Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin
Lớp Khoa học Tài năng 2017

Case Study Report

News Sarcasm Detection Through Titles

(Nhận diện tin tức châm biến thông qua tiêu đề)

Giáo viên hướng dẫn: Huỳnh Thị Thanh Thương

Sinh viên nghiên cứu:

Nguyễn Trường Phát 17520880

Vũ Đình Vi Nghiệm 17520805

Nguyễn Chí Bảo 17520271

Mục lục

1. Phát biểu bài toán

1.1. Mục đích

Xây dựng một mô hình dự đoán tin tức châm biếm thông qua tiêu đề của bài báo.

1.2. Đầu vào

- Một bộ dữ liệu gồm 26709 điểm được crawl từ 2 trang báo HuffingtonPost và TheOnion. Dữ liệu gồm 3 phần: Tiêu đề bài báo, đường dẫn tới bài báo và nhãn của bài báo là có châm biếm (1) hay không (0).

1.3. Đầu ra

- Một mô hình thể hiện sự ánh xạ của X->Y, của nội dung tiêu đề (X) và nhãn thể hiện việc có châm biếm hay không (Y).

2. Xây dựng mô hình từ tập huấn luyện

2.1. Xử lý chuỗi

Trong dữ liệu tồn tại rất nhiều kí tự lạ, như kí tự đặc biệt $%. Chúng ta sẽ loại bớt chúng ra trước khi tokenize để không làm ảnh hưởng tới mô hình của chúng ta

2.1.1. Loại bỏ và thay thế kí tự unicode

Trong dữ liệu tồn tại những kí tự unicode như là \u00a9, \u201d,.. chúng ta sẽ loại bỏ bằng cách mã hóa chúng ra ascii và giải mã về unicode lại.

Ví dụ: "donald trump wouldn't have had the ready cash to self-finance entire campaign **\u2014** analysis."

2.1.2. Xóa những kí tự đặc biệt

Trong dữ liệu tồn tại những kí tự đặc biệt như "$%?". Chúng ta sẽ loại bỏ chúng ra vì nó sẽ làm nhiễu dữ liệu text của chúng ta và chúng không mang nhiều ý nghĩa.

Ví dụ: "Actually**,** cnn**'**s jeffrey lord has been **'**indefensible**'** for a while."

2.1.3. In thường tất cả các kí tự

Trong dữ liệu tồn tại có kí tự in hoa và in thường lẫn lộn, việc không loại bỏ chúng sẽ làm chúng bị tính là các từ khác nhau.

2.1.4. Xóa dấu ở các kí tự

Những kí tự sau khi loại bỏ và thay thế kí tự khác unicode (2.2.1) thì một số kí tự có dấu. Chúng ta sẽ chuyển nó về dạng bình thường

Ví dụ: beyoncé -> beyonce .

2.1.5. Loại bỏ stopwords

Những từ xuất hiện nhiều nhất là stopwords và cũng không mang nhiều ý nghĩa nên ta sẽ cân nhắc xóa bỏ chúng ra khỏi dataset. Lưu ý rằng bộ corpus của chúng ta rất ít (26k từ), nên chúng ta sẽ không xóa stopwords dựa trên số lần mà nó xuất hiện, vì có những từ là stopwords

nhưng cũng không được sử dụng nhiều. Thay vào đó, chúng ta sẽ sử dụng bộ stopwords có sẵn trên mạng.

Ví dụ: top 50 từ của data chúng ta là: *'to', 'of', 'the', 'in', 'for', 'a', 'on', 'and', 'with', 'is', 'new', 'trump', 'man', 'from', 'at', 'about', 'you', 'this', 'by', 'after', 'be', 'how', 'out', 'it', 'that', 'as', 'up', 'not', 'are', 'your', 'what', 'his', 'he', 'just', 'who', 'us', 'has', 'will', 'more', 'all', 'report', 'into', 'one', 'why', 'have', 'area', 'over', 'donald', 'says', 'woman'.*

2.2. Tokenization

Tokenization là phương pháp tách 1 câu ra thành các đơn vị từ nhỏ nhất, phục vụ cho việc định lượng, tính toán trên phạm vi từ, ví dụ tính tf-idf, bag of words,.. Tokenization cũng có nhiều cách tách khác nhau.

Ví dụ: He didn't arrive. -> ["He", "didn't", "arrive".] hoặc ["He", "did", "n't", "arrive", "."]

2.3. Biểu diễn dữ liệu văn bản

2.3.1. Số hóa dữ liệu văn bản

Để ứng dụng được trên mô hình logistic regression (và các mô hình học máy khác), ta phải biểu diễn được văn bản dưới dạng số.

Ví dụ:

	I	don't	like	fish
"I don't like fish"	0	1	0	0
"I like fish"	0	0	0	0

2.3.2. TFIDF - Word Embedding (Nhúng)

Có rất nhiều thuật toán để đưa dữ liệu văn bản thành số. Ở đây chúng ta sử dụng thuật toán tf-if để embed (nhúng) các từ này về dạng số.

Tf-idf là tích của 2 số liệu thống kê: Term Frequency (tf) và Inverse Document Frequency (idf). Trong đó:

Term Frequency

TF thể hiện tần suất xuất hiện của từ đó trong văn bản mà chúng ta đang xét. Cách đơn giản nhất là đếm (raw count). Ngoài ra còn có rất nhiều cách chọn tiếp cận khác nhau.

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

Inverse Document Frequency

IDF thể hiện số lượng thông tin của một từ. Một từ xuất hiện càng nhiều trong nhiều văn bản khác nhau thì sẽ không mang nhiều ý nghĩa đặc trưng nào, điển hình như ví dụ "I don't like fish" và "I like fish" ở mục **2.3.1**, chỉ có từ "don't" giữ nhiều ý nghĩa nhất vì các từ còn lại đều xuất hiện ở cả 2 văn bản

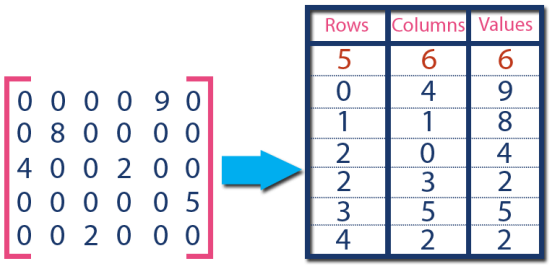
$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

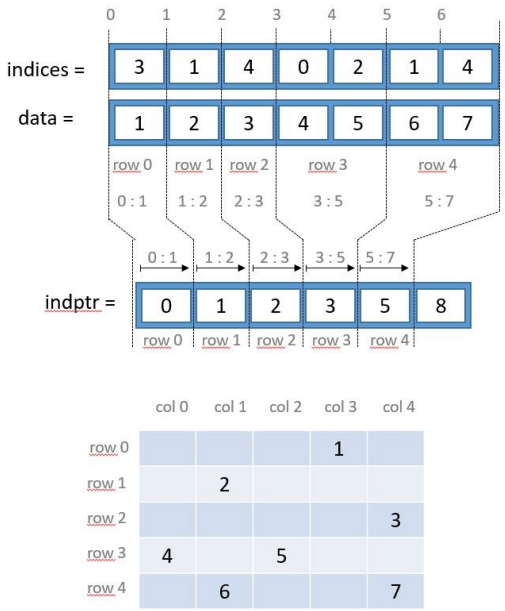
Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Ngoài ra, còn có các các biến thể khác của TFIDF

2.3.2. Làm việc ma trận thưa

Mục tiêu của chúng ta là biểu diễn được các dòng dữ liệu theo dạng ma trận để chúng ta để có thể sử dụng các phép toán trên ma trận để gradient descent và hơn thế nữa. Tuy nhiên nếu chúng ta sử dụng ma trận dạng thường để lưu trữ trong bài toán text embedding thì sẽ rất tốn tài nguyên, vì đa số các giá trị trong ma trận này là 0. Python có cung cấp kiểu dữ liệu sparse matrix (ma trận thưa), chúng ta chỉ cần lưu trữ những vị trí trong ma trận có gia trị khác 0, các vị trí còn lại sẽ hiển nhiên bằng 0.





Cấu trúc của ma trận thưa được cài đặt trong scipy.sparse.csr_matrix

2.4. Mô hình học

2.4.1. Hồi quy tuyến tính (Logistic Regression)

2.4.1.1 Gradient Descent

"Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó. Ví dụ như các hàm mất mát trong hai bài Linear Regression và K-means Clustering. Nhìn chung, việc tìm global minimum của các hàm mất mát trong Machine Learning là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán".- **MachineLearningCoBan.com**

2.4.1.2. Batch Gradient Descent

Bước 0: Thống nhất số chiều (shape) của các ma trận

Shape of W = (n,1) # n đã tính cột 1 được thêm vào

Shape of X = (m,n)

Shape of Y = (m,1)

Với m là số điểm dữ liệu, n là số đặc trưng (tính cả bias) của mô hình

Bước 1: Bắt đầu với 1 giả thuyết

$$H(x) = X * W$$

Ở "cảnh giới" này, ta tin rằng hàm đầu ra H(x) sẽ là 1 hàm tuyến tính phụ thuộc vào các biến đầu vào X

- Ban đầu ta khởi tạo bộ trọng số (weight) với những giá trị bất kì (ngẫu nhiên)

Bước 2: Chọn hàm độ lỗi

Ở đây chúng ta chọn hàm độ lỗi là Mean Squared Error

$$\mathcal{L}(w) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2$$

Bước 3: Tính đạo hàm (gradient) của bộ weights

Chúng ta tính đạo hàm của bộ trọng số để tối ưu hàm độ lỗi bên trên.

$$Grad(W) = \frac{1}{m}X^T(\hat{Y} - Y)$$

Bước 4: Cập nhật trọng số với đạo hàm (gradient)

Ở đây chúng ta cập nhật bộ trọng số thông qua đạo hàm (gradients), với "hi vọng" là hàm loss sẽ giảm sau **vài trăm lần** chạy bước này Với learning rate là tốc độ học

$$W = W - \alpha * \frac{\partial L}{\partial W}$$

Bước 5: "Bóp thắng"

Dừng lại khi thấy bộ trọng số không giảm nữa, hoặc hàm loss tạm chấp nhận được.

2.4.1.3. Stochastic Gradient Descent

Gần giống với Batch Gradient Descent, chỉ khác là phần tính loss độ lỗi chúng ta chỉ cần bỏ điểm dữ liệu vào khi tính

$$\mathcal{L}(w) = \frac{1}{2}(\hat{y} - y)^2$$

Vì lý do đó đạo hàm cũng khác đi 1 chút..

$$Grad(W) = X^T(\hat{Y} - Y)$$

2.4.1.4. Mini-Batch Gradient Descent

Mini-Batch cũng tương tự 2 dạng còn lại, chỉ khác là ta sẽ bỏ m (số dữ liệu/batch) vào mỗi lần tính và cập nhật trọng số.

2.4.1.5. Normal Equation (Phương trình chuẩn)

"Phương trình chuẩn để tính được ước lượng bình phương nhỏ nhất của các thông số trong một phân tích hồi quy, bao gồm tổng của các bình phương và tích chéo của các biến số trong phương trình hồi quy." -saga.vn

Áp dụng phương trình này, ta sẽ tìm được bộ tham số tối ưu của mô hình Machine Learning của chúng ta

$$W = (X^T X)^{-1} X^T y$$

Ở đây vì ma trận của chúng ta rất lớn. Nên chúng ta sẽ **không** sẽ dụng phương pháp này vì rất tốn tài nguyên

2.4.2. Căn chỉnh siêu tham số (Hyperparameter Tuning)

Chúng ta có 1 siêu tham số duy nhất ở trong mô hình Logistic Regression là ngưỡng kích hoạt, ở đây ngưỡng mặc định của ta là 0.5, ta sẽ thử sử dụng các ngưỡng khác xem kết quả có khả quan hơn không.

2.4.3. Phương pháp đánh giá (Metrics)

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

2.4.3.1. Accuracy

Ví dụ giải thích: Trong nguyên tập dataset, chúng ta đã dự đoán đúng bao nhiêu %

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

2.4.3.2. Precision

Ví dụ giải thích: Trong những điểm dữ liệu chúng ta đã gán nhãn "sarcastic", có bao nhiêu điểm thật sự "sarcastic"

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

2.4.3.3. Recall

Ví dụ giải thích: Trong những điểm "sarcastic", ta đã gán những "sarcastic" bao nhiêu điểm

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

2.4.3.4. Đánh đổi giữa Precision và Recall

Tùy vào những bài toán cụ thể thì chúng ta sẽ chọn tối ưu precision hay recall và ngược lại. Precision cao tăng độ chắc chắn mỗi khi chúng ta gán nhãn dương tính, ở đây là "sarcastic". Đối với những bài toán cần độ chính xác kiểu "chắc kèo" khi gán nhãn dương tính thì ta chọn tối ưu Precision.

Ví dụ: Việc ra lệnh một người thì sẽ thiên về 1 bài toán Precision Cao hơn, vì chúng ta cần chắc chắn rằng người bị ra lệnh bắt có khả năng cao là người phạm tội, chứ không phải bắt hết còn hơn bỏ sót.

Recall cao đảm bảo rằng chúng ta không bỏ lỡ bất kì

2.4.3.5. F1-score

F1 score dựa trên trung bình cộng điều hòa (harmonic mean) giữa Precision Score và Recall Score, chỉ số này sẽ giúp mình tìm được precision recall vừa tốt nhưng không bị đánh đổi 1 trong 2 quá nhiều

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

3. Thực nghiệm đánh giá

Phương thức : Sử dụng cả thư viện và code tay

Tự code (From scratch) [Level 1]

- Text Preprocessing

- TFDIF Embedders

Sử dụng thư viện [Level 2]

Lý do sử dụng thư viện: Logistic Regression em đã làm kì case study trước và thấy cũng không lý do gì để code lại và cảm thấy khá mất thời gian (em làm bài case study này gần như full-time và vẫn cảm thấy không đủ thời gian 😔). Thay vào đó, em sẽ dành thời gian đó cho xây dựng bộ embedders chỉnh chu hơn, thử nghiệm nhiều mô hình và hoàn chỉnh (fine-tune) cho các mô hình.

- sklearn.linear_model.LinearRegression

- sklearn.naive_bayes.MultinomialNB

- sklearn.metrics.accuracy_score/precision_score/recall_score/f1_score

3.1 Mô hình chưa hoàn chỉnh

Remove stopwords					
	TFIDF Method	Accuracy	Precision	Recall	F1 Score
Naive Bayes	natural	0.7749	0.7183	0.7973	0.7557
	log	0.7752	0.7186	0.7976	0.7561
	boolean	0.7757	0.7189	0.7986	0.7567
	augmented	0.7748	0.7178	0.7980	0.7558
Logistic Regression	natural	0.7854	0.7816	0.7058	0.7418
	log	0.7852	0.7815	0.7055	0.7415
	boolean	0.7834	0.7801	0.7020	0.7390
	augmented	0.7842	0.7786	0.7068	0.7410
Not remove stopwords					
	TFIDF Method	Accuracy	Precision	Recall	F1 Score
Naive Bayes	natural	0.8027	0.7735	0.7751	0.7743
	log	0.8025	0.7732	0.7751	0.7742
	boolean	0.8013	0.7719	0.7737	0.7728
	augmented	0.8030	0.7746	0.7741	0.7743
Logistic Regression	natural	0.8277	0.8271	0.7655	0.7951
	log	0.8275	0.8268	0.7655	0.7950
	boolean	0.8301	0.8322	0.7652	0.7973
	augmented	0.8286	0.8289	0.7655	0.7960

Có 1 điều rất lạ trong mô hình chúng ta là mô hình không bỏ stopwords cho ra kết quả khả quan hơn rất nhiều so với mô hình bỏ.

Điều này có thể được giải thích như sau:

- **Thiếu sót stopwords**: Bộ stopwords của chúng ta chưa thật sự tốt, có nhiều từ còn thiếu sót (chẳng hạn như từ will vẫn còn frequency rất cao), sẽ trở nên thống trị (dominant) trong mô hình.

- **Biến đổi về nghĩa**: Một số từ stopwords vẫn mang rất nhiều thông tin trong 1 câu. Đơn cử như từ “not”, “I’m not a cat” and “I’m a cat”, đây là 2 câu hoàn toàn khác nhau.

Giải pháp: Cần chọn 1 bộ stopwords khác hoàn chỉnh hơn

Not Remove stopwords - Not remove Special Char ($%#!)					
	TFIDF Method	Accuracy	Precision	Recall	F1 Score
Naive Bayes	natural	0.8016	0.7562	0.8055	0.7800
	log	0.8017588314	0.7564	0.8055	0.7802
	boolean	0.8022	0.7562	0.8075	0.7810
	augmented	0.8049	0.7595	0.8096	0.7837
Logistic Regression	natural	0.8363	0.8281	0.7891	0.8081
	log	0.8368	0.8285	0.7898	0.8087
	boolean	0.8374	0.8302	0.7891	0.8091
	augmented	0.8374	0.8297	0.7898	0.8092

Bất ngờ hơn cả, khi không xóa các kí tự đặc biệt kết quả lại cho ra tốt hơn. Điều này có thể giải thích như sau:

- **Biến đổi về nghĩa:** từ như son’s sẽ bị chuyển về sons, cat’s thành cats -> gây lẫn lộn nghĩa với những từ đã có trong dataset mặc dù ‘s ở đây mang nghĩa sở hữu.

- **Phức tạp hóa từ:** Những từ có dấu gạch nối như 18-dragon sẽ trở thành 18dragon thay vì tách ra thành 18 dragon thì sẽ mang nhiều ý nghĩa hơn rất nhiều và thật sự đồng bộ với từ “dragon” đã có sẵn trong từ điển.

Giải pháp: Đối với những từ mang tính sở hữu như ‘s thì nên để nguyên để tạo sự khác biệt cho những từ khác về nghĩa. Với những dấu gạch nối, nên tokenize thành 2 từ khác nhau thay vì 1.

Ngoài ra, ta có thể thấy rằng, logistic regression thì thiên về precision hơn còn naïve bayes thì thiên về recall hơn. Chúng ta sẽ cân nhắc khi chọn giữa 2 mô hình tùy vào bài toán mà chúng ta đang nghiên cứu.

3.2 Mô hình hoàn chỉnh (fine-tuned model)

Sau khi làm thí nghiệm thử các mô hình trên thì em sẽ chọn ra **3 mô hình tốt nhất** để căn chỉnh siêu tham số (hyperparameter tuning). Kết quả cho ra như sau.

Fine-Tuned Logistic Regression with Augmented TFIDF					
	TFIDF Method	Accuracy	Precision	Recall	F1 Score
Logistic Regression	log	0.8375	0.8203	0.8041	0.8121
	boolean	0.8384	0.8273	0.7962	0.8115
	augmented	0.8383	0.8259	0.7980	0.8117

Việc mình chọn mô hình nào là "tốt nhất" phụ thuộc vào việc mình muốn điều gì:

- Log TFIDF cho ra F1 score tốt nhất, nếu mình quan tâm tới dán nhãn các điểm dương tính ('is_sarcasm'=1) thì chọn Log

- Boolean cho ra accuracy cao nhất, nếu mình quan tâm tới độ chính xác trên tất cả các nhãn 0 và 1 thì chọn Boolean

- Augmented cho ra F1 score cao hơn Boolean và accuracy score cao hơn log, có thể nói là 1 mô hình có hiệu nặng "giữa giữa".

3.3. Mô hình sử dụng hoàn toàn thư viện có sẵn

Các công cụ đã sử dụng:

- sklearn.linear_model.LinearRegression

- sklearn.naive_bayes.MultinomialNB

- sklearn.metrics.accuracy_score/precision_score/recall_score/f1_score

- sklearn.feature_extraction.text.CountVectorizer/TfidfVectorizer (Có sẵn stopwords remover)

Remove stopwords - sklearn embedders					
	Method	Accuracy	Precision	Recall	F1 Score

Naive Bayes	CountVectorizer	0.8019	0.7921	0.7520	0.7715
	TfidfVectorizer	0.7904	0.8410	0.6521	0.7346
Logistic Regression	CountVectorizer	0.7912	0.7963	0.7128	0.7523
	TfidfVectorizer	0.7885	0.8084	0.6873	0.7430
Not Remove stopwords - sklearn embedders					
	Method	Accuracy	Precision	Recall	F1 Score
Naive Bayes	CountVectorizer	0.8480	0.8540	0.7939	0.8229
	TfidfVectorizer	0.8189	0.9019	0.6652	0.7657
Logistic Regression	CountVectorizer	0.8435	0.8374	0.8043	0.8205
	TfidfVectorizer	0.8420	0.8320	0.8080	0.8198

- Naïve Bayes với CountVectorizer cho kết quả cực kì ưu việt so các mô hình còn lại với **84.8% Accuracy** và **0.82 F1 Score**, tính cả accuracy và F1 score.

- Ở đây chúng ta có thể thấy, sau khi xóa stopwords bằng thư viện thì cũng không khá hơn nhiều so với code tay, kết quả cho tốt hơn code tay những vẫn tệ hơn không xóa stopwords.

3.4 Những điều cần cải thiện

- Cần lựa chọn kĩ càng hơn những stopwords cần xóa ra khỏi tập dataset

- Nên xử lí cân nhắc hơn phần xóa kí tự đặc biệt trong các trường hợp đặc biệt như ‘s và dấu gạch nối -.

- Nếu có nhiều thời gian hơn thì sẽ chú trọng hơn phần stemming, đưa 1 từ về từ gốc của nó. (dogs -> dog, ran -> run, etc)

4. Lập trình cài đặt

i) Phần lập trình cài đặt em đã để ở các file khác nhau, tuy nhiên chúng rất dễ đọc và đã được em đưa thành các file .pdf khác nhau để cô coi cách em làm.

ii) 7 bài experiment .ipnb sử dụng nhiều thủ thuật khác nhau để xử lí cũng như mô hình hóa bài toán này để hoàn thành được bài viết này.

Gồm:

./Experiment-1-Sarcasm-Detection-Removing-Stopwords.ipynb

./Experiment-2-Sarcasm-Detection-NotRemoving-Stopwords.ipynb

./Experiment-3-Sarcasm-Detection-NotRemovingStopwords-NotRemovingSpecial.ipynb

./Experiment-4-Sarcasm-Detection-Hyper-Parameter-Tuning-Augmented.ipynb

./Experiment-4-Sarcasm-Detection-Hyper-Parameter-Tuning-Augmented.ipynb

./Experiment-5-Sarcasm-Detection-Hyper-Parameter-Tuning-Boolean.ipynb

./Experiment-6-Sarcasm-Detection-Hyper-Parameter-Tuning-Log.ipynb

./Experiment-7-Sarcasm-Detection-Hyper-Parameter-sklearn.ipynb

iii) Bộ module **MyNLPTool** tự viết nằm ở "/MyNLPTool", bộ này bao gồm các công cụ phục vụ cho việc xử lí trừu tượng hóa và plot dữ liệu văn bản. Đặc biệt bộ TFIDF Embedder cho phép embed (nhúng) dữ liệu văn bản bằng nhiều kiểu TFIDF khác nhau (log, augmented, boolean)

./MyNLPTool

----/FilePickling.py || *Để lưu và tải các object.*

----/TextVisualizer.py || *Để plot số lượng xuất hiện các từ trong toàn bộ văn bản*

----/TextPreprocessor || *Xử lí dữ liệu văn bản (Xóa dấu, in thường, xóa stopwords, etc.)*

----/WordEmbedders || *Nhúng dữ liệu văn bản bằng TFIDF nhiều kiểu khác nhau*

5. Tham khảo

https://en.wikipedia.org/wiki/Tf%E2%80%93idf (tf-idf, Wikipedia)

https://www.researchgate.net/post/Does_Pre-processing_step_Remove_Stop_Word_effect_Sentiment_Analysis_result