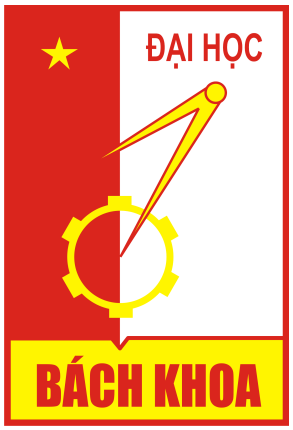
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

PHÂN TÍCH QUAN ĐIỂM



GIẢNG VIÊN HƯỚNG DẪN:
PGS.TS. LÊ THANH HƯƠNG

SINH VIÊN THỰC HIỆN:
ĐẶNG MẠNH CƯỜNG - 20130489
PHAN THỊ HỒNG HẠNH - 20131301
HÀ MINH CÔNG - 20130447

05-12-2016

TÓM TẮT

Phân tích quan điểm là một bài toán trong lớp bài toán xử lý ngôn ngữ tự nhiên. Bài toán được định nghĩa như sau, với một câu hoặc một đoạn văn đầu vào, mục đích của chúng ta là xác định lớp quan điểm của câu hay đoạn văn đó, là tích cực hay kém tích cực.
Có rất nhiều phương pháp giải quyết bài toán này, ví dụ như phân loại dựa trên Naive Bayes, Support Vector Machine... Chúng em xin trình bày một phương pháp mới, đã và đang phát triển rất mạnh mẽ đến thời điểm hiện tại, đó là Deep Learning.

Mục lục

Danh sách hình vẽ

Chương 1

Giới thiệu chung

1.1 Phát biểu bài toán

Phân tích quan điểm là một trong những bài toán xử lý ngôn ngữ tự nhiên, đã và đang được nghiên cứu rộng rãi, cũng giống như bài toán phân loại văn bản, theo đó mỗi văn bản sẽ thuộc về một trong các lớp được xác định trước. Trong phân tích quan điểm, các câu đầu vào sẽ được phân loại vào các nhãn quan điểm phù hợp, cụ thể được chia thành hai loại như sau:

- Binary class: Gồm 2 lớp quan điểm
  - Tiêu cực(negative)
  - Tích cực(positive)
- Multi class: Gồm 5 lớp quan điểm
  - Rất tiêu cực(very negative)
  - Khá tiêu cực(somewhat negative)
  - Không tích cực mà cũng không tiêu cực(neural)
  - Khá tích cực(somewhat positive)
  - Rất tích cực(very positive)

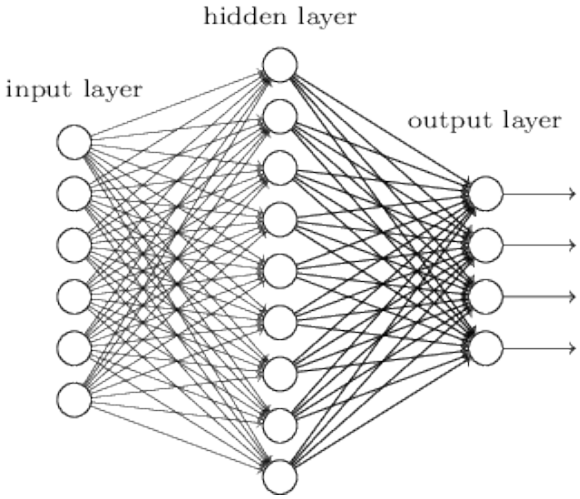
Hiện nay, bài toán đang được áp dụng vào nhiều hệ thống khác nhau:

- Hệ thống xếp hạng sản phẩm dựa trên các nhận xét, bình luận của khách hàng
- Hệ thống gợi ý
- Hệ thống lọc tin rác

1.2 Deep Learning là gì?

Deep learning là một kĩ thuật Machine Learning mạnh mẽ đang được nhiều người trong ngành biết đến và nghiên cứu. Kĩ thuật này nổi trội là do chúng thực hiện được hai việc cùng lúc: biểu diễn thông tin (represent problem/feature engineering) và học (learning). Do đó, kĩ thuật này còn được gọi là representation learning.

Bên cạnh các lĩnh vực đã gặt hái được nhiều thành công như Xử lý ảnh số và video số, hay Xử lý tiếng nói, Deep Learning cũng được áp dụng vào Xử lý ngôn ngữ tự nhiên. Với nền tảng là mạng nơ-ron(Neural Nerwork), Deep Learning hiện đang là một trong những kỹ thuật mạnh mẽ nhất và dữ liệu(Data) là một phần không thể thiếu trong kỹ thuật này. Dưới đây là kiến trúc cơ bản của mạng nơ-ron.



Hình 1.1: Kiên trúc Neral Network

Mạng nơ-ron cơ bản gồm 3 tầng chính:

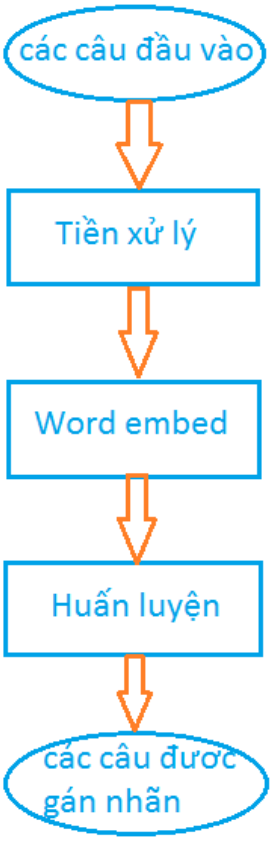
- Tầng đầu vào(input layer): Tẩng biểu diễn thông tin đầu vào, thường được biểu diễn bởi vector đặc trưng cho các thuộc tính của dữ liệu đầu vào
- Tầng ẩn: Biểu diễn, chắt lọc thông tin dữ liệu đầu vào qua các hàm kích hoạt(activation), được liên kết với tầng trước bởi ma trận trọng số.
- Tầng đầu ra(output layer): Biểu diễn thông tin đầu ra, kết quả mà chúng ta mong muốn.

Các nhà khoa học là đã chứng minh rằng trên lý thuyết, với bộ dữ liệu đủ lớn thì chỉ cần một lớp ẩn duy nhất chúng ta có thể giải quyết được tất cả các bài toán. Nhưng trên thực tế để làm được điều này, số lượng nơ-ron ở tẩng ẩn sẽ rất lớn, có thể lên tới hàng tỉ nơ-ron và với tốc độ xử lý của các siêu máy tính hiện tại thì chúng ta không thể giải quyết vấn đề với chỉ duy nhất 1 tầng ẩn, thay vào đó mạng nơ-ron sẽ có thêm nhiều lớp ẩn hơn để phù hợp với từng bài toán.

Chương 2

Mô hình

Mô hình xử lý được sử dụng trong nội dung bài tập lớn như sau



Hình 2.1: Mô hình xử lý

2.1 Tiền xử lý

Quá trình tiền xử lý như sau:

- Loại bỏ các kí tự không nằm trong bảng chữ cái
- Loại bỏ từ dừng
- Xây dựng tập từ điển nhằm phục vụ quá trình mã hóa từ(Word embedding)

Một câu văn đầu vào có thể chứa những kí tự không nằm trong bảng chữ cái, các kí tự này cần được loại bỏ để giảm độ dài câu văn đồng thời tránh học những thông tin không cần thiết. Tương tự với từ dừng, loại bỏ các từ này sẽ không ảnh hưởng đến ý nghĩa của câu.

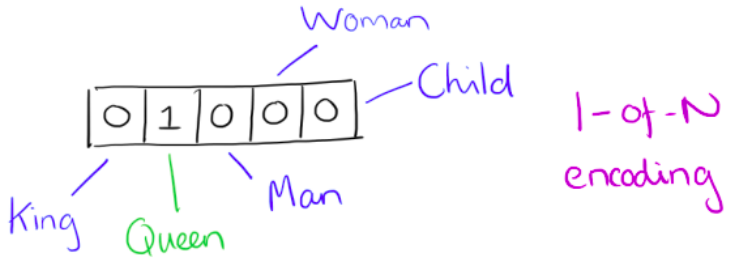
Xây dựng tập từ điển là công đoạn khá quan trọng, các từ xuất hiện trong các câu sau khi đã thực hiện việc loại bỏ kí tự thừa và từ dừng sẽ được đưa vào từ điển. Mỗi từ sẽ ứng với một vị trí duy nhất trong từ điển. Các từ sẽ được mã hóa thành vector tương ứng sẽ được trình bày ở phần tiếp theo.

Để thực hiện qúa trình tiền xử lý này chúng em sử dụng thư viện nltk, một công cụ mạnh mẽ được sử dụng trong xử lý ngôn ngữ tự nhiên

2.2 Word Embedding

Một vấn đề quan trọng trong xử lý ngôn ngữ tự nhiên là làm thể nào để máy tính có thể hiểu được ngôn ngữ của con người? Mã hóa từ (Word embedding) là một kỹ thuật giúp máy tính tiếp cận một cách dễ dàng với ngôn ngữ tự nhiên của chúng ta. Các từ sẽ được mã hóa bởi các vector tương ứng.

Về cơ bản, đây chỉ là một vector trọng số. Ví dụ, 1-of-N (one-hot vector) sẽ mã hoá (encoding) các từ trong từ điển thành một vector có chiều dài N (tổng số lượng các từ trong từ điển). Trong đó, có một phần tử mang giá trị 1 tương ứng với thứ tự của từ đó trong từ điển (ta có thể sắp từ điển tăng dần a-z, A-Z, 0-9) và các phần tử khác đều mang giá trị 0. Giả sử từ điển của chúng ta chỉ có 5 từ: King, Queen, Man, Woman, và Child. Ta có thể biểu diễn từ "Queen" như bên dưới:

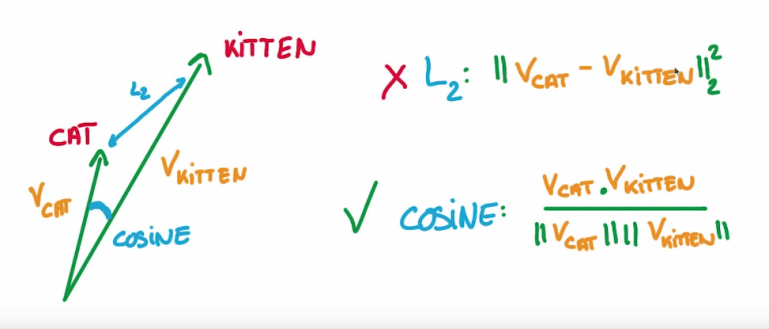


Hình 2.2: One-hot encoding

Với cách mã hoá đơn giản như vậy, ta không thu được nhiều ý nghĩa trong việc so sánh các từ với nhau ngoại trừ so sánh bằng. Mặt khác, nếu kích thước từ điển quá lớn vì việc biểu diễn sẽ vô cùng lãng phí. Word2vec được tìm ra để giải quyết các vấn đề này. Kỹ thuật này giúp chúng ta biểu diễn các từ bởi 1 vector có kích thước nhỏ hơn rất nhiều so với kích thước từ điển và các từ có quan hệ với nhau sẽ được biểu diễn bởi những vector gần nhau. Ví dụ chúng ta có 2 câu:

- The kitten jumps over table
- This cat hunts mice

Từ "kitten" và "cat" đều có nghĩa là "con mèo", vậy chúng cần được biểu diễn bởi 2 vector gần nhau



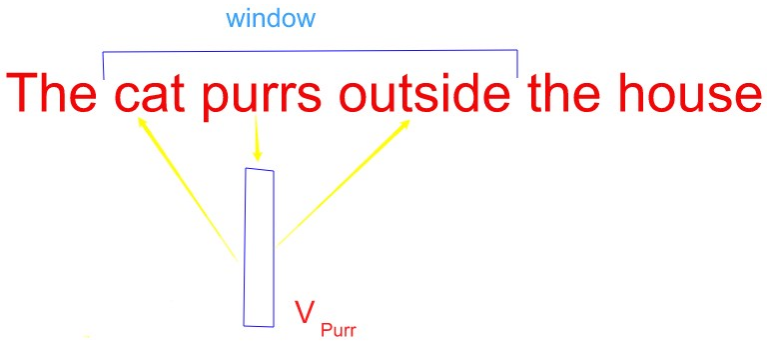
Hình 2.3: Khoảng cách giữa 2 vector

Độ đo *Cosin* sẽ được sử dụng để xác định khoảng cách giữa 2 vector thay vì khoảng cách *Euclid*. Và để có thể tìm ra cách biểu diễn cho các từ, chúng ta cần sử dụng một mô hình học như: $Bag\ of\ words$, $Skip-gram$.

Trong nội dụng bài tập lớn, chúng em sẽ áp dụng mô hình $Skip-gram$ và sẽ được trình bày ở phần tiếp theo.

2.2.1 The Skip-gram model

Xét câu ví dụ sau:

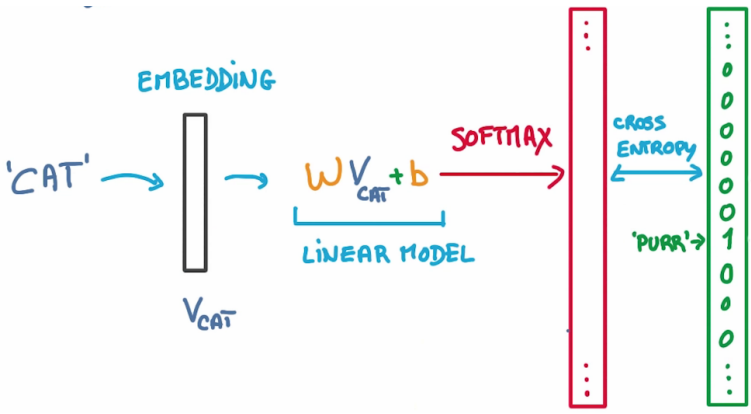


Hình 2.4: Câu đầu vào

Đầu tiên, chúng ta sẽ chuẩn hóa các câu thành các các từ (target) và ngữ cảnh (context) mà nó xuất hiện trong đó. Định nghĩa $Ng\ cnh$ (context) bao gồm những từ bên trái và bên phải của từ target. Với kích thước của sổ bằng 1, ta thu được tập dữ liệu mới là các cặp (context, target) từ cầu đầu vào như sau:

- (the, cat), (purr, cat), (cat, purr), (outside, purr)...

Công việc còn lại là tiến hành dự đoán *context* từ *target* tương ứng



Hình 2.5: The skip-gram model

Với một từ đầu vào, ta biểu diễn từ bởi một vector ngẫu nhiên $v \in R^D$, cho vector đi qua mô hình tuyến tính với 2 tham số (W, b) và kết quả đầu ra thu được bởi hàm $softmax$. Mô hình cần tối ưu các tham số v, w, b để cực tiểu hàm mục tiêu

$$C = -\frac{1}{n}\sum_{x}[ylna + (1-y)ln(1-a)]$$

Trong đó:

- a là vector đầu ra của mô hình
- y là kết quả thực tế, biểu diễn *context* bởi vector one-hot encoding

Quá trình tối ưu được thực hiện bằng ký thuật $Stochasticgradientdescent$. Sau quá trình này, mỗi từ trong tử điển sẽ được biểu diễn bởi các vector tương ứng học được từ mô hình. Sau khi biểu diễn các từ bởi các vector, dữ liệu sẽ được đưa vào quá trình huấn luyện để xác định quan điểm cho từng câu đầu vào. Để thực hiện điều này, chúng ta sẽ sử dụng $RecurrentNeuralNetworks$

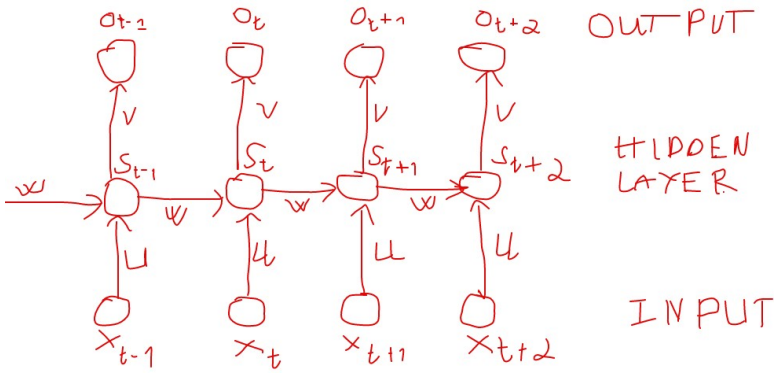
2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) là một trong những mô hình Deep learning được đánh giá có nhiều ưu điểm trong các tác vụ xử lý ngôn ngữ tự nhiên (NLP). Ý tưởng của RNNs đó là thiết kế một Neural Network sao cho có khả năng xử lý được thông tin dạng chuỗi (sequential information), ví dụ một câu là một chuỗi gồm nhiều từ. Recurrent có nghĩa là thực hiện lặp lại cùng một tác vụ cho mỗi thành phần trong chuỗi. Trong đó, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước đó.

Nói cách khác, RNNs là một mô hình có trí nhớ (memory), có khả năng nhớ được thông tin đã tính toán trước đó. Không như các mô hình Neural Network truyền thống đó là thông tin đầu vào (input) hoàn toàn độc lập với thông tin đầu ra (output). Về lý thuyết, RNNs có thể nhớ được thông tin của chuỗi có chiều dài bất kì, nhưng trong thực tế mô hình này chỉ nhớ được thông tin ở vài bước trước đó. Để có khả năng lưu trữ dài hạn, cần có một số cải tiến trong mỗi *cell* RNN và được thể hiện cụ thể trong mô hình *Long short term memory*

2.3.1 Mạng RNN truyền thống

Dưới đây là mô hình Recurrent neural networks truyền thống



Hình 2.6: Kiến trúc Recurrent neural networks truyền thống

- X_t là input tại thời điểm t.
- S_t là trạng thái ẩn (memory) tại thời điểm t.

$$S_t = f(UX_t + S_{t-1}W), \text{ f là hàm tanh hoặc Relu.}$$
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},\ Relu(x) = max(x, 0)$$

- O_t là output tại thời điểm t, $O_t = softmax(VS_t)$

Các tham số U, W, V cần được tối ưu để cực tiểu hàm mục tiêu

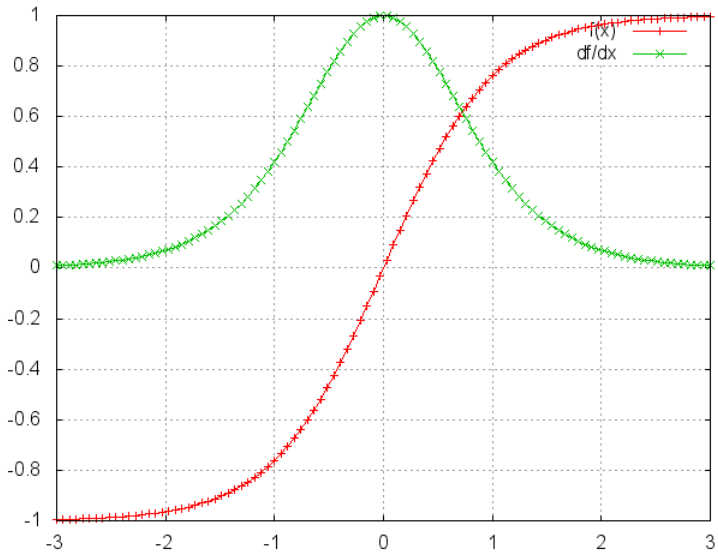
$$C = E(y_t, \hat{y}_t) = -\sum_t y_t log(\hat{y}_t)$$

Huấn luyện RNNs tương tự như huấn luyện Neural Network truyền thống. Ta cũng sử dụng đến thuật toán backpropagation (lan truyền ngược) nhưng có một chút tinh chỉnh. Gradient tại mỗi output không chỉ phụ thuộc vào kết quả tính toán của bước hiện tại mà còn phụ thuộc vào kết quả tính toán của các bước trước đó.

Cụ thể để tính Gradient của hàm mục tiêu đối với các tham số W, U, V ta sẽ thuật toán Backpropagation through time với các công thức như sau:

- $\frac{\partial E_t}{\partial V} = (\hat{y}_t - y_t) \otimes S_t$
- $\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}$
- $\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial U}$

Như đã trình bày ở trên, mạng *RNNs* truyền thống không có khả năng nhớ dài hạn, vấn đề này sẽ ảnh hưởng rất lớn đến kết quả của việc phân loại trong trường hợp các câu quá dài. Vấn đề này được gọi là *vanishing gradient* (biến mất gradient).



Hình 2.7: Đồ thị biểu diễn đạo hàm của hàm tanh

Trong công thức tính trạng thái ẩn $S_t = tanh(UX_t + S_{t-1}W)$, S_t được biến đổi bởi hàm *tanh*. Nhìn vào đồ thị, ta dễ thấy Gradient của hàm *tanh* luôn nằm trong khoảng (0, 1). Mặt khác

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

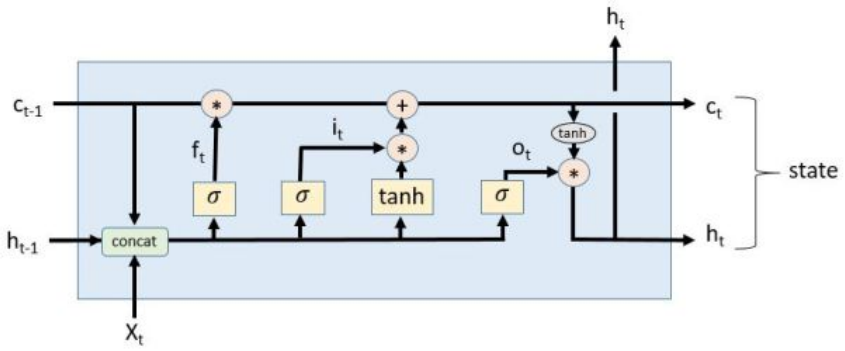
$$\Rightarrow \prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \text{ sẽ tiến tới 0 nếu độ dài câu quá dài.}$$

Vấn đề này sẽ dẫn đến việc huấn luyện sẽ bị dừng lại. *Long short term memory* sẽ giải quyết vấn đề này.

2.3.2 Long short term memory

Mô hình này có cấu trúc tương tự như RNNs nhưng có cách tính toán khác đối với các hidden state. Memory trong LSTMs được gọi là cells (hạt nhân). Ta có

thể xem đây là một hộp đen nhận thông tin đầu vào gồm hidden state s_{t-1} và giá trị x_t. Bên trong các hạt nhân này, chúng sẽ quyết định thông tin nào cần lưu lại và thông tin nào cần xóa đi, nhờ vậy mà mô hình này có thể lưu trữ được thông tin dài hạn.



Hình 2.8: Kiến trúc cell LSTMs

Các biến đổi trong cells cụ thể như sau:

- $i = \sigma(X_t U^i + S_{t-1} W^i)$
- $f = \sigma(X_t U^f + S_{t-1} W^f)$
- $o = \sigma(X_t U^o + S_{t-1} W^o)$
- $g = tanh(X_t U^g + S_{t-1} W^g)$
- $C_t = C_{t-1} \otimes f + g \otimes o$
- $S_t = o \otimes tanh(C_t)$

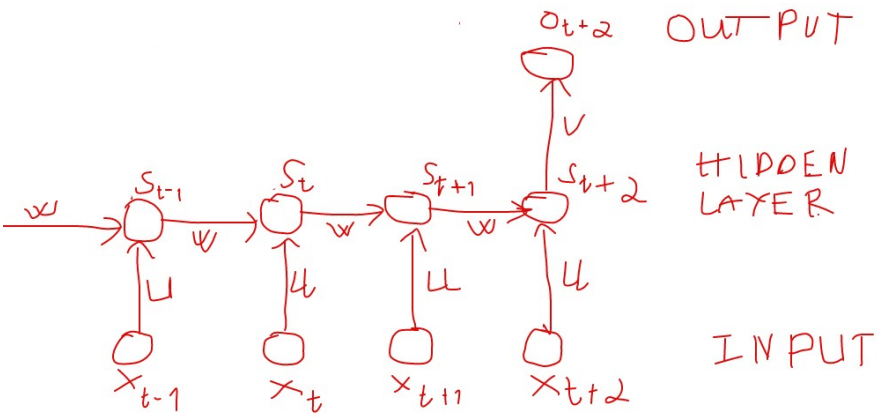
Ba cổng input, output, forget sẽ quyết định bao nhiêu lượng thông tin sẽ nên được lưu giữ lại.

- Cổng input xác định bao nhiêu lượng thông tin tại trạng thái hiện tại sẽ được lưu giữ
- Cổng forget xác định bao nhiêu lượng thông tin trong quá khứ sẽ được giữ lại

- Cổng output xác định bao nhiêu lượng thông tin tính đến thời điểm hiện tại sẽ được truyền đi tới nơ-ron tiếp theo

Mô hình này là một bước đột phá mà chúng ta đạt được từ mô hình RNNs và có thể trong tương lai chúng ta sẽ tim ra được những mô hình tốt hơn.

Dưới đây là mô hình cụ thể mà chúng em sử dụng trong bài toán phân tích quan điểm:



Hình 2.9: Mô hình sử dụng

- X_t là từ thứ t trong câu
- output là vector duy nhất $O \in R^D$
  - $D = 2$ trong trường hợp binary-class
  - $D = 5$ trong trường hợp multi-class

Chương 3

Kết quả thực nghiệm

Thực nghiệm trên 2 bộ dữ liệu Binary-class và multi-class

3.1 Dữ liệu thực nghiệm

Bộ dữ liệu UMICH SI650 (Binary class)

- Binary class
- Xây dựng bởi Đại Học Michigan
- Gồm 7086 câu được gán nhãn 0(negative) hoặc 1(positive)
  - Training dataset: 5000 câu
  - Test dataset: 2086 câu.

Bộ dữ liệu Rotten Tomatoes movie review (multi-class)

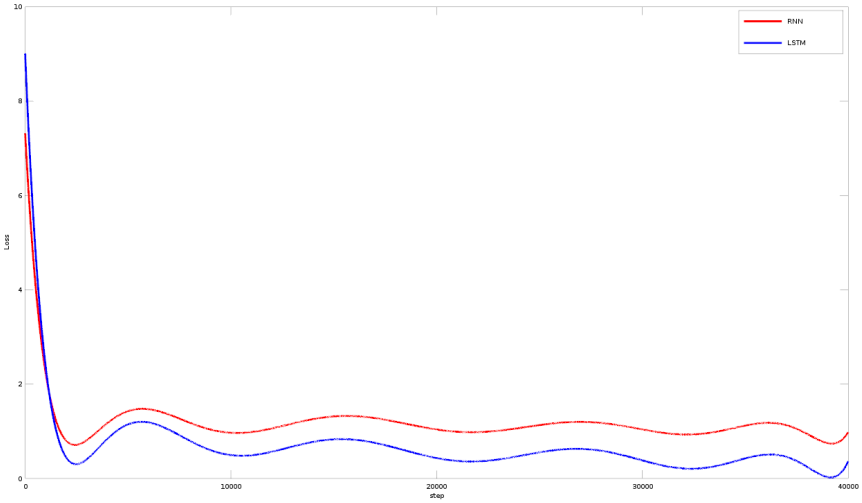
- Multi class
- Xây dựng bởi Pang và L.Lee
- Gồm 150000 câu được gán nhãn từ 0 đến 4
  - Training dataset: 140000 câu
  - Test dataset: 10000 câu.

3.2 Môi trường thực nghiệm

- Hệ điều hành ubuntu 14.04-64 bit
- Intel core i7 - 3540M, CPU 3.0Ghz x 4
- Ngôn ngữ lập trình: python
- Thư viện hỗ trợ: Tensorflow

3.3 Kết quả

So sánh giá trị hàm Loss giữa Standard-RNN vs LSTM trên bộ dữ liệu multi-class



Hình 3.1: Đồ thị hàm loss giữa Standard-RNNs vs LSTMs

Kết quả cho thấy mô hình LSTM có khả năng học tốt hơn so với RNNs
So sánh kết quả giữa Standard-RNN vs LSTM trên 2 bộ dữ liệu.

	Binary-class		Multi-class	
	Traing data	Test data	Training data	Test data
S-RNN	98,76%	95,12%	65,62%	55,50%
LSTM	99,34%	97,56%	85,93%	62,96%

Với bộ binary-class, kết quả thu được khá tốt. Còn với bộ Multi-class, trên tập huấn luyện LSTMs cho kết quả khá tốt, tuy nhiên trên tập test kết quả thu được chưa được tốt. Cần có một mô hình phù hợp hơn với bộ dữ liệu này.

Chương 4

Kết luận và hướng phát triển

Trong nội dung bài tập lớn, chúng em đã xây dựng mô hình để giải quyết bài toán phân tích quan điểm với việc sử dụng *word2vec* để học ra cách biểu diễn của các từ trong câu, kết hợp với mô hình mạng $RNNs$ cũng như $LSTMs$

Tuy nhiên, để cải tiến kết quả trên bộ dữ liệu multi-class sẽ cần phải tìm ra một mô hình phù hợp hơn. Sau khi kết thúc môn học chúng em sẽ tiếp tục cải thiện kết quả với việc sử dụng mô hình $Recursive\ Neural\ Network$, mô hình dựa trên cây phân tích cú pháp. Và hi vọng, kết quả sẽ được cải thiện.

Tài liệu tham khảo

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. & Dean, J., 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, (pp. 3111-3119).

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. & Khudanpur, S., 2010, September. Recurrent neural network based language model. *In Interspeech*, (Vol.2, p.3).

Colah. Understanding LSTM Networks. *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*.

Wildml. Recurrent Neural Network tutorial. *http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/*.