**1. Tính cấp thiết của vấn đề:**

Việc phát triển của công nghệ thông tin giúp mọi người có cơ hội chia sẻ và truyền thông tin lẫn nhau với phạm vi và tốc độ nhanh hơn bao giờ hết trong lịch sử. Điều này đem lại muôn vàn điều tốt song song với đó cũng là những thói xấu mà tin giả là một điển hình. Không cần kể nhiều chắc ta cũng biết tác động của tin giả đến xã hội đặc biệc là trong mùa dịch Covid-19 vừa qua. Và theo nghiên cứu [1] thì năm 2020 ở các nước phát triển tin giả sẽ còn nhiều hơn tin chính thống và việc phát hiện tin giả bằng việc xem xét từ ngữ được sử dụng trong bài viết là khả thi. Vì vậy sử dụng Deep Learning để dự đoán tin giả, những tin tức giả này có thể ảnh hưởng đến chính trị, kết quả bầu cử hoặc uy tín một doanh nghiệp nên việc **dự đoán tin giả là một yêu cầu cấp bách** trong thời đại công nghệ thông tin hiện nay.

**2. Tổng quan về các công nghệ phát hiện tin giả hiện nay và tính mới của sản phẩm và cơ sở nghiên cứu**

Do sự cấp bách đó, nhiều tổ chức và công ty công nghệ đã vào cuộc phát triển những sản phẩm trí tuệ nhân tạo (hầu hết là dùng công nghệ Deep Learning) để phát hiện tin giả có thể kể đến như: matdekoning với hybrid model, google với bert model ..

Các sản phẩm của các công ty lớn đã đạt kết quả tốt trong việc phát hiện tin giả nhưng có một vấn đề là lớp word embedding (đó đơn giản là một lớp giúp chuyển các từ trong ngôn ngữ tữ nhiên thành các vector để phục vụ cho việc dự đoán tin giả). **Lớp word embedding của họ không thể dùng dự đoán tin giả bằng ngôn ngữ nó chưa được huẩn luyện** và việc huấn luyện(training) một word embedding độ phức tạp tương đương với việc training một mô hình(model) hoàn chỉnh và nó cũng là phần quan trọng nhất trong một model. Hiện nay chỉ có một số word embedding của các ngôn ngữ phổ biến như Tiếng Anh hoặc Tiếng Trung đạt hiệu quả cao nên tương lai các công ty lớn như google hỗ trợ các model dự đoán tin giả Tiếng Việt là còn rất xa. Và như đã đề cập ở trên dự đoán tin giả là một yêu cầu cấp bách nên **việc xây dựng một model dự đoán tin giả Tiếng Việt là điều cấp bách.** Vì thế nhóm em đã làm extension (một phẩn mền gọn nhẹ có thể cài đặt tích hợp trên trình duyệt như Google) để dự đoán độ xác thực của thông tin đang đọc một cách đơn giản.

Cơ sở của việc phát hiện tin giả dựa trên cách sử dụng từ ngữ trong bài viết cũng đã được chứng minh là hiệu quả và được các công ty lớn, tổ chức nghiên cứu từ lâu nên Tiếng Việt cũng không nằm ngoài phạm vi đó (đặc biệt ta thấy những bài báo mang tính chất phản động hay quảng cáo thần dược thì lời lẽ rất đặc trưng và có thể dự đoán được).

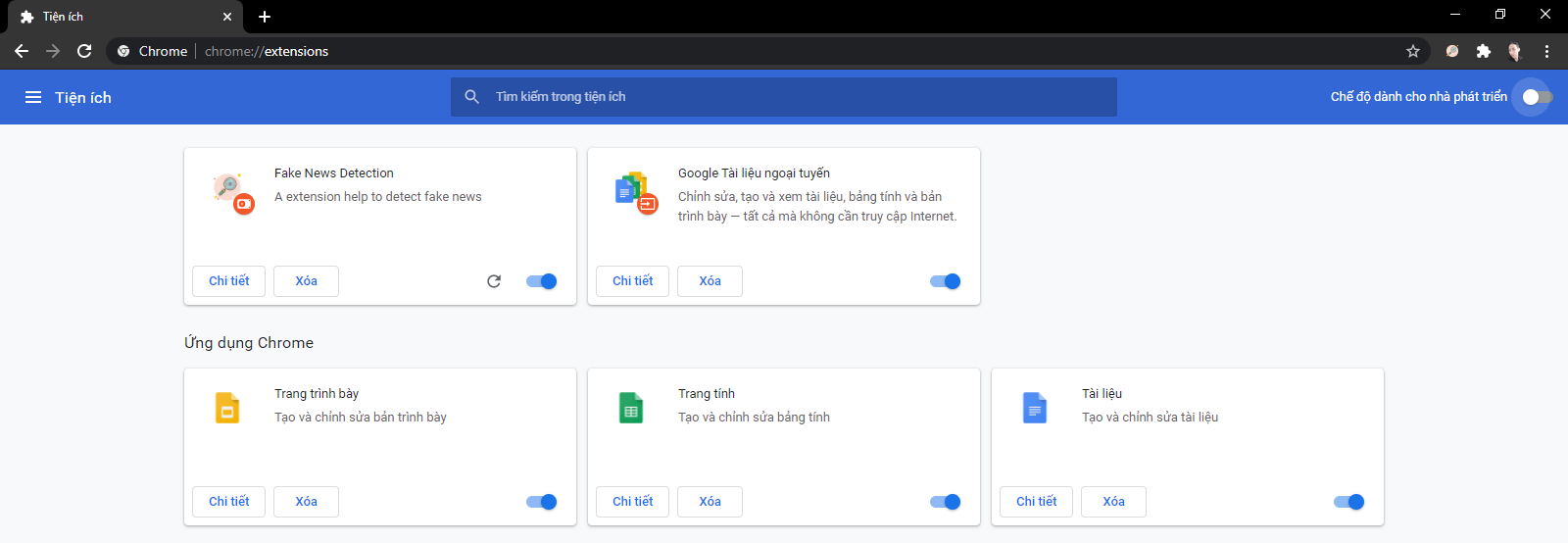
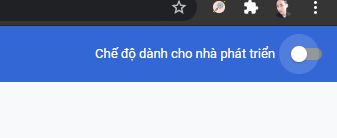
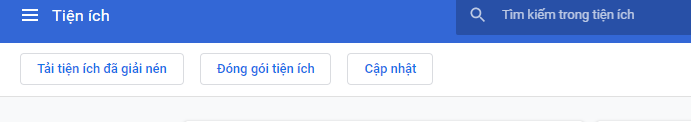
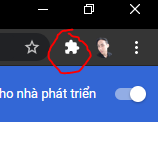
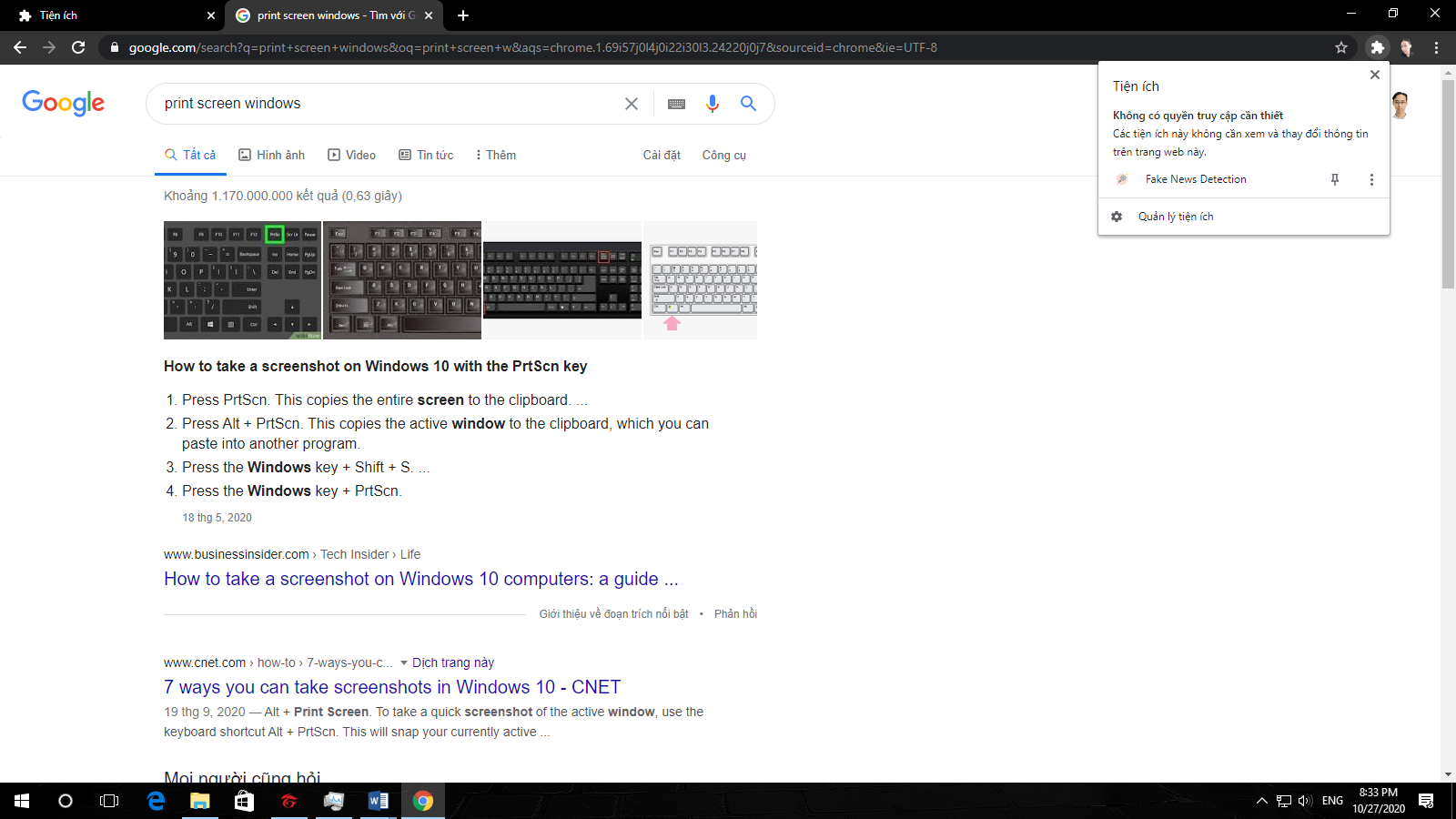
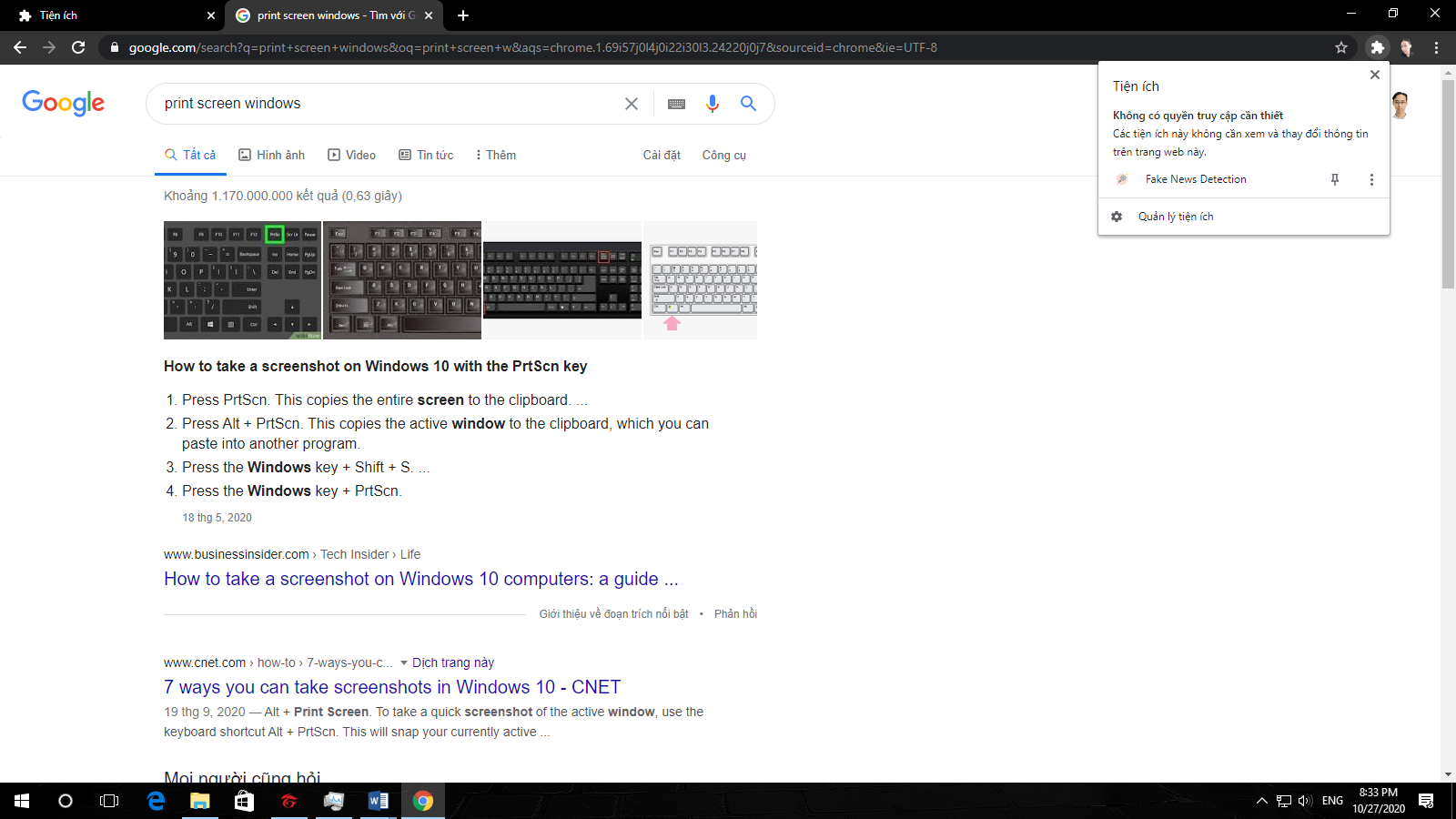
**2. Phương án thiết kế và thu thập dữ liệu**

Đầu tiên khi bắt tay vào làm sản phẩm nhóm nhận ra là hầu hết các tin giả đều được đọc từ trình duyệt, hầu hết tin giả trên facebook cũng dẫn nguồn đến bài báo khác nên việc chọn một phương án thiết kế sản phẩm cho thuận tiện với người dùng khi họ đang đọc bài báo trên trình duyệt là rất cần thiết. Từ đó nhóm đã quyết định chọn làm sản phẩm là một **extension** với ưu điểm gọn nhẹ tích hợp trên trình duyệt giúp người dùng có thể sử dụng xử lí thông tin ngay trên trình duyệt mà không cần chuyển sang một phần mền thứ ba.

Thời gian hạn chế và việc thu thập dữ liệu về tin giả khá khó khăn nhưng nhóm đã tìm ra bộ dữ liệu VIETNAMESE FAKE NEWS DATASET – VFND [2], đó là bộ dataset về các tin tức giả bằng ngôn ngữ tiếng Việt được tập hợp trong khoảng thời gian từ 2017 đến 2019, các tin tức được đưa vào đây được phân loại thật giả dựa trên một số nguồn tin, tham chiếu chéo đến các nguồn tin được dẫn hoặc được phân loại bởi cộng đồng. Dữ liệu được lưu dưới dạng các file json và nhóm đã dùng ngôn ngữ lập trình python để bóc tách dữ liệu từ đó chuyển vào để huấn luyện model. Đến nay nhóm vẫn chưa có thời gian để bổ sung thêm dữ liệu khác ngoài bộ VFND nhưng trong tương lai thì sẽ cố kiếm thêm để tối ưu model.

**3. Hướng dẫn cài đăt và sử dụng**

Hiện tại vì mục đích dự thi KHKT nên nhóm chưa upload sản phẩm lên google extension store nhưng trong tương lai nếu được tải lên rồi thì mọi người có thể lên đó tải về chỉ với một cú click chuột. Còn bây giờ nhóm sẽ hướng dẫn cài thủ công chỉ với vài bước đơn giản:

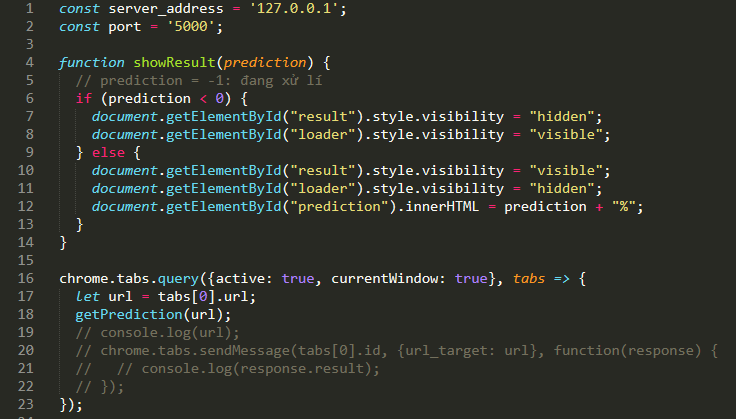
* B1: Mở trình duyệt lên và vào phần cài đặt extension, trong chrome bọn có thể truy cập nhanh bằng đường dẫn: chrome://extensions/
* 
* B2: Khi ra giao diện như trên trượt công tắc **Chế độ dành cho nhà phát triển** ở góc trên bên phải màn hình , khi thực hiện thì sẽ hiện thêm một thanh công cụ ở phần dưới dải màu xanh là: 
* B3: Chọn vào nút **Tải tiện ích đã giải nén** ở trên thanh công cụ vừa hiện ra ở trên và chọn tới folder chứa extension. Như vậy là đã cơ bản cài đặt xong nhưng muốn cho tiện sử dụng thì vui lòng đọc thêm B4.
* B4: Chọn vào biểu tượng chung của các extension ở trên thanh công cụ trình duyệt:  nó sẽ hiện ra như sauvà chọn vào biểu tượng  để ghim extension ra ngoài cho dễ sử dụng. Sau khi làm xong thì tiện ích của chúng ta đã hiện ra ngoài như sau:  và khi vào một trang báo bất kì muốn sử dụng chỉ cần nháy chuột vào đó và extension sẽ đưa ra độ xác thực của thông tin trên trang theo dự đoán.

**2. Cách thức hoạt động và kiến trúc model (kèm mã nguồn)**

**Cách thức hoạt động:**

Khi người dùng bấm vào extension thì qua thư viện **Tabs** lấy được đường dẫn của bài viết hiện tại đang đọc và dùng thư viện **XMLHttpRequest** để gửi đường dẫn đó đến server đang chạy model để đưa ra dự đoán:

*Lấy url, hiện thị khung loading, gọi hàm gửi dự đoán đến server*



*Gửi dự đoán tới server và khi có kết quả gọi hàm để ẩn khung loading và hiển thị kết quả*

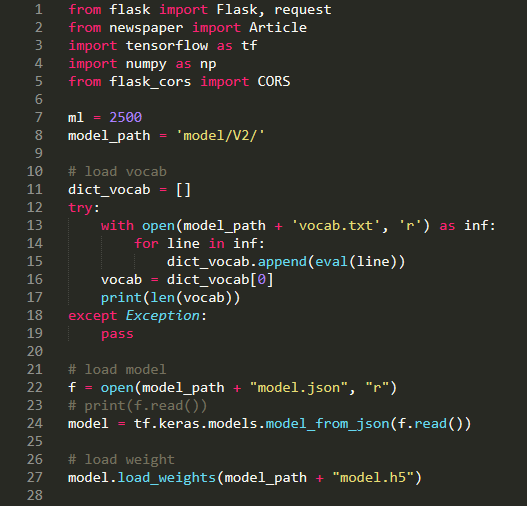


Server dùng thư viện **flask** để bắt các request sau đó dùng thư viện **newspaper** để lấy dữ liệu các bài viết và gọi các **hàm tiền xử lý** data để đưa về dạng phù hợp cho việc dự đoán, cuối cùng là gọi các **hàm dự đoán** từ model đã khởi chạy sẵn:



Các hàm tiền xử lý dữ liệu sử dụng **numpy**, khởi chạy và dự đoán model dùng **tensorflow**:

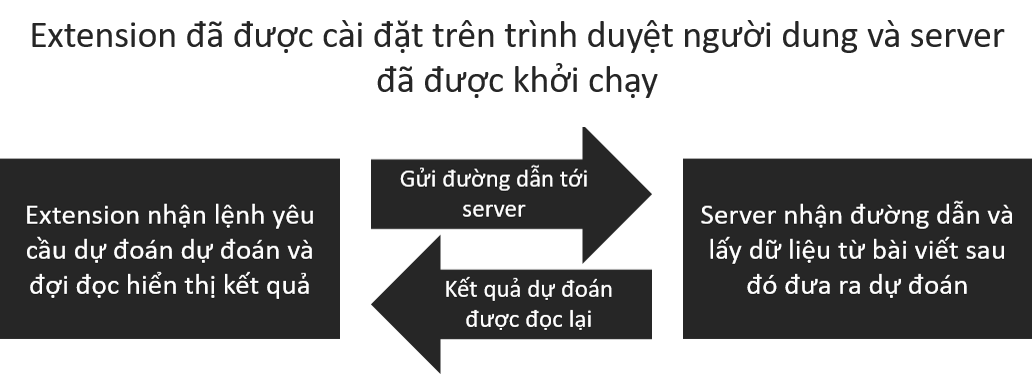
*Khởi chạy model cho dự đoán*



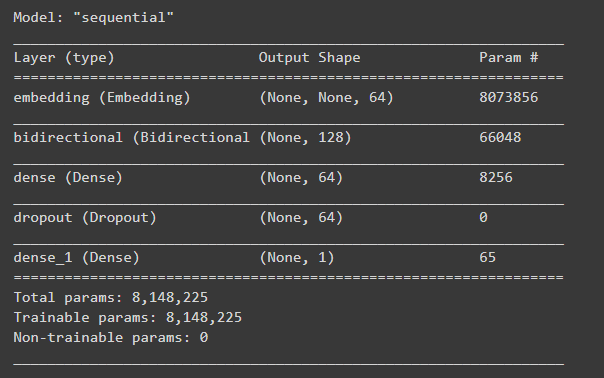
*Tiền xử lý*



Sau khi model đưa ra dự đoán sẽ được đọc lại từ extension và hiển thị kết quả. Đó là cách thức hoạt động cơ bản của thệ thống và có thể được biểu diễn đơn giản qua sơ đồ khối sau:



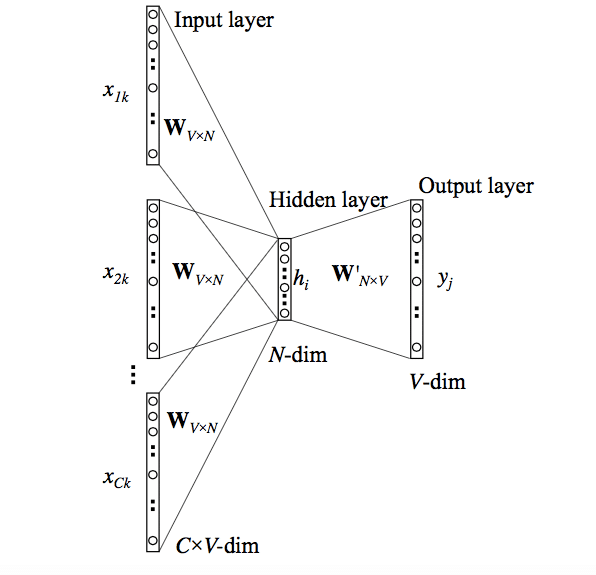
**Kiến trúc model: Gồm 5 lớp**



**Lớp đầu là lớp Word Embedding** [3]:

Lớp này giúp chuyển đổi từ ngữ trong ngôn ngữ tự nhiên sang word embedding. Word embedding được coi là cách tốt nhất để thể hiện các từ trong văn bản. Kỹ thuật này cũng gán mỗi từ với một vector, nhưng ưu việt hơn kỹ thuật vector ngẫu nhiên vì các vector này được tính toán để biểu diễn quan hệ tương đồng giữa các từ. Để tạo word embedding thì có hai phương pháp phổ biến là: **Skip Gram** và Continuous Bag of Word (**CBOW**). Ở đây nhóm em làm theo phương pháp thứ hai vì có ưu điểm là train nhanh và biểu hiện tốt trên bộ dữ hiệu lớn nhưng khi thu thập dữ liệu thì nhóm nhận ra là khá mất thời gian nên chưa tìm được nhiều nên kết quả vẫn chưa tối ưu lắm và trong tương lai có thể nhóm sẽ nghiên cứu thử dùng phương pháp thứ nhất. Vì vậy ở đây sẽ nói luôn về cả hai phương pháp:

Mô hình CBOW: Phương pháp này lấy ngữ cảnh của mỗi từ làm đầu vào và cố gắng dự đoán từ tương ứng với ngữ cảnh. Hãy xem xét ví dụ: Hôm nay tôi đi học. Chúng ta sẽ cố gắng dự đoán từ mục tiêu (đi) bằng cách sử dụng duy nhất một từ ngữ cảnh đầu vào (học). Cụ thể hơn, chúng tôi sử dụng mã hóa one-hot của từ đầu vào và đo lỗi đầu ra của mạng nơ ron đối với mã hóa one-hot của từ mục tiêu (đi). Ngoài ra, chúng ta có thể xây dựng các kiến trúc dự đoán một từ bằng nhiều từ xung quanh. Trong quá trình dự đoán từ mục tiêu, mô hình sẽ học được biểu diễn vectơ của từ mục tiêu.

Mô hình Skip Gram:

Mô hình Skip Gram còn được coi là phiên bản đảo ngược của mô hình CBOW. Cho trước một vị trí ngữ cảnh, mô hình cần đưa ra được phân bố xác suất của mỗi từ ở vị trí đó. Trong cả hai trường hợp, mạng sử dụng lan truyền ngược để học ra biểu diễn vector của từ.

**Lớp thứ hai là lớp Bidirectional RNN** [4]:

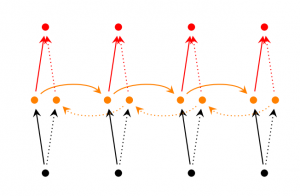
Để giải thích về lớp này em xin giải thích trước về một lớp RNN[5] trước vì BRNN là một lớp cải tiến của RNN. Thì ý tưởng cơ bản của việc sử dụng RNN là do trong ngôn ngữ ý nghĩa của một câu từ có thể phụ thuộc vào ngữ cảnh mà nó đang được đặt vào. VD trong một bài viết:



Ta có thể thấy sự việc Mr Bean qua đời là không chính xác vì các từ ngữ ở trước đã phủ định nó. Để xử lí vấn đề đó RNN sử dụng chuỗi các thông tin. RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó (ta cùng bàn cụ thể vấn đề này sau) mà thôi. Về cơ bản một mạng RNN có dạng như sau:



Nhưng RNN có một hạn chế là ý nghĩa của một từ không chỉ phụ thuộc vào các từ đứng trước nó mà còn là các từ đứng sau đó. VD khi ta muốn dùng RNN để đánh giá bình luận của người dùng về một bộ phim là tốt hay tệ, bình luận của người dùng đó như sau: “Phim hay thế, biết vậy khỏi xem” ta có thể thấy đó là một câu nói mỉa mai và ý nghĩ của từ “hay” phụ thuộc vào các từ ngữ đứng sau nó mà RNN chỉ có thể nhớ những từ ngũ đứng trước. Để giải quyết vấn đề đó thì người ta đã phát minh ra BRNN, đó cơ bản là cài đặt 2 mô hình RNN chạy ngược hướng nhau với mô hình chạy theo hướng đọc(thường là trái sang phải nhưng tùy theo ngôn ngữ) là chính và mô hình kia để hiệu chỉnh kết quả của mô hình chính. Mô hình BRNN:



Các lớp tiếp theo không có gì quá nổi bật nên sẽ giải thích đơn giản:

**Lớp thứ ba và thứ năm là lớp Dense**:

Dense layer hay Fully-connected layer là một lớp cổ điển trong mạng nơ ron nhân tạo. Mỗi nơ ron nhận đầu vào từ tất cả nơ ron lớp trước đó.

Có thể hiểu cơ bản là như đã nói ở phần **Cơ sở lý thuyết** thì đó là các lớp chứa các điểm nút trên đồ thị dựa vào đó các phần đồ thị giữa hai điểm nút kề nhau có thể được các hàm activation uốn đồ thị theo dữ liệu mẫu.

**Lớp thứ tư là lớp Dropout**:

Dropout là việc chúng ta sẽ bỏ qua một vài unit trong suốt quá trình train trong mô hình, những unit bị bỏ qua được lựa chọn ngẫu nhiên. Ở đây, chúng ta hiểu “bỏ qua - ignoring” là unit đó sẽ không tham gia và đóng góp vào quá trình huấn luyện (lan truyền tiến và lan truyền ngược).

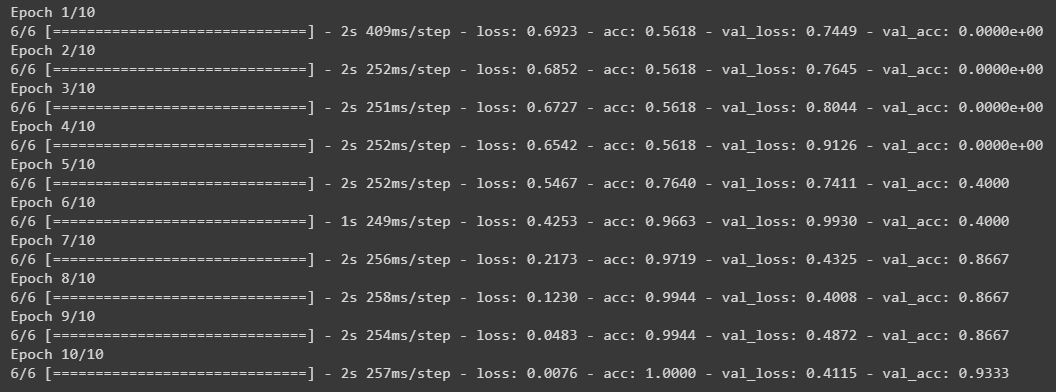
Dropout được sử dụng để chống over-fitting[6]. Nó cơ bản là vấn đề do mô hình cố gắng uốn đồ thị quá khớp bộ dữ liệu do train không phù hợp dẫn đến thể hiện rất tệ trong thực tế.

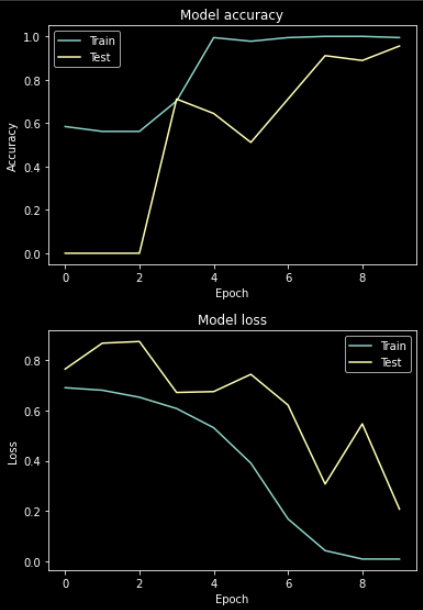
**3. Kết luận**

**3.1 Kết quả:**

Thiết kế chế tạo được sản phẩm đáp ứng mục tiêu nghiên cứu đặt ra, cụ thể:

* Cho kết quả khá tốt trên bộ dữ liệu kiểm thử(độ chính xác val\_acc thời điểm cao nhất đạt 93.33%):



Và không hề có dấu hiệu over-fitting(có thể thấy trong biểu đồ được chiết xuất từ mô hình, Accuracy tăng liên tục trong khi loss vẫn giảm tốt):

* Tích hợp thành công lên nền tảng extension và giao tiếp tốt với server AI.

**3.2 Tính ưu việt của đề tài**

* Kích thước nhỏ gọn, dễ cài đặt, không đòi hỏi cấu hình cao, đưa ra kết quả nhanh chóng(thời gian mỗi lần dự đoán trả về kết quả chỉ 2 – 3 giây), hoàn toàn không có hiện tượng giật lag, chiếm ít bộ nhớ(vì hoàn toàn xử lí trên server), tính bảo mật cao(vì cái được gửi đi chỉ là đường link của bài viết).
* Tính tái sử dụng là cao vì word embedding của tin giả khá phù hợp để transfer-learning trong các mô hình dự đoán độ hay dỡ, tốt xấu.. Có thể sử dụng cài đặt trên các nền tảng khác vì đa số phần xử lí là trên server.
* Hoạt động tốt trên các bài viết từ năm 2017 đến 2019.
* Có tính mới, tính sáng tạo cao.

**3.3 Hạn chế**

* Do thời gian hạn chế nên chưa tìm được thêm dữ liệu và trong bộ dữ liệu cũng có một số bài viết đặt sai nhãn dẫn đến sự khó khăn khi huấn luyện mô hình. Bên cạnh đó định dạng dữ liệu cũng khá tệ dẫn đến các bước tiền sử lý khó khăn.
* Cũng vì do vấn đề thời gian nên chưa tích hợp thử các mô hình tạo embedding mới như đã đề cập ở trên và chưa thử tích hợp lớp Pooling(Đây là một lớp khá tốt trong việc bắt nhũng từ hiểm vì trong các bài viết tin giả đặc biệt là các bài có nội dung phản động thì thường sử dụng nhũng từ ngữ đặc biệc).
* Do bộ dữ liệu nhỏ(chưa đến 150 bài viết) chỉ có các bài viết từ 2017 đến 2019 và kèm với sự phát triển của tin giả.

**=>** Việc dự đoán các tin giả trong nửa cuối năm 2019 đến 2020 là không ổn định lắm.

**3.4 Hướng phát triển của đề tài**

* Bổ sung thêm các tin giả trong thời gian gần đây, lọc và đánh nhãn lại các bài viết. Thực sự các nguồn tin giả của Việt Nam có rất nhiều, nhất là trong các trang phản động như: <https://danlambaovn.blogspot.com/> nên nếu được đi tiếp thì việc bổ sung thêm là khá dễ dàng.
* Cải thiện thuật toán, cấu trúc model. Cụ thể là thử tích hợp lớp Pooling và mô hình tạo word embedding mới.
* Vì bản chất hệ thống chỉ dự đoán một tin là thật hay giả dựa trên từ ngữ của bài viết nên việc dự đoán tin giả và huấn luyện thêm các sự kiện nội bật trong thời gian đó là việc phải đi song hành nhau thì mới có đạt được hiệu quả cao. Nên có thể sẽ tạo ra một hệ thống cho phép người dùng có thể đóng góp thêm cho sản phẩm bằng việc đánh giá một tin là thật hay giả bằng kiến thức của mình và sẽ được duyệt để đưa vào huấn luyện như mô hình làm việc của hệ thống Google Dịch chứ không phải chính tác giả phải đi tìm dữ liệu như hiện tại.

**Các nguồn tham khảo thêm:**

[1]: Titcomb, J., Carson, J.: www.telegraph.co.uk. Fake news: What exactly is it – and how can you spot it?

[2]: <https://github.com/thanhhocse96/vfnd-vietnamese-fake-news-datasets>

[3]: <https://trituenhantao.io/kien-thuc/word-embedding-la-gi-tai-sao-no-quan-trong/>

[4]: <https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks>

[5]: <https://dominhhai.github.io/vi/2017/10/what-is-rnn/>

[6]: <https://en.wikipedia.org/wiki/Overfitting>