Báo cáo Project cuối kì NLP

Tên thành viên:

* Phùng Dũng Quân: 22280073
* Trần Kiết Tường: 22280102
* Nguyễn Hồ Tuyên: 22280103

Nội dung bài thực hành:

Tuyên truyền và tin giả lan rộng gây ra nhiều rủi ro xã hội nghiêm trọng, như làm xói mòn niềm tin công chúng, gia tăng chia rẽ chính trị, thao túng bầu cử và lan truyền thông tin sai lệch trong các cuộc khủng hoảng như dịch bệnh hay xung đột.

Từ góc nhìn xử lý ngôn ngữ tự nhiên (NLP), việc phát hiện tin giả rất khó khăn vì nội dung này thường bắt chước ngôn phong và cấu trúc của báo chí chính thống, khiến việc phân biệt qua các đặc trưng bề mặt trở nên kém hiệu quả. Hơn nữa, thiếu hụt dữ liệu gán nhãn đáng tin cậy, đặc biệt là ở nhiều ngôn ngữ và khu vực, làm giảm hiệu quả của các mô hình học có giám sát, các mô hình NLP có nguy cơ khuếch đại thành kiến trong dữ liệu huấn luyện, dẫn đến phân loại sai và nguy cơ kiểm duyệt nhầm nội dung hợp pháp. Những thách thức này cho thấy cần có những phương pháp thận trọng, nhận biết ngữ cảnh rõ ràng, nhằm tránh việc hệ thống vô tình tiếp tay cho việc lan truyền thông tin sai lệch.

Tập dữ liệu được sử dụng trong bài là

MisinfoSuperset\_TRUE.csv, MisinfoSuperset\_FAKE.csv

Nguồn từ

American right-wing extremist websites (e.g., Redflag Newsdesk, Breitbart, Truth Broadcast Network)

Public dataset from:

      Ahmed, H., Traore, I., & Saad, S. (2017): "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques" \*(Springer LNCS 10618)\*

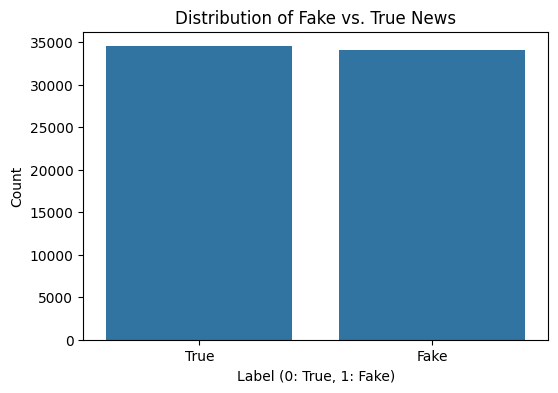
Trước tiên ta sẽ load model từ 2 bộ dữ liệu lại thành 1 tập dữ liệu lớn tên df và sẽ gán nhãn các tin thật là 0 và fake là 1. Ta thực hiện lần lượt các bước kiểm tra kiểu dữ liệu của tập dữ liệu, kiểm tra các phần tử rỗng và trùng nhau. Ở tập dữ liệu trên không có dữ liệu rỗng mà chỉ có phần từ trùng nhau nên ta sẽ bỏ các dữ liệu trùng.

DATA PREPROCESSING:

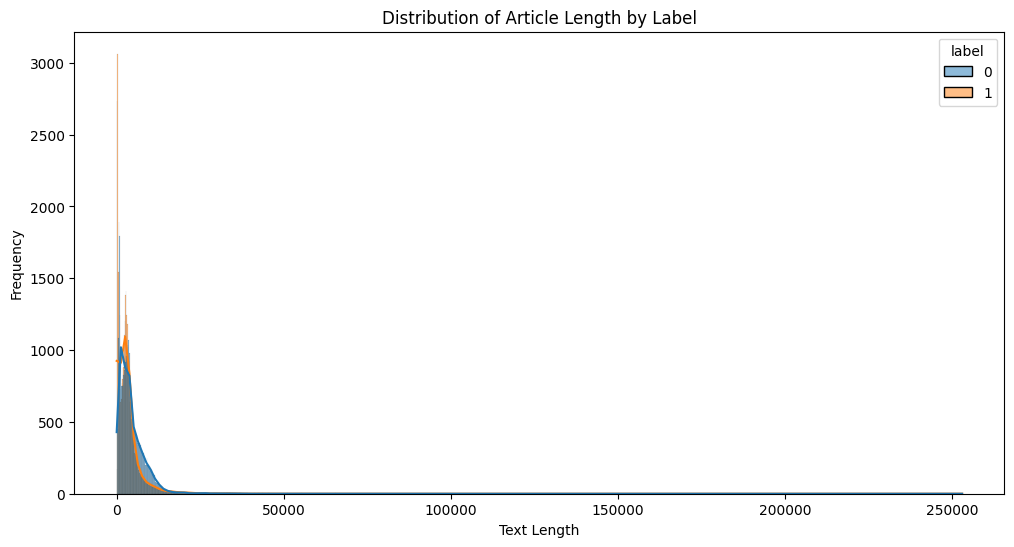
Ở bước tiêp theo ta sẽ thực hiện các bước tiền xử lý văn bản để làm sạch và chuẩn hoá đầu vào của bộ dữ liệu văn bản trước khi đưa vào các mô hình học máy với các bước đã làm trong bài bao gồm lặp văn bản, chuyển văn bản vể chữ thường, loại bỏ thẻ html và nội dung trong ngoặc, xoá URL, xoá kí tự đặc biệt và dấu câu, loại bỏ ký tự xuống dòng, xoá những từ chứa số, loại bỏ các stopwords ( là những từ xuất hiện nhiều trong văn bản nhưng không có ý nghĩa đặc biệt) và loại bỏ các từ ngắn ( dưới 3 kí tự). Các bước trên nhằm làm sạch bộ định dạng, ký tự nhiễu, giảm độ phức tạp và loại nhiễu cho bộ dữ liệu.

Để thuật tiện cho các thành viên làm nhóm sẽ lưu bộ dữ liệu với tên ‘cleaned’ để mọi người có thể chủ động gọi lại và sử dụng.

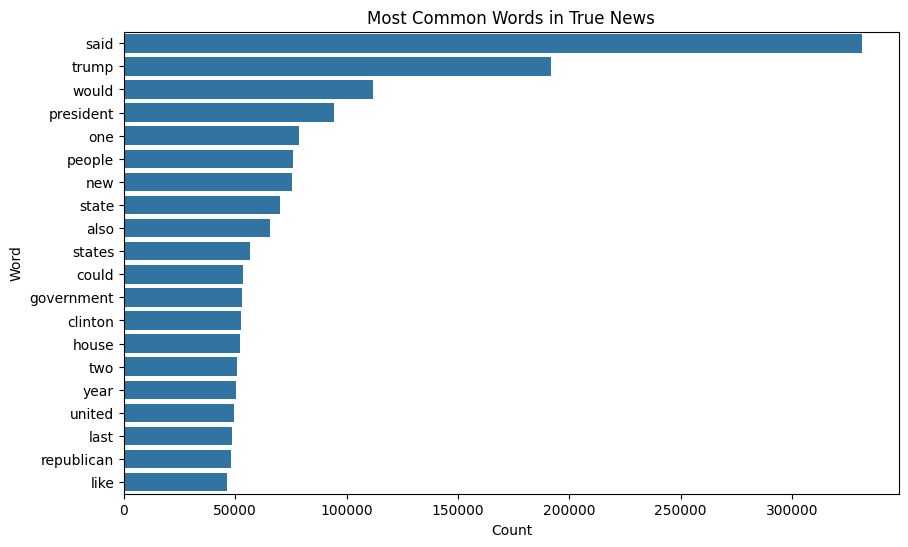
EDA:

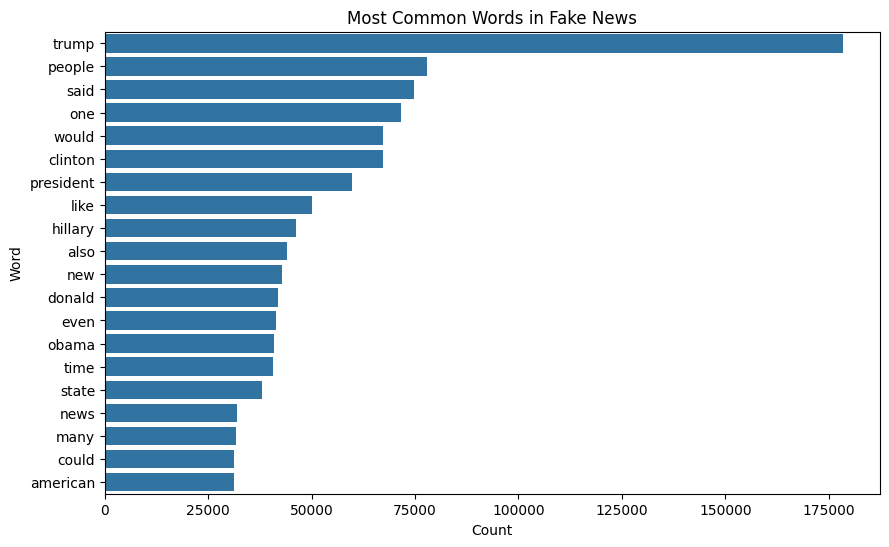


Đây là hình ảnh vẽ sự phân phối của fake và true news, ở đây ta có thể thấy dữ liệu không có hiện tượng bị lệch, cả 2 label phân phối đều nhau.



Biểu đồ này giúp ta hiểu về phân bố độ dài bài báo theo từng loại label, với cột x sẽ thể hiện độ dài văn bản và y là tần suất xuất hiện văn bản có độ dài đó, nhìn chung độ dài trung bình của các đoạn văn bản đã qua xử lý có độ dài dưới 50000 chữ tần 12000 - 20000 chữ. Các đoạn văn có label là 1 có tần suất xuất hiện với độ dài văn ngắn nhiều hơn label 0 thông qua hình ta thấy đỉnh của đường mật độ label 1 có đỉnh cao hơn nhiều so với label 0.. Label 0 có phân bố rộng hơn và có một bài bài viết dài, nên đường mật độ nó trải dài phía bên trái hơn. Biểu đồ cho thấy sự khác biệt rỗ rệt giữa tin giả và thật khi độ dài tin giả thương ngắn hơn.





2 đồ thị trên mô tả các từ xuất hiện nhiều nhất của 2 tin giả và thật:

Ở tập tin thật từ “said” dẫn đầu với 331,501 lần xuất hiện theo sau là “trump” 191,764 lần và “would” điều này có thể chỉ rằng tập dữ liệu tập trung vào các đoạn trích dẫn của một người nào đó hoặc 1 nhân vật chính trị. “trump” cũng được nhắc đến nhiều nhất trong các tin giả theo sau là “people” và “said” 74,629 lần. Ở đây ta có thể nói rằng từ “said” có thể là dấu hiệu của tin thật khi nó xuất hiện nhiều hơn gấp 4 lần so với tin giả ám chỉ rằng việt tin thật thường sẽ trích các đoạn văn thật tế nhiều hơn. Còn 1 điểm dễ nhìn thầy ở cả 2 tin là ở chữ “trump” và “clinton”, cả 2 chữ đều được xuất hiện với số lần xấp xỉ nhau ở cả 2 loại tin ám chỉ rằng chủ để chính trị sẽ là trung tâm ở cả 2 loại nhưng fake new lại khai thác về Clinton nhiều hơn 67,176 lần so với 52,628 lần. Ở tin thật: các từ government" (53,002), "house" (51,970), "two" (50,980), "year" (50,326), "united" (49,569), "last" (48,446), và "republican" (48,010) là đặc trung riêng của tin thật cho thấy rằng các tin thường nhấn mạnh yếu tố mang tính thể chế, thời gian, bối cảnh quốc già và thường đưa các tin tức chính phủ vào thời gian cụ thể. "hillary" (46,302), "donald" (41,772), "even" (41,483), "obama" (40,802), "time" (40,697), "news" (31,899), "many" (31,726), and "american" (31,271) là đặc trung của tin giả cho thấy tin giả có xu hướng tập trung vào các cá nhân nổi tiêng và sử dụng từ ngữ giật gân, cảm tính.

Nhận xét:

Tin thật: dùng từ ngữ có tính chất khách quan, phản ánh bối cảnh chính trị và cấu trúc xã hội.

Tin giả: khai thác nhiều yêu tố bất ngờ, tên người nổi tiêng nhằm thu hút tình cảm người đọc và cung cấp các tin chưa được kiểm chứng.

Huấn luyện các mô hình LinearSVC, LogisticRegression, XGBoost:

Chia tập dữ liệu thành: X\_train, X\_test, y\_train, y\_test với tập y sẽ chứa các label và X là phần còn lại với tỉ lệ train 0.8, test 0.2.

Sử dụng TfidfVectorizer chuyển đổi văn bản thành ma trận số dựa trên tần suất xuất hiện của từ và mức độ quan trọng của từ đó trong toàn bộ tập dữ liệu.

Với các tham số:

max\_features=5000, giới hạn số lượng đặc trưng (từ hoặc cụm từ) được sử dụng, chỉ lấy 5000 từ hoặc cụm từ có giá trị TF-IDF cao nhất. Nhằm kiểm soát kích thước ma trận chỉ tập trung vào các từ quan trọng.

ngram\_range=(1, 2): Xác định phạm vi n-gram (kích thước của các chuỗi từ liên tiếp) được xem xét, từ unigram (1 từ, e.g., "fake") đến bigram (2 từ, e.g., "fake news"). Để bắt được ngữ cảnh và tăng độ chính xác cho mô hình.

Tiếp theo ta sẽ scale bộ dữ liệu trên tập X\_train và fit X\_test theo cùng tỉ lệ.

Mô hình LinearSVC:

Dù kết quả của mô hình LinearSVC với các tham số mặc định kết quả cao f1-score: 0.92 nhưng ta vẫn sẽ cho chạy GridSearchCV để tìm các tham số tốt nhất của các tham số trong khoảng:

'C': [0.1, 1, 10],  'penalty': ['l1','l2'],   'loss': ['squared\_hinge'],

Kết quả trả về:

Best parameters found by GridSearchCV:

{'C': 0.1, 'loss': 'squared\_hinge', 'penalty': 'l1'}

Classification Report for Tuned LinearSVC:

precision recall f1-score support

0 0.93 0.93 0.93 7045

1 0.93 0.92 0.93 6675

accuracy 0.93 13720

macro avg 0.93 0.93 0.93 13720

weighted avg 0.93 0.93 0.93 13720

Ý nghĩa của các params:

C = 0.1: Tham số C là một hằng số điều chỉnh độ mạnh của regularization (quy tắc hóa) trong mô hình SVM. Nó kiểm soát sự cân bằng giữa việc tối đa hóa lề (margin) giữa các lớp và giảm thiểu lỗi phân loại trên tập huấn luyện, với C = 0.1, mô hình LinearSVC tránh học quá chi tiết các mẫu nhiễu trong tập dữ liệu tin giả/tin thật (như các từ không nhất quán hoặc văn phong khác biệt), giúp nó hoạt động tốt hơn trên dữ liệu kiểm tra.

loss = 'squared\_hinge': Tham số loss xác định hàm mất mát (loss function) được sử dụng để đo lường sai lệch giữa dự đoán của mô hình và nhãn thực tế. 'squared\_hinge' là một biến thể của hàm hinge loss (được định nghĩa max(0, 1 - y \* (w^T \* x + b)) ), thường được sử dụng trong SVM. Sử dụng 'squared\_hinge' giúp mô hình bạn phân biệt rõ ràng hơn giữa tin giả và tin thật, đặc biệt nếu dữ liệu có các điểm ngoại lai.

penalty = 'l1': Tham số penalty xác định loại regularization áp dụng cho mô hình, giúp ngăn chặn overfitting bằng cách giới hạn độ lớn của các trọng số (w). Với penalty='l1', mô hình LinearSVC có thể tự động chọn các đặc trưng quan trọng nhất (e.g., các từ hoặc cụm từ đặc trưng cho tin giả như "fake", "hoax") và bỏ qua các đặc trưng không liên quan. Điều này đặc biệt hữu ích với dữ liệu văn bản lớn như của bạn (78,588 bài báo), nơi không phải tất cả các từ đều mang ý nghĩa phân biệt.

Mô hình Logistic Regression:

Cũng như LinearSVC các performance metrics của Logistic regression cũng đã rất cao nhưng để đạt hiệu quả nhất ta sẽ cho tìm GridSerchCV các tham số:

   'C': [0.01, 0.1, 1, 10],

    'penalty': ['l2'],

    'solver': ['liblinear']

Và thu được: Best parameters found by GridSearchCV for Logistic Regression:

{'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}

Ý nghĩa các params:

C = 0.01: Tăng regularization để tổng quát hóa tốt, phù hợp với dữ liệu nhiễu như tin giả.

penalty = 'l2': Sử dụng regularization L2 để giữ tất cả đặc trưng với trọng số giảm đều, phù hợp với dữ liệu văn bản đa dạng.

solver = 'liblinear': Tối ưu hóa hiệu quả cho dữ liệu trung bình, phù hợp với kích thước hiện tại.

Classification Report for Tuned Logistic Regression:

precision recall f1-score support

0 0.92 0.94 0.93 7045

1 0.93 0.92 0.93 6675

accuracy 0.93 13720

macro avg 0.93 0.93 0.93 13720

weighted avg 0.93 0.93 0.93 13720

Mô hình XGBoost:

Kết quả mô hình:

Classification Report for XGBoost:

precision recall f1-score support

0 0.94 0.95 0.94 6913

1 0.94 0.94 0.94 6808

accuracy 0.94 13721

macro avg 0.94 0.94 0.94 13721

weighted avg 0.94 0.94 0.94 13721

do mô hình có kết quả ổn định nên ta không tìm các param khác mà giữ mặc định.

Huấn luyện Bert và Distilbert:

1. **Tạo bộ dữ liệu train, valid, test:**

* 60% dữ liệu đầu cho training.
* 20% dữ liệu tiếp theo cho validation.
* 20% dữ liệu cuối cùng cho testing.

1. **Xây dựng mô hình với số nhãn là 2:**

* Mô hình Bert: 'bert-base-uncased'.
* Mô hình XLNet: 'xlnet-base-cased'.

1. **Class Tokenize:**

class Tokenizer:

def \_\_init\_\_(self, model\_name, max\_length=128):

if 'bert' in model\_name:

self.tokenizer = BertTokenizerFast.from\_pretrained(model\_name)

 elif 'xlnet' in model\_name:

      self.tokenizer = XLNetTokenizerFast.from\_pretrained(model\_name)

  self.max\_length = max\_length

def tokenize\_function(self, examples):

return self.tokenizer(

  examples["text"],

      truncation=True,

      padding="max\_length",

      max\_length=self.max\_length

  )

1. **Thiết lập Hyperparameter cho training:**

* Output\_dir=’./bert\_results’. # Lưu kết quả huấn luyện vào file bert\_results.
* per\_device\_train\_batch\_size=8.
* per\_device\_eval\_batch\_size=8. # Thiết lập kích thước batch cho tập huấn luyện và tập đánh giá.
* num\_train\_epochs=1 # Số lần mô hình huấn luyện trên toàn bộ tập dữ liệu.
* weight\_decay=0.01 # Áp dụng kỹ thuật điều chuẩn L2 để giúp mô hình tránh overfitting.
* evaluation\_strategy="epoch" # Tần suất đánh giá mô hình.
* save\_strategy="epoch" # Tần suất lưu loss, learning\_rate, grad\_norm, trọng số của mô hình.

1. **Huấn luyện mô hình:**

* Sử dụng các thông số như accuracy, f1-score, recall, precision-score để đánh giá.

|  |  |  |
| --- | --- | --- |
| Chỉ số | Bert | Distilbert |
| Thời gian Training | ~13 phút (783s) | ~7.8 phút (471s) |
| Thời gian Đánh giá | ~1.5 phút (88.7s) | ~53.5s |
| Accuracy | 94.26% | 93.81% |
| F1-Score (Weighted) | **0.9427** | **0.9382** |
| Precision (Weighted) | 0.943 | 0.9389 |
| Recall (Weighted) | 0.9426 | 0.9381 |
| Loss trên tập Train | 0.242 | 0.2334 |
| Loss trên tập Validation | 0.1549 | 0.1641 |
| Loss trên tập Test | 0.1625 | 0.17 |
| Confusion Matrix | [[7174, 464],  [323, 5759]] | [[7095, 543],  [306, 5776]] |

1. **Đánh giá mô hình:**
2. **Nhận xét:**

* Mô hình Bert cho ra các thông số đánh giá tốt hơn nhưng đòi hỏi tài nguyên tính toán và thời gian huấn luyện đáng kể (13 phút cho 1 epoch).
* Mô hình Distilbert có thời gian huấn luyện và đánh giá **nhanh hơn khoảng 66%** và kích thước nhỏ hơn 40%, nó mang lại một sự đánh đổi gần như hoàn hảo giữa tốc độ và độ chính xác.
* Nếu mục tiêu là đạt được **độ chính xác tối đa** mà không quá quan tâm đến chi phí tính toán, **bert-base-uncased** là lựa chọn tốt nhất.
* Tuy nhiên, trong các ứng dụng thực tế nơi tốc độ inference và chi phí triển khai là yếu tố quan trọng, **distilbert-base-uncased** là một lựa chọn vượt trội. Nó chỉ hy sinh một phần rất nhỏ (chưa đến 1%) về hiệu năng để đổi lấy tốc độ nhanh hơn đáng kể và yêu cầu tài nguyên ít hơn.
* Cách cải thiện mô hình: ta có thể áp dụng thêm EarlyStopping để tránh hiện tượng overfit, và giảm thời gian huấn luyện và tài nguyên.

Đối với bài toán này, sự đánh đổi của Distilbert là hoàn toàn chấp nhận được và nó được xem là mô hình hiệu quả hơn về tổng thể.

Input Text

Tokenizer

Input IDs + Attention Mask + Label

DistilBERT

FAKE / REAL

-**Input**: Dữ liệu đầu vào là văn bản.

**-Tokenizer**: Chia văn bản thành các từ nhỏ (tokens) và ánh xạ chúng sang các số nguyên (ID).

**-Input IDs**: Chuỗi các ID số thể hiện văn bản gốc.

**-Attention Mask**: Dãy 0/1 xác định token nào cần được mô hình chú ý đến (1 = chú ý, 0 = padding).

**-Label**: Nhãn dữ liệu gồm giá trị 0 và 1.

**-DistilBert**: Mô hình học sâu là tiền huấn luyện của Bert, nhanh hơn, nhẹ hơn nhưng vẫn đủ khả năng phân biệt được ngữ cảnh, từ vựng trong câu.

**-Cross Binary Entropy**: Hàm loss đo lường sự khác biệt giữa xác suất dự đoán và nhãn thực tế.

**-Evaluation**: Đánh giá độ chính xác mô hình qua các thông số Accuracy, F1-score, Precision.

**-Output**: Phân loại nhãn.

Huấn luyện mô LSTM:

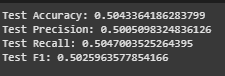
**\*Mạng RNN đơn tầng:**

-Nhận embedding làm đầu vào

-Trả về hidden state cuối (h\_n)

-Dự đoán nhị phân qua Linear + Sigmoid

Evaluate trên tập test:

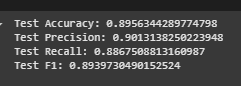


**\*Hạn chế của RNN (đơn tầng):**

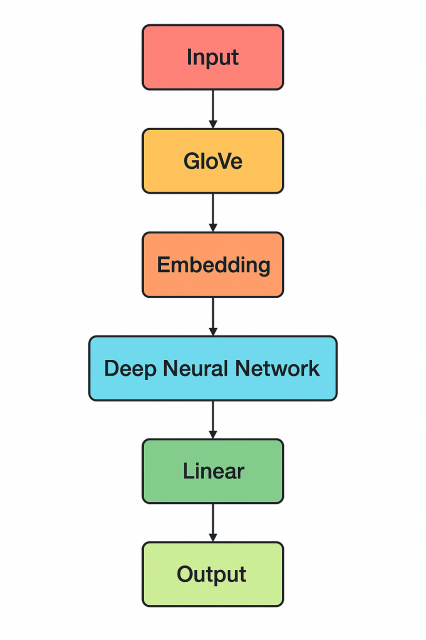
| **Hạn chế** | **Chi tiết** |
| --- | --- |
| **Không nhớ dài hạn tốt** | RNN dễ bị mất thông tin khi chuỗi quá dài (vấn đề "vanishing gradient") |
| **Chỉ xử lý từ quá khứ → hiện tại** | Không tận dụng được ngữ cảnh tương lai trong chuỗi |
| **Đơn tầng không đủ trừu tượng** | Một tầng không đủ để học các đặc trưng ngôn ngữ phức tạp như cú pháp, ngữ nghĩa sâu |
| **Dễ overfit** | Mạng nông khó đạt độ chính xác cao trên dữ liệu thực tế |

**\*Giải pháp: Deep BiLSTM:**

Evaluate trên tập test:

****

| **Cải tiến** | **Lợi ích** |
| --- | --- |
| **Bi-directional** | Xem được cả thông tin trước và sau từ hiện tại |
| **LSTM > RNN** | LSTM dùng cơ chế cổng (gates) để **ghi nhớ dài hạn** tốt hơn |
| **Deep (2 tầng)** | Nhiều tầng giúp học **đặc trưng trừu tượng** hơn: tầng 1 học đặc trưng ngữ nghĩa cơ bản, tầng 2 học mối liên hệ ngữ pháp phức tạp hơn |
| **Ổn định gradient** | LSTM và nhiều tầng có cơ chế giúp tránh gradient vanishing tốt hơn RNN truyền thống |



-**Input**: Dữ liệu đầu vào là văn bản

-**Glove(Word Embeddings)**: GloVe giúp biến từ thành vector có ý nghĩa ngữ nghĩa (semantic), giúp mô hình hiểu được các mối quan hệ giữa từ

**-Embedding Layer:** Dùng trọng số GloVe đã huấn luyện để ánh xạ văn bản thành dãy vector – cho đầu vào nhất quán cho mô hình học sâu.

**-Deep BiLSTM**:

Mô hình sâu học được đặc trưng ngữ cảnh phức tạp hơn.

BiLSTM khai thác thông tin cả chiều xuôi và ngược, hiểu tốt các nội dung câu dài, mập mờ – thường gặp trong tin giả.

**-Linear + Sigmoid:**

Linear biến đổi về 1 giá trị logit.

Sigmoid để xuất xác suất (giả hay thật).

**-Output:**

Phân loại nhị phân: 1 (giả), 0 (thật).

**\*Nhận xét:**

Tokenization còn đơn giản trong Glove khi chỉ dùng str.split() gây ảnh hưởng đến chất lượng embedding nên độ chính xác chưa như kì vọng.

Epochs của quá trình train Glove và Deep BiLSTM còn thấp nên ảnh hưởng đến độ chính xác:

-Glove chưa học đủ để hiểu rõ các mối quan hệ giữa từ nên embedding chưa tốt

-Deep BiLSTM chưa được huấn luyện đủ nên chưa khai thác được tối đa ngữ cảnh

**\*Cách cải thiện:**

-Nâng cấp tokenization để chẩun hóa đầu vào Glove.

-Tăng số epoch, kết hợp EarlyStopping, giúp embedding hội tụ tốt hơn và mô hình phân loại học đặc trưng ngữ cảnh sâu hơn.

**KẾT LUẬN:**

Bộ dữ liệu này tương đối đơn giản và rõ ràng nên khi ta khi thực hiện EDA đã có thể thấy được insight của từng bộ dữ liệu thật và giả. EDA cho thấy tin thật mang tính khách quan, còn tin giả thường giật gân và cá nhân hóa. BERT đạt độ chính xác cao nhất (94.26%) nhưng huấn luyện chậm và tốn tài nguyên. DistilBERT với độ chính xác (93.81%) nhưng tốc độ huấn luyện nhanh hơn khoảng 66%, rất phù hợp cho triển khai thực tế. Deep BiLSTM chưa đạt hiệu quả tối ưu do hạn chế về tokenization và số epoch thấp. Các mô hình như Logistic Regression, XGBoost, và SVC tuy hiệu suất cao nhưng lại không tận dụng được kiến thức ngôn ngữ học sâu. Nhờ kiến trúc transformer, DistilBERT linh hoạt mở rộng và thích ứng với dữ liệu mới tốt hơn.

→DistilBert là lựa chọn tối ưu cho bài toán này.