

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA: KHOA HỌC MÁY TÍNH**



BÁO CÁO ĐỒ ÁN MÔN MÁY HỌC

ĐỀ TÀI: DETECTION FAKE NEWS

Giảng viên: Phạm Nguyễn Trường An

: Lê Đình Duy

Sinh viên: Nguyễn Văn Thái An - 17520216

: Lê Hoàng Khánh - 17520623

: Ngô Anh Vũ – 17522172

: Nguyễn Hữu Nghĩa - 18521144

Tp HCM, tháng 1 năm 2021

Tên	Nhiệm vụ
Lê Hoàng Khánh	Tìm kiếm Tài liệu, Code
Nguyễn Văn Thái An	Tìm kiếm Tài liệu, Chỉnh sửa báo cáo, PowerPoint, Code, Trình bày và thuyết trình
Ngô Anh Vũ	Viết báo cáo

Abstract: Tin giả và các trò lừa đã tràn ngập trong nhiều thập kỷ xung quanh cuộc sống của chúng ta. Kể từ khi có sự xuất hiện của Internet, vấn đề này ngày càng nghiêm trọng hơn. Một số phương tiện truyền thông xã hội và báo chí cạnh tranh với nhau bằng cách xuất bản hoặc phát hành các tin tức rác để thu được nhiều người đọc hơn vì lợi nhuận, đây là một phần của chiến tranh tâm lý. Mục đích chính của họ là thu được lợi ích từ việc nhấp chuột của người dùng. Kết quả là có những chủ đề độc hại được đưa vào não người đọc, mọi người trên khắp thế giới đang đối mặt với thách thức to lớn này.

Ý tưởng: văn bản đã cho trong một mẫu tin cụ thể là thật hay giả. Trước hết, ta cần thực hiện một số bước xử lý trước để làm sạch dữ liệu văn bản và sau đó áp dụng trích xuất tính năng trên chúng. Cuối cùng, chúng em sẽ train mô hình của mình và xem xét độ chính xác.

Keywords— LSTM, word-embedding, fake news (keywords).

I. INTRODUCTION

Tin tức giả là tình trạng của việc thiếu thông tin hoặc thông tin sai lệch được lan truyền nhanh chóng trên các phương tiện truyền thông xã hội chính thống. Đặc biệt, như chúng em đã quan sát, tin tức giả tràn ngập trong thời gian bùng phát COVID-19 ở một số quốc gia và cuộc bầu cử năm 2016 ở Hoa Kỳ. Trong một số tình huống, người ta phát hiện ra rằng số lượng tin tức giả được chia sẻ nhiều hơn số tin tức chính xác. Hơn nữa, nó được hiểu bởi thực tế là khi một tin tức giả được thu thập, người khác sẽ ngay lập tức chia sẻ nó. Do đó, tin tức giả mạo trở thành một vấn đề nghiêm trọng, có nguy cơ cao dẫn đến việc cư xử sai, hiểu sai, thậm chí bạo lực đối với nguồn gốc báo cáo của tác giả.


Người ta cũng nhận thấy rằng tin nhắn rác và tin giả có những điểm giống nhau. Họ sử dụng những cách lôi kéo để thu hút ý kiến của người đọc. Hầu hết chúng đều mắc lỗi ngữ pháp và chúng chỉ cho người đọc thấy một nửa sự thật. Vì cả hai phương tiện truyền thông đều chia sẻ các thuộc tính tương tự như vậy, chúng ta có thể sử dụng các phương pháp tương tự để phát hiện tin tức giả một cách chính xác. Một cách để giải quyết tin tức giả là phân loại thủ công tin tức là thật hay giả. Nó có vẻ như là một giải pháp đơn giản nhất nhưng nó không thực tế với hàng triệu tin tức cập nhật liên tục hàng ngày, do đó rất khó để gán nhãn thủ công. Do đó, cần phải tìm kiếm một giải pháp kỹ thuật thực dụng để làm được điều tương tự. Phương pháp được đề xuất trong nghiên cứu này là khai thác sự tiến bộ trong máy học. Để làm điều này, mô hình phân loại đã được đào tạo với các thuật toán máy học khác nhau để gán nhãn dữ liệu. Các kết quả từ nghiên cứu đã chỉ ra mạng nơ-ron là cách mà tin tức đạt được độ chính xác cao nhất. Trong bài báo này, chúng em sẽ hướng tới việc lấy dữ liệu trong bộ thử nghiệm làm văn bản đầu vào và phân loại nó.

II. DATA

Mô tả ngắn gọn về tập dữ liệu (Brief description of the dataset)



Nguồn : <https://www.kaggle.com/clmentbisailon/fake-and-real-news-dataset>

Người thu thập data:



Clément Bisailon

Programmer consultant at Ma Gestion Plus
Montreal, Quebec, Canada
Joined a year ago · last seen 6 days ago

[Home](#) [Datasets \(1\)](#) [Code \(4\)](#) [Discussion \(8\)](#) [Followers \(12\)](#)

Cách thức mà tác giả thu thập data theo chúng em hiểu là tác giả sẽ collect trên các trang tin tức, chính trị và từ báo đài của Mỹ. Cụ thể trong bộ dữ liệu này chúng em thấy được đa số là ở trang REUTERS, dữ liệu tổng hợp lại trong file theo cách hiểu đơn giản của chúng em tác giả sẽ thu thập những title trước tiên và cho nó vào column title tiếp đến sẽ là phần text của title đó và đến subject và cuối cùng là ngày tháng của những tin tức đó. Và nguồn dữ liệu được thu thập ngày càng nhiều dựa trên cách tích góp từ mẫu tin tức lại với nhau.

Ở đây tác giả cũng đã phân lớp các mẫu data thành 2 phần fake và real. **Làm sao tác giả biết được fake hay real để mà phân lớp?** Trong đồ án này chúng em cũng rất đau đầu về câu hỏi này và cuối cùng có một data cho chúng em hiểu là làm sao tác giả có thể nhận biết đó là tin fake hay real mà phân lớp chúng. Tất nhiên tác giả sẽ không thể chắc chắn rằng đoạn tin tức đó là fake hay real thật sự để mà phân lớp. Nhưng có một data mà chúng em tìm hiểu được đó là tác giả có thể xác nhận lại mẫu tin tức đó bằng cách get expert. Vì những chuyên gia trong lĩnh vực này học sẽ có thể chắc chắn hơn và cũng có thể xác nhận với người thu thập dữ liệu về mẫu data mà tác giả cần xác nhận. Và theo một data khác thì cũng có thể chủ quan một chút về mặt tác giả khi tự mình thu thập nguồn dữ liệu lớn như vậy, và cũng tự phân lớp toàn bộ số dữ liệu này. Và đó cũng là một trong những vấn đề mà trong đồ án này chúng em chưa thể tối ưu hoá nó một cách tốt nhất có thể.

Tập dữ liệu của chúng em là tập hợp khoảng 40000 bài báo thô bao gồm tin tức giả và tin thật. Số lượng của mỗi nhãn dường như giống nhau. Nó đã được chia thành hai bộ dữ liệu riêng biệt gắn nhãn Fake và Real. Mỗi điểm dữ liệu trong tập dữ liệu này có bốn đặc điểm là tiêu đề, văn bản, chủ đề và ngày. Nội dung của những bài báo này tập trung chủ yếu vào tin tức chính trị Hoa Kỳ. Tập dữ liệu này, thu được trên Kaggle, chỉ là dữ liệu thô cần được lọc lại.

- **Tiền xử lý dữ liệu (Preprocessing data)**

Trong bước tiền xử lý dữ liệu: dữ liệu thô là một phần thông thường của tập dữ liệu này, chúng bao gồm nhiều tin giả mà chúng em phải làm sạch dữ liệu. Trước hết, chúng em sẽ kiểm tra các giá trị null trong tập dữ liệu của mình, sau đó xem xét tính năng nào là quan

trọng nhất để giữ chúng và xóa các bản ghi còn lại. Hơn nữa, chúng em cần thực hiện một số kỹ thuật tiền xử lý trước khi có thể bắt đầu thực hiện mô hình trong dự án của mình. [1] Dưới đây là một số kỹ thuật xử lý trước văn bản mà chúng em sẽ tiếp cận và đề cập đến vai trò của từng loại:

- **Lowercasing letters:** Để viết thường tất cả các chữ cái trong toàn bộ dữ liệu văn bản, đây là một trong những phương pháp xử lý trước văn bản hiệu quả nhưng dễ hiểu nhất. Bởi vì trong nhiều vấn đề NLP, chúng em nhận thấy rằng sự sai lệch lớn giữa đầu vào văn bản viết hoa và văn bản không viết hoa sẽ cho ra kết quả đầu ra khác nhau đáng kể (ví dụ: “USA” so với “usa”, “Many” so với “many”).
- **Stemming and Lemmatization:** Các phương pháp này dường như tương tự với nhau. Cả hai đều là những quy trình làm giảm bớt sự biến dạng trong từ ngữ về dạng ban đầu nhưng chúng vẫn có những điểm khác biệt. Stemming là sử dụng một quy trình cắt bỏ phần cuối của những từ đó với mong muốn đạt được mục tiêu này một cách chính xác, đôi khi các đơn vị dẫn xuất cũng bị loại bỏ. Đây là một ví dụ về các kỹ thuật này:

Original word	Stemmed or Lemmatization word
troubles	trouble
raised	raise
thought	think

- **Removing stop-words :** Các stop-words là một tập hợp các từ được sử dụng thường xuyên trong một ngôn ngữ cụ thể. Chúng không mang nhiều ý nghĩa trong toàn bộ văn bản và thường bị loại bỏ. Ví dụ về các từ dừng trong tiếng Anh là “am”, “is”, “are”, “it”. Một cách thực tế khi sử dụng phương pháp này là chúng em chỉ có thể tập trung vào những từ quan trọng. Hơn nữa, nó giúp chúng em cắt giảm số lượng các tính năng cho phép mô hình của chúng em có kích thước phù hợp. Chúng em sẽ

tải xuống các stop-words bằng bộ công cụ Ngôn ngữ Tự nhiên (NLTK) và sau đó loại bỏ chúng.

- **Noise removal** : Đó là việc loại bỏ một số đoạn văn bản có thể xâm nhập vào phân tích văn bản của chúng em. Đặc biệt, nhiều có thể là URL, một số chữ số thừa không liên quan đến phân tích hoặc dấu câu của chúng em, thậm chí là tập hợp các ký hiệu [! " # \$ % & ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~]. Vì thế, chúng em cần xóa chúng vì nó có thể tạo ra kết quả không nhất quán trong các tác vụ lưu của chúng em. Đây là một ví dụ về điều đó:

Từ gốc	Từ sau khi đã xử lý
@realDonaldTrump	realDonaldTrump
#(_)<) THAT'S /	THAT'S
stepsâ€	steps

Tokenization: Đây là quá trình chia văn bản đã cho thành các phần nhỏ hơn được gọi là tokens. Các tokens này sẽ được lưu trữ ở dạng tương tự như một “danh sách chuỗi”. Chúng em cũng sẽ sử dụng thư NLTK để thực hiện điều này trong đồ án của mình.

III. METHODS

A. Data Train:

Vì dữ liệu trong vùng Xử lý ngôn ngữ tự nhiên (NLP) là tổng dữ liệu văn bản thô mà máy tính không thể hiểu được. Do đó, chúng em cần chuyển chúng thành dạng biểu diễn số của các từ nắm bắt ý nghĩa, mối quan hệ ngữ nghĩa và các loại ngữ cảnh khác nhau mà chúng được sử dụng. Đó chính xác là định nghĩa của Word Embedding. Tóm lại, train trước embedding word là những cách nhúng đã được train trong một tác vụ được sử dụng để giải quyết một tác vụ tương tự khác. Trong đồ án này, chúng em sử dụng một kỹ thuật phổ biến là Word2vec. Đầu ra của Word2vec là một từ vựng trong đó mỗi từ có một vector gắn liền với nó, có thể được đưa vào mạng nơ-ron học sâu như LSTM.

B. Các mô hình dự đoán (Prediction Models):

Trong phần này, chúng em sẽ giới thiệu phương pháp để xây dựng bộ phân loại của chúng em và đánh giá chúng dựa trên một số chỉ số: accuracy, precision, recall. Chúng em sẽ đi sâu vào các lý thuyết chi tiết của LSTM dưới đây:

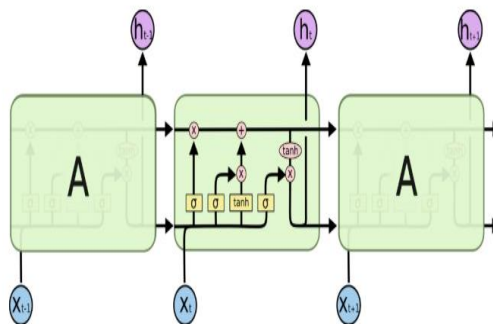
- **LSTM**

LSTM, viết tắt của Bộ nhớ dài ngắn hạn (Long short-term Memory), là một phần mở rộng được cải tiến của mô hình RNN (Recurrent Neural Network) nổi tiếng trước đây. Nó bao gồm ba cổng gọi là cổng input, cổng output và cổng forget. Cổng forget quyết định thông tin nào cần được loại bỏ khỏi cell state vì các tính năng này có giá trị hoặc trọng lượng kém. Cổng này có một chức năng sigmoid để đưa ra quyết định đó. Nó xem xét h_{t-1} và x_t trước đó để tính toán và xuất ra giá trị từ 0 đến 1 cho mỗi số ở cell state C_{t-1} . Giá trị gần đúng 1 thể hiện “nên giữ tính năng này”, trong khi giá trị gần đúng 0 thể hiện “nên loại bỏ tính năng này”. Do đó, nó có tác động rất lớn để có thể phù hợp nhất với mô hình này. Không giống như một số kiến trúc mạng nơ-ron liên kết với nhau, LSTM có các nơ-ron dạng vòng lặp. Dưới đây là i_t , f_t và O_t lần lượt đại diện cho phương trình của cổng input, cổng forget và cổng output. Ký tự w là biểu tượng của trọng lượng và hàm sigmoid được biểu thị bằng σ . Dưới đây giới thiệu một hình minh họa cách hoạt động của LSTM.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (3)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (4)$$

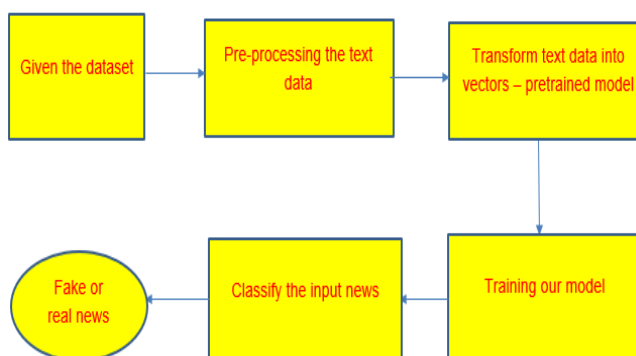
$$O_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad (5)$$



Hình 1: Mô-đun lặp lại trong LSTM chứa bốn lớp tương tác. [4]

IV. CLASSIFICATION PROCESS

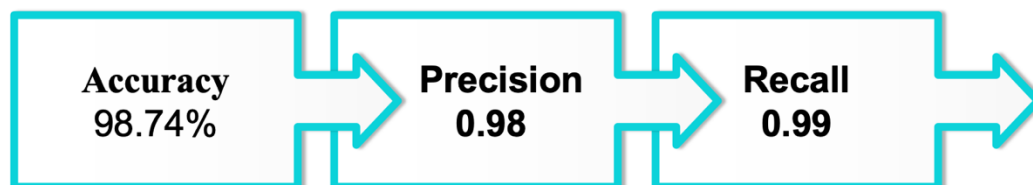
Trong phần này, chúng em sẽ tóm tắt cách thức hoạt động của trình phân loại bằng cách biểu diễn ngắn gọn một tập hợp các khối. Trong bước đầu tiên, chúng em đã thực hiện xử lý trước dữ liệu bằng các kỹ thuật được đề cập ở trên, bao gồm các stop-word, noise removal, lowercase toàn bộ văn bản và cuối cùng là stemming and lemmantizing. Sau đó, chúng em sử dụng Word2Vec với clean text thành các vector đại diện cho các tài liệu liên quan và chúng sẽ được đưa vào mô hình để train, đặc biệt rất quan trọng cho LSTM technique. Cuối cùng, bộ phân loại mà chúng em xây dựng sẽ dự đoán và quyết định tin tức đã cho thuộc về lớp fake or real. Dưới đây là hình ảnh minh họa cho quy trình phân loại của nhóm em trong đồ án lần này :



Hình 2: Quy trình phân loại

V. KẾT QUẢ

Dưới đây hiển thị kết quả bằng cách tính toán độ chính xác cũng như các chỉ số đánh giá của mô hình mà chúng em đã đề cập trước đây. Các giá trị hiển thị trong bảng là các giá trị khi chúng em triển khai các mô hình trong tập thử nghiệm.



Hình 3: Đánh giá các mô hình trong bộ thử nghiệm

VI. ERROR ANALYSIS

Mặc dù các giá trị accuracy, precision, recall khá cao, nhưng chúng em đã tìm thấy một số nhược điểm tiềm ẩn trong bài toán phân lớp này của mình. Giả sử rằng chúng em cung cấp văn bản được gán Real và tất nhiên, nó sẽ xuất ra "Real news". Tuy nhiên, chúng em đã thử sửa đổi tên của một nhân vật trong văn bản này, nó vẫn giữ đầu ra như trước đó. Chúng em nghĩ rằng vì tên của mọi người có trọng số thấp, mô hình của chúng em có xu hướng bỏ qua nó và đã để lại lỗi trong lúc đưa ra kết quả.

VII. CONCLUSION AND FUTURE DEVELOPMENT

Kết luận, chúng em đã tiến hành và trình bày một mô hình phát hiện tin tức giả thông qua phương pháp máy học LSTM.

Phát hiện tin tức giả là một lĩnh vực nghiên cứu đang phát triển có sẵn một số bộ dữ liệu. Tuy nhiên, một số tập dữ liệu có độ chệch cao và nhãn của chúng đôi khi không đáng tin cậy vì chúng phụ thuộc vào tư duy hoặc quan điểm của chủ sở hữu.

Trong quá trình phát triển thuật toán trong tương lai, chúng em sẽ cố gắng thu thập thêm dữ liệu từ các lĩnh vực khác để làm phong phú thêm nội dung cho mô hình của mình nhằm train hiệu quả hơn

VIII. BỔ SUNG PHẦN GIẢI THÍCH LẠI CODE:

```
def convertXToFormatAcceptable(data):
    X = []
    stop_words = set(nltk.corpus.stopwords.words("english"))
    tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')
    for par in data['fulltext'].values:
        tmp = []
        sentences = nltk.sent_tokenize(par)
        for sent in sentences:
            sent = sent.lower()
            tokens = tokenizer.tokenize(sent)
            filtered_words = [w.strip() for w in tokens if w not in stop_words and len(w) > 1]
            tmp.extend(filtered_words)
        X.append(tmp)
    return X
```

Phần này: Phần này: Thực hiện các bước tiền xử lý dữ liệu: lowercase toàn bộ text; tách từng chữ trong câu thành câu tokens và loại bỏ stopwords trước khi feed data vào mô hình. Thư viện hỗ trợ **nlTK**

```
def get_weight_matrix(model, vocab):
    vocab_size = len(vocab) + 1
    weight_matrix = np.zeros((vocab_size, EMBEDDING_DIM))
    for word, i in vocab.items():
        weight_matrix[i] = model[word]
    return weight_matrix

def tokenizerX(X):
    w2v_model = gensim.models.Word2Vec(sentences=X, size=EMBEDDING_DIM, window=5, min_count=1)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(X)
    X = tokenizer.texts_to_sequences(X)
    word_index = tokenizer.word_index
    X = pad_sequences(X, maxlen=maxlen)
    vocab_size = len(tokenizer.word_index) + 1
    return X, word_index, vocab_size, w2v_model, tokenizer
```

Phần này: chuyển đổi data dạng text thành các vectors bằng pretrained model Word2Vec (từ dòng w2v_model... đến texts_to_sequences(X)). pad_sequences nhằm để quy đổi các vectors về một độ dài cụ thể (ở đây độ dài bằng biến maxlen). Có sử dụng gensim để hỗ trợ. Link tìm hiểu Gensim: <https://helpex.vn/article/gensim-vector-hoa-van-ban-va-chuyen-doi-5c6645d7ae03f601287658fe>

```

def databefore(real):
    unknown_publishers = []
    for index, row in enumerate(real.text.values):
        try:
            record = row.split(" -", maxsplit = 1)
            assert (len(record[0]) < 260)
        except:
            unknown_publishers.append(index)
    real.iloc[unknown_publishers].text
    publisher = []
    tmp = []
    try:
        for index, row in enumerate(real.text.values):
            if index in unknown_publishers:
                tmp.append(row)
                publisher.append("Unknown")
                continue
            record = row.split(" -", maxsplit=1)
            publisher.append(record[0])
            tmp.append(record[1])
        real['publisher'] = publisher
        real['text'] = tmp
        del record
    except:
        pass
    del publisher, tmp, unknown_publishers
    real = real.drop([index for index, text in enumerate(real.text.values) if str(text).strip() == ''], axis=0)
    real['fulltext'] = real['title'] + ' ' + real['text']
    try:
        real = real.drop(["subject", 'text', "date", "title", "publisher"], axis=1)
    except:
        real = real.drop(["subject", 'text', "date", "title"], axis=1)
    return real

```

Phần này: Các mẫu tin được xem có publisher sẽ có format <Tên báo đài> + “-“ (dấu – là mẫu chốt để xác định). Ví dụ **WASHINGTON (Reuters) - The head of a conservat...** thì publisher ở đây là WASHINGTON (Reuters), nếu dòng data nào mà thiếu dạng như vậy thì đc xếp vào unknown publishers vì thường thì các news ko có publishers sẽ ko đáng tin cậy. Vì thế, cần được tách riêng ra để rồi loại bỏ chúng. Ở những dòng tiếp theo thì ta xoá những cột nào bị khuyết data dạng text; tạo ra một thuộc tính mới “full-text” bằng cách combine 2 thuộc tính “title” và “text” với nhau. Sau cùng là loại bỏ các thuộc tính khác như subject, date, etc. vì không còn quan trọng.

```

#Defining Neural Network
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state= 4353)
try:
    json_file = open('model.json', 'r')
    load_model = json_file.read()
    json_file.close()
    model = model_from_json(load_model)
    # load weights into new model
    model.load_weights("model.LSTM")
except:
    model = Sequential()
    # Non-trainable embedding layer
    model.add(Embedding(vocab_size, output_dim=EMBEDDING_DIM, weights=[embedding_vectors], input_length=maxlen,
                        trainable=False))
    # LSTM
    model.add(LSTM(units=128))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
    model.fit(X_train, y_train, validation_split=0.3, epochs=6)

    model_json = model.to_json()
    with open("model.json", "w") as json_file:
        json_file.write(model_json)
    # serialize weights to LSTM
    model.save_weights("model.LSTM")

```

Phần này: Trong lệnh “try”: khi file model đã được lưu sẵn khi thực hiện bước huấn luyện (training) xong, chỉ cần gọi lại file model để huấn luyện data mà không phải train lại từ đầu ở mỗi lần chạy code.

Trong lệnh “except”: Khởi tạo mạng LSTM với 128 units. Hàm kích hoạt qua lớp fully connected là hàm sigmoid, ta cần sử dụng hàm này với mục đích dự đoán xác suất cho model (cụ thể ở đây là xác suất real or fake) (vì hàm này luôn đưa ra các giá trị nằm trong đoạn [0;1]). Tiếp theo, tối ưu hoá bằng tham số Adam. Cuối cùng là gọi hàm model.fit() để bắt đầu train với tỉ lệ train/test là 70/30.

```

def GetPred(X_test):
    try:
        json_file = open('model_LSTM.json', 'r')
        load_model = json_file.read()
        json_file.close()
        model = model_from_json(load_model)
        # load weights into new model
        model.load_weights("model_LSTM.LSTM")
    except:
        model = Sequential()
        # Non-trainable embedding layer
        model.add(Embedding(vocab_size, output_dim=EMBEDDING_DIM, weights=[embedding_vectors], input_length=maxlen,
                             trainable=False))
        # LSTM
        model.add(LSTM(units=128))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
        model.fit(X, y, validation_split=0.3, epochs=6)

        model_json = model.to_json()
        with open("model_LSTM.json", "w") as json_file:
            json_file.write(model_json)
        # serialize weights to LSTM
        model.save_weights("model_LSTM.LSTM")
    y_pred = (model.predict(X_test) >= 0.5).astype("int")
    real = 0
    fake = 0
    for y_p in y_pred:
        if y_p == 1:
            real += 1
        else:
            fake += 1
    if real >= fake:
        return 1
    else:
        return 0

```

Phần này: dự đoán nhãn của text.

```

def run():
    score = GetPred(X_test_LSTM)
    # addResult(1, LSTM)
    if score >= 0.5:
        label_ans_fake.pack_forget()
        label_ans_real.pack(side="bottom", fill="both", expand="yes")
    else:
        label_ans_real.pack_forget()
        label_ans_fake.pack(side="bottom", fill="both", expand="yes")

```

Phần này: Gọi hàm GetPred() để lấy giá trị dự đoán. Nếu giá trị là 1 thì hiện hình nhãn Real lên màn hình. Ngược lại thì là nhãn Fake.

```

def run_predict():
    global X_test_LSTM, y_test_LSTM

    today = date.today()
    d = today.strftime("%d-%m-%y")
    link_ = entry1.get()
    if link_ != '':
        run()
        return

    title_data = title.get()
    text_data = text.get()
    subject_data = subject.get()
    if title_data != '' or text_data != '' or subject_data != '':
        if title_data == '':
            title_data = " "
        if text_data == '':
            text_data = " "
        if subject_data == '':
            subject_data = " "

        dataFile = pd.DataFrame([[title_data, text_data, subject_data, d]],
                                columns=['title', 'text', 'subject', 'date'])
        dataFile.to_csv('./demo.csv', index=False)
        dataFile = pd.read_csv('./demo.csv')
        os.remove('./demo.csv')
        global X_test_LSTM
        X_test_LSTM = getDataBeforeTrain(dataFile)
        run()

```

Phần này: Đọc dữ liệu từ file CSV và rồi thực hiện training, sau cùng là chạy hàm run() để hiển thị ra nhãn dự đoán.

REFERENCES

- [1] <https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html>
- [2] Poonam Tijare, Prannay S Reddy, Diana Elizabeth Roy, P. Manoj, M. Keerthana, “A Study on Fake News Detection Using Naïve Bayes, SVM, Neural Networks and LSTM”.

- [3] Pengfei Liu, Xipeng Qiu, Xuanjing Huang, “Recurrent Neural Network for Text Classification with Multi-Task Learning”.
- [4] Kelly Stahl, California State University Stanislaus, “Fake news detection in social media”.
- [5] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [6] Hands-on Machine Learning with Scikit-Learn Book.
- [7] <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [8] Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844. doi:10.1109/34.709601.
- [9] An Essential Guide to Pretrained Word Embeddings