TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT
VIỆN KỸ THUẬT CÔNG NGHỆ

ĐỒ ÁN MÔN HỌC
NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Đề tài

XÂY DỰNG NGƯỜI BẠN ẢO MIRA

GVHD: Ths NGUYỄN LÊ HIỀN DUYÊN
SVTH: Nguyễn Anh Hào
1824801030064
Trần Tiến Đạt
1824801030161

Tháng 4/2021

VIỆN KỸ THUẬT – CÔNG NGHỆ

NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN

Họ và tên giảng viên: **Nguyễn Lê Hiền Duyên**

Tên đề tài: **XÂY DỰNG NGƯỜI BẠN ẢO MIRA**

Nội dung nhận xét:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:

Bằng số: .....

Bằng chữ: .....

GIẢNG VIÊN CHẤM

(Ký, ghi rõ họ tên)

NGUYỄN LÊ HIỀN DUYÊN

MỤC LỤC

LỜI MỞ ĐẦU

Hiện nay, ở Nhật số người độc thân chiếm phần lớn người dân Nhật Bản. Nguyên nhân cũng chính vì sự phát triển kinh tế, đam mê công việc nên có rất nhiều người chọn phương pháp sống độc thân để dễ dàng cho cuộc sống. Hay cũng có rất nhiều người tuy đã có gia đình nhưng họ đã bỏ gia đình vì đam mê công việc. Nó thật sự ảnh hưởng rất lớn đối với cuộc sống của con người. Mà chúng ta đã biết con người là loài vật sống theo bầy đàng gọi là xã hội, nếu một người không được giao tiếp, nói chuyện với nhiều người khác, họ sẽ cảm thấy khó chịu và bực bội.

Và cũng vì lí do đó, số vụ tự tử ở Nhật tăng lên đáng kể không chỉ vì lí do áp lực công việc mà còn là sự cô đơn ăn xâu trong mỗi con người dù bề ngoài họ luôn mạnh mẽ để trang trải cho cuộc sống bên ngoài nhưng bên trong họ rất cần có người để tâm sự.

Và cũng chính vì những thực tiển trên, có rất nhiều người đã lựa chọn các dịch vụ tốn kém như thuê bạn trò chuyện, thuê bạn gái,... Nhưng thật sự, nó cũng chỉ giúp vơi đi trong họ phần nhỏ mà thôi. Vậy đâu mới là sự lựa chọn đúng đắng?

Đương nhiên việc mở rộng ngoại giao và giao tiếp giữa người với người vẫn tốt nhất nhưng hơn hết, đây cũng là một mãnh ghép lớn cần công nghệ giúp đở. Bỏ qua các công nghệ phần mềm giúp tìm kiếm bạn mới để trò chuyện, với công nghệ trí tuệ nhân tạo, có rất nhiều công ty đã bắt tay vào việc xây dựng người bạn ảo giúp mọi người có thể giải tỏa những tâm sự của mình với trí tuệ nhân tạo.

Và dự án Người bạn ảo Mira cũng là dự án hướng đến việc sử dụng trí tuệ nhân tạo để giải quyết vấn đề trên. Người bạn ảo Mira hoạt động gióng như một trợ lí ảo nhưng nó hướng về trở thành một người bạn thật sự hơn là một người trợ lí giúp ta việc vặt.

Báo cáo bao gồm các phần:

Chương 1. Tổng quan về đề tài

Chương 2. Phân tích thiết kế hệ thống

Chương 3. Vận hành và thử nghiệm

Chương 4. Kết quả đạt được

Tài nguyên: https://github.com/aokidai/Mirachan

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1. Ý tưởng dự án

- Xây dựng một chatbot có khả năng trả lời lại tất cả các câu hỏi tự nhiên của người dùng và trả về câu trả lời bằng ngôn ngữ tự nhiên.
- Học các cách tiếp nhận và phản hồi sao cho phù hợp nhất với câu hỏi
- Phân tích dữ liệu đầu vào và có cách trả lời, phản ứng phù hợp
- Phát triển thêm việc nhận dạng giọng nói sau đó trả về với dạn giọng nói trong tương lai

2. Định nghĩa bài toán

- Tên dự án: Japanese Neural Conversational Model (日本語神経会話モデル)
- Sử dựng dử liệu có sẳn để giúp máy có thể học và đưa ra giá trị đúng nhất có thể
- Input: Ngôn ngữ tự nhiên do con người nhập vào
- Output: Ngôn ngữ tự nhiên trả lời lại câu hỏi của người dùng
- Đây là phần mềm dành cho thị trường Nhật Bản nên ngôn ngữ mạc định là Tiếng Nhật (日本語) (ja)

3. Phương pháp thực hiện

- Dự án được phát triển bởi mô hình Machine Learning với công nghệ Norton Networks
- Sử dụng dử liệu đầu vào của Nagoya University Conversation Corpus (名大会話コーパス) (NUCC)
- Sử dựng TensorFlow và Mecab để phân tích, xây dựng, tính toán các dử liệu.
- Sử dựng NLTK (Natural Language Toolkit) để phân tích ngôn ngữ tự nhiên.
- Ngoài ra có thể sử dựng Nagisa để tách tiếng Nhật theo danh từ, trợ từ, tính từ, trạng từ.
- Thiết bị khởi chạy: Linux/Ubuntu

4. Giới thiệu công cụ

a. Python2

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu. Vào tháng 7 năm 2018, Van Rossum đã từ chức Leader trong cộng đồng ngôn ngữ Python sau 30 năm lãnh đạo.

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

b. TensorFlow

TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu

bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

c. MeCab

Mecab là 1 công cụ phân tách từ tiếng Nhật rất nổi tiếng và hiệu quả (trên 99% với tiếng Nhật).

Bản giới thiệu và hướng dẫn bằng tiếng Nhật tại đây http://code.google.com/p/mecab/ . (tôi không tìm thấy bản tiếng Anh).

Ưu điểm tuyệt với của Mecab là tính mềm dẻo và ứng dụng rất cao. Mecab được xây dựng với phương hướng là tách biệt hoàn toàn chương trình và dữ liệu (từ điển, corpus huấn luyện, các định nghĩa và tham số). Vì thế, chỉ cần thay đổi dữ liệu trong Mecab, ta có thể nhận được những ứng dụng mới một cách hiệu quả.

Trong phần giới thiệu về Mecab, tác giá Kudo Taku đã đưa ra 1 loạt các ví dụ về những ứng dụng của mecab như :

- Thêm nguyên âm cho tiếng Nhật, ví dụ "nhg" sẽ thêm nguyên âm thành "nihongo" (tiếng Nhật).

- Đổi từ bàn phím 9 số sang chữ. Ví dụ chuỗi "226066" sẽ đổi thành "cam on". (phím 2 tương ứng với abc, phím 6 tương ứng với mno).

Từ đặc điểm rất hiệu quả đó của mecab, tôi đang thực hiện việc Việt hoá Mecab bằng cách thay thế các dữ liệu của mecab từ tiếng Nhật sang tiếng Việt.

Khó khăn hiện tại là sự khan hiếm về dữ liệu, khi mà có rất ít từ điển và corpus huấn luyện được công khai trên mạng.

Bằng cách sử dụng dữ liệu của các phầm mềm mở cho tiếng Việt, tôi đã lấy được 1 số dữ liệu cần thiết như từ điển (khoảng 30.000 từ), corpus (khoảng 5000 câu đã tách từ - quá ít so với yêu cầu cần thiết).

Ngoài các ứng dụng tách từ, tôi sẽ tìm hiểu thêm về cách sử dụng mecab để thêm dấu cho tiếng Việt, hay ứng dụng tạo bàn phím gõ tiếng Việt trên điện thoại.

d. NLTK

Natural Language Toolkit (tạm dịch là Bộ công cụ Ngôn ngữ Tự nhiên, hay viết tắt là NLTK) là một bộ thư viện và chương trình dành cho xử lý ngôn ngữ tự nhiên (NLP) thống kê và biểu tượng tiếng Anh, được xây dựng bằng ngôn ngữ Python. Steven Bird and Edward Loper là hai tác giả phát triển NLTK tại Khoa Khoa học Thông tin và Máy tính, đại học Pennsylvania. NLTK bao gồm các minh họa đồ họa và dữ liệu mẫu, đi kèm một cuốn sách chứa các giải thích về khái niệm cơ bản đằng sau các tác vụ xử lý ngôn ngữ được hỗ trợ bởi bộ công cụ, và một cuốn sách hướng dẫn.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. Đối tượng dữ liệu

- Đối tượng dữ liệu được khởi tạo từ file Makefile (Chỉ chạy trên Terminal của Linux)
- Dữ liệu sẽ tạo ra sequence.txt, đây là nơi lưu trử các dữ liệu gốc.
- Dữ liệu Input là các câu hỏi bàng ngôn ngữ tự nhiên mà máy tính cần phải học.
- Dữ liệu Output là các câu trả lời tương ứng của câu hỏi trên.
- Học máy sẽ làm nhiệm vụ học tập các trường hợp chính sát và gần gióng của dữ liệu mẩu.
- Các đối tượng dử liệu không phân chia theo các nhóm. Vì ngôn ngữ tự nhiên không được phân chia nhưng được dùng NLTK sử lí trong bước xử lí.

2. Giải thuật thực hiện

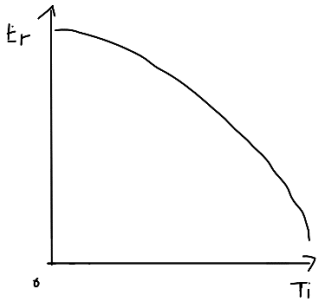
- Giải thuật được sử lí dữ liệu bao gờm 3 bước:
  + Khởi tạo dữ liệu mẫu
  + Tạo Input và Output của PreData
  + Phân tách dữ liệu theo 30000
  + Khởi tạo Data Text và Data Train cho hệ thống
- Khiết chế Wakati được thực hiện bổ sung để tương ứng với tiếng Nhật
- Sử dụng file translate.py với Norton Networks liên kết với 2 file data_utils.py và seq2seq_model.py để chạy Train cho máy học

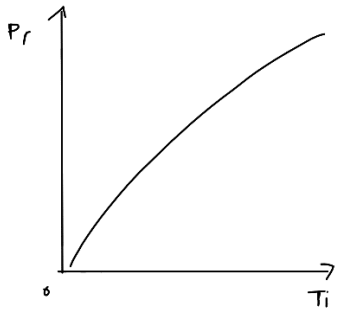
3. Phương pháp hoạt động

- Translate.py là trang khởi tạo chạy chính chương trình

3.1. Tổng quan phương thức hoạt động

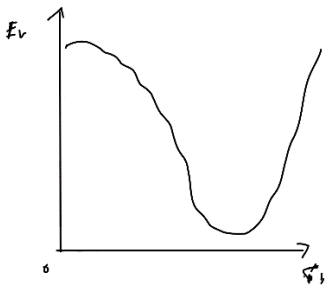
- Không gióng với phân tích tiếng Anh, tiếng Nhật không tách các từ với nhau thong qua dấu cách mà tiếng Nhật tách các từ thông qua cấu trúc ngữ pháp để chia ra thành Danh từ, Động từ, Tính từ và Trợ từ.
- Trong quá trình khởi chạy chương trình, hệ thống hoạt động theo sơ đồ sau

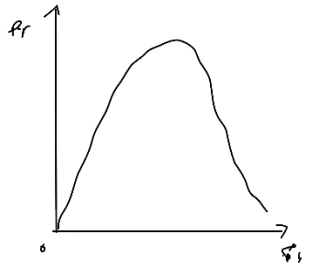




Sơ đồ độ sai lệch Sơ đồ độ chính sát

- Quá trình Training sẽ phải kết thúc cho đến khi độ chính sát và độ sai lệch thấp nhất.
- Tuy nhiên đến một mức độ nào đó, giá trị sai lệch xuống mức âm, sơ đồ sẽ chuyển biết như sau





Sơ đồ độ sai lệch Sơ đồ độ chính sát

- Về mặt dữ liệu, số dữ liệu song song đối ngược với độ sai lệch có nghĩa là nếu gọi a là số dữ liệu, b là trị số sai lệch thì nếu a tăng, b sẽ giảm và ngược lại. Nguyên tác của thuật toán là tìm câu trả lời khớp với câu hỏi nên số dữ liệu mẩu cần là rất lớn.
- Ví dụ: 「ご飯(はん)を食(た)べましたか。」(“Bạn ăn cơm chưa”) hệ thống sẽ tách và hiểu rằng:

ご飯	を	食べ	ました	か	。
Danh từ	Trợ từ	Động từ	Đuôi	Từ để hỏi	Dấu câu

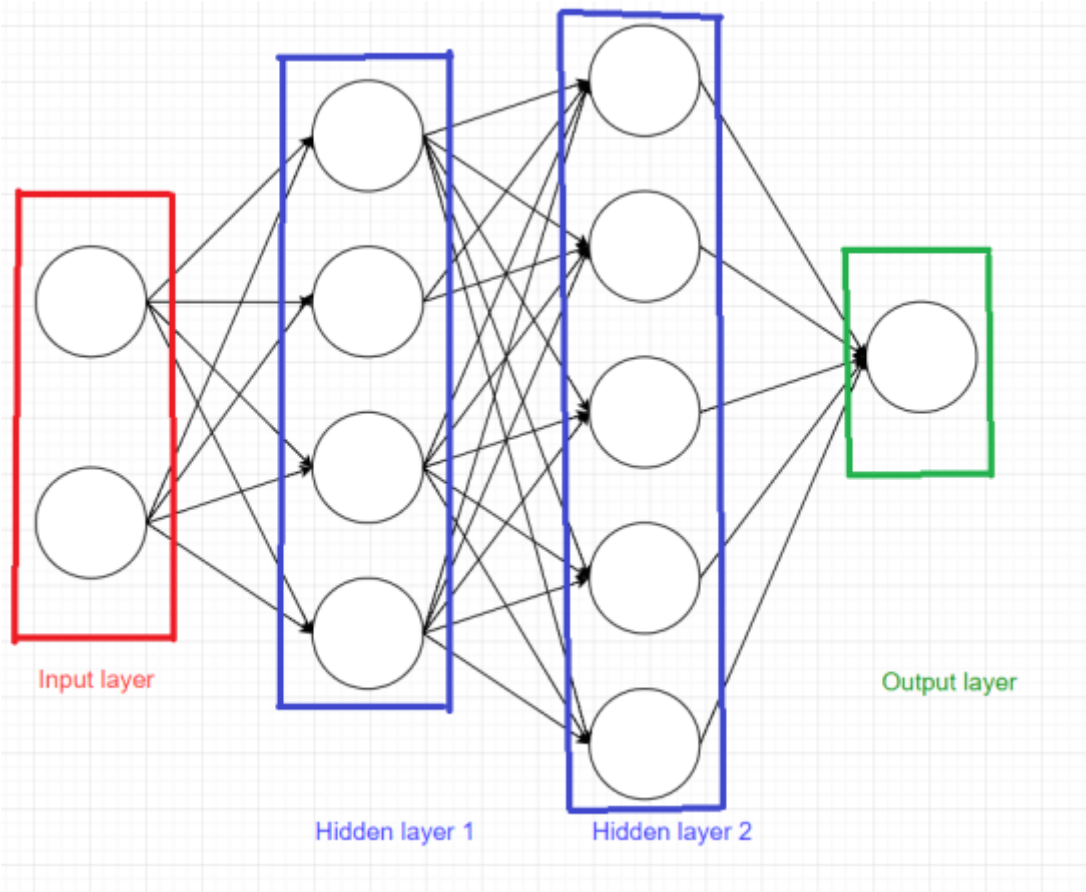
Hệ thống sẽ tìm câu trả lời thích hợp cho câu hỏi trên

ご飯	を	食べ	ました	か	。
Danh từ	Trợ từ	Động từ	Đuôi	Từ để hỏi	Dấu câu
-	-	食べ	ました	はい、	。
Lược bỏ		Động từ	Đuôi	Từ trả lời	Dấu câu

Sau khi sắp xếp lại ta được câu hoàn chỉnh: 「はい、食べました。」

(Vâng, tôi đã ăn.)

- Tương tự như ví dụ trên có rất nhiều mẫu để hệ thống học và hiểu được khi từ đó được hỏi thì từ vựng gì tương ứng để trả lời.
- Và cũng vì lí do đó, số lượng dữ liệu được Training cũng phải ở mức độ nào đó.
- Kì vọng dữ liệu Output, Input vào sẽ được chia ra thành 2 layer để sử lí trên tổng 256 thành phần.



- Phương thức tính toán bằng các công thức tính toán của mạng Norton Networks.

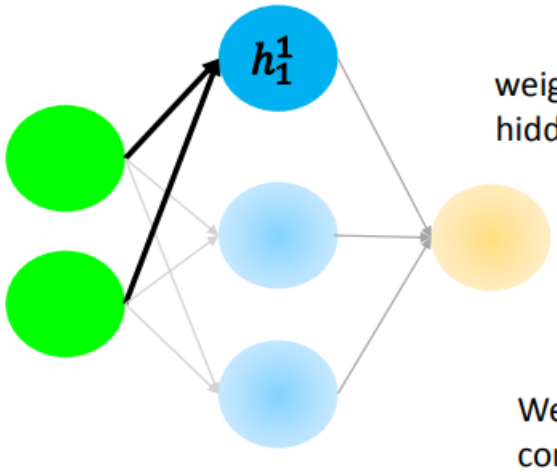
➔ Các phương trình điều chỉnh -> Các nguyên tắc là việc ảo -> TPE tối thiểu

- Quay lại với các layer của norton networks, chúng được tính theo công thức như sau:

For input dataset $X = \{x_1, x_2, \ldots, x_m\}$

➢ The first hidden node:

bias parameters $b^1 = \{b_1^1, b_2^1, \ldots, b_m^1\}$



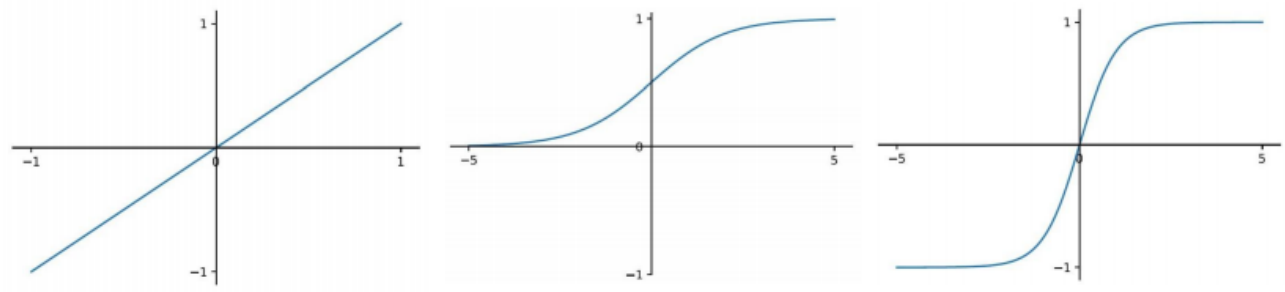
weight set to the 1st hidden neuron node $W^1 = \{w_1^1, w_2^1, \ldots, w_m^1\}$

We have $z^1 = \sum_{j=1}^{m} w_j^1 x_j + b_j^1$

We feed z^1 into activation function *f(.)* to compute hidden node

$$h_1^1 = f(z^1)$$

+ Các Activation function sẽ được biểu diễn như sau:



Identity

$f(x) = x$

Range: (-∞,+∞)

Sigmoid

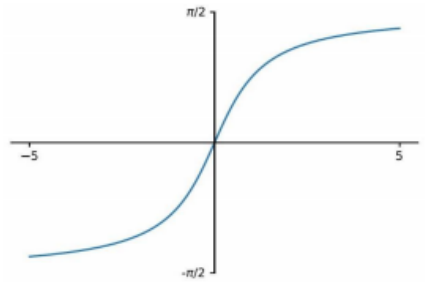
$f(x) = \frac{1}{1 + e^{-x}}$

(0,1)

Tanh

$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

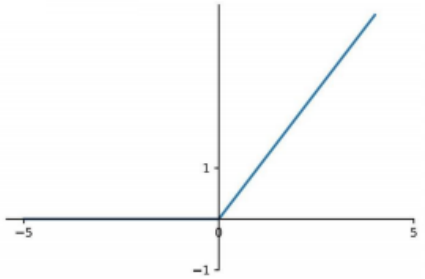
(-1,1)



Arctan

$f(x) = tan^{-1}(x)$

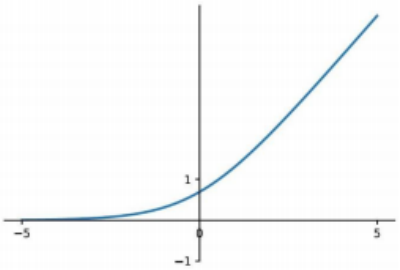
Range: $(-\frac{\pi}{2}, \frac{\pi}{2})$



Rectified linear unit (ReLu)

$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$

[0,∞)



Softplus

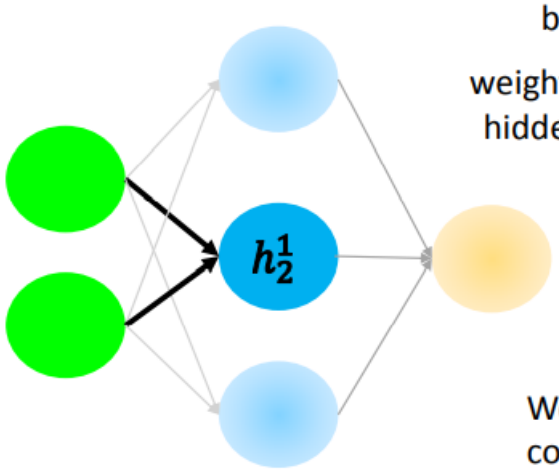
$f(x) = ln(1 + e^x)$

(0,∞)

+ Tương tự ở các nốt sau:

For input dataset $X = \{x_1, x_2, \ldots, x_m\}$

- The second hidden node:



bias parameters $b^2 = \{b_1^2, b_2^2, \ldots, b_m^2\}$

weight set to the 2nd hidden neuron node $W^2 = \{w_1^2, w_2^2, \ldots, w_m^2\}$

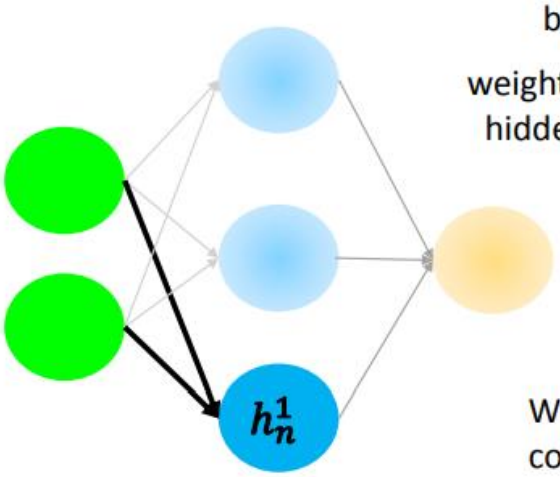
We have $z^2 = \sum_{j=1}^{m} w_j^2 x_j + b_j^2$

We feed z^2 into activation function *f(.)* to compute hidden node

$$h_2^1 = f(z^2)$$

For input dataset $X = \{x_1, x_2, \ldots, x_m\}$

➢ The *n-th* hidden node:



bias parameters $b^n = \{b_1^n, b_2^n, \ldots, b_m^n\}$

weight set to the *n-th* hidden neuron node $W^n = \{w_1^n, w_2^n, \ldots, w_m^n\}$

We have $z^n = \sum_{j=1}^{m} w_j^n x_j + b_j^n$

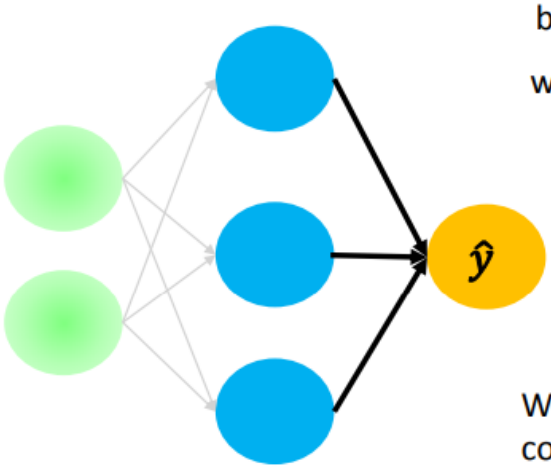
We feed z^n into activation function *f(.)* to compute hidden node

$$h_n^1 = f(z^n)$$

+ Dữ liệu thu được:

So in total, we have *n* hidden outputs $h^1 = \{h_1^1, h_2^1, \ldots, h_n^1\}$

➢ The *output* node:



bias parameters $b^h = \{b_1^h, b_2^h, \ldots, b_n^h\}$

weight set to the *output* node $W^h = \{w_1^h, w_2^h, \ldots, w_n^h\}$

We have $z = \sum_{j=1}^{n} w_j^h h_j^1 + b_j^h$

We feed *z* into activation function *f(.)* to compute the output

$$\hat{y} = f(z)$$

- Trị số sai số (Er) phải giảm dần dưới 0.0010 và trị số chính sát (Pr) phải luôn giữ ở mức ổn định cao trong những lần training.
- Vì số lượng dữ liệu cần là rất lớn và trị số chính sát và sai số cần phải có số lần training rất nhiều nên thời gian training cũng tăng lên đáng kể.

Theo ước tính kì vọng, ta mất 10 tiếng để traning được 10400 global step và learing rate được 0.0543 và preplexity là 2.16.

- Số lược training càng ít thì dữ liệu sẽ bị sai do giá trị bị hỗ loại trong quá trình học. Nếu xem việc training là quá trình học của đứa trẻ thì số tiết học càng ít thì kĩ năng phản xạ và vốn từ sẽ càng ít và ngược lại.

3.2. Cách thức file Translate.py hoạt động

- Các thư viện được sử dụng: Mecab, math, os, random, sys, time, tensorflow, numpy, six và hai file python đó là data_utils và seq2seq_model.
- Tôi sẽ phân tích các bước hoạt động của hệ thống thông qua các hàm

a. Main():

- Đối với hàm main, đây là hàm khởi chạy hệ thống.
- Khi file translate được khởi chạy, nó đặc cờ kiểm tra lại dữ liệu:

  + Nếu Flags.self_test (Cờ đang đặc ở self_test), chương trình sẽ chạy hàm self_test(); (Self_test() sẽ được mô tả tại mục f)

  + Nếu không nó sẽ kiểm tra Flags.decode (Cờ có ở decode()) hay không nếu có hàm decode sẽ khởi chạy; (decode() sẽ được mô tả tại mục e)

  + Nếu cờ không nằm ở đâu, hệ thống sẽ training (Khởi động hàm train); (train() sẽ được mô tả tại mục d)

b. Hàm đọc dử liệu():

- Trước đi vào hàm này ta phải vào file data_utils để xem file có quy tắt chuyển như thế nào.
- Phương pháp đánh số: Dựa trên bản chữ cái Hiragana và Katakana số sẽ được đánh, riêng Kanji, hệ thống sẽ đánh theo phiên âm Hiragana, các dấu câu và các từ Romaji được đánh đầu tiên theo quy tắt. Quy tắt tiên quyết: Số ids không được trùng.
- Ví dụ:

じゃあ	、	なん	か	じゃべっ	て	。
116	4	34	8	483	7	5

- Hàm khởi tạo các ID để phân loại nhóm bao gờm: PAD, GO, EOS, UNK; WORD_SPLIT để chứa các dấu câu _DIGIT_RE chứa giá trị.
- Các hàm chính để data_utils hoạt động:

  + Hàm Mã hóa cơ bản: Giúp phân mãnh không gian các câu thành các từ riêng lẽ và trả về dấu câu cùng với từ phân tách. Kết hàm, return về số lượng từ.

  + Hàm tạo từ vựng:

  - Nếu path từ vựng tồn tại, hệ thống sẽ khởi tạo nó, đưa vào biến data_path để lát ép vào file ids. Tiếp theo hàm sẽ mã hóa theo từng dòng và chuẩn hóa các chữ số. Cuối cùng hàm sẽ viết vào file của từ vựng.

  + Hàm khởi tạo từ vựng:

  - Hàm này sẽ quay lại file của từ vựng lúc nãy khi tạo từ vựng xong, nó sẽ review lại sau đó dùng dict và liệt kê của tensorflow để lấy ra lại giá trị từ vựng.

  + Hàm mã hóa cấu trú ids: gióng tên gọi của hàm, hàm sẽ cấu trúc lại dữ liệu đã mã hóa và lại các UNK và DIGIT_RE.

  + Hàm mã hóa dử liệu ids: Sau khi khởi chạy dử liệu từ data_path và targer_path, hàm sẽ ghi lại dữ liệu của hàm mã hóa cấu trúc ids sau đó ghi lại giá trị mã hóa vào tokens_file

  + Hàm chuẩn bị dữ liệu: Đây là nơi tạo, ghi các giá trị khi các hàm khác tạo ra theo cấu trúc hệ thống của tensorflow hổ trợ.
- Quay lại file translate và hàm đọc dữ liệu, đây là hàm lấy dữ liệu để khởi chạy. Nó mở dử liệu source và target lên để đọc từng dòng
- Sau đó nó sẽ biến dữ liệu thành ids (đánh số id cho từ trong câu). Hàm sẽ duyệt từng dòng và sau đó trả về các số ids ra file xxx_xxx_ids.txt.
- Giá trị trả về là data_set.

c. Tạo lớp model()

- Hàm sẽ sử dụng file seq2seq_model để hoạt động. Seq2seq là file bao gồm 3 phần: bộ mã hóa (Encoder), véc tơ mã hóa trung gian (Encoder vector) và bộ giải mã (Decoder) của Norton Networks.

- **Bộ mã hóa – Encoder**

Một ngăn xếp chứa các mạng con là phần tử của RNN (hoặc các ô nhớ của LTSM hay GRU) nơi nhận vào tín hiệu của một phần tử của chuỗi đầu vào và truyền tiếp về phía cuối mạng.

Trong bài toán hỏi đáp, chuỗi đầu vào là tập hợp tất cả các từ của câu hỏi. Mỗi từ được thể hiện bởi *x_i* với *i* là thứ tự của từ đó.

Các trạng thái ẩn *h_i* được tính với công thức:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

Công thức đơn giản này mô tả kết quả của một mạng nơ ron hồi quy (RNN) thông thường. Như chúng ta có thể thấy, các trạng thái ẩn được tính bởi đầu vào tương ứng (*x_t*) và trạng thái ẩn trước đó *h_(t-1)*.

- **Véc tơ mã hóa trung gian – Encoder vector**

Đây là trạng thái ẩn nằm ở cuối chuỗi, được tính bởi bộ mã hóa, nó cũng được tính bởi công thức phía trên.

Véc tơ này có chức năng gói gọn thông tin của toàn bộ các phần tử đầu vào để giúp cho bộ mã hóa dự đoán thông tin chính xác hơn.

Véc tơ này sau đó hoạt động như trạng thái ẩn đầu tiên của bộ giải mã.

- **Bộ giải mã – Decoder**

Một ngăn xếp các mạng con là phần tử của RNN có nhiệm vụ dự đoán đầu ra *y_t* tại thời điểm *t*.

Mỗi phần từ này nhận đầu vào là trạng thái ẩn trước đó và tạo kết quả đầu ra cũng như trạng thái ẩn của chính nó.

Trong bài toán hỏi đáp, chuỗi đầu ra là tập hợp các từ của câu trả lời. Mỗi từ được biểu diễn bởi *y*_i với *i* là thứ tự của từ.

Các trạng thái ẩn *h_i* được tính bởi công thức:

$$h_t = f(W^{(hh)} h_{t-1})$$

Như chúng ta thấy, các trạng thái ẩn được tính bởi trạng thái ngay trước đó.

Đầu ra *y_t* tại thời điểm *t* được tính bởi công thức:

$$y_t = softmax(W^S h_t)$$

Và kết quả mong đợi chính là mô hình các thông số mới vả retunt lại model.

d. Train()

- Đây là hàm quan trộng nhất của hệ thống.
- Trước tiên hàm sẽ chuẩn bị lại hết dữ liệu
- Sau đó hàm sẽ khởi tạo các layer của các đơn vị. Đối với lượng dữ liệu đầu vào hiện tại, hệ thống có thể chuẩn bị được 2 layer của 256 đơn vị
- Bây giờ hệ thống sẽ đọc dữ liệu phát triển và training dữ liệu thông qua vòng lập:

  + Dữ liệu sẽ được lấy random để training

  + Sau đó hệ thống sẽ thu lại được các trị số bao gờm: “Global step”, “Learining rate”, “Step-time” và “Perplexity”.

  + Dữ liệu sau khi training xong được lưu lại thành file Translate.ckpt

  + Sau cùng sẽ lấy ra được trị số sai số và độ chính sát để căng cứ vào đó để biết độ chính sát của dử liệu đầu ra.

e. Decode()

- Đây là hàm chạy chương trình. Tức là sau khi chương trình hoàng tất, ta có thể dùng lệnh: 「python translate.py –decode」 để chạy lên.
- Hàm lấy dử liệu model từ hàm tạo lớp model, dử liệu vào từ file “vocab_int.txt” và xuất dữ liệu từ file “vocab_out.txt”

- Vào vòng lập của cấu trúc, cấu trúc gán lại bằng hàm wakati() (wakati() sẽ được giải thích ở mục g) lấy dữ liệu cấu trúc sau đó lấy token_ids và bucket_id để xử lí đưa rra dữ liệu phù hợp.
- Sau cùng biến cấu trúc lại được gán thành sys.stdin.readline() để tiếp tục vòng lập.

f. Self_test()

- Đây là hàm tự kiểm tra lại mô hình thần kinh của hệ thống.
- Hệ thống sẽ gán model vào lại seq2seq_model và tiến hành chạy global_varibles_initializer theo thư viện tensorflow.
- Data_set được đưa về thành các ma trận dữ liệu để kiểm tra.
- Phương pháp: Chọn random giá trị sau đó đưa vào hàm get_batch các giá trị để cuối cùng thu được model.step()

g. Wakati()

- Đây là hàm của Mecab, nó có chức vụ tách từ, đối với giá trị đầu vào dạng string, giá trị trả về m.parse(wakatext) sẽ là văn bản được chia sẻ.

4. Các bước thực hiện

4.1. Tôi đã thực hiện

a. Chuẩn bị

- Trước hết ta cần phải đi sơ qua về TensorFlow để khai thác chúng
- Bản TensorFlow được thích nghi với hệ thống là bản v0.12
- Wakati và các hàm đặc biệt khác được thêm vào code để tương thích với tiếng Nhật
- Cấu hình của chương trình bao gờm Python 2.7, Ubuntu, pip.
- Để có thể chạy được code ta cần các thư viện cần thiết, các thư viện đó đã được tạo trong file “requirements.txt”, các thư viện bao gồm TensorFlow v0.12, NLTK v3.3 và Mecab-python v0.996.
- Ta có thể chạy lệnh pip install -r requirements.txt. Tuy nhiên đến hiện tại, Mecab-python đã không còn có thể cài đặc được trên Python2 nên để tối ưu nhất tôi khuyến nghị nên cài thủ công:

  + Cài TensorFlow:

```
pip install tensorflow==0.12
```

+ Cài NLTK:

```
pip install nltk==3.3
```

+ Cài Mecab:

```
sudo apt-get install swig limecab-dev mecab-ipadic-utf8
pip install unidic-lite
pip install –no-binary :all: mecab-python
pip install mecab-python==0.996
```

- Nếu vẫn không thể cài được có thể tham khảo: https://gist.github.com/nwiizo/35cd20f6b36a5934d2a8cd0967f59718

b. Tạo dữ liệu

- File dữ liệu góc được lấy trong file “make-meidai-dialogue”
- Khởi chạy file Makefile bằng Terminal của Ubuntu:

```
make -f Makefile
```

- Sau đó copy file “sequence.txt” vào mục /predata
- Khởi chạy file “data_prepro.py” trong thư mục Japanese-Neural-Conversational-Model/predata để tạo ra 2 file “input/output.txt”
- Phân tách dữ liệu bằng cách dùng lệnh:

```
split -l 30000 input.txt
split -l 30000 output.txt
```

- Đổi tên file “input/output.txt” thành 4 file “train/text_data_in/out.txt” rồi duy chuyển vào file “data”.
- Đổi tên file “data” thành “datas” và tạo thêm tệp ids và vocab trống.
- Ta thu được cấu trúc thư mục con như sau:

```
Japanese-Neural-Conversational-Model
├── datas
│   ├── test_data_ids_{in, out}.txt
│   ├── test_data_{in, out}.txt
│   ├── train_data_ids_{in,out}.txt
│   ├── train_data_{in, out}.txt
│   └── vocab_{in, out}.txt
│
├── predata
│   ├── data_prepro.py
│   └── sequence.txt
│
├── data_utils.py
├── seq2seq_model.py
└── translate.py
```

c. Training

- Để training dữ liệu, ta dùng lệnh:

  python translate.py

- Việc học sẽ được bắt đầu:

```
global step 82700 learning rate 0.0844 step-time 0.19 perplexity 2.46
  eval: bucket 0 perplexity 122.06
  eval: bucket 1 perplexity 242.77
  eval: bucket 2 perplexity 108.74
  eval: bucket 3 perplexity 126.07
global step 82800 learning rate 0.0836 step-time 0.17 perplexity 2.07
  eval: bucket 0 perplexity 21.42
  eval: bucket 1 perplexity 501.38
  eval: bucket 2 perplexity 145.67
  eval: bucket 3 perplexity 210.59
global step 82900 learning rate 0.0836 step-time 0.19 perplexity 2.49
  eval: bucket 0 perplexity 67.73
  eval: bucket 1 perplexity 66.10
  eval: bucket 2 perplexity 183.61
  eval: bucket 3 perplexity 26.17
global step 83000 learning rate 0.0827 step-time 0.18 perplexity 2.46
  eval: bucket 0 perplexity 100.90
  eval: bucket 1 perplexity 22.86
  eval: bucket 2 perplexity 57.99
  eval: bucket 3 perplexity 108.47
```

d. Khởi chạy

- Để có thể sử dụng được ta sẽ dùng lệnh:

  python translate.py –decode

- Kết quả mong đợi sẽ là:

```
> 名前は？
カーミちゃん。

> 何歳なの？
知らない。

> 元気にしてた？
してないね。

> 怒ってる？
ううん。

> カルシウム足りてないんじゃない？？
うん、たぶんね。

> 好きな食べ物って何かある？
おいしい、テレビで。
```

e. Giấy phép:

- Giấy phép của tập dữ liệu đã sử dụng tuân thỉ theo bản góc.
- 使用(しよう)したデータセットのライセンスはオリジナルに従(したが)います。

4.2. Sau khi đã training

- Dữ liệu và có thể khởi chạy bình thường bằng cách:

```
python translate.py --decode
```

CHƯƠNG 3. VẬN HÀNH VÀ THỰC NGHIỆM

1. Vận hành

a. Quá trình vận hành

- Quá trình training là quá trình mất rất nhiều thời gian để đến được giá trị kì vọng.
- Tuy nhiên cũng vì mất nhiều thời gian nên khi training được ít dữ liệu, kết quả trả về sẽ bị sai.
- Theo ước tính đến hiện tại, thời gian training trung bình 24 tiếng để đạt được 15% cho độ chính sát dữ liệu thì ta có thể tính ra được Er sẽ là 0.032 và Pr sẽ là 2.16 cho tổng quá trình 4 pha, 1 pha Pr trung bình sẽ là 136.5769 ≈13.6%. Tương đương 25000 bước chạy (Kết quả kì vọng)
- Khi dữ liệu càng được training nhiều độ chính sát sẽ tăng lên.

b. Kết quả vận hành

- Kết quả thực tế khi training 10 tiếng, ta đạt được Pr = 2.13 tương đương 7% kết quả dữ liệu, tỉ số Er = 0.0554.
- Training được 10400 bước chạy.
- Số lượng chạy ít nên kết quả chỉ có 0.001% là chính sát.
- Ví dụ:

- Ta nhận thấy dữ liệu sai hoàn toàn.

c. Nguyên nhân:

- Khi dữ liệu học ít, hệ thống không được nhận biết nhiều dữ liệu mẫu nên câu trả lời bị sai.
- Nhìn kĩ vào cấu trúc của Output sẽ bị sai theo nguyên tắt lập rất nhiều từ rồi tạo thành một câu không ý nghĩ. Nguyên tắt của sử lí ngôn ngữ tự nhiên là tách các từ để biết và để có câu trả lời chính sát tuy nhiên nếu từ vựng đó chưa được hệ thống nhận biết, nó sẽ không biết nó là gì thì câu trả lời sẽ sai.

- Cũng gióng như các hệ thống AI lớn của Google, Apple, Amazon, Microsoft,... ta cần cho hệ thống Training nhiều hơn để có lượng kiến thức cho máy tính phong phú hơn thì % sai lệch sẽ giảm.

2. Thực nghiệm

- Nếu số lượng Training tăng lên, kết quả kì vọng sẽ như sau:

>こんにちは

こんにちは！

>失礼(しつれい)ですが、お名前(なまえ)は？

みらと申(もう)します。

- Trường hợp nếu dữ liểu Input sai, Output sẽ là chia thành 2 trường hợp:

+ TH1: Input sai hoàn toàn và không thể hiểu, Output sẽ ở dạng 「すみません！よくわかりません。」

+ TH2: Input sai bán phần, Output sẽ dựa theo những từ trong câu hỏi để trả lời. Tuy nhiên kết quả có thể đúng có thể sai tùy theo mức độ hợp lí và tính đúng của Input.

KẾT LUẬN

1. Kết quả đạt được:

- Biết cách xây dựng, thiết kế kiến trúc Machine Learning với kĩ thuật Norton Networks
- Đã tạo ra được một chatbot có thể hoạt động và hỗ trợ cho người dùng
- Thành thạo thao tác trên Python và các thư viện TensorFlow, NLTK, MeCab và Nagisa
- Thành thạo việc sửa lỗi, xây dựng, thây đổi kiến trúc Norton

2. Việc chưa làm được:

- Phần trăm sai sót và phần trăm chính sát chưa cao (Do chưa Training được nhiều dữ liệu)
- Chưa có giao diện
- Chỉ có thể chạy được trên Terminal của Ubuntu

3. Hướng phát triển của đề tài

- Trong tương lai, hệ thống sẽ được training nhiều dữ liệu hơn
- Tạo ra giọng nói cho AI
- Cho AI có khả năng lắng nghe người dùng
- Giảm độ phức tạp của giải thuật và tăng các giá trị, các dữ liệu lên

➢ Giao diện đẹp, thân thiện với người dùng, có nhiều chức năng và tương tác tốt hơn với người dùng.

➢ Thiết kế, xây dựng và lập trình bổ sung các tính năng còn thiếu hay chưa thật sự hoàn chỉnh trong quá trình sử dụng.

TÀI LIỆU THAM KHẢO

- Sách Python と Keras によるディープラーニング
- Sách Python (パイソン) 機械学習 (きかいがくしゅう)
- Slide Time series forecasting for the solution of Poisson problems with crack singularities – Nguyen Xuan Hung
- Các webside:

  + https://www.tensorflow.org/?hl=ja

  + https://www.mlab.im.dendai.ac.jp/~yamada/ir/MorphologicalAnalyzer/MeCab.html

  + https://www.nltk.org/

  + https://nagisa-inc.jp/

  + https://mmsrv.ninjal.ac.jp/nucc/

  + https://qiita.com/kamihork/items/a875cb26ee2c1444f08b

  + https://qiita.com/Pu-of-Parari/items/62d7226814a6aba98354

  + https://yukituna.com/2597/

  + https://catindog.hatenablog.com/entry/2016/12/11/205637