# **Ứng dụng Deep Learning xây dựng bộ dịch Tiếng Việt mới về Tiếng Việt cũ**

Xin chào các bạn. Nếu các bạn đã theo dõi trong bài viết trước của mình [Thay đổi cách viết Tiếng Việt - là dân công nghệ thì không phải sợ](https://viblo.asia/p/thay-doi-cach-viet-tieng-viet-la-dan-cong-nghe-thi-khong-phai-so-ORNZqPaNK0n) thì có thấy là mình đã giới thiệu một ứng dụng nhỏ đó chính là chuyển đổi từ Tiếng Việt cũ về Tiếng Việt mới và trong phần cuối cùng của bài viết đó mình có đề cập đến một phần mới đó là ứng dụng chuyển đổi từ Tiếng Việt mới về Tiếng Việt cũ và cũng đã giới thiệu một mô hình Deep Learning được gợi ý để thực hiện vấn đề này đó chính là LSTM - Long Short Term Memory tuy nhiên mình vẫn chưa có thời gian viết rõ ràng cho các bạn. Và bài viết này giúp giải thích rõ ràng hơn về vấn đề đó. Mình tin rằng đây sẽ là một chủ đề hết sức thú vị đó. OK chúng ta bắt đầu thôi các bạn nhé.

# **Tổng quan về Tiếng Việt mới.**

Trước hết cần phải nói lại một chút về vấn đề này, dù rằng cộng đồng mạng phản đối khá gay gắt về vấn đề này tuy nhiên đứng trên khía cạnh kĩ thuật mà nói thì chúng ta thấy có rất nhiều ứng dụng hay ho có thể thực hiện từ đề xuất cải tiến Tiếng Việt mới này. Tất nhiên ý kiến này của mình chỉ dựa trên quan điểm kĩ thuật thôi nhé các bạn. Giống như [bài trước](https://viblo.asia/p/thay-doi-cach-viet-tieng-viet-la-dan-cong-nghe-thi-khong-phai-so-ORNZqPaNK0n) mình đã đề cập khá kĩ về phương pháp chuyển đổi của GS Bùi Hiển. Trên khía cạnh kĩ thuật các bạn có thể thực hiện nó dễ dàng bằng một Regex như sau:

class VnConverter(object):

def \_\_init\_\_(self, str):

self.str = str

self.maps = [

['k(h|H)', 'x'],

['K(h|H)', 'X'],

['c(?!(h|H))|q', 'k'],

['C(?!(h|H))|Q', 'K'],

['t(r|R)|c(h|H)', 'c'],

['T(r|R)|C(h|H)', 'C'],

['d|g(i|I)|r', 'z'],

['D|G(i|I)|R', 'Z'],

['g(i|ì|í|ỉ|ĩ|ị|I|Ì|Í|Ỉ|Ĩ|Ị)', r'z\1'],

['G(i|ì|í|ỉ|ĩ|ị|I|Ì|Í|Ỉ|Ĩ|Ị)', r'Z\1'],

['đ', 'd'],

['Đ', 'D'],

['p(h|H)', 'f'],

['P(h|H)', 'F'],

['n(g|G)(h|H)?', 'q'],

['N(g|G)(h|H)?', 'Q'],

['(g|G)(h|H)', 'g'],

['t(h|H)', 'w'],

['T(h|H)', 'W'],

['(n|N)(h|H)', 'n\'']

]

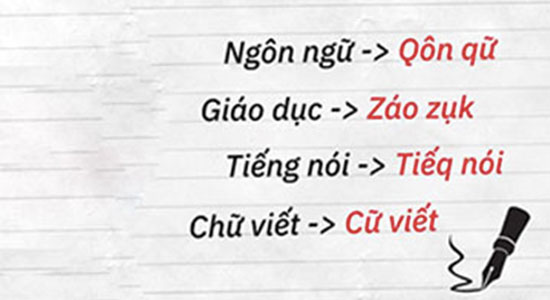
Về mặt toán học chúng ta có thể thấy đây là một ánh xạ từ một tập từ vựng vocabulary trong Tiếng Việt hiện hành sang một tập từ vựng mới nhỏ hơn tạm gọi là vocabulary Bùi Hiển. Với biểu thức chính quy như trên chúng ta có thể dễ dàng thực hiện việc chuyển đổi này bằng một hàm như sau. Các bạn viết trong cùng class VnConverter bên trên nhé:

def convert(self):

for map in self.maps:

self.str = re.sub(re.compile(map[0]), map[1], self.str)

return self.str

Việc chuyển đổi này được coi là một hàm tuyến tính và chúng ta có thể có kết quả như sau:  Tuy nhiên như chúng ta đã biết. Chính vì phương pháp này làm giảm đi số lượng kí tự nên rõ ràng việc chúng ta cố tình suy đoán xem chữ mới này biếu diễn bằng chữ cũ nào là một việc vô cùng khó khăn hay nói cách khác chúng ta không thể sử dụng một biểu thức chính quy để chuyển đổi một cách tuyến tính được. Đây là một biến đổi dạng phi tuyến và là một vấn đề khá là khoai tây nếu như chúng ta không áp dụng các phương pháp của học máy hiện đại. Chúng ta sẽ cùng thảo luận kĩ hơn về vấn đề này trong phần sau nhé

# **Chuyển chữ mới về chữ cũ - làm như thế nào???**

Như mình đã đề cập bên trên thì việc chuyển đổi từ chữ mới về chữ cũ có thể được biểu diễn bằng một hàm như sau:

old\\_string=f(new\\_string)

*old*\_*string*=*f*(*new*\_*string*)

Một câu hỏi cho các bạn là hàm f trên là gì. Tôi cũng không biết cụ thể nó là như thế nào và tôi tin rằng các bạn cũng vậy. Chính vì thế việc cần làm của chúng ta đó là tìm ra hàm f nêu trên. Các phương pháp đi tìm hàm f ẩn cho mỗi vấn đề thương làm người ta liên tưởng đến Machine Learning và thật đúng như vậy. Tất cả mọi thứ phức tạp nhất trên cuộc đời này để có thể biểu diễn bằng một hàm f dạng như thế. Trong Machine Learning chúng có thể là một matrix giống như Linear Regression cũng có thể là một hyperplane giống như trong SVM hoặc cũng có thể là một mạng nơ ron giống như trong cách tiếp cận của Deep Learning và còn là những thứ gì khác nữa mà hiện tại con người chưa khám phá ra được.

Vấn đề mà chúng ta đang đề cập đến tại đây là một dạng của machine translation hay còn gọi là dịch máy. Chắc có lẽ một công cụ đã không còn xa lạ gì với tất cả chúng ta đó chính là Google Dịch về bản chất nó cũng sử dụng các kĩ thuật Deep Learning mà tôi sẽ trình bày với các bạn trong bài viết này. Nhưng không phải là để dịch Tiếng Anh sang Tiếng Việt mà là để dịch Tiếq Việt sang Tiếng Việt

Graphical user interface

Description automatically generated

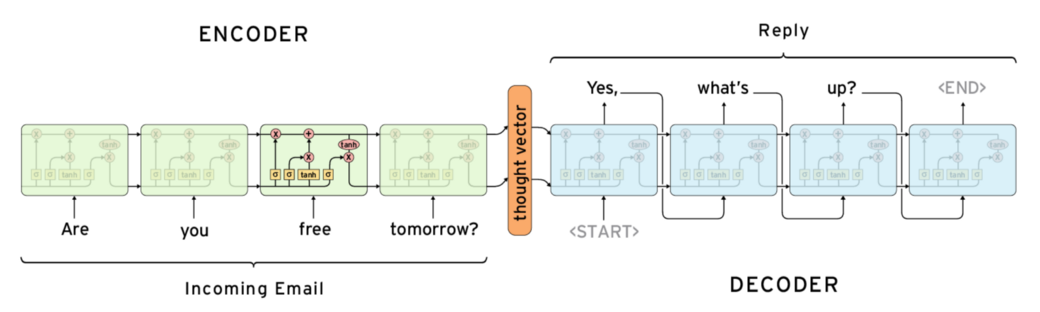
Việc dịch này được thực hiện dựa trên một mô hình gọi là Sequence to Sequence mà mình sẽ giới thiệu chi tiết cho các bạn sau đây. Nhưng trước hết có thể bạn sẽ có một câu hỏi rằng tại sao không sử dụng phương pháp dịch từng từ một rồi ghép chúng lại với nhau. Câu trả lời là việc dịch như thế đôi khi làm cho câu dịch bị mất đi ý nghĩa và đọc lên cảm thấy rất là vô hồn hơn nữa nó là mất đi các quy tắc về ngữ pháp trong ngôn ngữ đích nếu như chúng ta không xây dựng một tập luật - rules để xác định cách biểu diễn ngôn ngữ đích sao cho đúng ngữ pháp. Mô hình Sequence to Sequence giải quyết được hoàn toàn hai vấn đề trên. Bây giờ chúng ta sẽ cùng nhau tìm hiểu về mô hình thần thánh này nhé

## **Mô hình Sequence to Sequence**

Có thể nói mô hình Sequence to Sequence mang lại một hiệu quả vô cùng to lớn cho thế giới hiện đại của chúng ta. Rất nhiều những ý tưởng mang tầm cỡ thế kĩ được sinh ra từ chúng như các trợ lý ảo, các hệ thống dịch tự động, các hệ thống tự động sinh caption cho ảnh / video hay chatbot. Về bản chất mô hình này gồm có hai pha. Chúng ta có thể hiểu nôm na như sau:

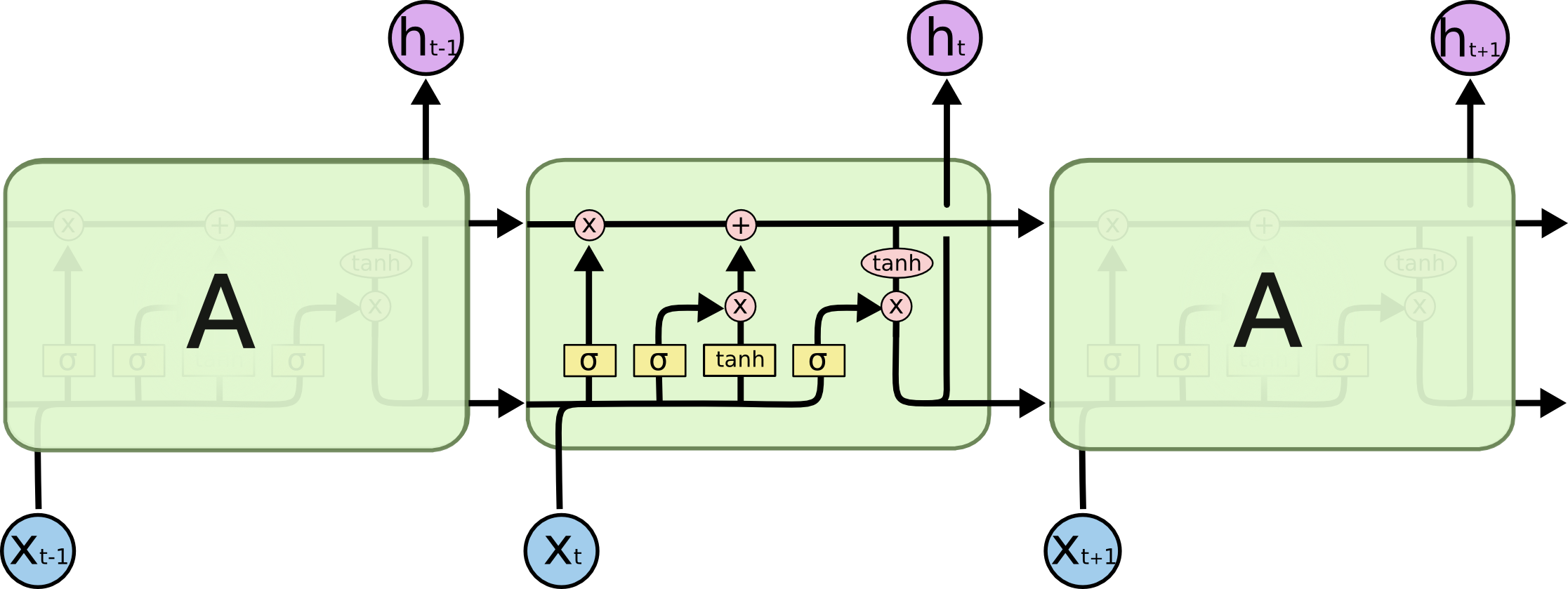
Pha 1 - Encoder: Pha encoder này có nhiệm vụ hiểu được thông tin của dữ liệu đầu vào. Việc hiểu này có nghĩa là mô hình hóa được bản chất của dữ lệu đầu vào. Chẳng hạn như trong bài toán của chúng ta đó chính là Tiếng Việt mới. Thông thường người ta sẽ sử dụng mạng nơ ron hồi quy - RNN để biểu diễn thông tin cho các dữ liệu trong NLP vì dữ liệu NLP có đặc tính là phụ thuộc theo chuỗi (từ sau phụ thuộc vào từ đứng trước đó). Đây cũng chính là lý do người ta lựa chọn RNN để mô hình hóa ngôn ngữ trong dịch máy. Tránh tình trạng sinh ra những câu vô nghĩa như bên trên mình đã đề cập

Pha 2 - Decoder Pha decoder này có nhiệm vụ chuyển đổi từ output của pha 1. Tức là một thought vector - vector suy diễn biểu diễn ngữ nghĩa của câu nguồn - sang cấu trúc của câu đích (câu cần dịch). Chúng ta có thể thấy rõ điều này trong sơ đồ sau:



## **Mạng nơ ron LSTM - Long Short Term Memory**

Trong các bước encoder và decoder trong mô hình seq2seq người ta thường sử dụng một kiến trúc mở rộng của mạng nơ ron hồi quy RNN đó chính là LSTM. Mạng LSTM được sinh ra với mục đích giúp cho RNN có thể nhớ được những phụ thuộc xa trong chuỗi dữ liệu. Điều này cũng giống như cách mà bộ nhớ của con người hoạt động, chúng ta không bắt đầu suy nghĩ của chúng ta từ con số 0 (đây là cách mà các mạng nơ ron truyền thống áp dụng) mà chúng ta sẽ liên tưởng và ghép nối các thông tin từ các sự kiện đã xảy ra trước đó. Đây chính là tư tưởng của mạng RNN. Tuy nhiên trong mạng RNN truyền thống chỉ lấy đầu ra của bước trước đó làm đầu vào của bước hiện tại. Tuy nhiên do một số vấn đề về mất mát hoặc bùng nổ đạo hàm - vanishing / exploding gradient trong quá trình tính toán back propagation mà mạng RNN truyền thống không thể nhớ được các phụ thuộc xa trong dữ liệu. Tức là nó bị đãng trí. Để khắc phục tình trạng đãng trí này người ta đề xuất mô hình LSTM bằng cách thêm vào một số cải tiến như sau:



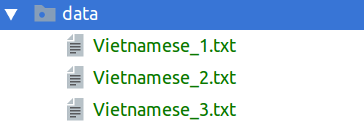
* Thêm vào mạng một trạng thái tế bào - cell state sử dụng để lưu trữ và lan truyền các thông tin có ích trong mạng, nó tương tự như một bộ nhớ cục bộ của mạng.
* Bổ sung thêm cổng quên - forget gate để loại bỏ những thông tin không quan trọng. Việc quyết định thông tin nào được giữ lại thông tin nào được loại bỏ đi chính là ở bước này. Và đương nhiên là chúng ta cần tìm ra một ma trận
* W\_f
* *W*
* *f*
* ​
* để khi nhân ma trận này với thông tin đầu vào chúng ta sẽ quyết định được thông tin đó có đi vào trạng thái tế bào ở lần lặp tiếp theo hay không.
* Bổ sung một cổng input - input gate và cũng với một ma trận
* W\_i
* *W*
* *i*
* ​
* để thể hiện thông tin mới được đưa vào (ví dụ như từ hiện tại mà mạng LSTM đang chạy qua).

Mình chỉ nói ở mức cơ bản như vậy thôi nhé. Các bạn muốn tìm hiểu có thể tham khảo thêm trong [blog này](https://dominhhai.github.io/vi/2017/10/what-is-lstm/). Giờ chúng ta sẽ tìm hiểu xem cách mà chúng ta cần triển khai một bộ dịch Tiếng Việt như thế nào.

# **Xây dựng bộ dịch Tiếng Việt mới sang Tiếng Việt cũ**

## **Bước 1: Thu thập dữ liệu**

Dữ liệu là một thứ không thể thiếu được với bất kì một ứng dụng Machine Learning nào. Chúng ta sẽ không thể làm được gì nếu như không có dữ liệu phải không nào. Các bạn có thể viết các tool crawler để crawl các dữ liệu từ trên mạng về rồi lưu vào file. Tuy nhiên trong đây mình có sử dụng một tập dữ liệu nhỏ mà mình crawl được từ trên các trang báo mạng về rồi tổng hợp thành một file các bạn có thể download nó về [tại đây](https://github.com/thandongtb/convert_tieq_viet/blob/master/data/Vietnamese.txt). Tiếp theo, do mô hình LSTM sẽ nhận đầu vào là từng câu trong tập dữ liệu nên chúng ta cần phải thực hiện bước tách câu cho tiếng việt và thêm kí tự kết thúc (ví dụ như dấu chấm) vào cuối câu đó. Các bạn có thể download thêm nhiều dữ liệu khác và đặt vào cùng một thư mục data như hình sau:



Sau khi chuẩn bị xong dữ liệu chúng ta tiến hành load dữ liệu nhé

## **Bước 2: Xây dựng tập dữ liệu**

Bước này khá đơn giản cũng không có gì phải bàn nhiều. Các bạn tiến hành đọc tất cả các file dữ liệu của các bạn trong thư mục data/ vừa rồi

DATA\_PATH = 'data/'

class DataLoader(object):

def \_\_init\_\_(self, path = DATA\_PATH):

self.path = path

def load\_source\_data(self):

files = glob.glob('{}\*.txt'.format(self.path))

data = [open(f, 'rb').read().decode('utf-8') for f in files]

return data

Tiếp theo chúng ta cần viết một hàm tách tập dữ liệu txt của chúng ta thành các câu đơn lẻ. Kết quả là chúng ta sẽ có một list các câu đơn lẻ trong tập dữ liệu. Ở đây để cho đơn giản mình sử dụng một công cụ tách câu của thư viện nltk. Các bạn thực hiện như sau

def load\_source\_sentences(self):

source\_data = self.load\_source\_data()

return [sent.replace('\n', ' ') for d in tqdm(source\_data) for sent in sent\_tokenize(d)]

Chúng ta thử xem kết quả thực hiện của hàm trên.

if \_\_name\_\_ == '\_\_main\_\_':

old\_sentences = DataLoader().load\_source\_sentences()

print(old\_sentences[:2])

>>> ['Đối với tôi, câu chuyện này bắt đầu 15 năm trước, khi tôi còn là một bác sĩ tế bần tại Đại học Chicago.', 'Tôi chăm sóc những người đang hấp hối và gia đình của họ ở Nam Chicago.']

Đây chính là các câu ở trong tiếng việt của chúng ta sử dụng. Giờ chúng ta sẽ sử dụng hàm convert trong class VnConverter phía trên để thực hiện chuyển đổi câu này sang Tiếng Việt mới nhé.

def load\_dict\_sentences(self):

source\_sentences = self.load\_source\_sentences()

return [VnConverter(sent).convert() for sent in source\_sentences]

chúng ta cũng thử xem kết quả của hàm này xem sao:

if \_\_name\_\_ == '\_\_main\_\_':

new\_sentences = DataLoader().load\_dict\_sentences()

print(new\_sentences[:2])

>>> ['Dối với tôi, kâu cuyện này bắt dầu 15 năm cướk, xi tôi kòn là một bák sĩ tế bần tại Dại họk Cikago.', "Tôi căm sók n'ữq qười daq hấp hối và za dìn' kủa họ ở Nam Cikago."]

Vậy là chúng ta đã có hai tập dữ liệu việc còn lại là dump nó ra thành từng file. Chúng ta sử dụng hàm sau

def dump\_to\_txt(self, sentences, filename):

with open(file=filename, mode='a') as file:

for sent in sentences:

file.write(sent + '\n')

Như vậy sau bước chuẩn bị dữ liệu chúng ta đã có hai file, file nguồn và file đích tương ứng với nhau về số dòng. File nguồn chính là Tiếng Việt mới của Bùi Hiển, file nguồn là Tiếng Việt của chúng ta đang dùng. Chúng ta tiến hành chia dữ liệu thành hai phần traning và tessting nữa là xong. Các bạn có thể thực hiện như sau:

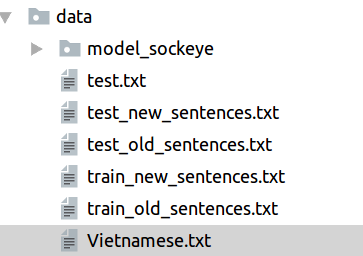
ratio = 0.8

DataLoader().dump\_to\_txt(sentences=old\_sentences[:int(len(old\_sentences) \* ratio)], filename=DATA\_PATH + 'train\_old\_sentences.txt')

DataLoader().dump\_to\_txt(sentences=new\_sentences[:int(len(new\_sentences) \* ratio)], filename=DATA\_PATH + 'train\_new\_sentences.txt')

DataLoader().dump\_to\_txt(sentences=old\_sentences[int(len(old\_sentences) \* ratio):], filename=DATA\_PATH + 'test\_old\_sentences.txt')

DataLoader().dump\_to\_txt(sentences=new\_sentences[int(len(new\_sentences) \* ratio):], filename=DATA\_PATH + 'test\_new\_sentences.txt')

Sau bước này chúng ta sẽ có tập dữ liệu như sau:

Tập dữ liệu này sẽ được sử dụng để training mô hình học máy của chúng ta.

## **Training mô hình**

Về mặt lý thuyết chúng ta có thể tự impliment một mô hình Machine Translation với các thư viện của Tensorflow tuy nhiên trong phạm vi của bài viết quá dài mình không thể trình bày hết các bước impliment được nên mình quyết định sử dụng một thư viện hỗ trợ khá tốt việc training các mô hình dịch máy đó chính là [Sockeye](https://github.com/awslabs/sockeye). Các bạn cài đặt theo như hướng dẫn và tiến hành trainng bằng câu lệnh như sau:

cd data

python -m sockeye.train --source train\_new\_sentences.txt \

--target train\_old\_sentences.txt \

--validation-source test\_new\_sentences.txt \

--validation-target test\_old\_sentences.txt \

--num-embed 256 \

--rnn-num-hidden 512 \

--rnn-attention-type dot \

--max-seq-len 60 \

--decode-and-evaluate 500 \

--use-cpu \

--output model\_sockeye

Sau khi training một thời gian đủ dài (khoảng 20 epochs) các bạn sẽ thu được một model dịch máy (khoảng 10GB) và sử dụng model đó trong việc dịch ngược từ Tiếng Việt mới về Tiếng Việt cũ.

## **Dịch ngược Tiếng Việt mới về Tiếng Việt cũ**

Sau khi có mô hình các bạn có thể dịch ngược một câu Tiếng Việt mới bất kì như sau:

echo "Căm năm coq kõi qười ta." | python -m sockeye.translate --models model\_sockeye --use-cpu

>>> Trăm năm trong cõi người ta.

echo "Và nó xôq cỉ zõ kái zì daq xảy za Một số kâu hỏi dượk dặt za qay lập tứk." | python -m sockeye.translate --models model\_sockeye --use-cpu

>>> Và nó không chỉ rõ cái gì đang xảy ra Một số câu hỏi được đặt ra ngay lập tức.

Độ chính xác của mô hình hiện tại mình đang sử dụng khoảng 70% do thời gian training không nhiều cũng như dữ liệu training không đủ. Bạn đọc có thể tự mình tạo thêm dữ liệu với các bước ở trên đồng thời có thể tự impliment một bộ Machine Translation của riêng mình mà không phụ thuộc vào thư viện của bên thứ 3. Chúc các bạn cuối tuần vui vẻ.

# **Source code**

Các bạn có thể tham khảo source code của bài viết tại [đây](https://github.com/thandongtb/convert_tieq_viet/blob/master/data/Vietnamese.txt)

# **Kết luận**

Với mô hình Deep Learning chúng ta hoàn toàn có thể xây dựng được một bộ dịch từ Tiếng Việt của Bùi Hiển về Tiếng Việt hiện tại. Tuy nhiên mình không khuyến khích điều này, mặc dù nó chứng minh một điều rằng nếu như nắm trong tay công nghệ, chúng ta hoàn toàn có thể giải quyết được đa số vấn đề của thế giới. Vậy nên, Dù là thay đổi bất cứ điều gì, chúng ta là dân công nghệ thì không phải sợ.