Quora Question Pairs Report

1. Đặt vấn đề

a. Mô tả

Quora là nơi để đạt được và chia sẻ kiến thức — về bất cứ điều gì. Đó là một nền tảng để đặt câu hỏi và kết nối với những người đóng góp thông tin chi tiết độc đáo và câu trả lời chất lượng. Điều này cho phép mọi người học hỏi lẫn nhau và hiểu rõ hơn về thế giới.

Hơn 100 triệu người truy cập Quora mỗi tháng, vì vậy không có gì ngạc nhiên khi nhiều người đặt những câu hỏi tương tự. Nhiều câu hỏi có cùng mục đích có thể khiến người tìm kiếm mất nhiều thời gian hơn để tìm câu trả lời tốt nhất cho câu hỏi của họ và khiến người viết cảm thấy họ cần phải trả lời nhiều lần của cùng một câu hỏi. Quora đánh giá cao các câu hỏi chuẩn vì chúng cung cấp trải nghiệm tốt hơn cho những người tìm kiếm và người viết tích cực, đồng thời mang lại nhiều giá trị hơn cho cả hai nhóm này về lâu dài.

b. Vấn đề

- Xác định những câu hỏi được hỏi trên Quora là trùng lặp của những câu hỏi đã được hỏi.
- Điều này có thể hữu ích để cung cấp ngay lập tức câu trả lời cho các câu hỏi đã được trả lời.
- Chúng ta có nhiệm vụ dự đoán xem một cặp câu hỏi có trùng lặp hay không.

2. Vấn đề Học Máy (Machine Learning problem)

a. Dữ liệu

- Dữ liệu nằm trong file train.csv
- Train.csv bao gồm 5 columns: qid1, qid2, question1, question2, is_duplicate
- Kích thước của file train.csv: 60MB
- Số lượng các mẫu (samples): 404 290

b. Bài toán trong Học Máy

- Đây là một bài toán phân loại nhị phân, đối với một cặp câu hỏi nhất định, chúng ta cần dự đoán xem chúng có trùng lặp hay không.

c. Phương pháp đánh giá

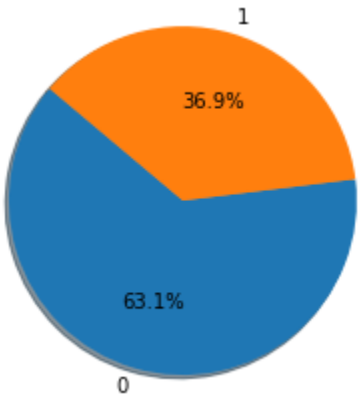
- Precison, Recall, F1-score, Accuracy
- Ma trận nhầm lẫn (confusion matrix)

3. Khám phá dữ liệu (Exploratory data)

Ở đây chúng ta có thể thấy dữ liệu bị missing value tại question1 (1) và question2(2). Số lượng missing không đáng kể, vì thế chúng ta sẽ xử lý bằng cách xóa row dữ liệu bị missing.

```
data.isnull().sum()

id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
dtype: int64
```



Ở đây chúng ta có thể thấy tỷ lệ số câu hỏi trùng lặp và không trùng lặp là không cân bằng, cụ thể trùng lặp (is_duplicate) chiếm 37%, không trùng lặp (non_duplicate) chiếm 63%.

Duplicate Question Pairs

Non-Duplicate Question Pairs

Ở đây chúng ta có thể có những từ hay xuất hiện ở cả trong những câu trùng lập và cả những câu không trùng lặp (best way, get…), chúng ta sẽ có thể cân nhắc loại bỏ những từ này ra khỏi tập từ của chúng ta.

4. Xử lý dữ liệu (Data Processing)

a. Tạo một vài thuộc tính số cơ bản

- freq_qid1: Số lần xuất hiện qid1
- freq_qid2: Số lần xuất hiện qid1
- q1len: chiều dài của question1
- q2len: chiều dài của question2
- q1_n_words: số lượng từ trong question1
- q2_n_words: số lượng từ trong question1
- word_Common: Số từ unique hay gặp trong question1 và question2
- word_Total: tổng số từ trong question1 và question2
- word_share: (word_common)/(word_Total)
- freq_q1+freq_q2: tổng số lần xuất hiện của qid1 và qid2
- freq_q1-freq_q2: hiệu trị tuyệt đối lần xuất hiện của qid1 và qid2
- cosin similarity: mức độ tương đồng của question1 và question2 bằng độ đo cosin

	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	word_Total	word_share	freq_q1+q2	freq_q1-q2
0	0	1	1	66	57	14	12	10.0	23.0	0.434783	2	0
1	0	4	1	51	88	8	13	4.0	20.0	0.200000	5	3

b. Làm sạch văn bản (clean text)

Sau khi tạo một vài đặc trưng số cơ bản, tiếp theo chúng ta sẽ clean text

- Removing html tags
- Word repetition
- Removing Punctuations
- Performing Lemmatization
- Removing Stopwords
- Expanding contractions etc.

```
=====BEFORE CLEANING TEXT=====:
 Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say
 about me? I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What
 does this say about me?
=====AFTER CLEANING TEXT=====:
 astrology capricorn sun cap moon cap rising say im triple capricorn sun moon
 ascendant capricorn say
```



```
=====TOKENIZATION=====:
['astrology', 'capricorn', 'sun', 'cap', 'moon', 'cap', 'rising', 'say', 'im', 'triple',
'capricorn', 'sun', 'moon', 'ascendant', 'capricorn', 'say']
```

c. Vecto hóa (Vectorization)

Trước khi chúng ta thực hiện vecto hóa các tokens để thuật toán học máy có thể học, chúng ta hãy xem qua một lượt các từ có tần xuất xuất hiện nhiều ở cả các cặp câu hỏi duplicate và non-duplicate.

Ở đây chúng ta có thể thấy từ ‘best’, ‘get’, ‘way’, ‘people’, ‘life’ xuất hiện với tần xuất rất lớn và xuất hiện đều ở cả hai loại cặp câu hỏi, vì thế chúng ta sẽ cân nhắc liệt các từ này vào stopwords để loại bỏ ra khỏi bộ vocabulary của chúng ta.

Bên cạnh đó, những từ xuất hiện với tần xuất <20 lần, chúng ta có thể cân nhắc loại luôn vì số lần xuất hiện quá ít thì nhiều khả năng là sẽ không mang lại nhiều ý nghĩa.

Sau khi đã cân nhắc bỏ đi những từ không thực sự mang lại nhiều thông tin, chúng ta sẽ tiến hành vecto hóa, trong dự án này, em sử dụng 2 kỹ thuật phổ biến là Bag-of-Word và TF-IDF với n-gram = (1,2), bên cạnh đó em còn sử dụng thêm đặc trưng về tần số xuất hiện để so sánh.

```
{('best', 0): 27067,
 ('best', 1): 25766,
 ('get', 0): 18908,
 ('like', 0): 14605,
 ('good', 0): 13110,
 ('india', 0): 12223,
 ('people', 0): 11846,
 ('would', 0): 11670,
 ('get', 1): 11216,
 ('one', 0): 10225,
 ('india', 1): 9734,
 ('make', 0): 9547,
 ('way', 0): 9449,
 ('way', 1): 9184,
 ('quora', 1): 9168,
 ('difference', 0): 8378,
 ('people', 1): 8013,
 ('year', 0): 7587,
 ('time', 0): 7482,
 ('much', 0): 7404,
 ('use', 0): 7131,
 ('know', 0): 6899,
 ('would', 1): 6414,
 ('thing', 0): 6370,
 ('money', 1): 6338,
 ('job', 0): 6256,
 ('many', 0): 6248,
 ('work', 0): 6216,
 ('someone', 0): 6191,
 ('want', 0): 6130,
 ('life', 1): 6102,
 ('make', 1): 6089,
 ('life', 0): 5980,
 ('new', 0): 5932,
 ('question', 1): 5885,
 ('note', 1): 5860,
 ('one', 1): 5810,
 ('mean', 0): 5762,
 ('good', 1): 5708,
```

5. Huấn luyện mô hình

a. Chọn mô hình

Sau khi đã tiến hành các bước xử lý dữ liệu (feature engineering), tiếp theo em sẽ chọn một mô hình để huấn luyện, trong dự án này em chọn RandomForestClassifier(n_estimators = 100, max_features = 'log2') vì thuật toán này cho ra kết quả khả quan nhất trong các thuật toán mà em đã thử nghiệm, cùng với các hyperparameter em đã chọn được từ phương pháp tìm kiếm GridSearchCV().

b. Kết quả

Accuracy for Frequency :

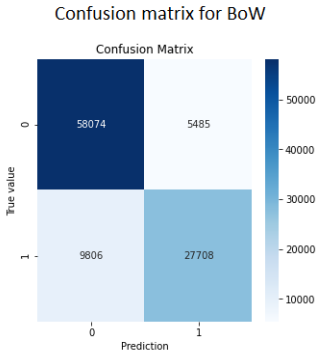
	precision	recall	f1-score	support
0	0.84	0.89	0.86	63559
1	0.80	0.71	0.75	37514
accuracy			0.82	101073
macro avg	0.82	0.80	0.81	101073
weighted avg	0.82	0.82	0.82	101073

Accuracy for Bag of Word :

	precision	recall	f1-score	support
0	0.86	0.92	0.89	63559
1	0.84	0.75	0.79	37514
accuracy			0.85	101073
macro avg	0.85	0.83	0.84	101073
weighted avg	0.85	0.85	0.85	101073

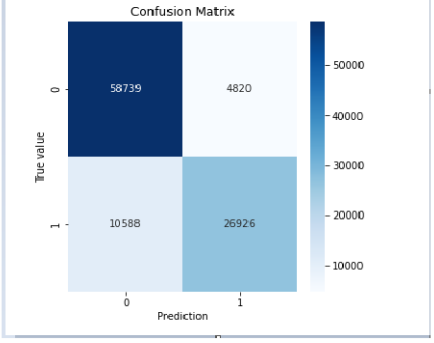
Accuracy for TF-IDF :

	precision	recall	f1-score	support
0	0.86	0.91	0.89	63559
1	0.84	0.74	0.79	37514
accuracy			0.85	101073
macro avg	0.85	0.83	0.84	101073
weighted avg	0.85	0.85	0.85	101073



Accuracy for All feature:

	precision	recall	f1-score	support
0	0.85	0.92	0.88	63559
1	0.85	0.72	0.78	37514
accuracy			0.85	101073
macro avg	0.85	0.82	0.83	101073
weighted avg	0.85	0.85	0.84	101073



c. Kết luận

Từ kết quả trên, chúng ta có thể rút ra một vài nhận xét như sau:

- Việc bỏ đi các từ có thể gây nhầm lẫn cho thuật toán làm cho thuật toán nhận diện lớp dương (is_duplicate) tốt hơn rất nhiều, từ 70% trước đó lên 75% (recall). Đồng thời cũng tang giá trị F1-score cho cả 2 lớp và Accuracy lên 85%. Tuy nhiên, thì kết quả này cũng có 1 phần đóng góp từ việc scaling và thay đổi n_gram từ (1,1) lên (1,2).
- Trong 3 phương pháp vecto hóa, thì Bag-of-Word cho kết quả khả quan nhất, TF-IDF thì cũng tương tự, điều này đến từ nguyên nhân là các đoạn text đầu vào của chúng ta không được dài lắm (điều mà TF-IDF có thể phát huy sở trường của mình đó là đánh giá mức độ quan trọng của từ dựa trên số lần xuất hiện trong đoạn này nhưng ít xuất hiện trong đoạn khác).
- Việc kết hợp tất cả các đặc trưng có vẻ như không mang lại nhiều tác dụng trong bài toán này, điều này có thể xuất phát từ việc các đặc trưng tuy xử lý bằng các phương pháp khác nhau, nhưng nhìn chung thông tin mà các từ mang lại thì đều giống nhau, vì thế thuật toán không học được gì mới từ việc gộp các đặc trưng lại.

- Việc tang n_gram lên (1,3) khiến cho số lượng đặc trưng lên đến hơn 2 triệu, tuy nhiên thì kết quả cho ra vẫn không cải thiện so với trước đó n_gram=(1,2) mặc dù huấn luyện lâu hơn rất nhiều, chứng tỏ với các phương pháp truyền thống BoW hay TFIDF thì với n_gram = (1,2), thuật toán đã học hết những gì cần học, vì thế khi chúng ta tang n_gram lên (1,3) thì cũng không mang lại nhiều ý nghĩa.

d. Cải tiến

Hầu hết các vấn đề về Học Máy nói chung đều đến từ 2 vấn đề: thứ nhất là thuật toán tệ, thứ 2 là dữ liệu xấu. Vì vậy, để có thể cải tiến thêm về chất lượng của bài toán, chúng ta có thể tiếp cận theo các hướng:

- Nghiên cứu sâu hơn nữa về dữ liệu để có thể có được nhiều thông tin nhất, sử dụng các kỹ thuật vecto hóa nâng cao hơn như Word2vec, Glove, Fasttext… nhằm tăng tính biểu diễn cho từ.
- Thử nghiệm với các mô hình phức tạp hơn, với hy vọng rằng có thể tang chất lượng mô hình nhưng không bị quá khớp.
- Kết hợp cả hai.