# Neural Machine Translation Overview Report



## 1.Tổng quan cấu trúc NMT và BLEU score

### 1.1. NMT

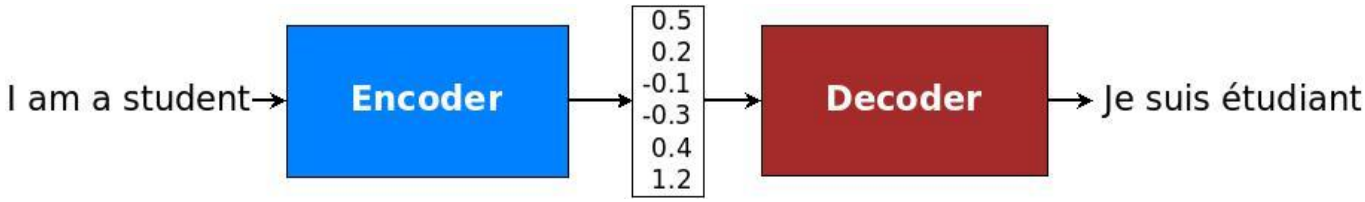
NMT (Neural Machine Translation) là sự kết hợp của dịch máy (Machine Translation - MT) và mạng nơ-ron nhân tạo (Artificial Neural Network - NN). Và cụ thể trong bài báo cáo này thì mạng NN được sử dụng trong mô hình NMT là mạng nơ-ron hồi quy (hoặc truy hồi) (Recurrent Neural Network - RNN) và mô hình NMT được sử dụng là mô hình xây dựng theo kiến trúc NMT của Google, trên nền tảng của thư viện Tensorflow dành cho Python theo Source code.

Khởi nguồn của MT hoạt động theo cách chia nhỏ câu thành các cụm từ và tiến hành dịch trên từng cụm từ một. Kết quả cuối cùng sẽ là một câu ghép lại từ các cụm từ đã được dịch. Cách tiếp cận này được gọi là dịch theo cụm (pharase-based), và kết quả thì không được ấn tượng lắm vì cách tiếp cận của phương pháp này không thực sự giống với cách mà con người sử dụng trong dịch thuật là đọc toàn bộ câu, nắm ý nghĩa của câu và đưa ra câu dịch tương ứng. Và NMT được xây dựng hoàn toàn dựa trên cách làm này. NMT là cách tiếp cận MT phổ biến trong khoảng 4 năm gần đây và đã cho ra các kết quả thực sự tốt, tới mức ngang hoặc hơn cả con người.

Cụ thể về kiến trúc thì NMT là sự kết hợp của 2 thành phần chính là **Seq2Seq** và **Attention**

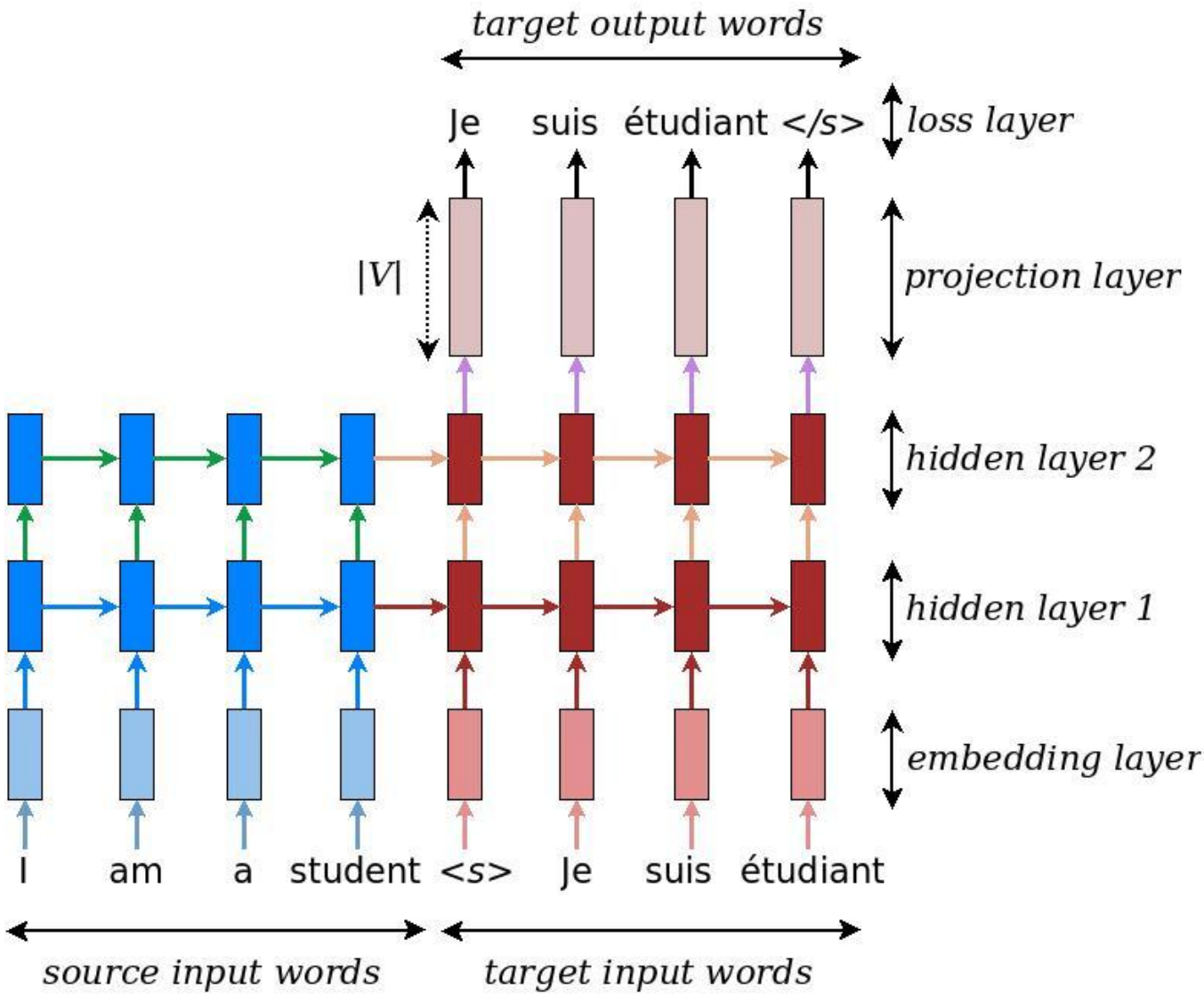
**Seq2Seq**

Kiến trúc Seq2Seq được lấy từ tên của bài nghiên cứu **Sequence to Sequence Learning with Neural Networks** (Sutskever et al., 2014) trình bày một kiến trúc MT mà ở đó, một câu (**Seq**) sẽ được dịch bằng cách đưa vào một bộ mã hóa (**Encoder**) và nối tiếp với một bộ giải mã (**Decoder**) để dịch sang một câu (**2Seq**) ở ngôn ngữ khác.




*Hình 1. Cấu trúc Encoder-Decoder.*

Cụ thể thì NMT sẽ đưa toàn bộ câu ở ngôn ngữ gốc vào Encoder để *nén* ý nghĩa của câu thành một vector gọi là **context** (hoặc **thought**), và đưa vector đó sang cho bộ giải mã (Decoder) để chuyển vector thành câu thuộc ngôn ngữ khác (ngôn ngữ đích) mang ý nghĩa tương ứng với câu ở ngôn ngữ gốc. Bài viết này sẽ sử dụng RNN, gated recurrent unit (GRU) hoặc Long Short-term Memory (LSTM), để làm encoder và decoder.




*Hình 2. Ví dụ cấu trúc NMT.*

Ở ví dụ cấu trúc NMT trình bày ở Hình 2, bộ encoder và decoder đều được cấu tạo từ 2 lớp RNN cùng chiều, chồng lên nhau, tương ứng với màu xanh dương và màu đỏ. Ở đây, ký hiệu `<s>` sử dụng để báo hiệu bắt đầu

quá trình decode và ký hiệu `</s>` sử dụng để báo hiệu cho decoder dừng quá trình decode lại.

Ngoài 2 lớp RNN chồng lên nhau ở 2 bộ encoder và decoder thì trong Hình 2 còn có sự xuất hiện của 3 lớp NN khác là:

- 1 lớp embedding ở encoder
- 1 lớp embedding, 1 lớp projection ở decoder (2 lớp này dùng chung bộ trọng số, chỉ ngược chiều).

Đây là các lớp NN embedding và ứng với mỗi ngôn ngữ sẽ có một bộ NN embedding riêng biệt.

Trong khi embedding sẽ có vai trò chuyển một từ trong **không gian từ điển** (vocab) của ngôn ngữ, sang không gian *vector* (có chiều tương ứng với không gian của **vector context**). Thì projection ở decoder sẽ có vai trò chuyển ngược lại một từ thuộc không gian *vector* sang **không gian từ điển** (vocab) của ngôn ngữ.

Tại chiều không gian *vector* này thì encoder sẽ đóng vai trò *nén* ngữ nghĩa của một câu (tập hợp các từ thuộc không gian *vector*) của ngôn ngữ gốc thành một **vector context** và đưa sang cho decoder tiến hành *giải nén* **vector context** thành một câu (tập hợp các từ thuộc không gian *vector*) của ngôn ngữ đích.

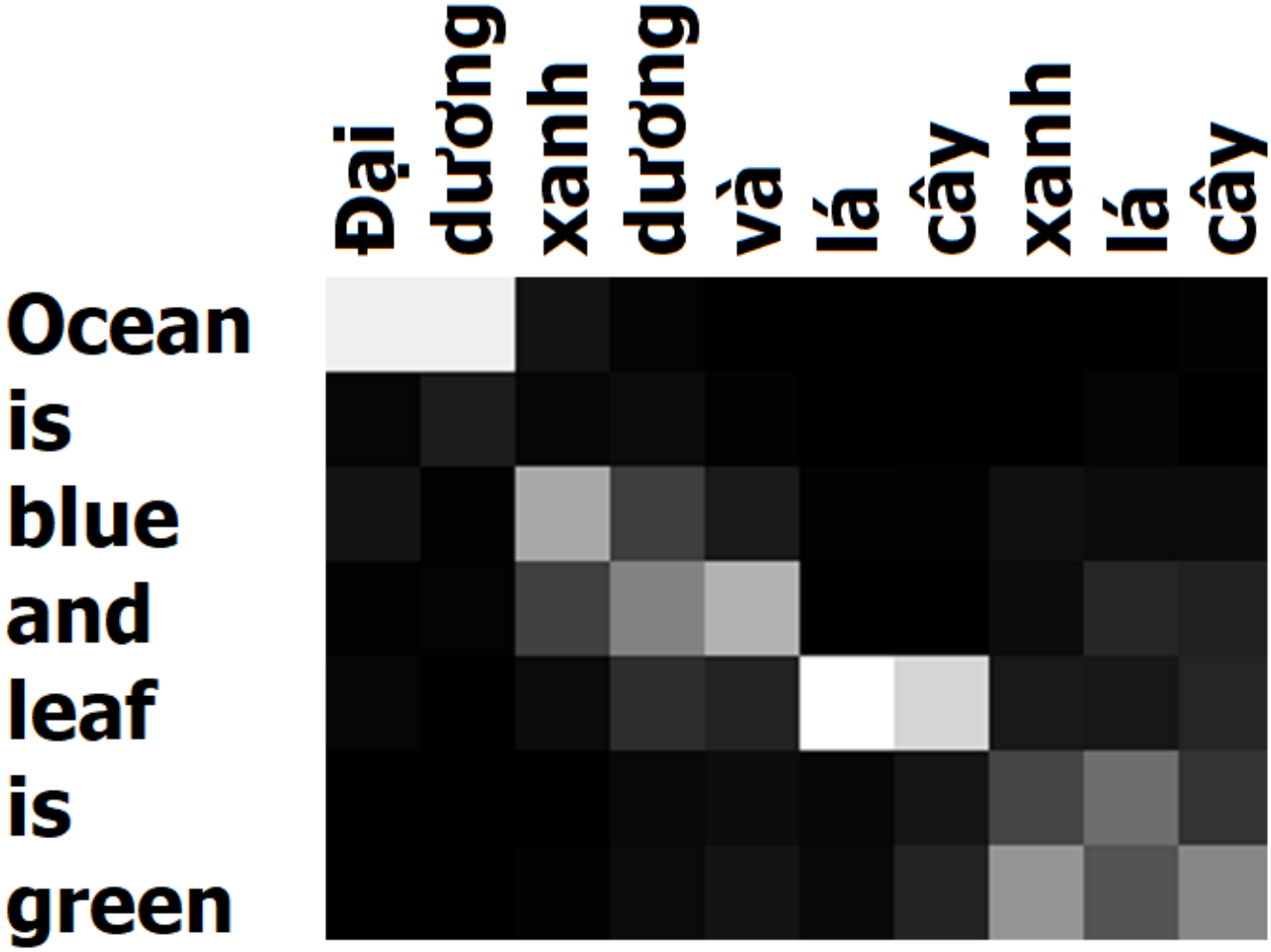
Để cho thuận tiện trong việc tính toán và chuyển đổi qua lại thì các chiều không gian *vector* này sẽ đều có chung chiều là chiều của **vector context** (kích thước của mạng RNN).

Mặc dù RNN ban đầu được nghĩ ra để phù hợp với các bài toán trải dài theo trình tự thời gian (time-step) nhưng với những quãng quá dài thì khả năng nắm bắt thông tin của RNN ở các time-step xa vẫn còn rất hạn chế và đã được nêu ra trong 2 bài nghiên cứu của Hochreiter (1991) [tiếng Đức] và Bengio, et al. (1994).

Giai đoạn đó cũng là lúc kiến trúc LSTM ra đời qua nghiên cứu của Hochreiter & Schmidhuber (1997). LSTM đã phần nào khắc phục được sự mất mát thông tin theo thời gian dài ở RNN cơ bản.

**Attention**

Vào năm 2015, để tăng độ chính xác trong việc nắm bắt thông tin ở các mô hình **Encoder-Decoder** thì cơ chế *Attention* đã được giới thiệu với ý tưởng chính là tạo một *đường tắt* liên kết trực tiếp từng từ ở ngôn ngữ đích với từ tương ứng cần *chú ý* ở ngôn ngữ gốc, bên cạnh ý nghĩa được mã hóa của **vector context** (Bahdanau et al., 2015 & Luong et al., 2015).




*Hình 3. Mô tả về sự chú ý ở ngôn ngữ gốc và ngôn ngữ đích*

Attention là 1 cơ chế giúp mô hình tập trung vào các phần quan trọng tương ứng của thông tin, bằng cách tạo 1 **alignment model** để tính các **alignment score** αij\alpha_{ij}αij nhằm cân chỉnh lại mức độ liên quan của các hidden state tại encoder với các output của decoder tại timestep tương ứng. Với NMT và Seq2Seq, việc này giúp mô hình hiểu được mức độ liên quan giữa *input* và từ cần *predict* tiếp theo tại decoder.

Từ 2 nghiên cứu đã đề cập ở trên, có một số cách tính alignment score khác nhau như sau:

- Từ nghiên cứu của Bahdanau et al., 2015 có Additive Attention

  $score(s_{i−1},h_{j})=va^{T}tanh(W_{a}s_{i−1}+U_{a}h_{j})$

- Từ nghiên cứu của Luong et al., 2015
  - Multiplicative Attention hay General Attention

    $score(s_{i−1},h_{j})=s^{T}{i−1}W{a}h_{j}$

  - Dot Product (simple mechanism)

    $score(s_{i−1},h_{j})=s^{T}{i−1}h{j}$

### 1.2. BLEU Score

Bilingual Evaluation Understudy Score hay ngắn gọn là BLEU score là một thang điểm được dùng phổ biến trong đánh giá MT. BLEU được Kishore Papineni và cộng sự đề xuất lần đầu vào năm 2002 qua bài nghiên cứu "a Method for Automatic Evaluation of Machine Translation".

BLEU được tính dựa trên số lượng n-grams giống nhau giữa câu dịch của mô hình (output) với các câu tham chiếu tương ứng (label) có xét tới yếu tố độ dài của câu.

Số n-grams tối đa của BLEU là không giới hạn, nhưng vì xét về ý nghĩa, cụm từ quá dài thường không có nhiều ý nghĩa, và nghiên cứu cũng đã cho thấy là với 4-gram, điểm số BLEU trung bình cho khả năng dịch thuật của con người cũng đã giảm khá nhiều nên n-grams tối đa thường được sử dụng là 4-gram.

Xét bộ câu dịch và tham chiếu có cùng độ dài như sau:

- Output: Đây là cái ghế
- Label 1: Đây là cái bàn
- Label 2: Kia có cái ghế

| **1-gram** | **Output** | **Label 1** | **Label 2** |
|---|---|---|---|
| Đây | 1 | 1 | 0 |
| là | 1 | 1 | 0 |
| cái | 1 | 1 | 1 |
| ghế | 1 | 0 | 1 |

BLEU độc lập 1-gram:

- 1-gram và Label 1: **3/4** = 0.75
- 1-gram và Label 2: **2/4** = 0.5
- 1-gram và Label 1 + 2: **4/4** = 1

| **2-grams** | **Output** | **Label 1** | **Label 2** |
|---|---|---|---|
| Đây là | 1 | 1 | 0 |
| là cái | 1 | 1 | 0 |
| cái ghế | 1 | 0 | 1 |

Tương tự, BLEU độc lập cho 2-gram sẽ được tính như sau:

- 2-grams và Label 1: **2/3** = 0.67
- 2-grams và Label 2: **1/3** = 0.33
- 2-grams và Label 1 + 2: **3/3** = 1

Công thức tổng quát độ chính xác cho mỗi n-grams:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}$$

*Hình 4. Độ chính xác của n-gram.*

Với cách tính này, thì sẽ dễ dàng thấy rằng nếu output càng ngắn, thì độ chính xác được tính theo công thức sẽ càng cao. Ví dụ với cùng 2 Label trên, nếu output chỉ ra duy nhất chữ **"cái"** thì BLEU sẽ bằng 1. Lúc này, để hạn chế việc câu quá ngắn thì BP (brevity penalty) được sử dụng để so sánh độ dài output **c** và độ dài câu tham chiếu **r**.

BLEU Score lúc này sẽ có công thức:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Then,

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

*Hình 5. BLEU score.*

Quay lại ví dụ ban đầu, BLEU chung cho cả 1-gram và 2-gram với trọng số ***wn*** ngang nhau $\tfrac {1}{2}$ sẽ là :

- 1-gram + 2-grams và Label 1:

  $\LARGE{e^{(0.5 \times log\tfrac {3}{4}+0.5 \times log\tfrac {2}{3})}} = 0.71$

- 1-gram + 2-grams và Label 2:

  $\LARGE{e^{(0.5 \times log\tfrac {2}{4}+0.5 \times log\tfrac {1}{3})}} = 0.41$

- 1-gram + 2-grams và Label 1 + 2:

$\LARGE{e^{(0.5 \times log\tfrac {4}{4}+0.5 \times log\tfrac {3}{3})}} = 1$

Với trường hợp có nhiều hơn 1 Label và các Label có độ dài khác nhau thì lúc này, BP sẽ được tính theo Label nào có chênh lệch độ dài so Output là ít nhất.

---

# 2. Siêu tham số

Việc lựa chọn siêu tham số cho mô hình NMT sẽ tác động rất lớn tới độ phức tạp và độ *nặng* của hệ thống khi huấn luyện và chạy mô hình. Một số lựa chọn chính bao gồm:

## 2.1. Cấu trúc NN

- Loại RNN (unit_type): LSTM, GRU
- Số lớp (nums_layer): độ *sâu* của mạng RNN, có thể là 2 lớp, 4 lớp hoặc thậm chí là 8 lớp RNN chồng lên nhau. Tuy nhiên thì trong nhiều trường hợp, mô hình sâu quá 4 lớp không đem lại sự thay đổi đáng kể trong kết quả huấn luyện (Reimers & Gurevych, 2017 và Britz et al., 2017)
- Số đơn vị (nums_unit): đặc trưng cho *kích thước* của mạng RNN, kích thước của **vector context**. Kích thước 2048 mang lại kết quả tốt nhất, nhưng rất khó áp dụng rộng rãi vì cực kỳ **nặng** khi chạy mô hình. Và kích thước 128 mang lại kết quả tương đối tốt với tốc độ huấn luyện nhanh và nhẹ hơn tới 16 lần (Britz et al., 2017).
- Chiều của encoder (encoder_type): 1 chiều, 2 chiều hoặc kết hợp một lớp 2 chiều và nhiều lớp một chiều. Và encoder 2 chiều đã được nghiên cứu là mang lại hiệu quả tốt hơn nhiều so với 1 chiều (Sutskever et al., 2014 & Britz et al., 2017)

## 2.2. Cơ chế Tối ưu hóa

- Thuật toán tối ưu hóa (optimizer): ta có 2 sự lựa chọn là Adam (Kingma & Ba, 2015) và SGD (stochastic gradient descent). Mặc dù Adam mới được ra mắt gần đây và được cộng đồng nghiên cứu NLP sử dụng thường xuyên vì sự vượt trội rõ ràng của Adam so với SGD thì một số nghiên cứu đã cho thấy là với nhưng tùy chỉnh hợp lý thì SGD đã có thể nhỉnh hơn Adam một chút (Wu et al., 2016) hoặc vượt trội hơn hẳn với việc áp dụng momentum (Zhang & Mitliagkas, 2017).
- Cơ chế khởi động tốc độ học ban đầu và giảm dần khi về cuối (warmup_scheme & decay_scheme). Mô hình cho phép lựa chọn một số cơ chế tích hợp sẵn như là:
    - Tensor2Tensor's warmup_scheme, khởi đầu với tốc độ học lr nhỏ hơn 100 lần và tăng dần cho đến khi đạt được con số mong muốn.
    - luong234 decay_scheme, sau 2/3 bước huấn luyện, bắt đầu giảm tốc độ học lr 4 lần, mỗi lần 50%
    - luong5 decay_scheme, sau 1/2 bước huấn luyện, bắt đầu giảm tốc độ học lr 5 lần, mỗi lần 50%
    - luong10 decay_scheme, sau 1/2 bước huấn luyện, bắt đầu giảm tốc độ học lr 10 lần, mỗi lần 50%
- Bước huấn luyện (trainnig_step): Dao động phụ thuộc vào độ phức tạp và khả năng huấn luyện, thường từ 10.000 lần cho tới hàng trăm nghìn lần.

## 2.3. Một số tùy chỉnh khác

- Giới hạn tối đa Gradient norm (gradient clipping): Trong một số trường hợp tốc độ học quá lớn, khiến gradient bùng nổ và nhảy quá xa khỏi điểm hội tụ thì việc giới hạn trên cho Gradient norm là cần thiết và thậm chí cải thiện hiệu năng. (Pascanu et al., 2013)

- Chiến thuật chọn output cho decoder (infer_mode):
    - Mặc định thì ở mỗi time_step,output của decoder sẽ cho ra phân phối xác suất của toàn bộ từ thuộc **không gian từ điển** (vocab) của ngôn ngữ đích và chiến thuật phổ biến sẽ là chọn từ có xác suất cao nhất (greedy) để làm output cho timestep đó, đưa vào timestep sau.
    - Tuy nhiên thì xác suất cao nhất ở bước hiện tại chưa chắc sẽ cho ra xác xuất cao nhất ở bước tiếp theo, vậy nên thay vì chỉ giữ 1 kết quả có xác xuất cao nhất, beam search sẽ giữ lại *k* kết quả cao nhất để tiếp tục đưa vào decoder ở bước tiếp theo và tiến hành lựa chọn *k* xác suất cao nhất ở bước tiếp theo này. *k* = 10 thường được sử dụng và mang lại hiệu quả đủ tốt.
- Sennrich et al., 2016 đề xuất một kỹ thuật chia nhỏ bộ từ điển ngôn ngữ thành các đơn vị từ nhỏ hơn (tiếp đầu ngữ, tiếp vĩ ngữ, v.v...) bằng cách mã hóa theo cặp Philip Gage, 1994 qua đó giúp mô hình NMT thích ứng tốt hơn trước các từ hiếm gặp, và tăng hiệu quả dịch thuật lên tới 1.1 % và 1.3 % BLEU score.
- residual: là một dạng NN tạo ra để khắc phục hiện tượng gradiend vanishing bằng cách trực tiếp cộng thêm input vào output (y = f(x) + x). Đây là kỹ thuật tương đối hiệu quả với các NN có độ sâu lớn (> 100 layer) (He el al., 2015).

## 2.4. Kết quả thử nghiệm

Mô hình NMT đã được huấn luyện thử nghiệm trên 11 bộ thông số khác nhau trên cùng tập dữ liệu, nhưng với cấu hình máy tính khác nhau nên thời gian huấn luyện là không . Vậy nên bước tổng kết này sẽ chỉ liệt kê điểm BLEU (%) tương ứng với các bộ thông số khác nhau. Phần tên thông số sẽ gồm tên mặc định của bộ thông số mặc định và thông số điều chỉnh so với mặc định.

**Thông số NMT Mặc định**

```
attention: None
unit_type: lstm
encoder_type: uni (1 chiều)
num_layers: 2
num_units: 128
num_train_steps: 12000
optimizer: sgd
learning_rate: 1.0
warmup_scheme: t2t
infer_mode: greedy
vi > en
```

| Default NMT | BLEU |
|---|---|
| Default | 5.9 % |
| + attention scaled_luong | 16.2% |
| + GRU | 6.1 % |
| + GRU + num_units=512 | 8.9 % |

**WMT16 Mod**

```
attention: normed_bahdanau
attention_architecture: gnmt_v2
encoder_type: gnmt (kết hợp 2 chiều và 1 chiều)
beam_search_width: 10
length_penalty_weight: 1.0
residual: true
subword_option: bpe
```

| Default WMT16 Mod | BLEU |
|---|---|
| Default | 15.8 % |
| + num_layers: 4 | 18.1 % |
| + infer_mode: greedy | 13.5 % |

**IWSLT15**

```
attention: scaled_luong
encoder_type: bi (2 chiều)
num_units: 512
decay_scheme: luong234
```

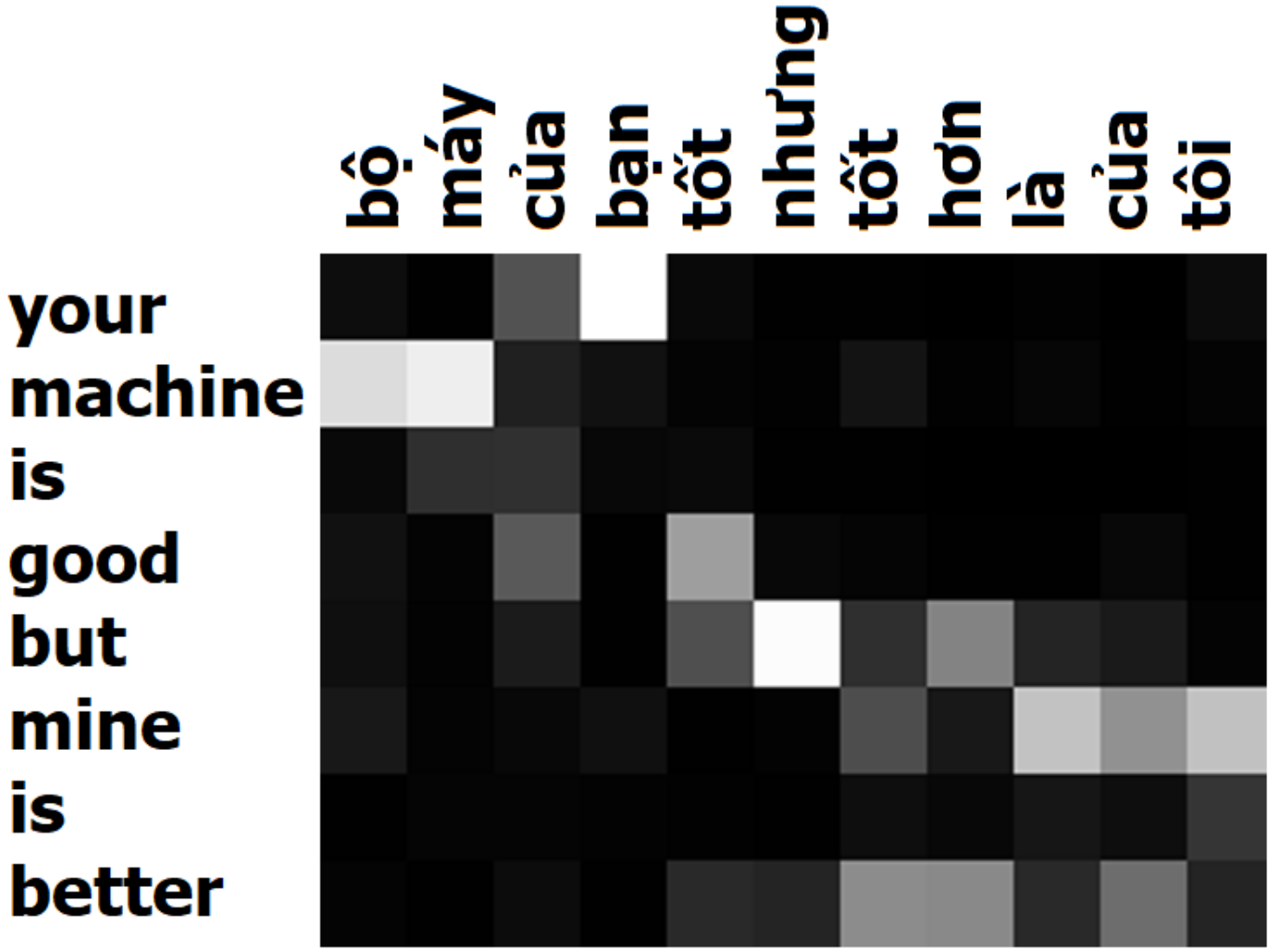
| Default IWSLT15 | BLEU |
|---|---|
| Default | 20.6 % |
| + num_train_steps: 20000 | 21.0 % |
| + en > vi + num_layers: 4 | 23.1 % |
| + en > vi + adam 0.001 | 22.2 % |

---

# 3. Attention matrix

Các Attention matrix dưới đây được tạo ra từ cùng 1 model với BLEU score 23.1

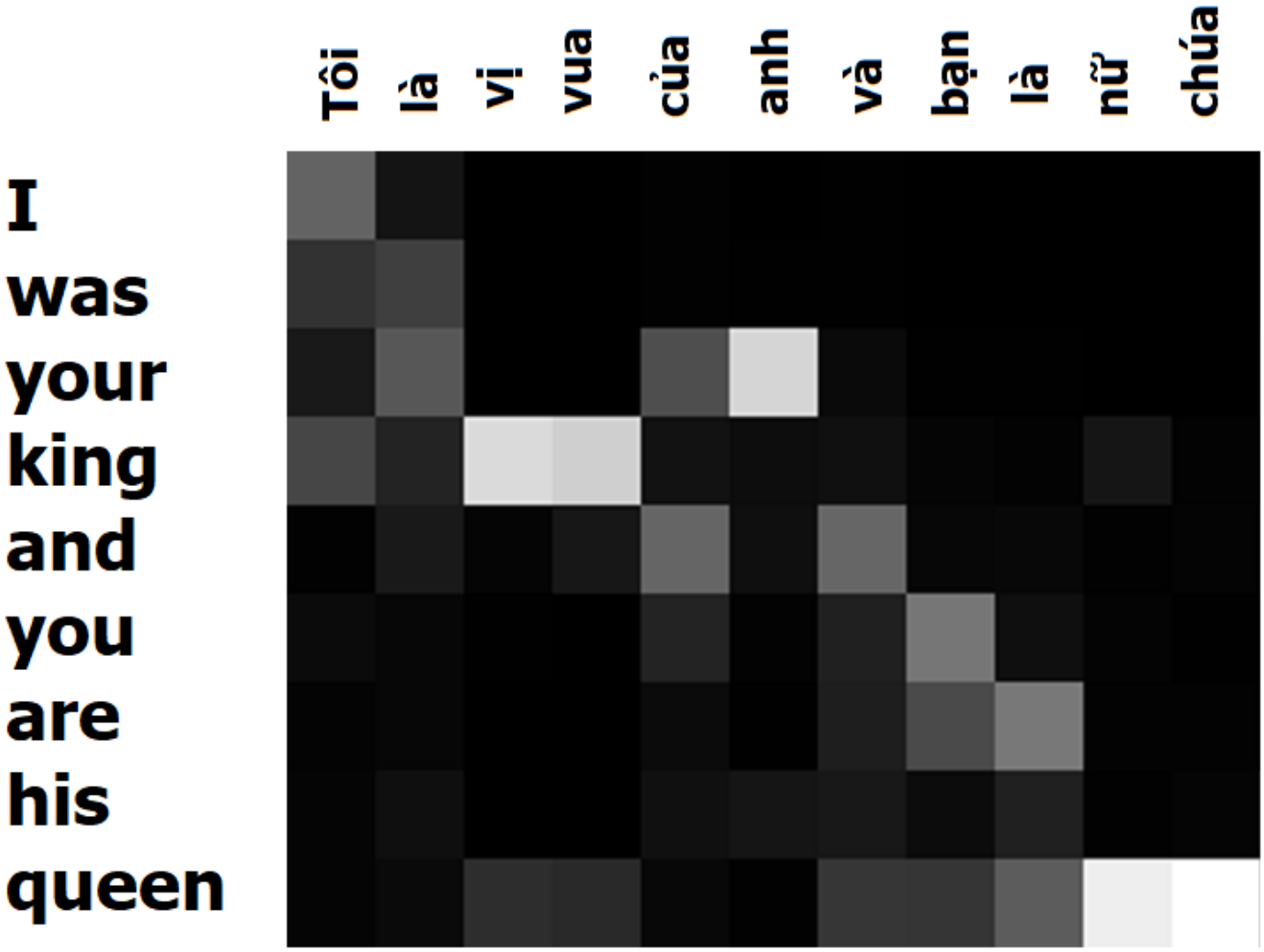
*Ma trận 1*




Mô hình đã nắm được mối quan hệ sở hữu ở hai từ **"của"**,

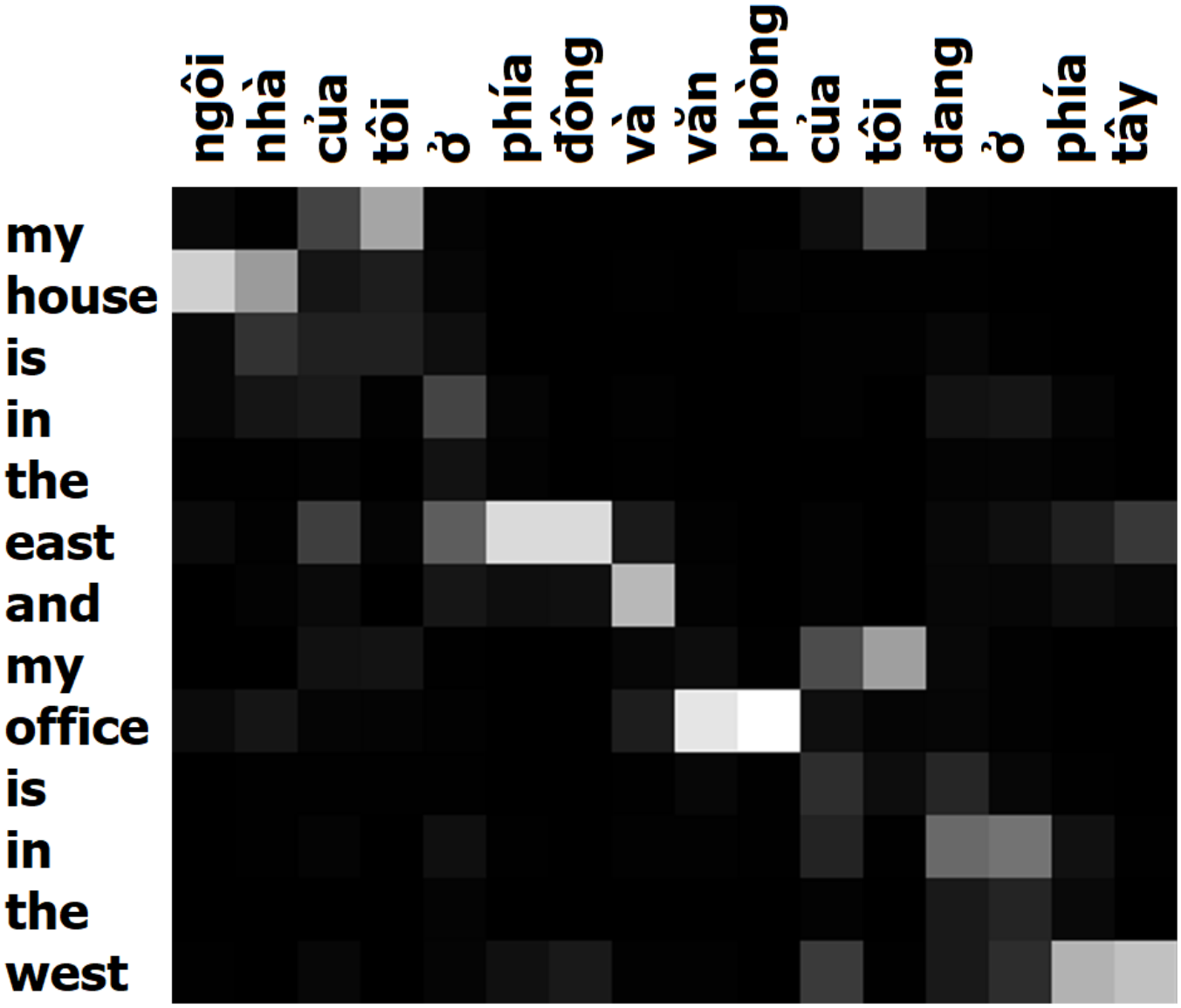
- Từ **"của"** đầu tiên đã tạo được sự liên kết giữa **"your"** và **"good"**
- Từ **"của "** thứ hai đã tạo được sự liên kết giữa **"mine"** và **"better"**.

*Ma trận 2*




- Từ **"Tôi"** đã tạo được mối liên hệ giữa từ **"I"** và từ **"king"**.
- Từ **"vị vua"** cũng có mối liên hệ với **"king"** và **"queen"**, điều này là hợp lý vì trên thực tế thông dụng trong tiếng Việt, từ vua có thể dùng để chỉ cả nữ hoàng (nếu là nữ) hoặc hoàng đế (nam)

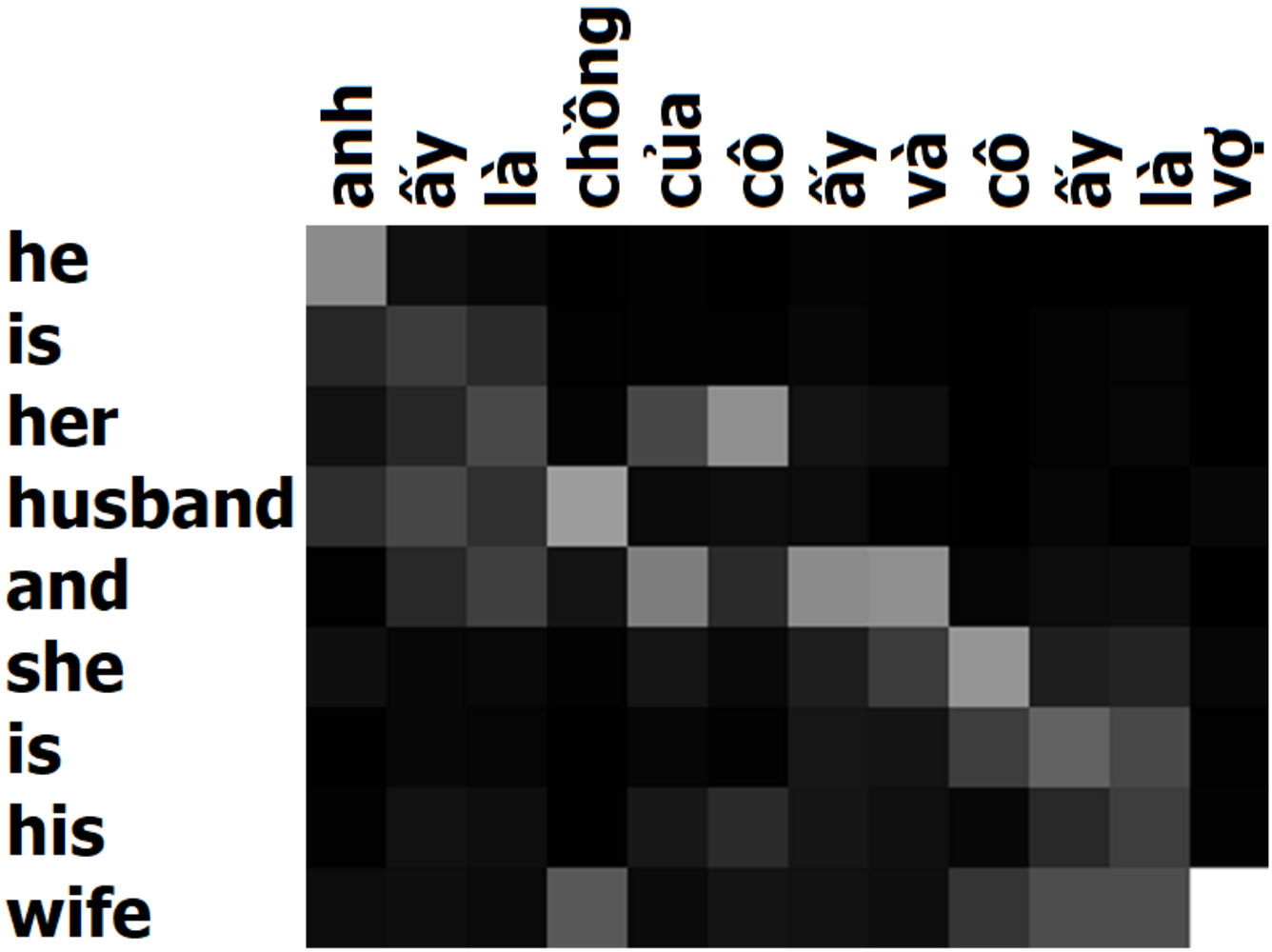
*Ma trận 3*




---

- Từ **"east"** và **"west"** đã có được mối liên hệ với nhau vì đều là từ để chỉ phương hướng.
- Trong ví dụ này, từ từ **"my"** lặp lại 2 lần thì ở từ **"my"** đầu tiên đã tạo được mối liên hệ với từ **"tôi"** đầu tiên và cả từ **"tôi"** thứ 2

*Ma trận 4*




- Tại ví dụ này, từ **"anh"** đã bắt được mối liên kết của từ **"he"** và **"husband"**
- Từ **"chồng"** đã nắm được mối liên quan chính là từ **"husband"** và cũng đã tìm được sự tương quan tới từ **"wife"**

## Another Reference

- Huan Phan's GitHub
- Neural Machine Translation (seq2seq) Tutorial
- Deep Learning for NLP Best Practices
- [Machine Learning] Attention, Attention, Attention, ...!
- A Gentle Introduction to Calculating the BLEU Score for Text in Python
- Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)