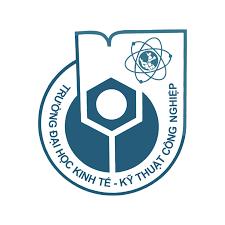


BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP HÀ NỘI

🕮🕮🕮 ---KHOA CÔNG NGHỆ THÔNG TIN--- 🕮🕮🕮



**BÁO CÁO THỰC TẬP**

|  |  |  |
| --- | --- | --- |
| *Sinh viên thực hiện* | : | Lưu Vũ Tùng , Đào Cư Thanh , Lưu Đức Huy, Đỗ Thanh Hà, Phạm Thị Thái Hà |
| *Lớp* | : | Tin11A3 |
| *Nhóm* | : | 10 |
| *Giáo viên hướng dẫn* | : | Cao Thụy Ánh & Trần Hồng Việt |

Mục lục

Contents

[Chương 1.Trí tuệ nhân tạo là gì ? 3](#_Toc62795851)

[Chương 2. Mô hình Transformer cho câu tiếng Việt 7](#_Toc62795852)

[Chương 3 Sử dụng Phobert để nhúng câu Tiếng Việt 20](#_Toc62795853)

[Chương 4 : ĐƯỢC GÌ SAU CHẶNG ĐƯỜNG 5 TUẦN ? 27](#_Toc62795854)

[Chương 5 : Cài đặt môi trường 28](#_Toc62795855)

### Chương 1.Trí tuệ nhân tạo là gì ?

Trong [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), **trí tuệ nhân tạo** hay [**AI**](https://vi.wikipedia.org/wiki/Artificial_intelligence) ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *Artificial Intelligence*), đôi khi được gọi là **trí thông minh nhân tạo**, là [trí thông minh được](https://vi.wikipedia.org/wiki/Tr%C3%AD_th%C3%B4ng_minh) thể hiện bằng [máy móc](https://vi.wikipedia.org/wiki/M%C3%A1y_m%C3%B3c), trái ngược với **trí thông minh tự nhiên** của con người. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc (hoặc máy tính) có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với [tâm trí](https://vi.wikipedia.org/wiki/T%C3%A2m_tr%C3%AD), như "học tập" và "giải quyết vấn đề".[[1]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-FOOTNOTERussellNorvig2009-1)

Trí tuệ nhân tạo có thể được phân thành ba loại hệ thống khác nhau: trí tuệ nhân tạo phân tích, lấy cảm hứng từ con người và nhân tạo.[[6]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-6) AI phân tích chỉ có các đặc điểm phù hợp với [trí tuệ nhận thức](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_th%E1%BB%A9c); tạo ra một đại diện nhận thức về thế giới và sử dụng học tập dựa trên kinh nghiệm trong quá khứ để thông báo các quyết định trong tương lai. AI lấy cảm hứng từ con người có các yếu tố từ [trí tuệ](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_x%C3%BAc_c%E1%BA%A3m) nhận thức và [cảm xúc](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_x%C3%BAc_c%E1%BA%A3m); hiểu cảm xúc của con người, ngoài các yếu tố nhận thức và xem xét chúng trong việc [ra quyết định](https://vi.wikipedia.org/wiki/Ra_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh). AI nhân cách hóa cho thấy các đặc điểm của tất cả các loại năng lực (nghĩa là trí tuệ nhận thức, cảm xúc và xã hội), có khả năng tự ý thức và tự nhận thức được trong các tương tác.

#### 1.1 Lịch sử trí tuệ nhân tạo

Trí tuệ nhân tạo được thành lập như một môn học thuật vào năm 1956, và trong những năm sau đó đã trải qua nhiều làn sóng lạc quan,[[7]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Optimism_of_early_AI-7)[[8]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-AI_in_the_80s-8) sau đó là sự thất vọng và mất kinh phí (được gọi là " mùa đông AI "),[[9]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-First_AI_winter-9)[[10]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Second_AI_winter-10) tiếp theo là cách tiếp cận mới, thành công và tài trợ mới.[[8]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-AI_in_the_80s-8)[[11]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-AI_in_2000s-11) Trong phần lớn lịch sử của mình, nghiên cứu AI đã được chia thành các trường con thường không liên lạc được với nhau.[[12]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Fragmentation_of_AI-12) Các trường con này dựa trên các cân nhắc kỹ thuật, chẳng hạn như các mục tiêu cụ thể (ví dụ: " [robot học](https://vi.wikipedia.org/wiki/Robot_h%E1%BB%8Dc) " hoặc "học máy"),[[13]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Problems_of_AI-13) việc sử dụng các công cụ cụ thể ("logic" hoặc [mạng lưới thần kinh nhân tạo](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron_nh%C3%A2n_t%E1%BA%A1o)) hoặc sự khác biệt triết học sâu sắc.[[14]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Biological_intelligence_vs._intelligence_in_general-14)[[15]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Neats_vs._scruffies-15)[[16]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Symbolic_vs._sub-symbolic-16) Các ngành con cũng được dựa trên các yếu tố xã hội (các tổ chức cụ thể hoặc công việc của các nhà nghiên cứu cụ thể).[[12]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-Fragmentation_of_AI-12)

Lĩnh vực này được thành lập dựa trên tuyên bố rằng [trí thông minh của con người](https://vi.wikipedia.org/wiki/Tr%C3%AD_th%C3%B4ng_minh_c%E1%BB%A7a_con_ng%C6%B0%E1%BB%9Di) "có thể được mô tả chính xác đến mức một cỗ máy có thể được chế tạo để mô phỏng nó".[[17]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-17) Điều này làm dấy lên những tranh luận triết học về bản chất của [tâm trí](https://vi.wikipedia.org/wiki/T%C3%A2m_tr%C3%AD) và đạo đức khi tạo ra những sinh vật nhân tạo có trí thông minh giống con người, đó là những vấn đề đã được [thần thoại](https://vi.wikipedia.org/wiki/L%E1%BB%8Bch_s%E1%BB%AD_ng%C3%A0nh_tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o), viễn tưởng và [triết học](https://vi.wikipedia.org/wiki/Tri%E1%BA%BFt_h%E1%BB%8Dc_v%E1%BB%81_tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o) từ [thời cổ đại](https://vi.wikipedia.org/wiki/Th%E1%BB%9Di_k%E1%BB%B3_c%E1%BB%95_%C4%91%E1%BA%A1i) đề cập tới.[[18]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-McCorduck's_thesis-18) Một số người cũng coi AI là [mối nguy hiểm cho nhân loại](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%83m_k%E1%BB%B3_d%E1%BB%8B_c%C3%B4ng_ngh%E1%BB%87) nếu tiến triển của nó không suy giảm.[[19]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-19) Những người khác tin rằng AI, không giống như các cuộc cách mạng công nghệ trước đây, sẽ tạo ra [nguy cơ thất nghiệp hàng loạt](https://vi.wikipedia.org/w/index.php?title=Th%E1%BA%A5t_nghi%E1%BB%87p_c%C3%B4ng_ngh%E1%BB%87&action=edit&redlink=1).[[20]](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o#cite_note-guardian_jobs_debate-20)

Trong thế kỷ 21, các kỹ thuật AI đã trải qua sự hồi sinh sau những tiến bộ đồng thời về sức mạnh máy tính, [dữ liệu lớn](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u_l%E1%BB%9Bn) và hiểu biết lý thuyết; và kỹ thuật AI đã trở thành một phần thiết yếu của [ngành công nghệ](https://vi.wikipedia.org/w/index.php?title=C%C3%B4ng_nghi%E1%BB%87p_c%C3%B4ng_ngh%E1%BB%87&action=edit&redlink=1), giúp giải quyết nhiều vấn đề thách thức trong học máy, [công nghệ phần mềm](https://vi.wikipedia.org/wiki/C%C3%B4ng_ngh%E1%BB%87_ph%E1%BA%A7n_m%E1%BB%81m) và [nghiên cứu vận hành](https://vi.wikipedia.org/wiki/V%E1%BA%ADn_tr%C3%B9_h%E1%BB%8Dc).

#### 1.2 Phân loại

Trí tuệ nhân tạo (AI) chia thành hai trường phái tư duy: Trí tuê nhân tạo truyền thống và [trí tuệ tính toán](https://vi.wikipedia.org/w/index.php?title=Tr%C3%AD_tu%E1%BB%87_t%C3%ADnh_to%C3%A1n&action=edit&redlink=1).

Trí tuê nhân tạo truyền thống hầu như bao gồm các phương pháp hiện được phân loại là các phương pháp [học máy](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_m%C3%A1y) (*machine learning*), đặc trưng bởi [hệ hình thức](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_h%C3%ACnh_th%E1%BB%A9c&action=edit&redlink=1) (*formalism*) và [phân tích thống kê](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_Th%E1%BB%91ng_k%C3%AA). Nó còn được biết với các tên Trí tuê nhân tạo [biểu tượng](https://vi.wikipedia.org/wiki/Bi%E1%BB%83u_t%C6%B0%E1%BB%A3ng), Trí tuê nhân tạo [logic](https://vi.wikipedia.org/wiki/Logic), [Trí tuê nhân tạo ngăn nắp](https://vi.wikipedia.org/w/index.php?title=TTNT_ng%C4%83n_n%E1%BA%AFp&action=edit&redlink=1) (*neat AI*) và [Trí tuê nhân tạo cổ điển](https://vi.wikipedia.org/w/index.php?title=TTNT_c%E1%BB%95_%C4%91i%E1%BB%83n&action=edit&redlink=1) (*Good Old Fashioned Artificial Intelligence*). (Xem thêm [ngữ nghĩa học](https://vi.wikipedia.org/wiki/Ng%E1%BB%AF_ngh%C4%A9a_h%E1%BB%8Dc).) Các phương pháp gồm có:

* [Hệ chuyên gia](https://vi.wikipedia.org/wiki/H%E1%BB%87_chuy%C3%AAn_gia): áp dụng các khả năng suy luận để đạt tới một kết luận. Một hệ chuyên gia có thể xử lý các lượng lớn thông tin đã biết và đưa ra các kết luận dựa trên các thông tin đó. [Clippy](https://vi.wikipedia.org/w/index.php?title=Clippy&action=edit&redlink=1) chương trình trợ giúp có hình cái kẹp giấy của [Microsoft Office](https://vi.wikipedia.org/wiki/Microsoft_Office) là một ví dụ. Khi người dùng gõ phím, Clippy nhận ra các xu hướng nhất định và đưa ra các gợi ý.
* [Lập luận theo tình huống](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_lu%E1%BA%ADn_theo_t%C3%ACnh_hu%E1%BB%91ng).
* [Mạng Bayes](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_Bayes).

Trí tuệ tính toán nghiên cứu việc học hoặc phát triển lặp (ví dụ: tinh chỉnh tham số trong hệ thống, chẳng hạn hệ thống [connectionist](https://vi.wikipedia.org/w/index.php?title=Connectionist&action=edit&redlink=1)). Việc học dựa trên dữ liệu kinh nghiệm và có quan hệ với Trí tuệ nhân tạo phi ký hiệu, [Trí tuê nhân tạo lộn xộn](https://vi.wikipedia.org/w/index.php?title=TTNT_l%E1%BB%99n_x%E1%BB%99n&action=edit&redlink=1) (*scruffy AI*) và [tính toán mềm](https://vi.wikipedia.org/w/index.php?title=T%C3%ADnh_to%C3%A1n_m%E1%BB%81m&action=edit&redlink=1) (*soft computing*). Các phương pháp chính gồm có:

* [Mạng neural](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron): các hệ thống mạnh về [nhận dạng mẫu](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_m%E1%BA%ABu) (*pattern recognition*).
* [Hệ mờ](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_m%E1%BB%9D&action=edit&redlink=1) (*Fuzzy system*): các kỹ thuật [suy luận không chắc chắn](https://vi.wikipedia.org/w/index.php?title=Suy_lu%E1%BA%ADn_kh%C3%B4ng_ch%E1%BA%AFc_ch%E1%BA%AFn&action=edit&redlink=1), đã được sử dụng rộng rãi trong các hệ thống công nghiệp hiện đại và các hệ thống quản lý sản phẩm tiêu dùng.
* [Tính toán tiến hóa](https://vi.wikipedia.org/w/index.php?title=T%C3%ADnh_to%C3%A1n_ti%E1%BA%BFn_h%C3%B3a&action=edit&redlink=1) (*Evolutionary computation*): ứng dụng các khái niệm sinh học như [quần thể](https://vi.wikipedia.org/wiki/Qu%E1%BA%A7n_th%E1%BB%83), [biến dị](https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn_d%E1%BB%8B_sinh_h%E1%BB%8Dc) và [đấu tranh sinh tồn](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A5u_tranh_sinh_t%E1%BB%93n) để sinh các lời giải ngày càng tốt hơn cho bài toán. Các phương pháp này thường được chia thành các [thuật toán tiến hóa](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_ti%E1%BA%BFn_h%C3%B3a&action=edit&redlink=1) (ví dụ [thuật toán gene](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_di_truy%E1%BB%81n)) và [trí tuệ bầy đàn](https://vi.wikipedia.org/w/index.php?title=Tr%C3%AD_tu%E1%BB%87_b%E1%BA%A7y_%C4%91%C3%A0n&action=edit&redlink=1) (*swarm intelligence*) (chẳng hạn [hệ kiến](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_ki%E1%BA%BFn&action=edit&redlink=1)).
* [Trí tuê nhân tạo dựa hành vi](https://vi.wikipedia.org/w/index.php?title=TTNT_d%E1%BB%B1a_h%C3%A0nh_vi&action=edit&redlink=1) (*Behavior based AI*): một phương pháp module để xây dựng các hệ thống Trí tuê nhân tạo bằng tay.

Người ta đã nghiên cứu các [hệ thống thông minh lai](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_th%C3%B4ng_minh_lai) (*hybrid intelligent system*), trong đó kết hợp hai trường phái này. Các luật suy diễn của hệ chuyên gia có thể được sinh bởi mạng neural hoặc các [luật dẫn xuất](https://vi.wikipedia.org/w/index.php?title=Lu%E1%BA%ADt_d%E1%BA%ABn_xu%E1%BA%A5t&action=edit&redlink=1) (*production rule*) từ việc học theo thống kê như trong kiến trúc [ACT-R](https://vi.wikipedia.org/w/index.php?title=ACT-R&action=edit&redlink=1).

#### 1.3 Ví dụ về dự án trí tuệ nhân tạo

1. [Twenty Questions](http://www.20q.net/) - Một trò chơi 20 câu hỏi, trong đó sử dụng mạng neural
2. [The Start Project](http://start.csail.mit.edu/) - một chương trình trả lời các câu hỏi bằng tiếng Anh.
3. [Brainboost](http://www.brainboost.com/) - một hệ thống trả lời câu hỏi khác
4. [Cyc](https://vi.wikipedia.org/w/index.php?title=Cyc&action=edit&redlink=1), một cơ sở tri thức với rất nhiều kiến thức về thế giới thực và khả năng suy luận logic.
5. [Jabberwacky](https://vi.wikipedia.org/w/index.php?title=Jabberwacky&action=edit&redlink=1), một [chatterbot](https://vi.wikipedia.org/w/index.php?title=Chatterbot&action=edit&redlink=1) có khả năng học
6. [ALICE](https://vi.wikipedia.org/w/index.php?title=ALICE&action=edit&redlink=1), một [chatterbot](https://vi.wikipedia.org/w/index.php?title=Chatterbot&action=edit&redlink=1)
7. [Alan](http://www.a-i.com/alan1), một chatterbot khác
8. [Albert One](http://www.cybermecha.com/Studio), chatterbot nhiều mặt
9. [ELIZA](https://vi.wikipedia.org/w/index.php?title=ELIZA&action=edit&redlink=1), một chương trình giả làm bác sĩ tâm lý, phát triển năm [1966](https://vi.wikipedia.org/wiki/1966)
10. PAM (Plan Applier Mechanism) - một hệ thống hiểu được chuyện kể, phát triển bởi John Wilensky năm [1978](https://vi.wikipedia.org/wiki/1978).
11. SAM (Script applier mechanism) - một hệ thống hiểu được chuyện kể, phát triển năm [1975](https://vi.wikipedia.org/wiki/1975).
12. [SHRDLU](https://vi.wikipedia.org/w/index.php?title=SHRDLU&action=edit&redlink=1) - một chương trình hiểu ngôn ngữ tự nhiên, phát triển năm [1968](https://vi.wikipedia.org/wiki/1968)-[1970](https://vi.wikipedia.org/wiki/1970).
13. [Creatures](https://vi.wikipedia.org/w/index.php?title=Creatures&action=edit&redlink=1), một trò chơi máy tính với các hoạt động nhân giống, tiến hóa các sinh vật từ mức gien trở lên, sử dụng cấu trúc sinh hóa phức tạp và các bộ não là mạng neural.
14. [BBC news story](http://news.bbc.co.uk/1/hi/wales/3521852.stm) on the creator of *Creatures* latest creation. [Steve Grand](https://vi.wikipedia.org/w/index.php?title=Steve_Grand&action=edit&redlink=1)'s *Lucy*.
15. [AARON](http://www.kurzweilcyberart.com/KCATaaron/STAFsample) - chương trình vẽ tranh, phát triển bởi Harold Cohen.
16. [Eurisko](https://vi.wikipedia.org/w/index.php?title=Eurisko&action=edit&redlink=1) - một ngôn ngữ giúp giải quyết các bài toán, trong đó có sử dụng các phương pháp heuristics, gồm cả heuristics cho việc sử dụng và thay đổi các phương pháp heuristics. Phát triển năm 1978 bởi Douglas Lenat.
17. [X-Ray Vision for Surgeons](http://www.ai.mit.edu/projects/medical-vision/) - một nhóm nghiên cứu xử lý ảnh y học tại đại học MIT.
18. [Các chương trình trò chơi backgammon và cờ vây sử dụng mạng neural](http://www.jellyfish-ai.com/).
19. [Talk to William Shakespeare](http://www.shakespearebot.com/) - William Shakespeare chatbot
20. [Chesperito](irc://irc.dal.net/windows95) - Một chat/infobot về #windows95 channel trên mang DALnet IRC.
21. [Drivatar](https://vi.wikipedia.org/w/index.php?title=Drivatar&action=edit&redlink=1), một chương trình học cách lái xe đua bằng cách xem các xe đua khác, phát triển cho trò chơi điện tử [*Forza Motorsport*](https://vi.wikipedia.org/w/index.php?title=Forza_Motorsport&action=edit&redlink=1)
22. Tiểu Độ - một [Robot](https://vi.wikipedia.org/wiki/Robot) có trí tuệ nhân tạo thuộc hãng [Baidu](https://vi.wikipedia.org/wiki/Baidu) từng tham gia [chương trình Siêu Trí Tuệ Trung Quốc (mùa 4)](https://vi.wikipedia.org/wiki/Danh_s%C3%A1ch_c%C3%A1c_t%E1%BA%ADp_c%E1%BB%A7a_Si%C3%AAu_Tr%C3%AD_Tu%E1%BB%87_Trung_Qu%E1%BB%91c_(m%C3%B9a_4)) và đoạt giải

#### 1.4 Các bài toán điển hình

* [Nhận dạng mẫu](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_m%E1%BA%ABu)
  + [Nhận dạng chữ cái quang học](https://vi.wikipedia.org/w/index.php?title=Nh%E1%BA%ADn_d%E1%BA%A1ng_ch%E1%BB%AF_c%C3%A1i_quang_h%E1%BB%8Dc&action=edit&redlink=1) (*Optical character recognition*)
  + [Nhận dạng chữ viết tay](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_ch%E1%BB%AF_vi%E1%BA%BFt_tay)
  + [Nhận dạng tiếng nói](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_ti%E1%BA%BFng_n%C3%B3i)
  + [Nhận dang khuôn mặt](https://vi.wikipedia.org/w/index.php?title=Nh%E1%BA%ADn_dang_khu%C3%B4n_m%E1%BA%B7t&action=edit&redlink=1)
* [Xử lý ngôn ngữ tự nhiên](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_ng%C3%B4n_ng%E1%BB%AF_t%E1%BB%B1_nhi%C3%AAn), [Dịch tự động](https://vi.wikipedia.org/wiki/D%E1%BB%8Bch_t%E1%BB%B1_%C4%91%E1%BB%99ng)(dịch máy) và [Chatterbot](https://vi.wikipedia.org/w/index.php?title=Chatterbot&action=edit&redlink=1)
* [Điều khiển phi tuyến](https://vi.wikipedia.org/w/index.php?title=%C4%90i%E1%BB%81u_khi%E1%BB%83n_phi_tuy%E1%BA%BFn&action=edit&redlink=1) và [Robotics](https://vi.wikipedia.org/wiki/Robotics)
* [Computer vision](https://vi.wikipedia.org/w/index.php?title=Computer_vision&action=edit&redlink=1), [Thực tại ảo](https://vi.wikipedia.org/wiki/Th%E1%BB%B1c_t%E1%BA%BF_%E1%BA%A3o) và [Xử lý ảnh](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%BD_%E1%BA%A3nh)
* [Lý thuyết trò chơi](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_tr%C3%B2_ch%C6%A1i) và [Lập kế hoạch](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_k%E1%BA%BF_ho%E1%BA%A1ch) (*Strategic planning*)
* [Trò chơi Trí tuê nhân tạo](https://vi.wikipedia.org/w/index.php?title=Tr%C3%B2_ch%C6%A1i_TTNT&action=edit&redlink=1) và [Computer game bot](https://vi.wikipedia.org/w/index.php?title=Computer_game_bot&action=edit&redlink=1)

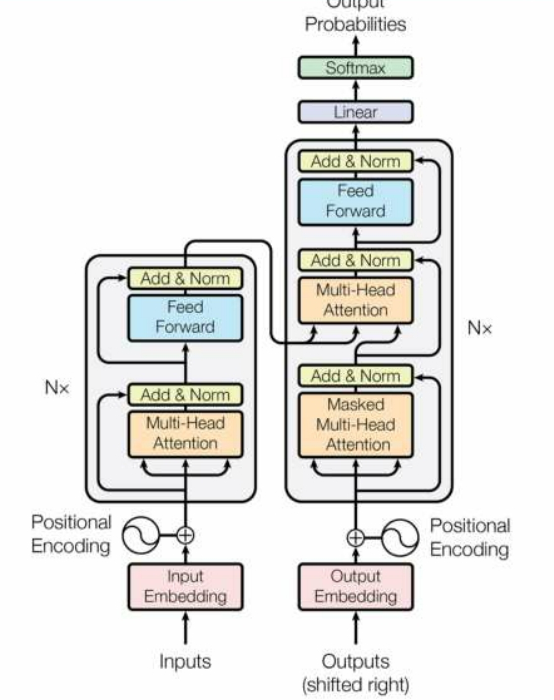
#### 1.5Các lĩnh vực khác cài đặt các phương pháp Trí tuê nhân tạo

* [Tự động hóa](https://vi.wikipedia.org/wiki/T%E1%BB%B1_%C4%91%E1%BB%99ng_h%C3%B3a)
* [Bio-inspired computing](https://vi.wikipedia.org/w/index.php?title=Bio-inspired_computing&action=edit&redlink=1)
* [Điều khiển học](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u_khi%E1%BB%83n_h%E1%BB%8Dc)
* [Hệ thống thông minh lai](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_th%C3%B4ng_minh_lai)
* [Agent thông minh](https://vi.wikipedia.org/wiki/T%C3%A1c_t%E1%BB%AD_ph%E1%BA%A7n_m%E1%BB%81m)
* [Điều khiển thông minh](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u_khi%E1%BB%83n_th%C3%B4ng_minh)
* [Suy diễn tự động](https://vi.wikipedia.org/w/index.php?title=Suy_di%E1%BB%85n_t%E1%BB%B1_%C4%91%E1%BB%99ng&action=edit&redlink=1)
* [Khai phá dữ liệu](https://vi.wikipedia.org/wiki/Khai_ph%C3%A1_d%E1%BB%AF_li%E1%BB%87u)
* [Cognitive robotics](https://vi.wikipedia.org/w/index.php?title=Cognitive_robotics&action=edit&redlink=1)
* [Developmental robotics](https://vi.wikipedia.org/w/index.php?title=Developmental_robotics&action=edit&redlink=1)
* [Evolutionary robotics](https://vi.wikipedia.org/w/index.php?title=Evolutionary_robotics&action=edit&redlink=1)
* [Chatbot](https://vi.wikipedia.org/wiki/Chatbot)

### Chương 2. Mô hình Transformer cho câu tiếng Việt

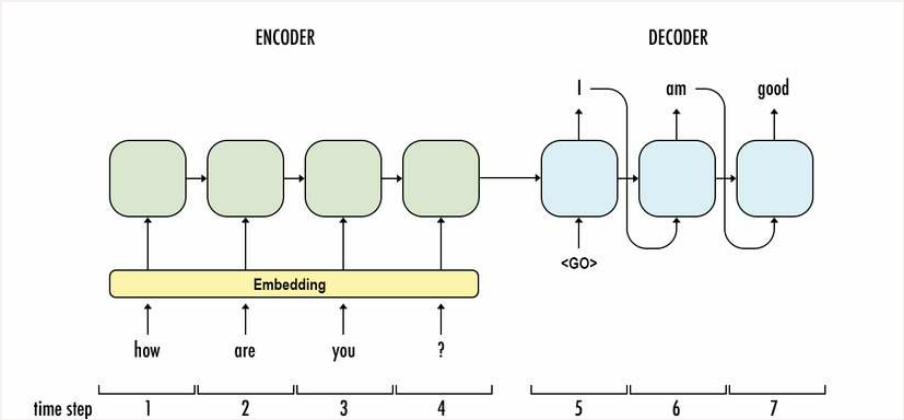
#### 2.1 Transformer là gì ?

Transformer là một mô hình học sâu được giới thiệu năm 2017, được dùng chủ yếu ở lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Có thể coi là SOTA – State Of The Art



Hình 2.1 Mô hình của RNN

#### 2.2 RNN phiên bản lỗi của Transformer

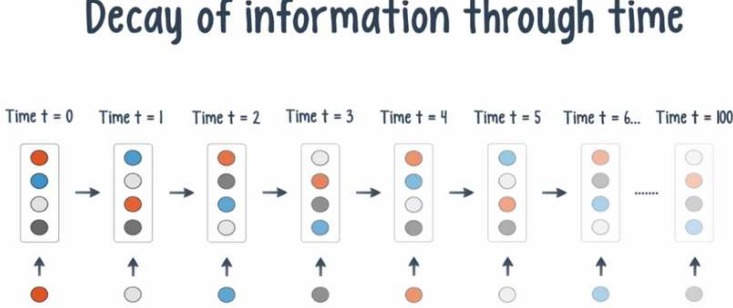
Nhận input là một sequence và trả lại output cũng là một sequence. Ví dụ bài toán Q&A, input là câu hỏi "how are you ?" và output là câu trả lời "I am good". Phương pháp truyền thống sử dụng RNNs cho cả encoder (phần mã hóa input) và decoder (phần giải mã input và đưa ra output tương ứng).

Hình 2.2

#### 2.3 Nhược điểm RNN

- Không thể tính toán song song , train chậm chạp và không tận dụng sức mạnh của GPU

-Bị giảm thông tin , mỗi liên hệ qua các bước do triệt tiêu/ bủng nổ qua Gradient . Đặc biệt với các câu dài

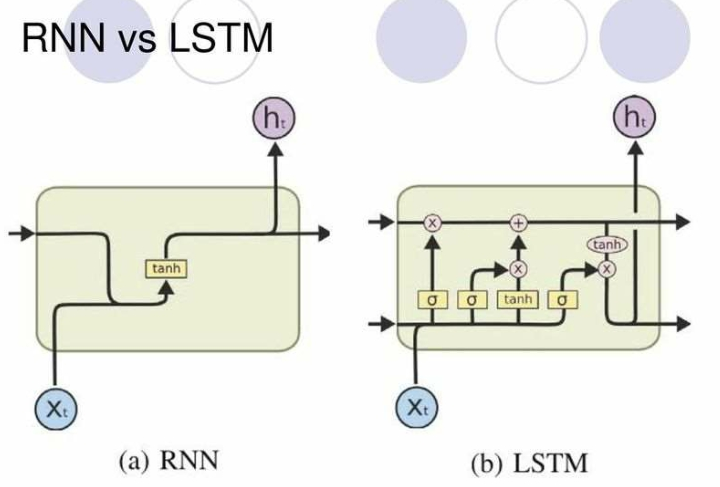


Hình 2.3

#### 2.4 LSTM vs RNN

LSTM có thể khắc phục đôi chút do có

cổng cho thông tin đi qua giúp duy trì thông tin khi qua các step nhưng lại dẫn tới train chậm hơn nữa do LSTM khá phức tạp



Hình 2.4

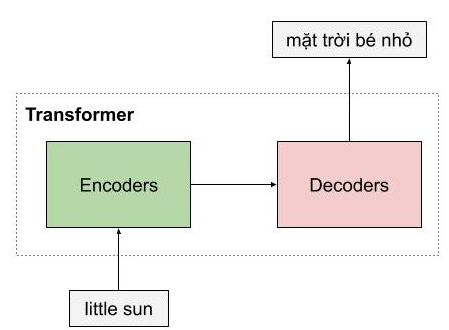
#### 2.5 Tổng quan mô hình Transformer

Để cho dễ cảm nhận được cách mà mô hình hoạt động, mình sẽ trình bày trước toàn bộ kiến trúc mô hình ở mức high-level và sau đó sẽ đi chi tiết từng phần nhỏ cũng như công thức toán của nó.

Giống như những mô hình dịch máy khác, kiến trúc tổng quan của mô hình transformer bao gồm 2 phần lớn là encoder và decoder. Encoder dùng để học vector biểu của câu với mong muốn rằng vector này mang thông tin hoàn hảo của câu đó. Decoder thực hiện chức năng chuyển vector biểu diễn kia thành ngôn ngữ đích.

Trong ví dụ ở dưới, encoder của mô hình transformer nhận một câu tiếng anh, và encode thành một vector biểu diễn ngữ nghĩa của câu *little sun*, sau đó mô hình decoder nhận vector biểu diễn này, và dịch nó thành câu tiếng việt *mặt trời bé nhỏ*

Một trong những ưu điểm của transformer là mô hình này có khả năng xử lý song song cho các từ. Như các bạn thấy, Encoders của mô hình transformer là một dạng feedforward neural nets, bao gồm nhiều encoder layer khác, mỗi encoder layer này xử lý đồng thời các từ. Trong khi đó, với mô hình LSTM, thì các từ phải được xử lý tuần tự. Ngoài ra, mô hình Transformer còn xử lý câu đầu vào theo 2 hướng mà không cần phải stack thêm một hình LSTM nữa như trong kiến trúc Bidirectional LSTM.

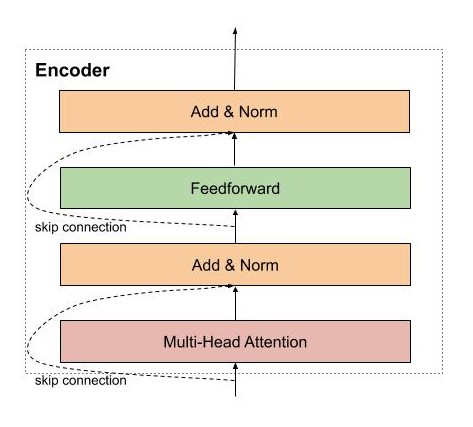


Hình 2.5

#### 2.6 Encoder

Encoder của mô hình transformer có thể bao gồm nhiều encoder layer tượng tự nhau. Mỗi encoder layer của transformer lại bao gồm 2 thành phần chính là multi head attention và feedforward network, ngoài ra còn có cả skip connection và normalization layer.

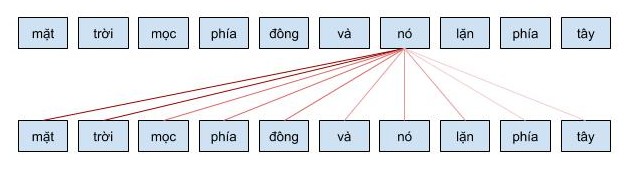
Trong 2 thành phần chính này, các bạn sẽ hứng thú nhiều hơn về multi-head attention vì đó là một layer mới được giới thiệu trong bài báo này, và chính nó tạo nên sự khác biệt giữa mô hình LSTM và mô hình Transformer mà chúng ta đang tìm hiểu.



Encoder đầu tiên sẽ nhận ma trận biểu diễn của các từ đã được cộng với thông tin vị trí thông qua positional encoding. Sau đó, ma trận này sẽ được xử lý bởi Multi Head Attention. Multi Head Attention thật chất là self-attention, nhưng mà để mô hình có thể có chú ý nhiều pattern khác nhau, tác giả đơn giản là sử dụng nhiều self-attention.

Self Attention Layer

Self Attention cho phép mô hình khi mã hóa một từ có thể sử dụng thông tin của những từ liên quan tới nó. Ví dụ khi từ **nó** được mã hóa, nó sẽ chú ý vào các từ liên quan như là **mặt trời**. Cơ chế self attention này có ý nghĩa tương tự như cơ chế attention mình đã chia sẻ ở [bài trước](https://pbcquoc.github.io/attention/) và những công thức toán học cũng tương ứng với nhau.

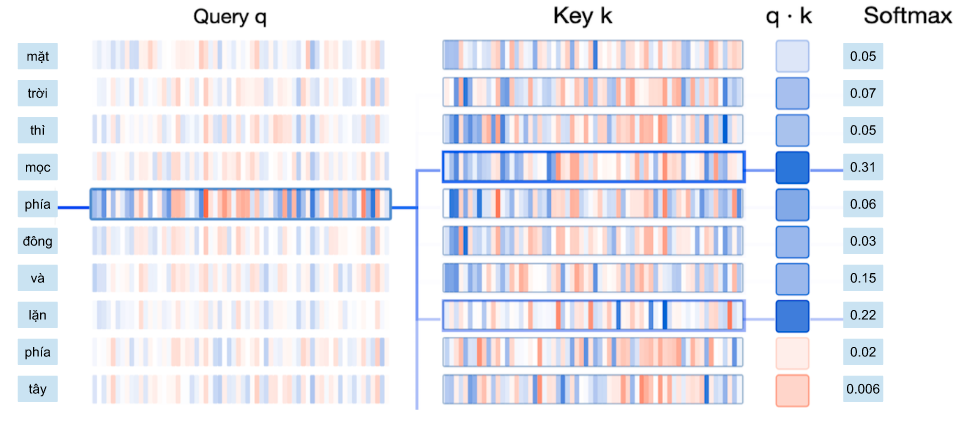


Bạn có thể tưởng tượng cơ chế self attention giống như cơ chế tìm kiếm. Với một từ cho trước, cơ chế này sẽ cho phép mô hình tìm kiếm trong cách từ còn lại, từ nào “giống” để sau đó thông tin sẽ được mã hóa dựa trên tất cả các từ trên.

Đầu tiên, với môi từ chúng ta cần tạo ra 3 vector: query, key, value vector bằng cách nhân ma trận biểu diễn các từ đầu vào với ma trận học tương ứng.

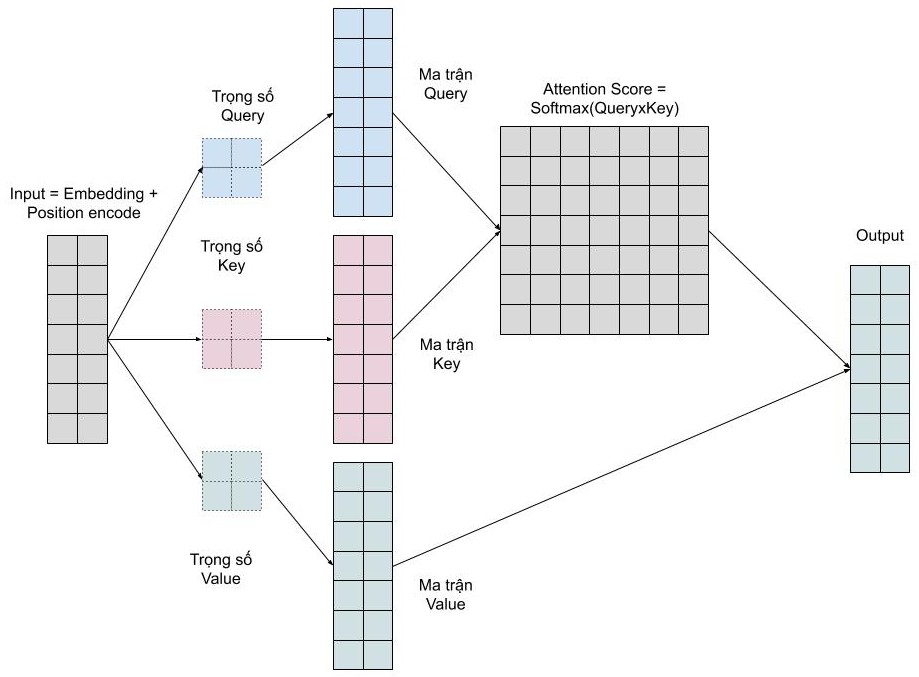
* query vector: vector dùng để chứa thông tin của từ được tìm kiếm, so sánh. Giống như là câu query của google search.
* key vector: vector dùng để biểu diễn thông tin các từ được so sánh với từ cần tìm kiếm ở trên. Ví dụ, như các trang web mà google sẽ so sánh với từ khóa mà bạn tìm kiếm.
* value vector: vector biểu diễn nội dung, ý nghĩa của các từ. Các bạn có thể tượng tượng, nó như là nội dung trang web được hiển thị cho người dùng sau khi tìm kiếm.

Để tính tương quan, chúng ta đơn giản chỉ cần tính tích vô hướng dựa các vector query và key. Sau đó dùng hàm softmax để chuẩn hóa chỉ số tương quan trong đoạn 0-1, và cuối cùng, tính trung bình cộng có trọng số giữa các vector values sử dụng chỉ số tương quan mới tính được. Quá dễ !!!



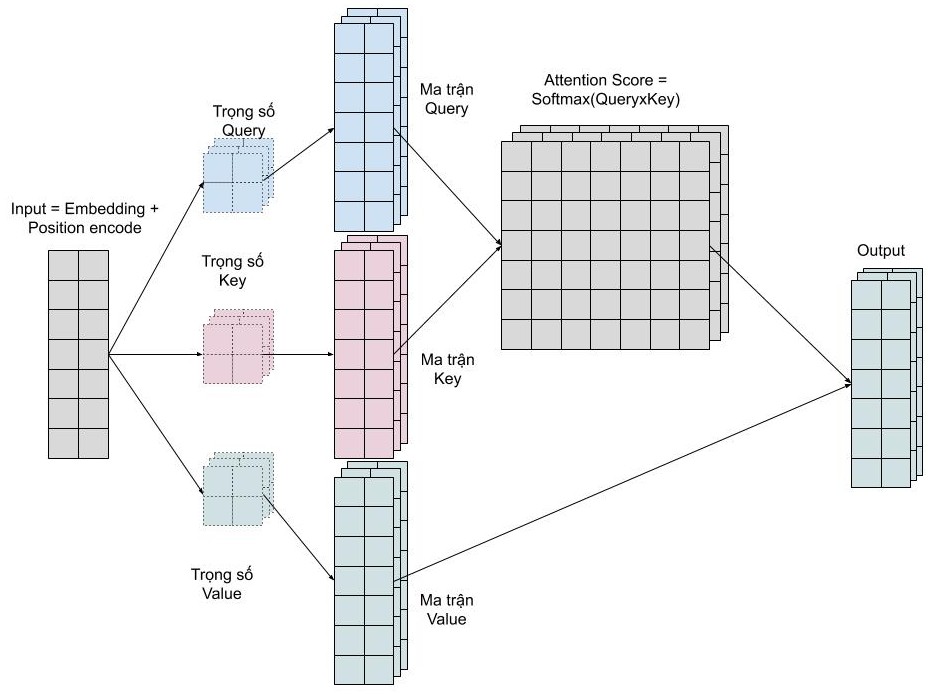
Cụ thể hơn, quá trình tính toán attention vector có thể được tóm tắt làm 3 bước như sau:

* Bước 1: Tính ma trận query, key, value bằng cách khởi tạo 3 ma trận trọng số query, key, vector. Sau đó nhân input với các ma trận trọng số này để tạo thành 3 ma trận tương ứng.
* Bước 2: Tính attention weights. Nhân 2 ma trận key, query vừa được tính ở trên với nhau để với ý nghĩa là so sánh giữa câu query và key để học mối tương quan. Sau đó thì chuẩn hóa về đoạn [0-1] bằng hàm softmax. 1 có nghĩa là câu query giống với key, 0 có nghĩa là không giống.
* Bước 3: Tính output. Nhân attention weights với ma trận value. Điều này có nghĩa là chúng ta biểu diễn một từ bằng trung bình có trọng số (attention weights) của ma trận value.



Multi Head Attention

Chúng ta muốn mô hình có thể học nhiều kiểu mối quan hệ giữa các từ với nhau. Với mỗi self-attention, chúng ta học được một kiểu pattern, do đó để có thể mở rộng khả năng này, chúng ta đơn giản là thêm nhiều self-attention. Tức là chúng ta cần nhiều ma trận query, key, value mà thôi. Giờ đây ma trận trọng số key, query, value sẽ có thêm 1 chiều depth nữa.



Multi head attention cho phép mô hình chú ý đến đồng thời những pattern dễ quan sát được như sau.

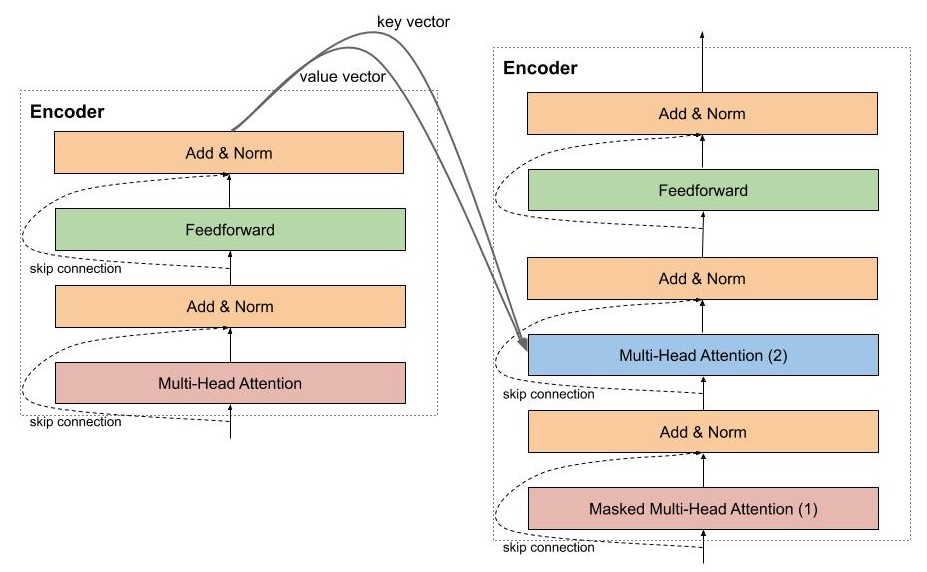
* Chú ý đến từ kế trước của một từ
* Chú ý đến từ kế sau của một từ
* Chú ý đến những từ liên quan của một từ

Residuals Connection và Normalization Layer

Trong kiến trúc của mô hình transformer, residuals connection và normalization layer được sử dụng mọi nơi, giống như tinh thần của nó. 2 kỹ thuật giúp cho mô hình huấn luyện nhanh hội tụ hơn và trách mất mát thông tin trong quá trình huấn luyện mô hình, ví dụ như là thông tin của vị trí các từ được mã hóa.

#### 2.7 Decoder

Decoder thực hiện chức năng giải mã vector của câu nguồn thành câu đích, do đó decoder sẽ nhận thông tin từ encoder là 2 vector key và value. Kiến trúc của decoder rất giống với encoder, ngoại trừ có thêm một multi head attention nằm ở giữa dùng để học mối liên quan giữ từ đang được dịch với các từ được ở câu nguồn.



Masked Multi Head Attention

Masked Multi Head Attention tất nhiên là multi head attention mà chúng ta đã nói đến ở trên, có chức năng dùng để encode các từ câu câu đích trong quá trình dịch, tuy nhiên, lúc cài đặt chúng ta cần lưu ý rằng phải che đi các từ ở tương lai chưa được mô hình dịch đến, để làm việc này thì đơn giản là chúng ta chỉ cần nhân với một vector chứa các giá trị 0,1.

Trong decoder còn có một multi head attention khác có chức năng chú ý các từ ở mô hình encoder, layer này nhận vector key và value từ mô hình encoder, và output từ layer phía dưới. Đơn giản bởi vì chúng ta muốn so sánh sự tương quan giữa từ đang được dịch với các từ nguồn.

Final Fully Connected Layer, Softmax và Loss function

Giống như nhiều mô hình khác, chúng ta cần thêm một fully connected layer để chuyển output từ layer phía trước thành ma trận có chiều băng số từ mà các bạn cần dự đoán. Sau đó thì đến softmax để các bạn tính được xác suất của từ xuất hiện tiếp theo là bao nhiêu.

Loss function thì tất nhiên là cross-entropy mà thôi, giống như ở các mô hình phân loại khác mà các bạn đã làm quen.

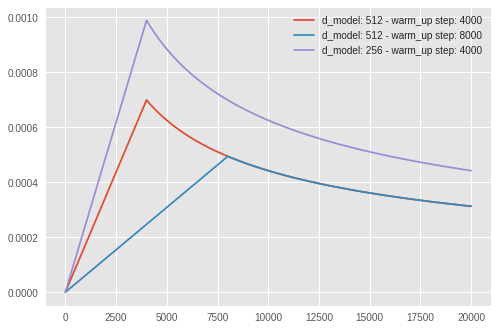
#### 2.8 Các kỹ thuật đặc biệt để huấn luyện Transformer

Để huấn luyện mô hình transformer, các bạn cần phải biết đến 2 kỹ thuật rất thú vị này. Nếu không sử dụng kỹ thuật đầu tiên về optimizer thì mô hình transformer sẽ **không hội tụ** được luôn đấy :))

Optimizer

Để huấn luyện mô hình transformer, các bạn vẫn sử dụng Adam, tuy nhiên, learning rate cần phải được điều chỉnh trong suốt quá trình học theo công thức sau

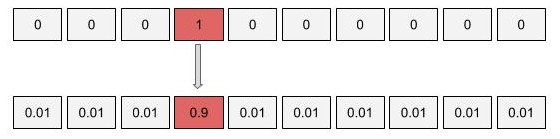
lr\\_rate=d^{-0.5}\_{d\\_model}\*min(step\\_num^{-0.5}, step\\_num\*warmup\\_steps^{-1.5})*lr*\_*rate*=*dd*\_*model*−0.5​∗*min*(*step*\_*num*−0.5,*step*\_*num*∗*warmup*\_*steps*−1.5)



Cơ bản thì learning rate sẽ tăng dần trong các lần cập nhật đầu tiên, các bước này được gọi là warm up step, lúc này mô hình sẽ ‘chạy’ tẹt ga. Sau đó learning rate lại giảm dần, để mô hình hội tụ.

Label Smoothing

Với mô hình nhiều triệu tham số của transformer, thì việt overfit là chuyện dễ dàng xảy ra. Để hạn chế hiện tượng overfit, các bạn có thể sử dụng kỹ thuật label smoothing. Về cơ bản thì ý tưởng của kỹ thuật này khá đơn giản, chúng ta sẽ phạt mô hình khi nó quá tự tin vào việc dự đoán của mình. Thay vì mã hóa nhãn là một one-hot vector, các bạn sẽ thay đổi nhãn này một chút bằng cách phân bố một tí xác suất vào các trường hợp còn lại.



Giờ thì các bạn sẽ an tâm khi có thể để số epoch lớn mà không lo rằng mô hình sẽ overfit nặng nề.

Implementation

Với mình lý thuyết mà không có thực hành có nghĩa là vẫn chưa hiểu rõ được lý thuyết đó. Cho nên mình đã cài đặt và mô tả chi tiết các bước làm trong notebook [tại đây](https://github.com/pbcquoc/transformer/blob/master/transformer.ipynb).

Dataset song ngữ anh-việt

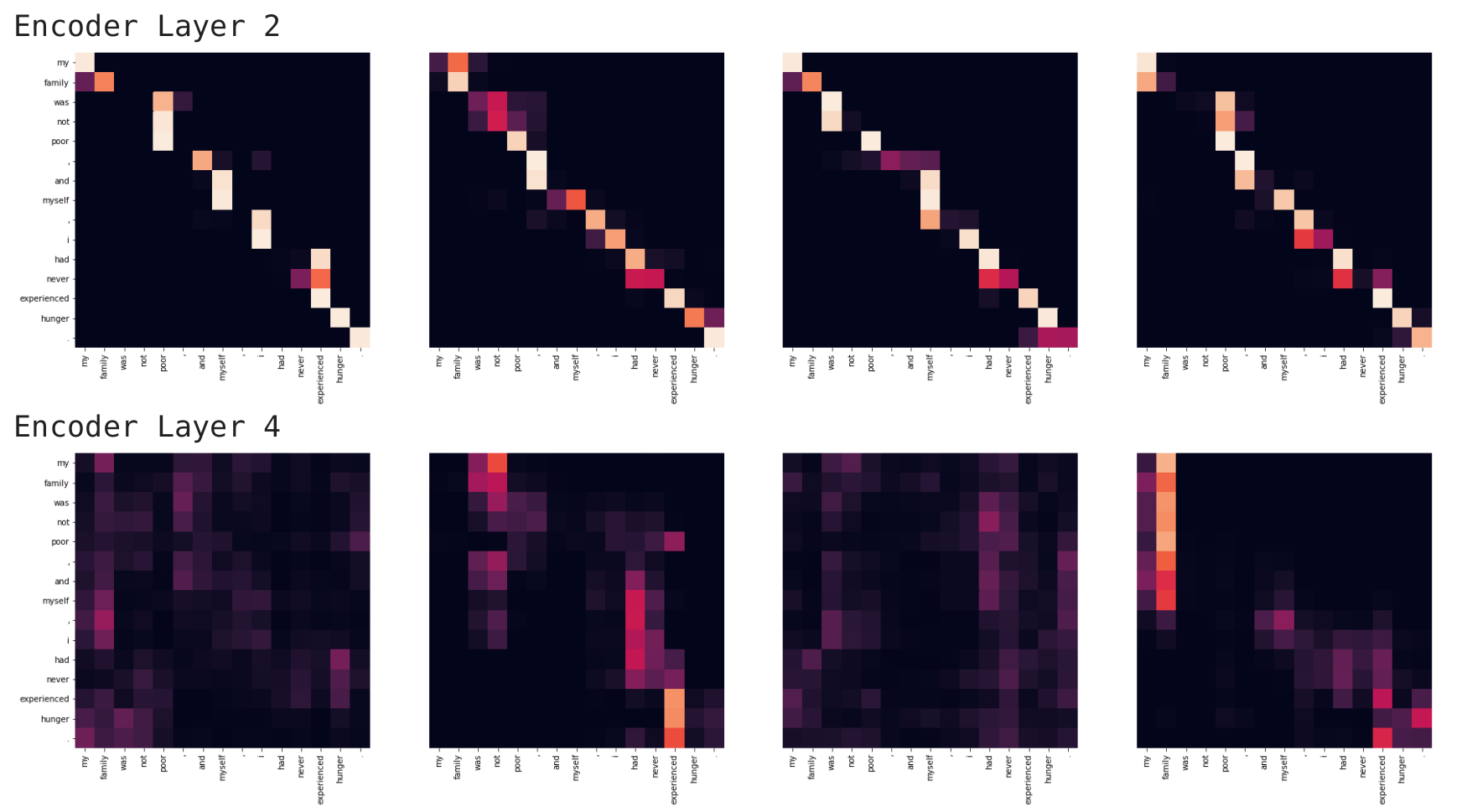
Đồng thời mình cũng cung cấp bộ dữ liệu song ngữ được thu thập trên TED bao gồm hơn 600k câu song ngữ anh-việt [tại đây](https://drive.google.com/file/d/1Fuo_ALIFKlUvOPbK5rUA5OfAS2wKn_95/view?usp=sharing). Với bộ dữ liệu mình hy vọng các bạn sẽ không gặp phải khó khăn khi thử nghiệm những kiến thức mới.

Visualization

Visualize trong số của các mô hình sử dụng cơ chế attention thực sự rất thú vị. Trong mô hình transformer, chúng ta visualize tại encoder và tại decoder. Các bạn có thể visualize đồng thời tại các heads của multi-head attentions, và tại layers khác nhau.

Encoder Visualize

Các bạn có thể dùng heatmap để visualize giá trị attention, sẽ cho chúng ta biết khi encode một câu mô hình chú ý từ gì ở lân cận

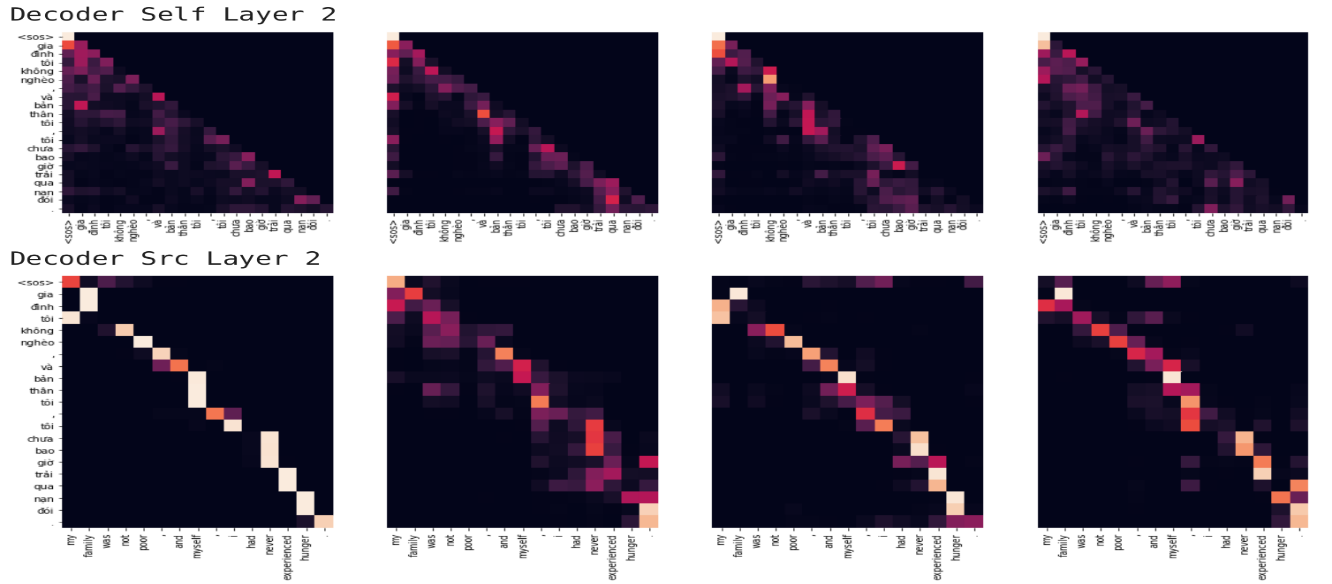


Ở đây mình visualize giá trị attention của encoder layer số 2 và 4, tại các head 0,1,2,3 (trong cài đặt các bạn có tổng cộng 6 encoder layer và 8 heads nhé). Nhìn vào các heatmaps ở trên, các bạn có thể thấy được rằng khi encode một từ mô hình sẽ nhìn vào các từ liên quan xung quanh. Ví dụ từ **family** có thể được mã hóa bằng 2 từ liên quan như **my** và **family**.

Decoder Visualize

Ở decoder, các bạn có 2 loại visualization

* self attention: giá trị attention khi mô hình decoder mã hóa câu đích lúc dịch
* src attention: giá trị attention khi mô hình decoder sử dụng câu src lúc dịch



Ở ví dụ này mình visualize decoder layer số 2, tại 4 heads 0,1,2,3. Các bạn có thể quan sát được khi encode từ **đình** mô hình sẽ nhìn vào các từ kế cạnh là **gia** và **tôi**, (và còn nhiều kiểu pattern khác nữa nhé). Còn khi dự đoán từ **tôi** mô hình sẽ nhìn vào từ **my**.

### Chương 3 Sử dụng Phobert để nhúng câu Tiếng Việt

#### 3.1 BERT là gì ?

BERT là một model biểu diễn ngôn ngữ (Language Model- LM) được google giới thiệu vào năm 2018. Trước khi BERT ra đời thì các tác vụ như: phân loại cảm xúc văn bản (tốt hay xấu, tích cực hay tiêu cực), sinh văn bản, dịch máy,…. đều sử dụng kiến trúc RNN. Kiến trúc này có nhiều nhược điểm như train chậm, mất quan hệ giữa các từ xa nhau…., cái này mình đều đã nói trong video về transformer [**tại đây**](https://www.miai.vn/2020/12/01/mi-transformer-tim-hieu-transformer-theo-cach-de-hieu-de-nho/), các bạn xem lại phần đầu nhé.

Tại thời điểm công bố, BERT đã nhanh chóng trở thành bá đạo trong mảng NLP bởi những cải tiến chưa từng có ở những model trước đó như: tăng độ chính xác, GLUE score (General Language Understanding Evaluation score)… Bạn nào cần chi tiết thì đọc thêm [**paper gốc**](https://arxiv.org/abs/1810.04805) nhé.

Túm lại cái là BERT là một LM ngon nhất hiện nay (thời điểm viết bài), được train sẵn rồi, anh em cứ vác về mà thực hiện các bài toán riêng của mình.

#### 3.2 – Làm thế quái nào mà BERT nó ngon hơn RNN?

Lý do chính là bởi nó có nhúng thêm ngữ cảnh (Context) vào trong các vector embedding các bạn ah. Ngữ cảnh là một thứ vô cùng quan trọng trong ngôn ngữ. Với các ngữ cảnh khác nhau thì các từ trong câu được hiểu theo ý nghĩa hoàn toàn khác nhau, các LM bỏ qua ngữ cảnh thì khó có thể đạt được chất lượng tốt.

Ngược về quá khứ ta sẽ thấy được sự phát triển của các phương pháp nhúng từ (word embbeding) qua thời gian.

**Đầu tiên là món embedding không ngữ cảnh**

Các bạn có để ý trước giờ mình dùng Word2Vec hay FastText không? Mỗi từ trong vocab sẽ có một vector để dại diện cho nó và trong bất cứ câu nào, đoạn văn nào thì từ đó vẫn chỉ được biểu diễn bởi vector đó.

Ví dụ ta có 2 câu:

* Hôm nay em đi **chơi** bóng đá
* Thằng kia nó **chơi** em anh ạ

Đó, 2 cái từ chơi kia rõ ràng nghĩa khác nhau (mình khỏi giải thích nghĩa nha kaka). Nhưng với việc nhúng từ không ngữ cảnh thì cả 2 từ này sẽ đều ánh xạ ra chung 1 vector word embedding và điều này đã làm giảm đi sự mềm mại và đa nghĩa của ngôn ngữ.

**Tiếp theo là món embedding có ngữ cảnh một chiều**

Người ta sử dụng các kiến trúc mạng RNN để có thể tạo ra mối quan hệ thứ tự giữa các từ trong câu, từ đó tạo ra vector nhúng từ có ngữ cảnh. Tuy nhiên việc này chỉ thực hiện được theo một chiều left-to-right hoặc right-to-left mà thôi. Một số mạng phức tạp hơn đã sử dụng BiLSTM để chạy dọc theo câu theo 2 hướng ngược nhau nhưng 2 hướng này lại độc lập, chả liên quan gì đến nhau nên có thể xem là một chiều mà thôi.

Ví dụ một case mà chiêu nhúng từ một chiều này fail nhé. Giả sử ta có bài toàn như sau:

* Câu văn gốc: “Hôm nay Nam đưa bạn gái đi chơi”
* Sau đó chúng ta che từ bạn gái đi và câu trên trở thành “Hôm nay Nam đưa**[mask]**đi chơi”.
* Yêu cầu bài toán là dự đoán ra từ đã được masking.

Rồi, với mặt thường của chúng ta thì ngay khi nhìn thấy câu này thì nghĩ ngay đến từ bạn gái =)). Thề luôn! Nhưng model thì không như vậy, do nó chỉ được training một chiều, nó sẽ dự đoán [mask] từ các word trước đó là “Hôm nay Nam đưa” và thế là kết quả [mask] có thể sẽ được dự là: “tiền”,”mắt”, “hàng”…. (tuỳ vào corpus của chúng ta).

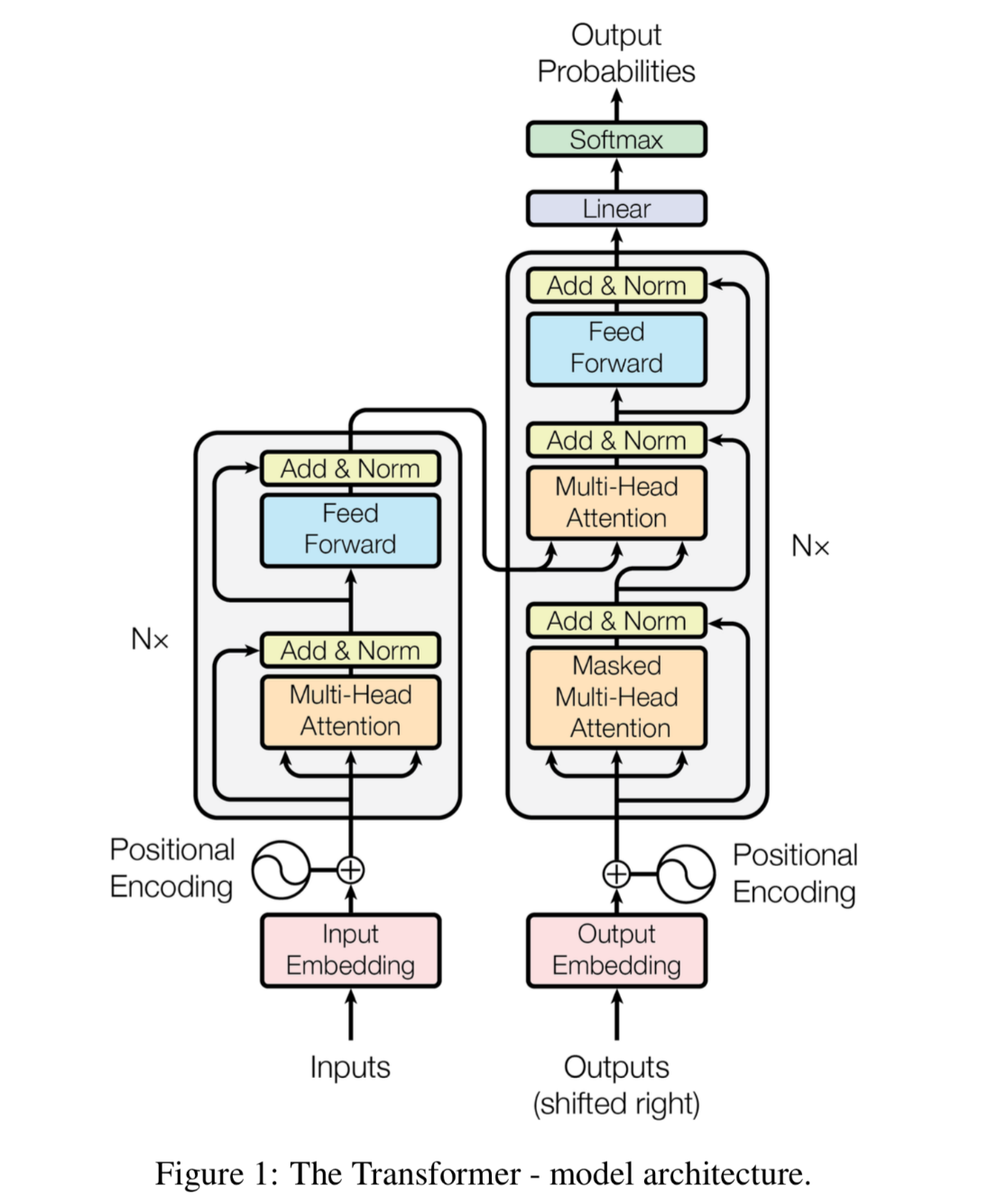
Toang rồi! Trong khi đó nếu nó kết hợp thêm từ đi chơi ở sau thì có phải ngon không cơ chứ =)). Cơ mà đưa hàng đi chơi cũng có nghĩa ấy chứ nhỉ anh em =)) (fun tý các bạn nhá).

**Và BERT đã đến và làm điều chúng ta cần**

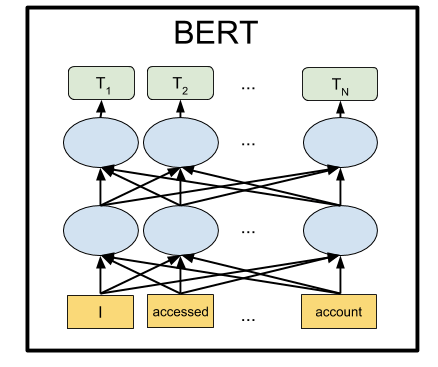
Ngay trong cái tên của BERT đã thấy ngay chữ Bidirectional (2 chiều) rồi. Tóm lại là một từ trong câu sẽ được biểu diễn một cách có liên quan đến cả từ trước lẫn từ sau, hay nói cách khác là liên quan đến tất cả các từ còn lại trong câu. Do đó khi ta che 1 từ trong câu đi, ví dụ như từ “bạn gái” bên trên thì lập tức model có thể predict ra khá chính xác vì dựa vào cả đoạn “Hôm nay Nam đưa” và “đi chơi” kaka.

#### 3.3 – Kiến trúc của BERT ra sao?

Yeah, bạn nào đã xem bài trước của mình về transformer chắc còn nhớ cái mô hình mạng transformer này:



Rồi, bây giờ ông BERT nhà mình chặt transformer làm đôi, chỉ lấy phần Encoder bên trái và bỏ đi phần Decoder bên phải. Tóm lai chỉ còn cái đoạn mà nhét câu văn bản vào và đầu ra là các encoder output như hình:

Nguồn: [Tại đây](https://habrastorage.org/getpro/habr/post_images/2bd/0ba/1c4/2bd0ba1c4fb80fe4d771f555168c9ff0.png)

Bạn nào chưa rõ về output Encoder là gì thì lại xem lại bài trước của mình về Transformer nha!

#### 3.4 – Chặt một nửa vậy rồi train kiểu gì?

Chắc hẳn đây cũng là câu hỏi nhiều bạn thắc mắc. Mình cũng vậy! Đang học mạng Transformer, quen cách train bên đó giờ chặt một nửa thì train sao. Okie, mình sẽ cùng tìm hiểu cách train ngay sau đây.

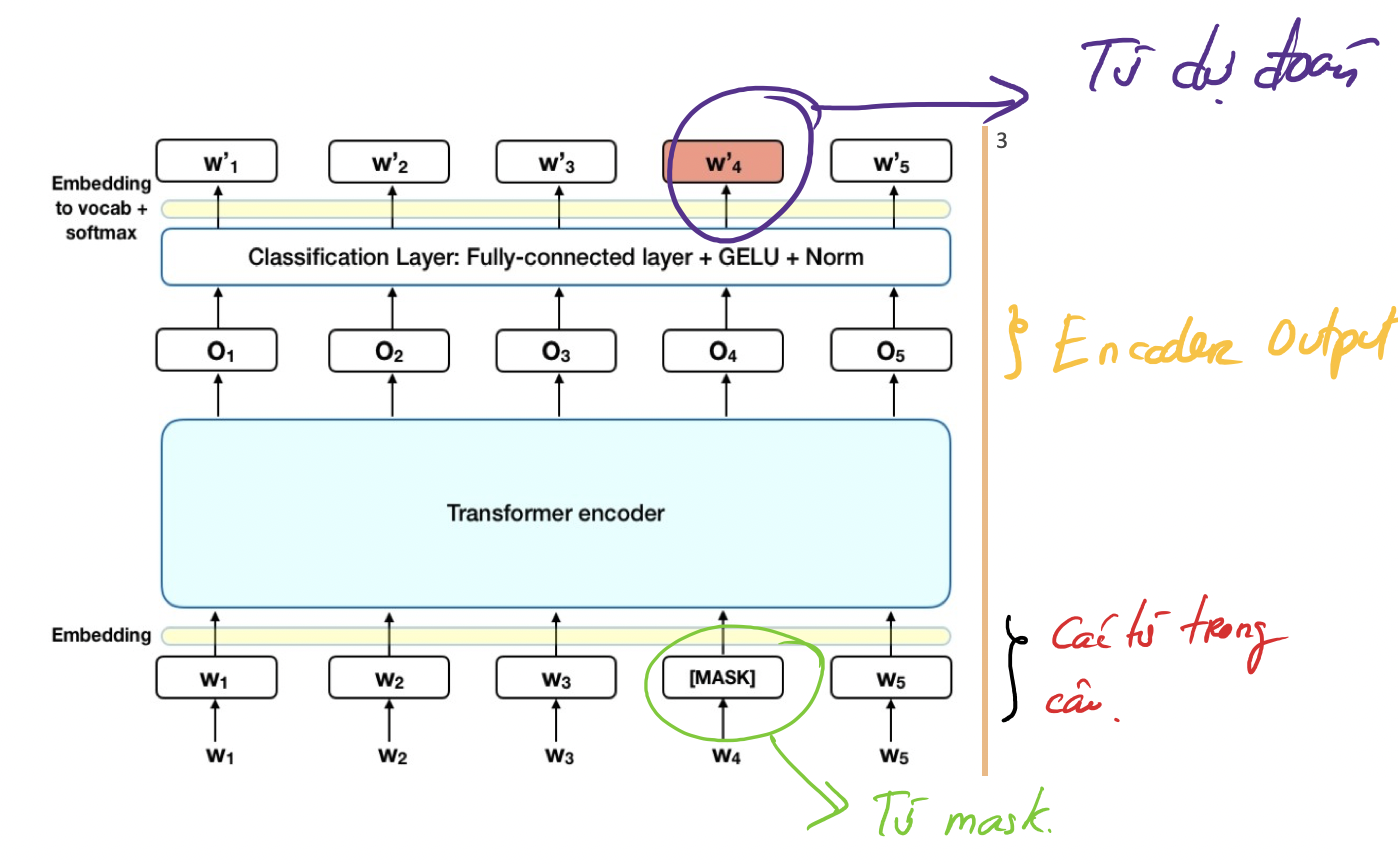
BERT được train đồng thời 2 task gọi là **Masked LM** (để dự đoán từ thiếu trong câu) và **Next Sentence Prediction** (NSP – dự đoán câu tiếp theo câu hiện tại). Hai món này được train đồng thời và loss tổng sẽ là kết hợp loss của 2 task và model sẽ cố gắng minimize loss tổng này. Chi tiết 2 task này như sau:

**Masked LM**

Với task này, ta train sẽ thực hiện che đi tầm 15% số từ trong câu và đưa vào model. Và ta sẽ train để model predict ra các từ bị che đó dựa vào các từ còn lại (đúng như câu Nam đưa bạn gái đi chơi ở trên đó).

Cụ thể là:

* Thêm một lớp classification lên trên encoder output
* Đưa các vector trong encoder ouput về vector bằng với vocab size, sau đó softmax để chọn ra từ tương ứng tại mỗi vị trí trong câu.
* Loss sẽ được tính tại vị trí masked và bỏ qua các vị trí khác (để đánh giá xem model dự đoán từ mask đúng/sai ntn mà, các từ khác đâu có liên quan).



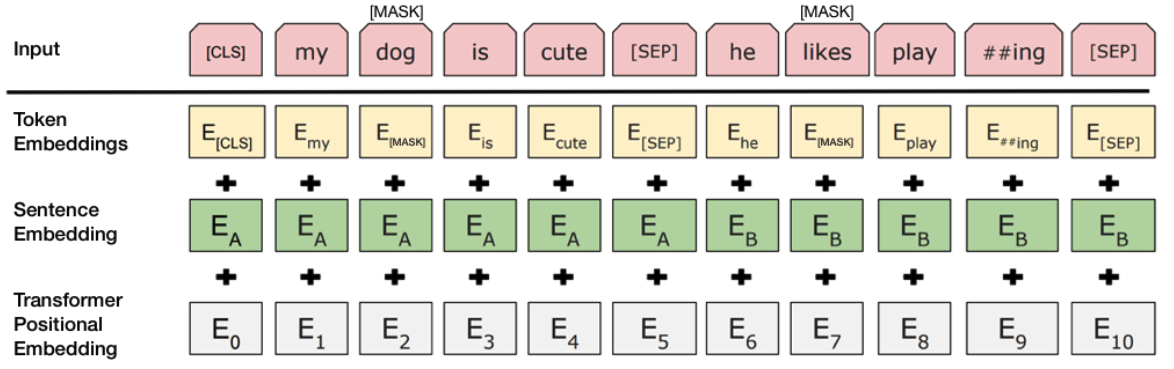
**Next Sentence Prediction (NSP)**

Với task này thì model sẽ được feed cho một cặp câu và nhiệm vụ của nó là output ra giá trị 1 nếu câu thứ hai đúng là câu đi sau câu thứ nhất và 0 nếu không phải. Trong quá trinh train, ta chọn 50% mẫu là Positive (output là 1) và 50% còn lại là Negative được ghép linh tinh (output là 0).

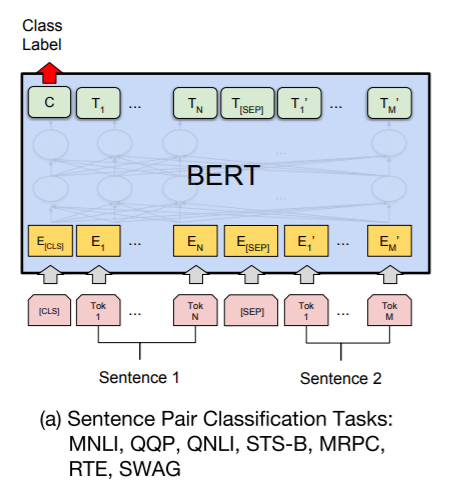
Cụ thể cách train như sau:

* Bước 1: Ghép 2 câu vào nhau và thêm 1 số token đặc biệt để phân tách các câu. Token [CLS] thêm vào đầu cầu Thứ nhất, token [SEP] thêm vào cuối mỗi câu. Ví dụ ghép 2 câu “Hôm nay em đi học” và “Học ở trường rất hay” thì sẽ thành [CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP]
* Bước 2. Mỗi token trong câu sẽ được cộng thêm một vector gọi là Sentence Embedding, thực ra là đánh dấu xem từ đó thuộc câu Thứ nhất hay câu thứ 2 thôi. Ví dụ nếu thuộc câu Thứ nhất thì cộng thêm 1 vector toàn số “0” có kích thước bằng Word Embedding, và nếu thuộc câu thứ 2 thì cộng thêm một vector toàn số “1”.
* Bước 3. Sau đó các từ trong câu đã ghép sẽ được thêm vector Positional Encoding vào để đánh dấu vị trí từng từ trong câu đã ghép (bạn nào chưa biết thì xem lại bài về Transformer nhé).
* Bước 4. Đưa chuỗi sau bước 3 vào mạng.
* Bước 5. Lấy encoder output tại vị trí token [CLS] được transform sang một vector có 2 phần tử [c1 c2].
* Bước 6. Tính softmax trên vector đó và output ra probality của 2 class: Đi sau và Không đi sau. Để thể hiện câu thứ hai là đi sau câu thứ nhất hay không, ta lấy argmax là okie

Các bước tạo Input:

Nguồn: [Tại đây](https://miro.medium.com/max/1400/0*m_kXt3uqZH9e7H4w.png)

Và đây là cách lấy output đầu ra:

Nguồn: [Tại đây](https://user-images.githubusercontent.com/4702353/47341717-56a70f80-d6d4-11e8-9b9d-6b31d0dfc4de.png)

Rồi, thế rồi người ta đưa dữ liệu vào, vô số dữ liệu và train trông một khoảng thời gian dài với bao nhiêu công sức, tiền điện, hao mòn máy móc thì chúng ta đã có một Pretrain Language Model ngon nghẻ.

Mô hình BERT bằng tiếng Anh ban đầu đi kèm với hai dạng tổng quát được đào tạo trước:[[1]](https://vi.wikipedia.org/wiki/BERT_(m%C3%B4_h%C3%ACnh_ng%C3%B4n_ng%E1%BB%AF)#cite_note-:0-1): (1) mô hình the BERTBASE, kiến trúc mạng NN chứa 12-lớp, 110M tham số, và (2) mô hình BERTLARGE model, kiến trúc chứa 24-lớp, 340M tham số. Cả hai đều được huấn luyện từ [BooksCorpus](https://vi.wikipedia.org/w/index.php?title=BooksCorpus&action=edit&redlink=1)[[4]](https://vi.wikipedia.org/wiki/BERT_(m%C3%B4_h%C3%ACnh_ng%C3%B4n_ng%E1%BB%AF)#cite_note-4) với 800M từ, và một phiên bản của [Wikipedia tiếng Anh](https://vi.wikipedia.org/wiki/Wikipedia_ti%E1%BA%BFng_Anh) với 2,500M từ.

### Chương 4 : ĐƯỢC GÌ SAU CHẶNG ĐƯỜNG 5 TUẦN ?

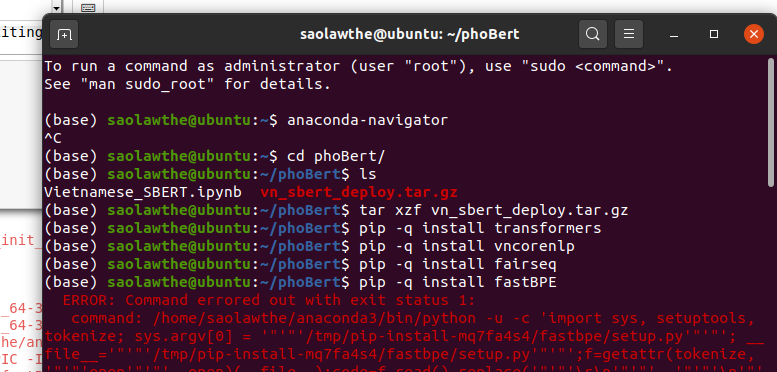
Nhóm 10 sau 5 tuần nghiên cứu đề tài “Tìm hiểu kỹ thuật nhúng câu Tiếng Việt và mô hình Transformer” đây là mô hình được tạo và phát triển bởi cựu sinh viên trường Uneti là Huy developer hiện đang công tác và làm việc cho tập đoàn VinGroup và sau 5 tuần đã thu được thành quả.

**Ý nghĩa :**

-Có cơ hội tiếp cận vào công nghệ Ai- machine learning mảng *Nhận dạng mẫu-nhận dạng câu văn* như đã phổ biến ở mục 1.4

-Có cơ hội học thêm framework mới trong python và lập trình python cơ bản

-Có cơ hội học thêm các lệnh cơ bản trọng môi trường ubuntu ( mở ứng dụng , mở foder , mở file , chạy file , cài thư viện …) mừng tượng ra cách giao cấu các phần source code với cửa sổ Terminal lệnh của ubuntu



-Chạy trên nền tảng ubuntu sử dụng local host để làm gì ? – Để dễ dàng quản lí source code cũng như có thể can thiệp vào code đó sửa đổi bổ dung nâng cấp hệ thống theo ý của mình ( tính tùy chỉnh) . Tích kiệm chi phí vận hành của con sever trực tuyến , chạy trên máy cá nhân ta thích mở lúc nào cũng được nhưng cũng có nhược điểm là dễ xảy ra lỗi khi xử lí source từ người khác kể cả môi trương window lẫn ubuntu.

-Đề tài nghiên cứu cho nhóm thêm kiến thức về Ai – macinelearning về mảng nhận dạng câu văn đồng nghĩa – câu trái nghĩa và thuật toán tính toán mức độ tương thích của câu văn đo đạc thông số cụ thể

-Thuật toán biểu đạt hệ vector trong không gian 3 chiều với hệ tọa độ x,y,z

-Chạy trên nền tảng google colab để làm gì ? -Để tích kiệm thời gian chạy code vì nó có sẵn môi trường cài đặt và con sever host trực tuyến là google colab , ưu điểm là nhanh nhưng khó có tính custermer như Anaconda.

### Chương 5 : Cài đặt môi trường

#### 5.1 Cài Anaconda

#### Anaconda là gì?

Nói một cách học thuật thì Anaconda là nền tảng mã nguồn mở về Khoa học dữ liệu trên Python thông dụng nhất hiện nay. Anaconda với hơn 11 triệu người dùng, Anaconda là cách nhanh nhất và dễ nhất để học Khoa học dữ liệu với Python hoặc R trên Windows, Linux và Mac OS X.

Nói một cách nhanh gọn theo phong cách Mì thì bạn chỉ cần nhớ 5 điểm sau về Anaconda:

* Miễn phí, không mất tiền mua.
* Là một phần mềm dùng để tạo ra các môi trường ảo, bạn sẽ làm việc trên môi trường ảo đó, đập phá thoải mái mà không gây lỗi cho môi trường python trong máy chính của bạn. Mình cũng đã có bài nói về môi trường ảo [**tại đây**](https://www.miai.vn/2020/07/15/virutal-env-moi-truong-ao-la-gi/)
* Nếu môi trường ảo bị bạn phá, bị lỗi, bạn remove đi tạo môi trường ảo khác. Nhanh cực!
* Giúp cài đặt ngon lành những thư viện khó cài như: tensorflow, dlib, cmake, numpy…
* Có thể tích hợp vào Pycharm giúp cho việc code đơn giản hơn rất nhiều.

#### Cài Anaconda ra làm sao?

Cài Anaconda dễ như cài Game và tất cả các phần mềm khác =)). Các bạn làm theo các step sau đây:

* Truy cập trang chủ [**https://www.anaconda.com/products/individual**](https://www.anaconda.com/products/individual), kéo xuống dưới và chọn phiên bản Anaconda dành cho Windows (chú ý check xem máy bạn 64bit hay 32bit để tải cho phù hợp).
* Sau khi tải được file exe về thì chạy file đó và Next liên tục ho đến khi nó cài xong.

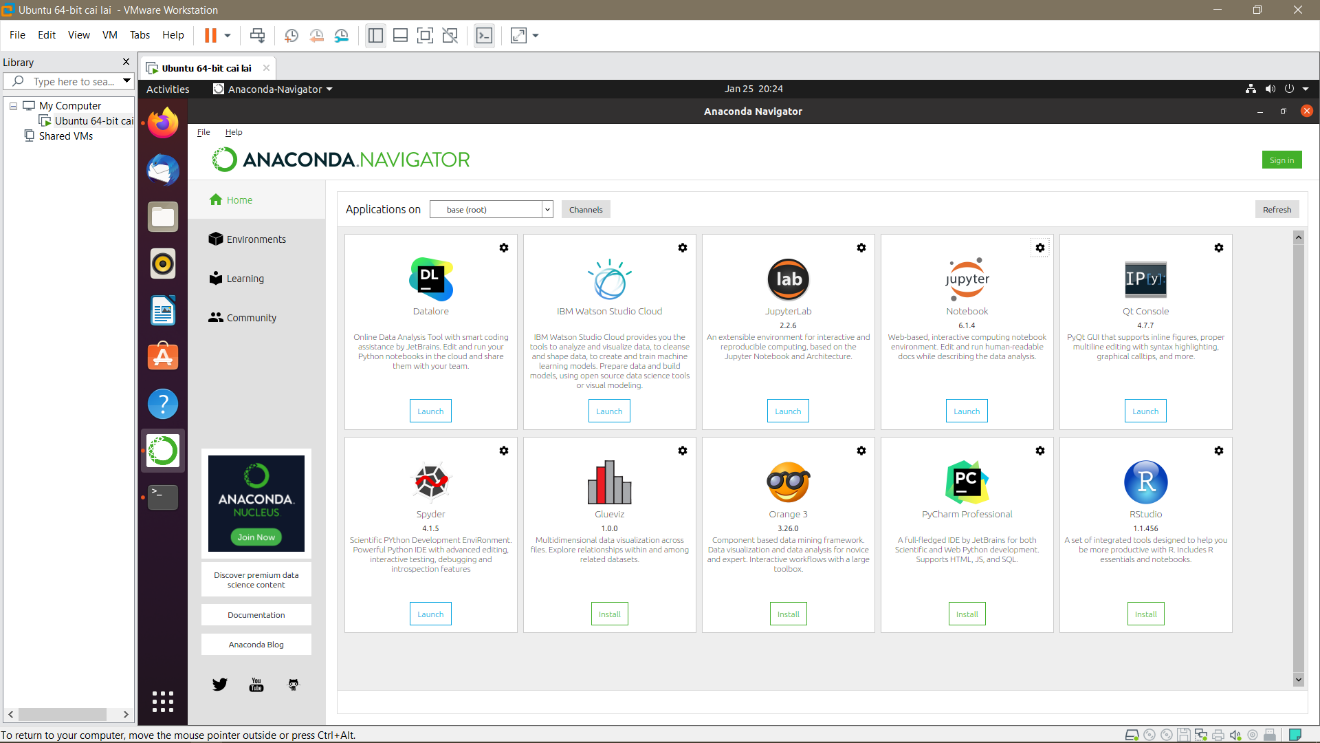
**Các thư viện cơ bản trên anaconda port :**

pip install numpy

pip install pandas

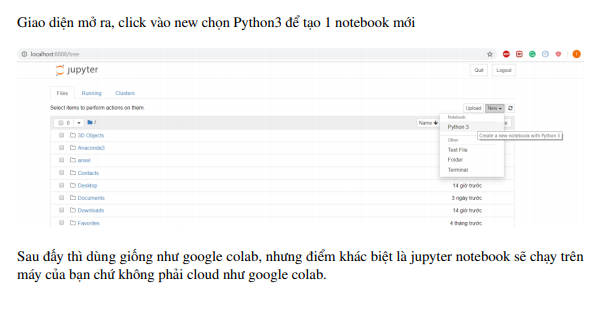
pip install matplotlib

pip install Keras

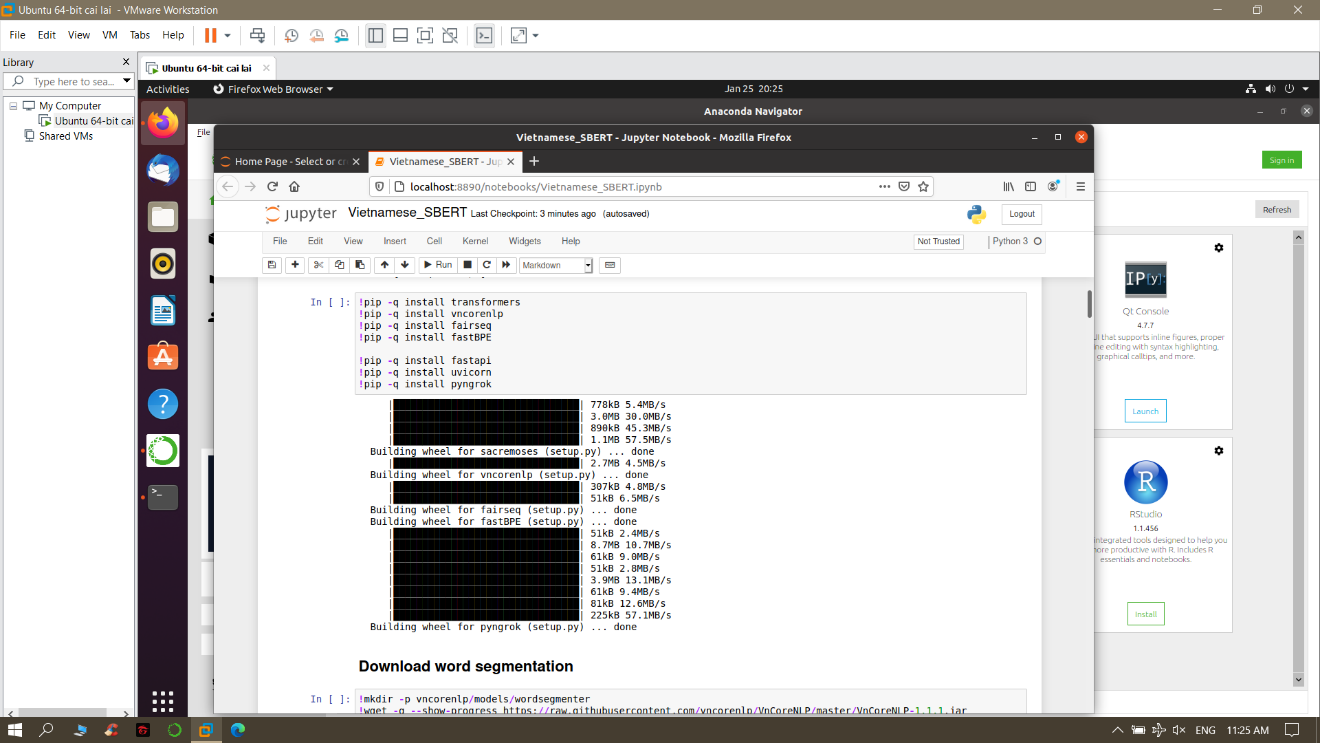


Hình 4.1 Mở giao diện Anaconda

Chọn mở Notebook

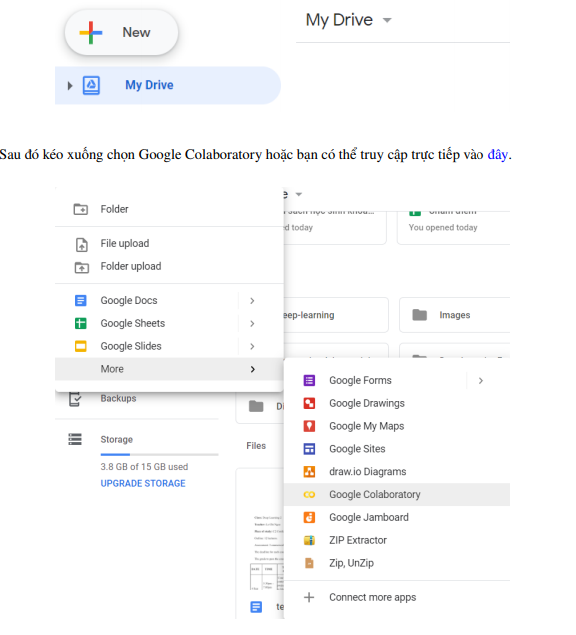


Hình 5.1.1 Đẩy file source lên sever

 Hình 4.1.2 Chạy chương trình trên host

#### 5.2 Môi trường Google Colab

5.2.1 Tạo foder trên google driver



Hình 5.2.1

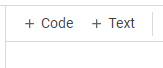
Trong trường hợp bạn không tìm thấy Google Colaboratory, hãy chọn "Connect more apps" ở phía bên dưới, sau đó search Google Colaboratory trên thanh tìm kiếm và cài đặt.

Tiếp đó, bạn click vào phần tên trên cùng của file để đổi tên file cho phù hợp

Tiêp theo chọn GPU bằng cách Bước này để chọn GPU chạy, bạn chọn Runtime -> Change runtime type ->GPU

5.2.2 Liên kết với google driver ở google colab

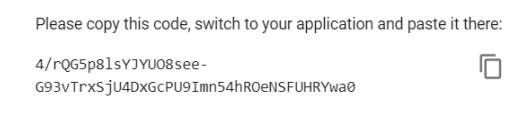
Vì code Python được chia thành từng khối (block) để chạy, bạn nhìn hình bên dưới có: • nút +Code để thêm 1 block code python • nút +Text để thêm 1 khối text (giống như comment nhưng có thể format được màu style hơn) • biểu tượng hình thùng rác để xóa khối code/text đi • nút mũi tên xoay ngang để chạy khối code/text đấy. Bên cạnh đó các bạn cũng có thể chạy khối code bằng các phím tắt cho nhanh hơn: – Cmd/Ctrl + Enter (Cmd cho Mac, Ctrl cho Win): chạy khối code/text. – Shift + Enter: Chạy và chuyển đối tượng xuống khối code bên dưới. – Alt + Enter: Chạy và thêm một khối code bên dưới.



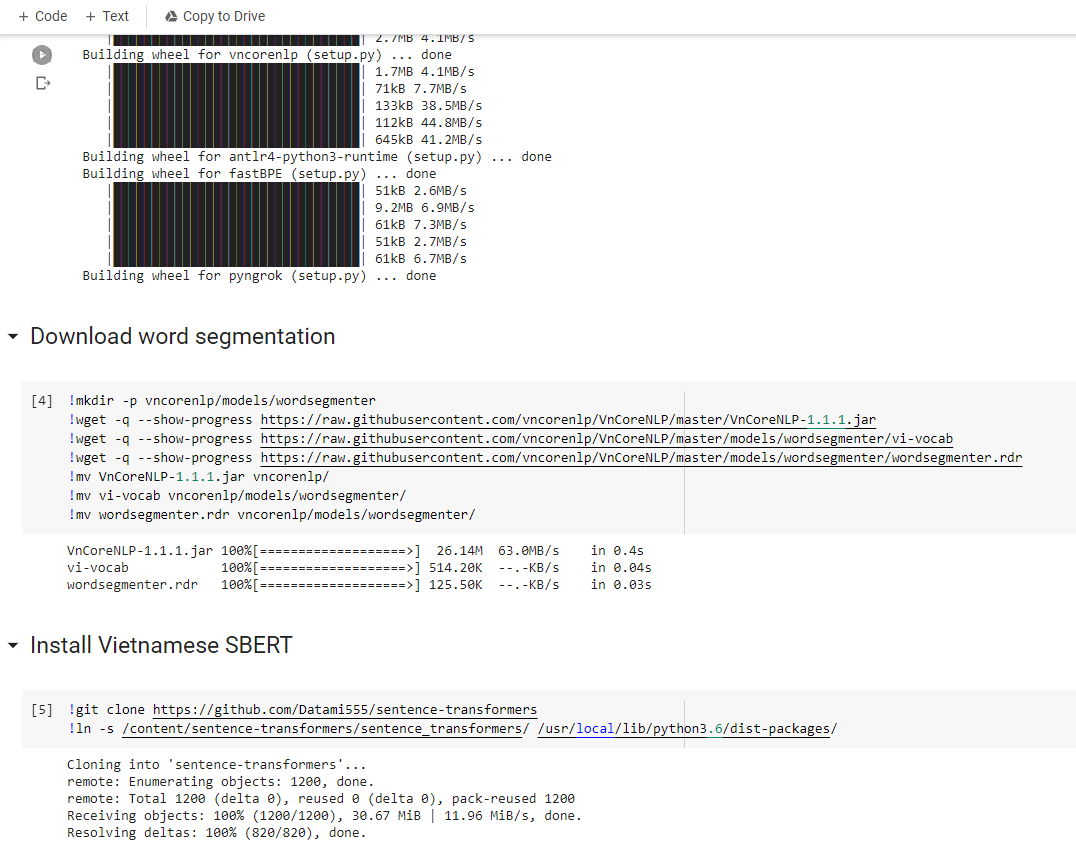
ấn vào “+code” và điền đoạn code này :

from google.colab import drive drive.mount('/content/gdrive')

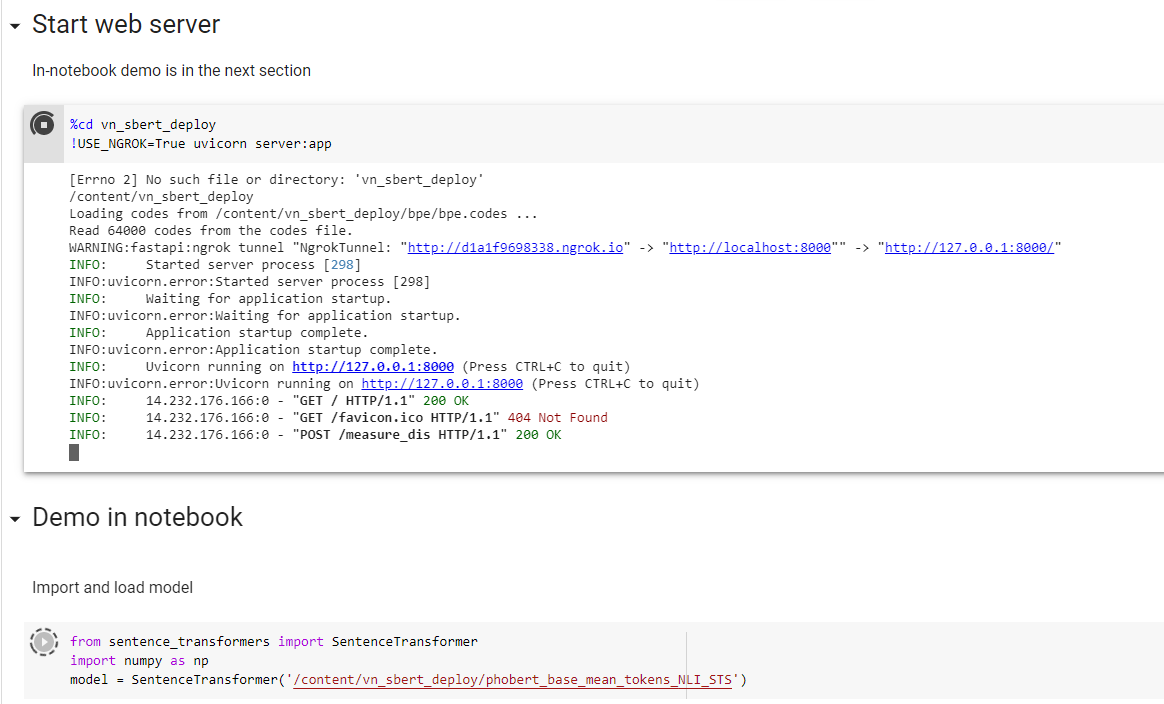
Sau khi ấn chạy đoạn code đấy, bạn click vào link trên, chọn tài khoản google bạn đang dùng, rồi chọn accept bạn sẽ có mã code như ở dưới.



Copy đoạn mã này và paste vào phần chạy của block hiện tại ở google colab rồi Enter . Sau khi link (mount) thành công bạn sẽ thấy dòng chữ **Mounted at /content/gdrive** là bạn đã thành công . Và giờ là chạy chương trình.



Hình 5.2.2 Chạy chương trình thành công

Hình 5.2.3 Start con sever thành công

Rồi click vào đường link "[http://d1a1f9698338.ngrok.io](http://d1a1f9698338.ngrok.io/)"

Để mở trang web demo

