

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────



**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC MÁY**

**Phân loại văn bản tiếng anh bằng thuật toán**

**Neural Network**

**Giáo viên hướng dẫn: TS. Nguyễn Nhật Quang**

**Mã học phần: IT4866, Mã lớp: 95090, Học kỳ: 20162**

**Sinh viên thực hiện – Nhóm: 06**

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **MSSV** | **Lớp** |
| Võ Mạnh Cường | 20140635 | CNTT2.4-K59 |
| Lưu Hữu Phước | 20143546 | CNTT2.3-K59 |

***Hà Nội, tháng 5 năm 2017***

# LỜI NÓI ĐẦU

Sự phát triển nhanh chóng của mạng Internet cùng với những bước tiến mạnh mẽ của công nghệ lưu trữ, lượng thông tin cũng vì vậy nhiều lên theo cấp số mũ. Với các thông tin sinh ra liên tục mỗi ngày trên mạng Internet, con người có thể khai thác và mang lại lợi ích không nhỏ cho nhiều lĩnh vực. Tuy nhiên, nó cũng khiến chúng ta khó khăn trong việc tìm kiếm và tổng hợp thông tin. Giải pháp cho vấn đề này có thể phân loại, tóm tắt, trích rút… Với việc phân loại các tài liệu, người dùng sẽ dễ dàng tập trung vào các chủ đề mà mình quan tâm hơn. Với việc tóm tắt văn bản, người dùng có thể tối giản thời gian hơn nữa cho thời gian đọc các tài liệu. Các phương pháp này cũng đều mang lại lợi ích to lớn cho các bộ máy tìm kiếm có thể đánh chỉ mục hiệu quả hơn.

Trong phạm vi môn học cũng như báo cáo này, chúng em xin trình bày phương pháp Neural network để phân loại văn bản có thể áp dụng vào các công việc như phân loại nội dung trang web.

Để hoàn thành bài tập lớn này, nhóm chúng em đã vận dụng các kiến thức được truyền đạt từ môn học, chúng em xin chân thành cảm ơn sự hỗ trợ của thầy TS. Nguyễn Nhật Quang. Quá trình thực hiện bài tập chắc chắn sẽ còn nhiều thiếu sót, chúng em mong được thầy hướng dẫn thêm để khắc phục và hiểu rõ hơn các vấn đề.

***Hà Nội, tháng 5 năm 2017***

# PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

1, Võ Mạnh Cường

* Lập trình thuật toán
* Thử nghiệm cải tiến thuật toán.

2, Lưu Hữu Phước

* Tiền xử lý
* Kiểm thử, so sánh độ chính xác.

MỤC LỤC

[LỜI NÓI ĐẦU 2](#_Toc483177552)

[PHÂN CÔNG THÀNH VIÊN TRONG NHÓM 3](#_Toc483177553)

[CHƯƠNG 1 GIỚI THIỆU 5](#_Toc483177554)

[1.1 Lý do chọn đề tài 5](#_Toc483177555)

[1.2 Phạm vi thực hiện 5](#_Toc483177556)

[1.3 Tổng quan báo cáo 5](#_Toc483177557)

[CHƯƠNG 2 MÔ TẢ TIỀN XỬ LÝ THUẬT TOÁN 6](#_Toc483177558)

[2.1 Dữ liệu The 20 Newsgroups data set 6](#_Toc483177559)

[2.2 Các bước tiền xử lý 6](#_Toc483177560)

[CHƯƠNG 3 PHƯƠNG PHÁP NEURAL NETWORK 11](#_Toc483177561)

[3.1 Giới thiệu phương pháp 11](#_Toc483177562)

[3.2 Mô hình bài toán áp dụng cho bài toán 14](#_Toc483177563)

[CHƯƠNG 4 CHƯƠNG TRÌNH CÀI ĐẶT 16](#_Toc483177564)

[4.1 Cấu trúc chương trình 16](#_Toc483177565)

[CHƯƠNG 5. KIỂM THỬ VÀ ĐÁNH GIÁ 17](#_Toc483177566)

[5.1 Phương pháp kiểm thử 17](#_Toc483177567)

[5.2. Kiểm thử 17](#_Toc483177568)

[5.2 Đánh giá 19](#_Toc483177569)

[5.3. Hướng phát triển 19](#_Toc483177570)

[CHƯƠNG 6. KẾT LUẬN 20](#_Toc483177571)

[TÀI LIỆU THAM KHẢO 20](#_Toc483177572)

# CHƯƠNG 1 GIỚI THIỆU

## 1.1 Lý do chọn đề tài

Phân loại văn bản là một trong những bài toán kinh điển cho các giải thuật học máy, là bài toán thường được những người mới bắt đầu lựa chọn để tìm hiểu về học máy. Có nhiều giải thuật được dùng để xử lý bài toán phân loại văn bản như K-Nearest Neighbors, Navie Bayes, Support Vector Machine, Neural Network. Trong đó Navie Bayes, Support Vector Machine, Neural Network là có kết quả tốt nhất. Hiện tại, học sâu Deep Learning đang trở thành xu hướng trên toàn cầu mà bản chất chính là Neural Network. Vì vậy bọn em lựa chọn giải thuật Neural Network để giải quyết bài toán phân loại văn bản trong khuôn khổ bài tập lớn này.

## 1.2 Phạm vi thực hiện

Chúng em lựa chọn giải thuật Neural Network để phân loại văn bản dựa trên bộ dữ liệu The 20 Newsgroups data set từ địa chỉ <http://qwone.com/~jason/20Newsgroups/>

## 1.3 Tổng quan báo cáo

Báo cào này gồm năm chương.

Chương thứ nhất là Giới thiệu.

Chương thứ hai là Mô tả phần tiền xử lý đầu vào trước khi đưa vào áp dụng thuật toán xử lý.

Chương thứ ba là Giới thiệu phương pháp Neural Network.

Chương thứ tư trình bày chương trình cài đặt.

Chương thứ năm là đánh giá các kết quả đạt được, khó khăn, hạn chế và hướng phát triển.

Chương thứ sáu là kết luận.

# CHƯƠNG 2 MÔ TẢ TIỀN XỬ LÝ THUẬT TOÁN

## 2.1 Dữ liệu The 20 Newsgroups data set

The 20 Newsgroups data set là một bộ gần 20,000 tài liệu, tương đương với 20 nhóm tài liệu. Bộ dữ liệu được sưu tập từ Ken Lang, sau đó trở thành bộ dữ liệu phổ biến cho các cuộc thí nghiệm các giải thuật học máy như phân loại văn bản text classification hay bài toán phân cụm văn bản text clustering.

Bộ dữ liệu được tổ chức thành 20 nhóm, mỗi nhóm tương ứng với một chủ đề khác nhau. Một vài các nhóm rất gầm với nhau (ví dụ comp.sys.inm.pc.hardware/comp.sys.mac.hardware) trong khi một số khác không liên quan gì đến nhau. Đây là danh sách của 20 nhóm dữ liệu:



## 2.2 Các bước tiền xử lý

Bước 1: Tách mỗi văn bản ra thành một danh sách các từ riêng lẻ. Trong đó mỗi văn bản được xử lý qua các bước sau:

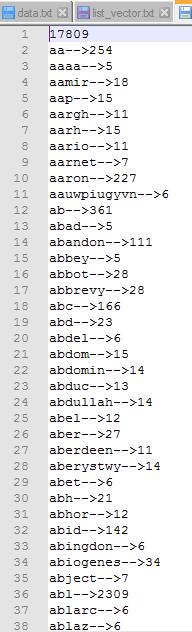
* Đưa các kí hiệu tiền về xâu ‘currency’.
* Bỏ các link, email ra khỏi văn bản.
* Bỏ các xâu kí tự số.
* Thay các xâu "'s", "'re", "'ve", "'d", "n't", "'ll" bởi dấu cách (trong trường hợp viết tắt của các từ tiếng anh)
* Thay các kí tự (symbol) có trong mỗi văn bản bằng dấu cách.
* Tách văn bản thành một danh sách các xâu (tách bởi dấu cách)

Bước 2: Đưa các từ trong danh sách xâu của mỗi văn bản về dạng gốc (ví dụ: talking, talked->talk) và loại bỏ các từ stopword.

* Thường thì chúng ta sẽ loại các từ stopword đầu tiên, nhưng bọn em thấy là do sự hạn chế của danh sách các từ stopword mà cùng với 1 từ gốc nhưng trong danh sách các từ stopword không chứa hết các từ loại của từ gốc ấy, dẫn đến việc nếu loại các từ stopword ở đầu thì sẽ dẫn đến việc để sót 1 số loại từ của từ stopword không lọc được. Vì vậy bọn em đưa các từ về từ gốc trước sau đó mới loại các từ stopword sau.
* Bọn em sử dụng lemmatizer (WordNetLemmatizer) và stemmer (LancasterStemmer) từ thư viện nltk để đưa các từ về từ gốc. Ban đầu bọn em chỉ sử dụng lemmatizer nhưng sau đó kiểm tra dictionary thu được thì thấy hạn chế của lemmatizer là bỏ sót rất nhiều các loại từ (một từ gốc nhưng có những 2,3 từ loại của từ gốc đó xuất hiện trong dictionary).
* Từ 2 nhận xét trên, bọn em đầu tiên là kết hợp lemmatizer và stemmer để đưa các từ trong danh sách từ về từ gốc và loại từ ấy nếu nó chứa trong danh sách các từ stopword (đã lemmatizer và stemmer).

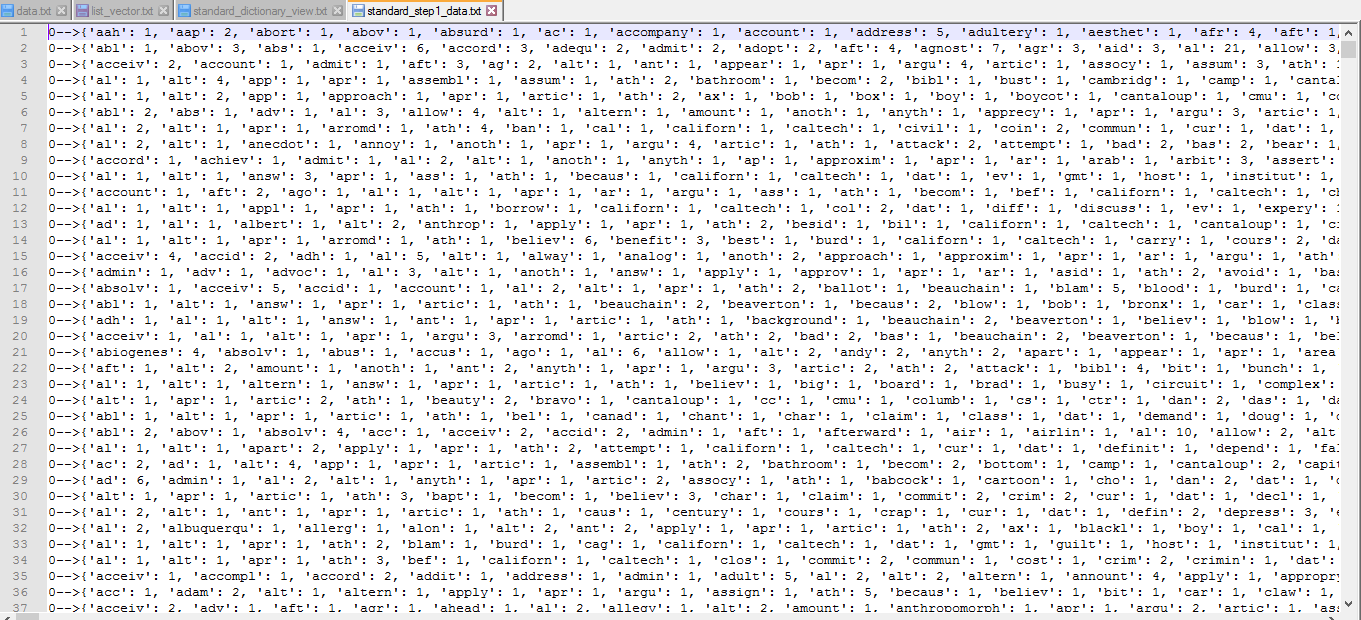
Bước 3: Tạo dictionary

* Đầu tiên bọn em tổng hợp tất cả các từ có trong toàn bộ tập dữ liệu.
* Sau đó bọn em lọc bớt những từ trong từ điển xuất hiện quá ít hoặc quá nhiều (nếu ít hơn 4 lần hoặc lớn hơn 4000 lần). Những từ xuất hiện quá ít hoặc quá nhiều tức là ít có khả năng đặc trưng cho từng chủ đề hay nhóm chủ đề. Vì trung bình mỗi chủ đề chưa khoảng 1000 văn bản, nên 4000 cũng là con số đủ để nó không loại đi những từ đặc trưng cho 1 chủ đề hay nhóm chủ đề.
* Đếm số văn bản mà mỗi từ trong dictionary xuất hiện. Nếu số văn bản mà từ đó xuất hiện quá lớn hoặc quá nhỏ thì nó không đặc trưng cho từng nhóm chủ đề hay nhóm chủ đề. Vì vậy tương tự nhận xét trên ta sẽ loại những từ trong từ điển mà số văn bản mà từ đó xuất hiện nhỏ hơn 4 hoặc lớn hơn 3000.



**Hình 1 Dictionary**

Bước 4: Dựa vào từ điển đã lọc, chúng ta lọc đi dữ liệu của các file chỉ giữ lại các từ có trong từ điển và bỏ đi những file sau khi lọc có số từ quá ít (nhỏ hơn 3) (tránh trường hợp 1 văn bản chỉ chứa 1 câu không liên quan đến nội dung gì của văn bản mà sau khi lọc chỉ còn 2,3 từ)



**Hình 2 Dữ liệu sau khi lọc qua dictionary**

Bước 5: Tạo ma trận dày sử dụng tf-idf

* Tf là term frequency là tấn số xuất hiện của từ xuất hiện trong văn bản. Người ta thường tính tf bằng cách chia số lần xuất hiện của từ trong văn bản cho số lần xuất hiện lớn nhất trong tất cả các từ trong văn bản. tf càng lớn thì từ càng đặc trưng cho văn bản.
* Idf là inverse document frequency tạm dịch là nghịch đảo của tần suất xuất hiện của từ trong toàn bộ văn bản. Người ta thường sử dụng công thức tính idf là lấy lograrit cơ số 10 của thương tổng số các file chia cho số file mà từ xuất hiện. idf càng lớn thì sự đại diện cho văn bản càng lớn.
* Khi tạo đầu vào người ta sử dụng tf-idf với tf-idf =tf\*idf để vecto hóa đầu vào. Dễ thấy ta hoàn toàn có thể làm như vậy vì if-idf tương đương với đầu vào là một văn bản khác mà vẫn giữ nguyên vai trò của các từ đại diện chủ đề (có tỉ lệ xuât hiện cao trong văn bản và tấn suất xuât hiện ít trong các văn bản khác).

Bước 6: Lưu ma trận dày vào file.



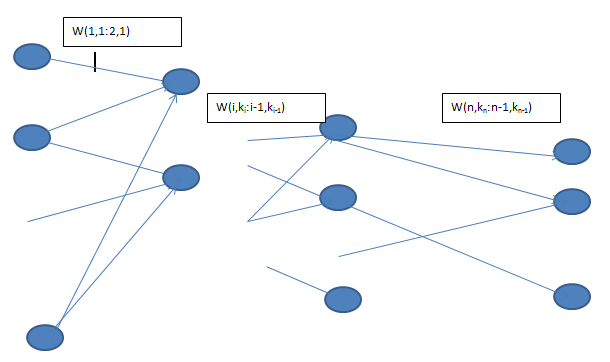
**Hình 3 Ma trận dày**

# CHƯƠNG 3 PHƯƠNG PHÁP NEURAL NETWORK

## 3.1 Giới thiệu phương pháp

Mục tiêu chung của các giải thuật học máy đó là tìm được một hàm F(x) sao cho với đầu vào x thì tìm được đầu ra y mong muốn F(x)

Với phương pháp Neural Network thì người ta tạo ra hàm F thông qua 1 mạng.



**Hình 4 Mô hình mạng**

Với x = [x1, x2, …] là giá trị đầu vào cũng là giá trị của các nút ở tầng đầu vào Input.

* Kí hiệu Neron<i, ki > là neron vị trí thứ ki ở tầng i.

Lưu ý khi viết x<i> chính là xi do giới hạn của word không biết được 2 chỉ dưới!

* Kí hiệu W<i, ki: i-1, ki-1 > là trọng số từ Neron<i, ki> và Neron<i-1, ki-1>
* Kí hiệu z<i, ki> và a<i, ki> lần lượt là giá trị đầu vào tổng thể và giá trị đầu ra của Neron<i, ki >
* Kí hiệu Ki, j = {ki, ki-1} (i chạy từ 2 đến n) là tập hợp tất cả các bộ ki, ki-1 sao cho tồn tại liên kết giữa Neron và Neron.
* Kí hiệu K = {k1, k2, …, kn} là toàn bộ k1, …kn sao cho mỗi bộ k1, …kn tồn tại đồng thời

W<n, kn: n-1, kn-1>, W<n-1, kn-1: n-2, kn-2>, …, W<2, k2:1, k1>

* Kí hiệu Di = { ki / Neron<i,ki> }

Hàm của mạng có dạng:

F(x) = F ({x1, …xn}) = {a<n, kn>, a<n-1,kn-1>, …,a<1, k1>}

Với

a<i,ki> = f(z<i,ki>) = f( W<n,kn:n-1,kn-1> f(

W<n-1,kn-1:n-2,kn-2>…f( W<2,k2:1,k1> x<k1> )…))

f(x) là hàm số liên tục, khả vi trên R, còn gọi là hàm tác động.

Ta cần tìm bộ trọng số W sao cho hàm

F(x) = F({x1,x2, …,xn}) L({x1,x2, …,xn}) = {L1,L2, …,Ln} (1)

Điều này tương đương với tìm bộ trọng số W sao cho hàm:

E(x) = ( L<kn> – a<n,kn> )2 0

E(x) gọi là hàm lỗi. Tất nhiên (1) tương đương với nhiều hàm lỗi E(x) khác.

dE(x) = - (L<kn> – a<n,kn> ) da<n,kn>

= - (L<kn> – a<n,kn> ) f’( z<n,kn> ) dz<n,kn>

dE(x) / dz<n, kn> = -(L<kn> – a<n, kn>) f’ (z<n, kn>) (2)

Xét W<i, ki: i-1, ki-1> bất kì. Ta có:

da<n, kn> = f’(z<n, kn>) W<n, kn: n-1, kn-1> f’ (z<n-1, kn-1>) W<n-1, kn-1: n-2, kn-2>

…W<i+1, ki+1: i, ki> f’ (z<i, ki>) dz<i, ki>

Tương tự, ta có:

da<n, kn> = f’(z<n, kn>) W<n, kn: n-1, kn-1> f’(z<n-1,kn-1>) W<n-1,kn-1: n-2,kn-2>…

W<i+1, ki+1: i, ki> f’ (z<i, ki>) dz<i, ki>

W<i, ki: i-1, ki-1> f’ (z<i-1, ki-1>) dz<i, ki>

dE(x) / dz<i-1, ki-1> =

( (dE(x) / dz<i, ki>) W<i, ki: i-1, ki-1> f’ (z<i-1, ki-1>)) với (3)

Nhìn vào công thức trên ta có thể thấy là khi tính được đạo hàm của hàm lỗi đối với đầu ra của các neron ở lớp sau thì chúng ta sẽ tính được đạo hàm của hàm lỗi đối với các đầu ra của các neron tầng trước.

Mà công thức tính đạo hàm của hàm lỗi đối với đầu ra của tầng cuối cùng ta đã có. Vậy ta có thể tính được tất cả các đạo hàm của các đầu ra ở bất kỳ tầng ẩn nào.

Mặt khác ta có:

dz<i, ki> = a<i-1, ki-1> dW<i, ki: i-1, ki-1> với

Suy ra: dE(x) / dW<i, ki: i-1, ki-1> = a<i-1, ki-1> ( dE(x) / dz<i, ki> ) với (4)

Như vậy ta tính được dW<i, ki: i-1, ki-1> với , ki , ki-1

Hay ta tính được đạo hàm của hàm lỗi đối với tất cả các trọng số W.

Từ đây dựa vào Gradient descent ta hoàn toàn có thể đưa về cực tiểu địa phương.

“Đi theo ngược chiều dấu của đạo hàm, hàm số có xu hướng đi đến được cực tiểu địa phương trong lân cận đấy!”

Vậy ta chỉ cần cập nhật W=W- cho tất cả các trọng số W là có thể cực tiểu hóa hàm lỗi E. ( với trường hợp đơn giản nhất là hằng số dương, còn gọi là learning rate )

Ta có phát biểu sau:

* Bất cứ hàm liên tục bị giới hạn nào cũng có thể học (biểu diễn được) (xấp xỉ) bởi một mạng ANN sử dụng 1 tầng ẩn [Cybenko,1989; Hornik et al, 1988].

Vậy hàm F là đủ tổng quát cho mọi hàm số liên tục!

## 3.2 Mô hình bài toán áp dụng cho bài toán

Trước hết bọn em xin trình bày ý tưởng đề giải quyết bài toán.

Trước hết bọn em xin đưa ra vài nhận xét và mong muốn của mô hình:

* Mô hình sẽ có 20 đầu ra. Tương ứng với mỗi label có hai mươi chiều.
* Mỗi từ đều có trọng số khác nhau đối với mỗi chủ đề (từ này có trọng số cao với chủ đề này nhưng lại có trọng số thấp với chủ đề khác, có nghĩa là trọng số đó là đặc trưng cho chủ đề này mà không đặc trưng cho chủ đề khác). Và trọng số đó là hằng số, không thay đổi.
* Việc của chúng ta là cần tìm các trọng số của mỗi từ ứng với mỗi chủ đề

Để thiết kế được mạng phù hợp thì bọn em xin đưa ra một số nhận định của bản thân như sau:

* Mô hình (hàm số) mình cần học mình cần học là mô hình tuyến tính (hàm số tuyến tính. Vecto từ và vecto trọng số nhân tích vô hướng sẽ được tỉ lệ với đầu ra. Như vậy chúng ta cần 1 hàm kích hoạt đồng biến (đảm bảo tương đối tỉ lệ trước và sau khi đi qua hàm kích hoạt)
* Những từ độc lập với chủ đề thì sẽ có trọng số xấp xỉ không. Vì mạng là feed-forward nên đầu vào là 0 thì đầu ra mong muốn cũng là 0. Điều này tương tương với hàm kích hoạt phải đi qua gốc tọa độ.
* Đầu ra là vecto 20 chiều gồm các số 0 và 1 suy ra hàm kích hoạt phải và bị chặn trên bởi 1.

Với những yêu cầu như trên bọn em lựa chọn hàm tanh (thỏa mãn đồng biến, bị chặn trên bởi 1, đi qua gốc tọa độ). Hàm tanh: f(x) =

* Một số các tham số: mini-batch size = 10, khởi tạo tất cả các trọng số ban đầu bằng 0.

# CHƯƠNG 4 CHƯƠNG TRÌNH CÀI ĐẶT

## 4.1 Cấu trúc chương trình

* Chương trình gồm:
* 1 file tiền xử lý main\_dense.py
* 1 file chạy chương trình chính all.py
* 1 file chứa các hàm bổ trợ my\_lib.py
* 1 file để xử lý stopword (lemmatize, stem) merge\_stop\_words\_file.py
* 1 thư mục stopword (stop-word-list-dir)
* và 2 file stopword lemmatized-merged-stop-word-list.txt và merged-stop-word-list.txt
* 1 thư mục data chứa file từ điển, các file văn bản chỉ còn các từ trong dictionary, file data lưu dữ liệu sau tiền xử lý.
* Yêu cầu cài đặt
* Python 3.6
* Thư viện nltk, matplotlib, numpy.
* Chạy chương trình:
* Chạy file main\_dense.py trước để tiền xử lý
* Chạy file all.py để bắt đầu giai đoạn học.
* Chi tiết file all.py
* Gồm 1 class network (chứa mạng)
* Hàm tanh, và các hàm bổ trợ khác

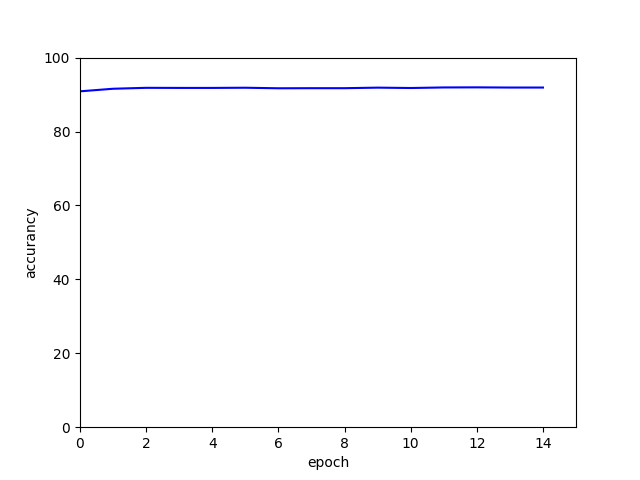
# CHƯƠNG 5. KIỂM THỬ VÀ ĐÁNH GIÁ

## 5.1 Phương pháp kiểm thử

Để đánh giá hiệu năng của hệ thống, chúng em lựa chọn phương pháp đánh giá Cross-validation với thông số k bằng 5.

Ý tưởng của cách đánh giá này là chia tập dữ liệu mẫu thành k phần không giao nhau có kích thước xấp xỉ nhau, sau đó mỗi lần thực hiện thì chọn một trong số k tập làm kiểm thử còn k-1 tập còn lại làm tập học. Như vậy dữ liệu kiểm thử sẽ không bị trùng lặp.

## 5.2. Kiểm thử



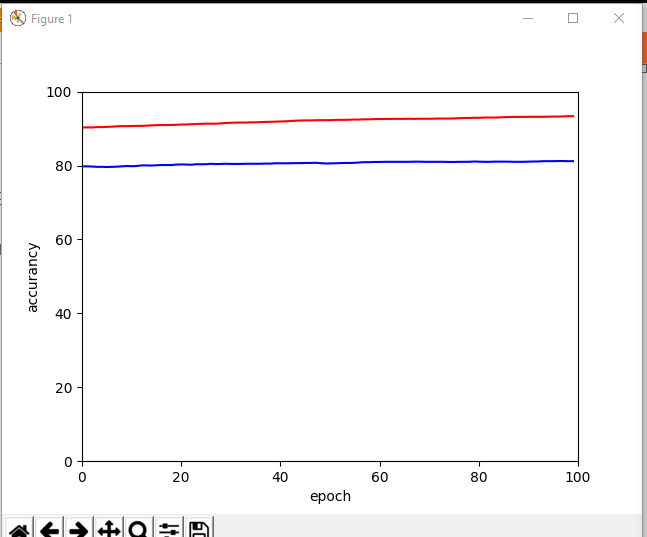
**Hình 5 Accurancy**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bước test | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Trung bình |
| Accurancy | 90,94% | 91,89% | 92,14% | 91,52% | 91,34% | 91,57% |

* Một vài nhận xét:
* Dùng lemmitizer kết hợp stemmer giúp giảm đáng kể kích thước dictionary

Cụ thể giảm từ 24000 từ xuống còn 17000 từ.

* Tách từ tính thêm từ ghép nối (girl-boy, ex-girl) làm từ điển tăng lên đáng kể. Cụ thể từ 17000 từ lên 32000.
* Sau khi cải tiến tiền xử lý tốt hơn (thay thế các dấu câu bởi dấu cách, loại bỏ email, xâu số)



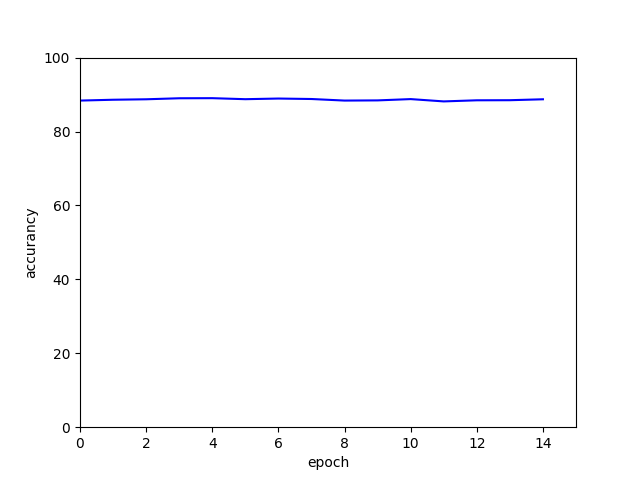
**Hình 6 Accurancy trước và sau khi cải tiến tiền xử lý**

Màu đỏ là sau khi cải tiến, màu xanh là trước khi cải tiến.

* So sánh với các giải thuật khác, cùng dữ liệu tiền xử lý.
* Svm: khoảng 90 % (thư viện svm python)
* KNN: khoảng 86% (bạn Nguyễn Thế Linh lập trình)

**Mạng neron không có tầng ẩn hội tụ nhanh hơn rất nhiều so với mạng neron 1 tầng ẩn mà kết quả không sai khác nhiều!**

* Thí nghiệm giữ nguyên các giá trị khác, thay đổi mini-batch size từ 10 xuống 1. Accurancy giảm xuống còn 88,72%



**Hình 7 Accurancy mini-batch size = 1**

## 5.2 Đánh giá

* Ưu điểm:
* Kết quả thử nghiệm cho kết quả chính xác cao ngang và cao hơn những giải thuật khác, thời gian chạy nhanh hơn so với mô hình mạng neron có tầng ẩn.
* Nhược điểm:
* Tách từ mới chỉ dựa vào từ điển, chưa áp dụng tách từ theo ngữ nghĩa.

## 5.3. Hướng phát triển

* Tách từ theo ngữ nghĩa.
* Thay đổi một vài hàm phát động.

# CHƯƠNG 6. KẾT LUẬN

Chúng em thực sự thấy hài lòng với kết quả thực nghiệm và những trải nghiệm trong quá trình thực hiện. Và một điều quan trọng không kém đó là được áp dụng những gì mình học được vào một bài toán thật sự mà hứa hẹn có nhiều ứng dụng tốt trong nhiều lĩnh vực.

Qua đây, chúng em xin chân thành cảm ơn thầy đã truyền đạt những kiến thức vô cùng bổ ích để chúng em thực hiện tốt đề tài này.

# TÀI LIỆU THAM KHẢO

[1] Bài giảng môn học máy, TS. Nguyễn Nhật Quang, bộ môn Hệ thống thông tin, viện công nghệ thông tin và truyền thông.

[2] http://neuralnetworksanddeeplearning.com/