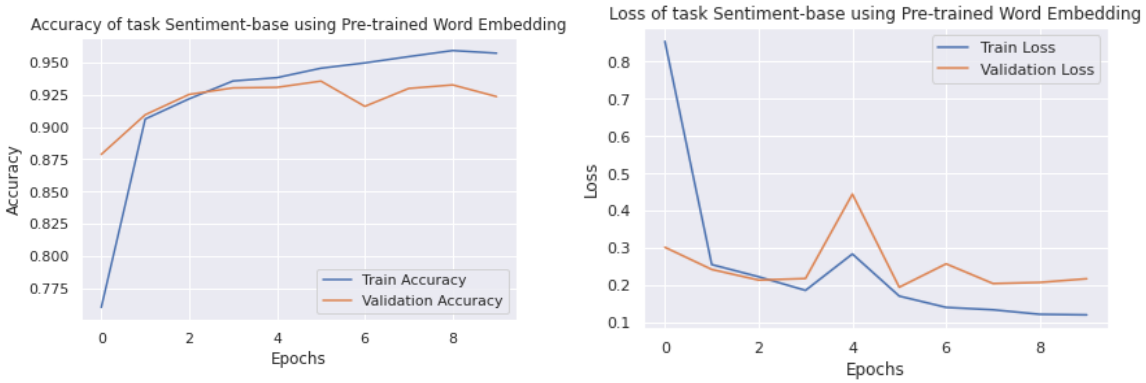
3. Mạng neural, sử dụng pre-trained embedding

   (a) Tiền xử lý dữ liệu

   - Đọc dữ liệu từ bộ word embedding (sử dụng bộ pre-trained embedding **W2V_ner.vec**) (Nguồn: https://github.com/vietnlp/etnlp)
   - Xây dựng embedding matrix cho pre-trained embedding
   - Tiến hành encode dữ liệu

   (b) Huấn luyện mô hình

   - Sử dụng 1 lớp Embedding trong mạng neural kết hợp với 1 lớp đầu ra.
   - Accuracy và loss của mô hình (Hình 9)



(a) Accuracy

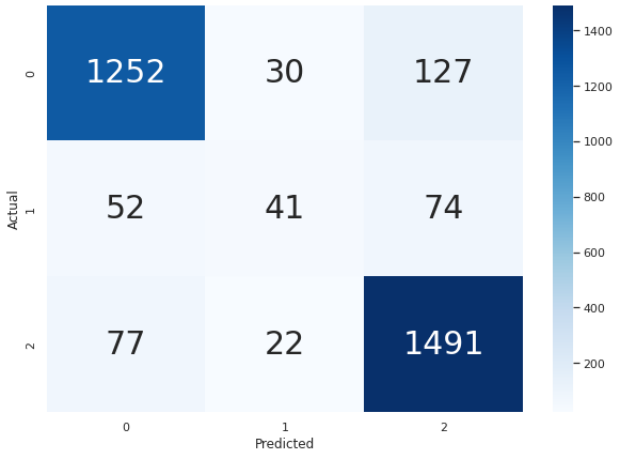
(b) Loss

Hình 9: Accuracy and Loss of task Sentiment-base using Pre-trained Embedding

   - **Nhận xét**: Mô hình đã giảm đi hiện tượng overfitting.

   (c) Đánh giá mô hình

   - Confusion matrix (Hình 10)
     - Mô hình dự đoán tốt với label = {0,2} và dự đoán không tốt với label = {1}.
     - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 10: Confusion matrix of test set

   - F1-score: Mô hình đạt được độ chính xác F1-score macro = 70.72% và F1-score micro = 87.93%.

- **Kết luận**:

  – Mô hình sử dụng thuật toán Pre-trained Embedding có độ chính xác F1-score macro cao nhất trong 3 mô hình (F1-score macro = 70.72%).

– Mô hình sử dụng thuật toán Word Embedding có độ chính xác F1-score micro cao nhất trong 3 mô hình (F1-score macro = 89.01%).
– Mô hình sử dụng thuật toán máy học cơ bản có độ chính xác F1-score macro và F1-score micro thấp nhất trong 3 mô hình (F1-score macro = 65.45%, F1-score micro = 87.08%).
– Nhìn chung ta có thể đánh giá được là mô hình sử dụng thuật toán Pre-trained Embedding là tốt nhất trong 3 mô hình khi mà độ chính xác F1-score khá cao và dựa vào confusion matrix có thể thấy label = {1} đã được dự đoán tốt hơn. Ngoài ra, còn khắc phục được hiện tượng overfitting của model khi sử dụng thuật toán Word Embedding.

Bài 2: Tác vụ topic-based

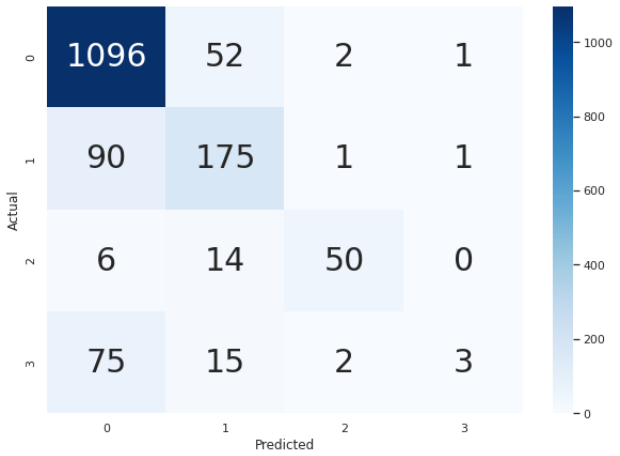
1. Thuật toán máy học cơ bản

(a) Naive Bayes

i. Tiền xử lý dữ liệu
- Sử dụng phương thức **CountVectorizer()** với đối số **analyzer='word'** và **ngram_range=(2,2)**, tức là sử dụng mô hình 2-gram.
- Tiếp theo chúng ta sẽ sử dụng phương thức **transform()** để tiến hành mã hóa văn bản train set, dev set và test set.

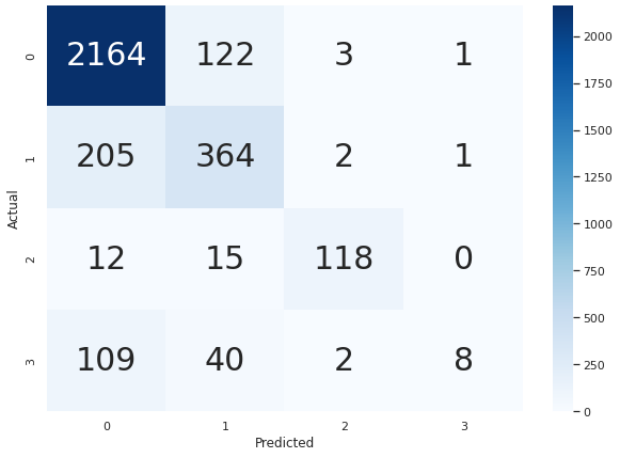
ii. Huấn luyện mô hình
- Sử dụng phương thức **MultinomialNB()** mặc định của **sklearn.naive_bayes** để khởi tạo model.
- Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train_topic**.

iii. Đánh giá mô hình
- Trên tập dev:
  – Confusion matrix (Hình 11)
    * Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
    * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 11: Confusion matrix of dev set

  – F1-score: Mô hình đạt được độ chính xác F1-score macro = 60.89% và F1-score micro = 83.64%.
- Trên tập test:
  – Confusion matrix (Hình 12)
    * Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.



Hình 12: Confusion matrix of test set

* Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.

– F1-score: Mô hình đạt được độ chính xác F1-score macro = 63.21% và F1-score micro = 83.83%.

(b) Logistic Regression

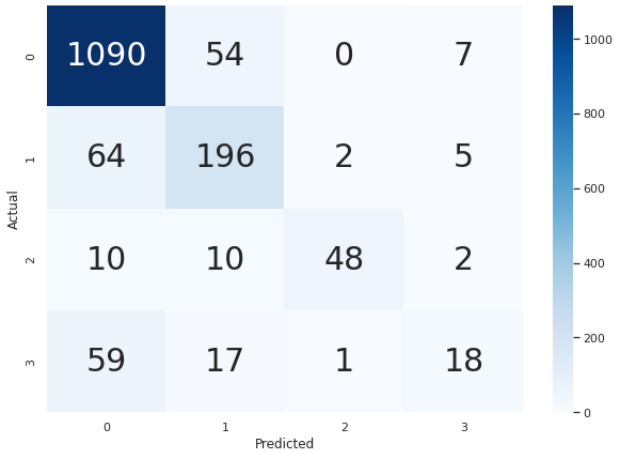
i. Tiền xử lý dữ liệu (tương tự phần 2a)

ii. Huấn luyện mô hình

- Sử dụng phương thức **LogisticRegression()** mặc định của **sklearn.linear_model** để khởi tạo model.
- Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train_topic**.

iii. Đánh giá mô hình

- Trên tập dev:
  – Confusion matrix (Hình 13)
    * Mô hình dự đoán tốt với nhãn có label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
    * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.

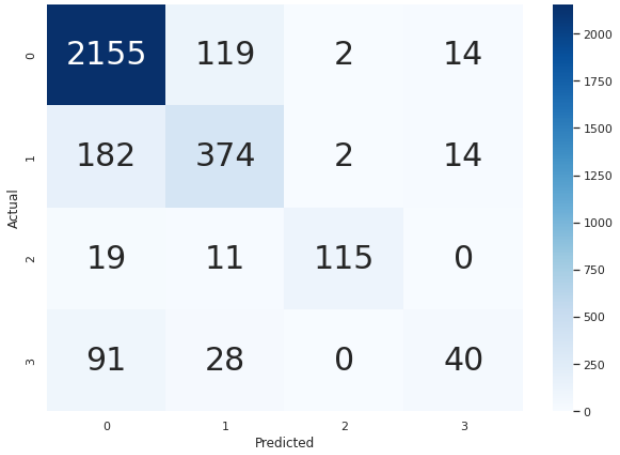


Hình 13: Confusion matrix of dev set

– F1-score: Mô hình đạt được độ chính xác F1-score macro = 67.89% và F1-score micro = 85.41%.

- Trên tập test:

– Confusion matrix (Hình 14)
  * Mô hình dự đoán tốt với nhãn có label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
  * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 14: Confusion matrix of test set

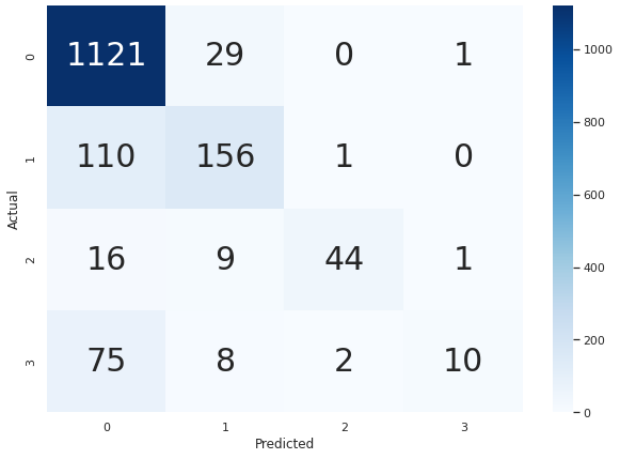
– F1-score: Mô hình đạt được độ chính xác F1-score macro = 70.28% và F1-score micro = 84.78%.

(c) Support Vector Machine

i. Tiền xử lý dữ liệu (tương tự phần a)

ii. Huấn luyện mô hình
- Sử dụng phương thức **SVC()** mặc định của **sklearn.svm** để khởi tạo model.
- Tiếp theo chúng ta sẽ sử dụng phương thức **fit()** để tiến hành training model với đối số là **X_train_encoded**, **y_train_topic**.

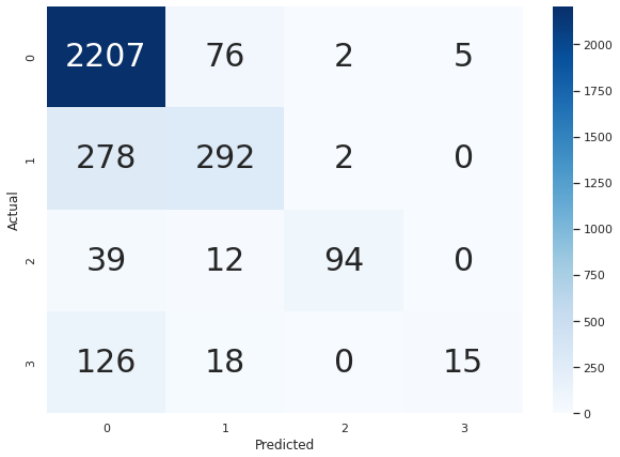
iii. Đánh giá mô hình
- Trên tập dev:
  – Confusion matrix (Hình 15)



Hình 15: Confusion matrix of dev set

  * Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.

        * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.
      - F1-score: Mô hình đạt được độ chính xác F1-score macro = 62.77% và F1-score micro = 84.08%.
    - Trên tập test:
      - Confusion matrix (Hình 16)
        * Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
        * Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 16: Confusion matrix of test set

      - F1-score: Mô hình đạt được độ chính xác F1-score macro = 60.92% và F1-score micro = 82.38%.

- **Kết luận về 3 thuật toán máy học cơ bản (dựa trên test set)**:
  - Mô hình sử dụng thuật toán Logistic Regression có độ chính xác F1-score macro và F1-score micro cao nhất trong 3 mô hình (F1-score macro = 70.28%, F1-score micro = 84.78%).
  - Mô hình sử dụng thuật toán Support Vector Machine có độ chính xác F1-score macro và F1-score micro thấp nhất trong 3 mô hình (F1-score macro = 60.92%, F1-score micro = 82.38%).
  - Nhìn chung ta có thể đánh giá được là mô hình sử dụng thuật toán Logistic Regression là tốt nhất trong 3 mô hình khi mà độ chính xác F1-score khá cao.

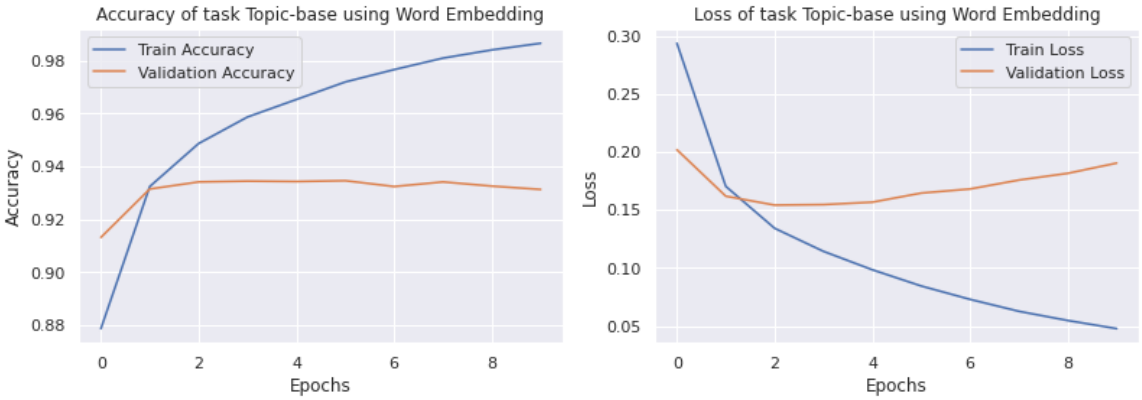
2. Mạng neural, sử dụng word embedding

(a) Tiền xử lý dữ liệu

- Cách 1: Tự xây dựng bộ từ vựng và từ điển tương ứng:
  - Xây dựng tập từ vựng $\mathbf{V}$
  - Xây dựng word_to_index và index_to_word
  - Tiến hành encode dữ liệu văn bản
- Cách 2: Dùng thư viện có sẵn trong keras. Sử dụng thư viện Tokenizer

(b) Huấn luyện mô hình

- Sử dụng 1 lớp Embedding trong mạng neural kết hợp với 1 lớp đầu ra.
- Accuracy và loss của mô hình (Hình 17)



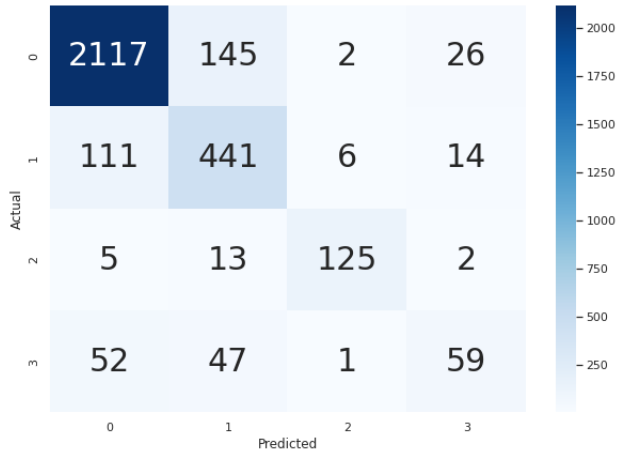
(a) Accuracy (b) Loss

Hình 17: Accuracy and Loss of task Topic-base using Word Embedding

- **Nhận xét**: Mô hình bị overfitting. Accuracy trên tập train được cải thiện qua các epoch, ngược lại accuracy trên tập dev lại đi ngang và có xu hướng giảm ở các epoch sau. Loss giảm ở vài epoch đầu nhưng sau đó lại có xu hướng tăng cao.

(c) Đánh giá mô hình

- Confusion matrix (Hình 18)
  - Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
  - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 18: Confusion matrix of test set

- F1-score: Mô hình đạt được độ chính xác F1-score macro = 74.99% và F1-score micro = 86.61%.

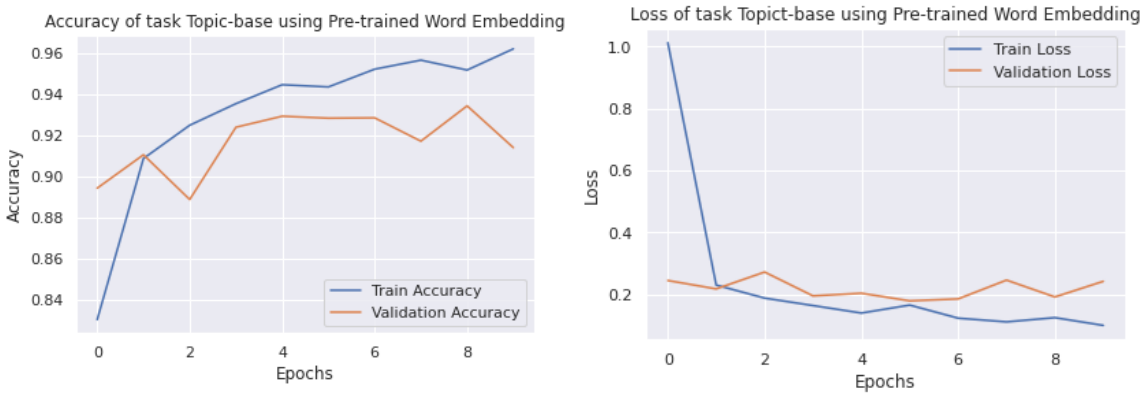
3. Mạng neural, sử dụng pre-trained embedding

(a) Tiền xử lý dữ liệu

- Đọc dữ liệu từ bộ word embedding (sử dụng bộ pre-trained embedding **W2V_ner.vec**) (Nguồn: https://github.com/vietnlp/etnlp)
- Xây dựng embedding matrix cho pre-trained embedding
- Tiến hành encode dữ liệu

(b) Huấn luyện mô hình

- Sử dụng 1 lớp Embedding trong mạng neural kết hợp với 1 lớp đầu ra.
- Accuracy và loss của mô hình (Hình 19)



(a) Accuracy

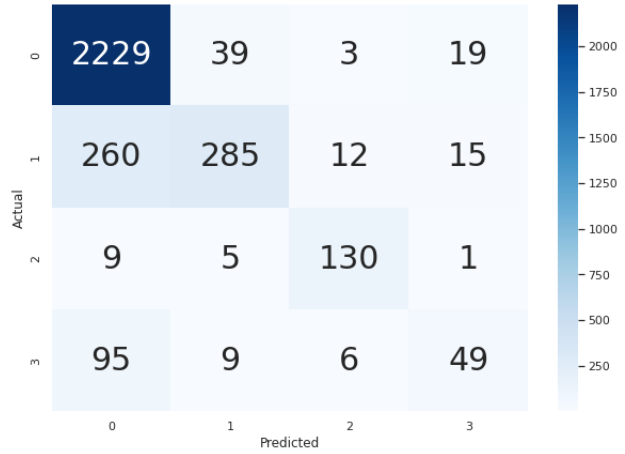
(b) Loss

Hình 19: Accuracy and Loss of task Topic-base using Pre-trained Embedding

- **Nhận xét**: Mô hình đã giảm đi hiện tượng overfitting.

(c) Đánh giá mô hình

- Confusion matrix (Hình 20)
  - Mô hình dự đoán tốt với label = {0,1,2} và dự đoán không tốt với label = {3}. Các nhãn sai đa số là dự đoán nhầm vào label = {0}, nguyên nhân là do label này chiếm đa số trong tập dữ liệu nên có hiện tượng bias với label = {0}.
  - Nguyên nhân có thể là vì bộ dữ liệu bị mất cân bằng dữ liệu giữa các label.



Hình 20: Confusion matrix of test set

- F1-score: Mô hình đạt được độ chính xác F1-score macro = 70.53% và F1-score micro = 85.06%.

- **Kết luận**:
  - Mô hình sử dụng thuật toán Word Embedding có độ chính xác F1-score macro và F1-score micro cao nhất trong 3 mô hình (F1-score macro = 74.99% và F1-score micro= 86.61%).
  - Mô hình sử dụng thuật toán máy học cơ bản có độ chính xác F1-score macro và F1-score micro thấp nhất trong 3 mô hình (F1-score macro = 70.28%, F1-score micro = 84.78%).
  - Nhìn chung ta có thể đánh giá được là mô hình sử dụng thuật toán Word Embedding là tốt nhất trong 3 mô hình khi mà độ chính xác F1-score khá cao và dựa vào confusion matrix có thể thấy label = {3} đã được dự đoán tốt hơn. Nhưng có một vấn đề là model này bị overftting khá nghiêm trọng.