ĐẠI HỌC QUỐC GIA HÀ NỘI

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

****

**Phạm Văn Dương**

**XÂY DỰNG CHATBOT HỖ TRỢ SỬ DỤNG PHẦN MỀM IQ-TREE**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

Ngành: Công nghệ thông tin định hướng thị trường Nhật Bản

**Hà Nội - 2021**

ĐẠI HỌC QUỐC GIA HÀ NỘI

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Phạm Văn Dương**

**XÂY DỰNG CHATBOT HỖ TRỢ SỬ DỤNG PHẦN MỀM IQ-TREE**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

Ngành: Công nghệ thông tin định hướng thị trường Nhật Bản

**Cán bộ hướng dẫn: Tiến sĩ Hoàng Thị Điệp**

**Cán bộ đồng hướng dẫn:**

**HÀ NỘI - 2021**

# TÓM TẮT

**Tóm tắt:** IQ-TREE là một hệ thống xử lý tin sinh học giải quyết được một lượng lớn những bài toán thường gặp trong giới tin sinh như Partitioned analysis, Bootstrapping partition model, Tree topology tests, .... Với sự phát triển mạnh của hệ thống IQ-TREE, vấn đề đặt ra là cần có một chatbot hỗ trợ giải đáp những thắc mắc của người dùng liên quan đến hệ thống này. Một chatbot thường gồm 3 thành phần là thành phần hiểu ngôn ngữ, thành phần quản lý hội thoại và thành phần sinh ngôn ngữ với thành phần hiểu ngôn ngữ là nòng cốt. Khóa luận: “Xây dựng chatbot hỗ trợ sử dụng hệ thống IQ-TREE” được nghiên cứu để xây dựng chatbot hỗ trợ hệ thống IQ-TREE với nhiều cách tiếp cận khác nhau trong việc hiểu ngôn ngữ tự nhiên mà tiêu biểu là dựa trên dựa trên một mô hình trí tuệ nhân tạo xử lý ngôn ngữ tự nhiên đang bùng nổ hiện nay là Transformer. Đồng thời khóa luận đánh giá từng ưu, nhược điểm của từng mô hình và tính phù hợp của nó với bài toán.

***Từ khóa:*** Chatbot, Xử lý ngôn ngữ tự nhiên, Transformer.

# LỜI CAM ĐOAN

Tôi xin cam đoan đây là kết quả nghiên cứu của tôi trong thời gian qua. Những số liệu và kết quả nghiên cứu là trung thực, không sao chép bất kỳ nguồn nào khác. Ngoài ra, trong bài báo cáo có sử dụng một số nguồn tài liệu tham khảo đã được trích dẫn nguồn và chú thích rõ ràng. Tôi xin hoàn toàn chịu trách nhiệm trước bộ môn, khoa và nhà trường về sự cam đoan này.

Hà Nội, ngày… tháng… năm…

Sinh Viên

# MỤC LỤC

[TÓM TẮT 1](#_Toc91552487)

[LỜI CAM ĐOAN 2](#_Toc91552488)

[MỤC LỤC 3](#_Toc91552489)

[BẢNG KÝ HIỆU VIẾT TẮT 5](#_Toc91552490)

[DANH MỤC HÌNH VẼ 6](#_Toc91552491)

[DANH MỤC BẢNG BIỂU 7](#_Toc91552492)

[MỞ ĐẦU 8](#_Toc91552493)

[Tính cấp thiết của vấn đề: 8](#_Toc91552494)

[Bố cục khóa luận: 9](#_Toc91552495)

[CHƯƠNG 1. TỔNG QUAN 10](#_Toc91552496)

[1.1 Hệ thống xử lý tin sinh học IQ-TREE 10](#_Toc91552497)

[1.2 Chatbot 10](#_Toc91552498)

[1.2.1 Trí tuệ nhân tạo 10](#_Toc91552499)

[1.2.2 Xử lý ngôn ngữ tự nhiên 12](#_Toc91552500)

[1.2.3 Chatbot 13](#_Toc91552501)

[1.3 Phát biểu bài toán 16](#_Toc91552502)

[CHƯƠNG 2. CÁC CÔNG NGHỆ SỬ DỤNG 18](#_Toc91552503)

[2.1 Các kỹ thuật trong xử lý ngôn ngữ tự nhiên 18](#_Toc91552504)

[2.1.1 Vector hóa 18](#_Toc91552505)

[2.1.2 TF-IDF và biến thể BM25: 18](#_Toc91552506)

[2.2 Kiến trúc Transformer 20](#_Toc91552507)

[2.2.1 Mạng nơ-ron nhân tạo, mạng nơ-ron truyền thẳng 21](#_Toc91552508)

[2.2.2 Mạng nơ-ron hồi quy và mạng nhớ dài hạn 22](#_Toc91552509)

[2.2.3 Kiến trúc Transformer 24](#_Toc91552510)

[2.3 Mạng nơ-ron kép 28](#_Toc91552511)

[2.4 Những mô hình và thư viện được sử dụng 29](#_Toc91552512)

[2.4.1 Beautifulsoup 29](#_Toc91552513)

[2.4.2 Pytorch 29](#_Toc91552514)

[2.4.3 Hugging Face 29](#_Toc91552515)

[2.4.4 Mô hình BERT 30](#_Toc91552516)

[2.4.5 Mô hình SBERT 31](#_Toc91552517)

[2.4.6 Mô hình T5 32](#_Toc91552518)

[2.3.6 Reactjs: 33](#_Toc91552519)

[CHƯƠNG 3. CÁCH TIẾP CẬN CHUNG CHO CHATBOT: 35](#_Toc91552520)

[3.1 Hiểu ngôn ngữ tự nhiên (NLU) 35](#_Toc91552521)

[3.2 Quản lý hội thoại 36](#_Toc91552522)

[3.3 Mô hình sinh ngôn ngữ (NLG) 38](#_Toc91552523)

[CHƯƠNG 4. GIẢI PHÁP XÂY DỰNG CHATBOT CHO PHẦN MỀM IQ-TREE 40](#_Toc91552524)

[4.1 Dữ liệu và khai phá dữ liệu 40](#_Toc91552525)

[4.2 Chi tiết hệ thống 41](#_Toc91552526)

[4.2.1 Server 42](#_Toc91552527)

[4.2.2 Client 51](#_Toc91552528)

[CHƯƠNG 5. HỆ THỐNG CHATBOT THỬ NGHIỆM 53](#_Toc91552529)

[5.1 Chương trình thử nghiệm 53](#_Toc91552530)

[5.2 Thiết kế đánh giá hệ thống 54](#_Toc91552531)

[5.2.1 Đánh giá trong kịch bản 54](#_Toc91552532)

[5.2.2 Đánh giá ngoài kịch bản 54](#_Toc91552533)

[5.3 Kết quả 55](#_Toc91552534)

[5.3.1 Kết quả đánh giá trong kịch bản 55](#_Toc91552535)

[5.3.2 Kết quả đánh giá ngoài kịch bản 57](#_Toc91552536)

[CHƯƠNG 6. KẾT LUẬN 68](#_Toc91552537)

[TÀI LIỆU THAM KHẢO 69](#_Toc91552538)

# BẢNG KÝ HIỆU VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Từ chuẩn | Diễn giải |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| ADN | Acid Deoxyribo nucleic | Thành phần của nhân tết bào |
| ARN | Acid Ribo nucleic | Bản sao của AND nhằm tổng hợp protein |
| ANN | Artificial Nerual Network | Mạng nơ-ron nhân tạo |
| LSTM | Long short-term memory | Mạng nhớ dài hạn – mạng nơ-ron cải tiến để giải quyết vấn đề phụ thuộc quá dài |
| ML | Machine Learning | Học máy, máy có khả năng học tập |
| NLG | Natural Language Generation | Thành phần sinh ngôn ngữ |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| NLU | Natural Language Understanding | Hiểu ngôn ngữ tự nhiên |
| TF-IDF | Term Frequency – Inverse Document Frequency | Chỉ số mức độ quan trọng của từ trong văn bản |
| RNN | Recurrent Neural Network | Mạng nơ-ron hồi quy |
| SNN | Siamese Nerual Network | Mạng nơ-ron kép |
| SVM | Vector Support Machine | Máy vector hỗ trợ |

# DANH MỤC HÌNH VẼ

Hình 1.1 Hệ thống IQ-TREE 10

Hình 1.3 Chatbot Eliza 11

Hình 1.4 Chatbot shopee 16

Hình 2.1 IDF với BM25 và IDF cổ điển 20

Hình 2.2 Mạng nơ-ron truyền thẳng 21

Hình 2.3 Mạng nơ-ron hồi quy 22

Hình 2.4 Mạng nhớ dài hạn 23

Hình 2.5 Trực quan hóa Self- Attention 25

Hình 2.6 Kiến trúc mô hình Transformer 26

Hình 2.7 Sự phát triển của các mô hình xử lý ngôn ngữ tự nhiên 27

Hình 2.8 Sự phát triển của kiến trúc Transformer về lượng tham số 27

Hình 2.9 Mạng nơ-ron kép trong nhận dạng chữ ký với constrative loss 29

Hình 2.10 Kết quả SQuAD 1.1 năm 2019 31

Hình 2.11 Đa tác vụ với mô hình T5 33

Hình 3.1 Mô hình FSA cho phản ánh chất lượng internet 36

Hình 4.1 Khai phá dữ liệu 40

Hình 4.2 Kiến trúc mô hình 40

Hình 4.3 Kiến trúc mô hình BERT 42

Hình 4.4 Phân lớp ý định bằng cosine và BM25 44

Hình 4.5 Kết quả STSB năm 2021 45

Hình 4.6 Phân lớp ý định dựa trên STS-B với T5 46

Hình 4.7 Kiến trúc mô hình T5 48

Hình 4.8 Nhiệm vụ STS-B đối với mô hình T5 49

Hình 5.1 Ứng dụng thử nghiệm 52

# DANH MỤC BẢNG BIỂU

Biểu đồ 4.1 Phân phối mẫu cho từng nhãn với phương pháp sử dụng BERT 43

Biểu đồ 4.2 Thống kê số lượng mẫu huấn luyện cho từng điểm số với STS-B 47

Biểu đồ 5.1 Độ chính xác mô hình T5 khi đóng băng từ trái sang phải 54

Biểu đồ 5.2 Độ chính xác mô hình T5 khi đóng băng tất cả các tổ hợp 55

Biểu đồ 5.3 Độ chính xác cho từng mô hình tính trên tập thử nghiệm lý thuyết 56

Biểu đồ 5.4 Thống kê kết quả các phương pháp phân lớp ý định trên thực nghiệm 65

Bảng 3.1 Minh họa mô hình Frame-base 35

Bảng 3.2 Minh họa sinh ngôn ngữ dựa trên luật sinh mẫu 36

Bảng 4.1 Các tham số huấn luyện mô hình T5 48

Bảng 5.1 Kết quả thử nghiệm các mô hình trên lý thuyết 54

Bảng 5.2 Kết quả thử nghiệm các mô hình trên thực nghiệm 55

# MỞ ĐẦU

## Tính cấp thiết của vấn đề:

Ngày nay, người dùng ngày càng có thói quen hướng đến những dịch vụ phục vụ 24/24 để được hỗ trợ nhanh nhất tất cả các hoạt động từ lớn tới nhỏ. Để đáp ứng nhu cầu này mà chatbot ngày càng được phổ biến và trở nên mạnh mẽ hơn. Nó được áp dụng vào mọi lĩnh vực từ ngân hàng, y tế, mua sắm, ... Đồng thời chatbot ngày càng có thể được tích hợp trên nhiều nền tảng như Facebook, WhatsApp, WeChat, … hay bất kỳ ứng dụng nhắn tin nào. Việc này đã mang lại sự phát triển lớn của ngành công nghiệp chatbot phục vụ việc chăm sóc khách hàng, tư vấn mua sắm hay giải trí, ... Chatbot đã và đang trở thành một nhu cầu thiết yếu trong nhiều ngành nhất định.  
Tin sinh học là sự kết hợp giữa công nghệ thông tin với sinh học mà ở đó người ta dùng thành tựu của những mô hình, thuật toán nhằm giải quyết những vấn đề mang tính thuần sinh học. Một số ứng dụng tiêu biểu của tin sinh học như dự đoán cấu trúc, dự đoán tương tác của protein, xác định cây gia phả, phân tích ADN/ARN, ... Với sự phát triển của tuệ nhân tạo, việc áp dụng công nghệ thông tin vào những vấn đề sinh học ngày càng dễ dàng và phổ biến. Đồng thời, việc đại dịch Covid-19 liên tục có những diễn biến phức tạp và liên tục xuất hiện những biến chủng mới càng thúc đẩy việc nghiên cứu tin sinh học nhằm giải quyết những vấn đề này.

Mô hình IQ-TREE là một trong những mô hình tin sinh học phổ biến hiện nay. Việc nhu cầu sử dụng hệ thống này ngày càng nhiều dẫn đến nhà phát triển liên tục nhận được những câu hỏi về hệ thống. Công cụ thường được sử dụng cho việc hỏi đáp này là google group. Tuy nhiên không thể có người túc trực 24/7 để giải quyết những câu hỏi của người dùng. Đồng thời việc nhiều câu hỏi được lặp đi lặp lại gây lãng phí thời gian của nhà phát triển. Do đó vấn đề cấp thiết là cần có một chatbot có khả năng tiếp thu câu hỏi từ người dùng và tìm kiếm những nội dung liên quan nhằm phản hồi thắc mắc của người dùng với độ chính xác cao nhất và tốc độ nhanh nhất

Nội dung tiến hành:

Mục tiêu chính của khóa luận này là tìm hiểu về hệ thống chatbot và những cách tiếp cận chatbot đang được áp dụng cũng như ứng dụng của chatbot trong đời sống thông qua việc xây dựng ứng dụng chatbot hỗ trợ trả lời câu hỏi liên quan tới hệ thống IQ-TREE.

## Bố cục khóa luận:

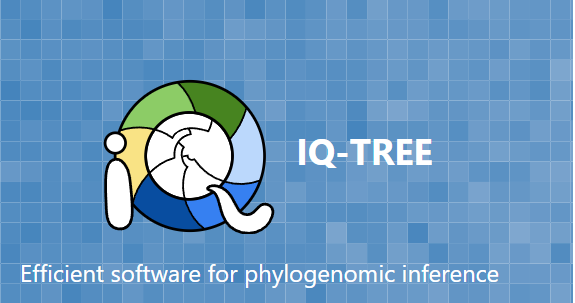
Khóa luận gồm 6 chương:

* Chương 1. Tổng quan: Chương này trình bày về hệ thống IQ-TREE, lý thuyết về chatbot, cấu trúc, phân loại chatbot và vấn đề xây dựng chatbot hỗ trợ hệ thống IQ-TREE.
* Chương 2. Cơ sở lý thuyết: Chương này trình bày về những lý thuyết liên quan sẽ được áp dụng trong chatbot:  Word embeddings, TF-IDF, BM25, Transformer, Phân lớp ý định, và một số thư viện, mô hình thường được sử dụng trong xử lý ngôn ngữ tự nhiên và xây dựng chatbot.
* Chương 3. Cách tiếp cận chung đối với chatbot: Chương này trình bày những phương pháp thường được áp dụng cho một chatbot dựa trên kiến trúc 3 phần của chatbot.
* Chương 4. Giải pháp Xây dựng chatbot IQ-TREE: Chương này trình bày việc áp dụng lý thuyết nêu trên vào thực tiễn xây dựng chatbot hỗ trợ hệ thống IQ-TREE.
* Chương 5. Hệ thống chatbot thử nghiệm: Chương này trình bày về kết quả thử nghiệm chatbot trên lý thuyết và thực tiễn.
* Chương 6. Kết luận: Chương này tóm tắt những kết quả đạt được, nêu lên những hạn chế và hướng phát triển trong tương lai.

# TỔNG QUAN

## Hệ thống xử lý tin sinh học IQ-TREE

IQ-TREE là một hệ thống xử lý tin sinh học được phát triển từ năm 2011 dưới dạng phần mềm mã nguồn mở theo giấy phép GNU-GPL. Được khai sinh và phát triển bởi nhóm nghiên cứu của giáo sư Bùi Quang Minh trường đại học quốc gia Australia. Nó giải quyết được một lượng lớn những bài toán thường gặp trong giới tin sinh như Partitioned analysis, Bootstrapping partition model, Tree topology tests, .... Trong thời gian gần đây, với động lực nghiên cứu và tìm kiếm giải pháp cho dịch bệnh Covid-19, hệ thống IQ-TREE ngày càng trở nên cần thiết và phổ biến hơn

. 

Hình 1.1 Hệ thống IQ-TREE (nguồn IQ-TREE homepage, http://iqtree.org/)

## Chatbot

### Trí tuệ nhân tạo

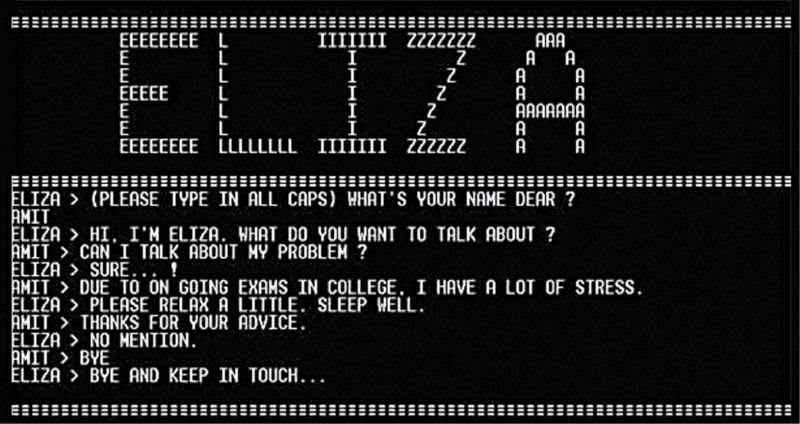
Trí tuệ nhân tạo hay trí thông minh nhân tạo (tiếng Anh: artificial intelligence hay machine intelligence, thường được viết tắt là AI) là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào.

Thuật ngữ này được bắt đầu đề xuất và nghiên cứu từ những năm 1950 dựa trên Thử nghiệm Turning (Turning test). Phép thử Turing là một bài kiểm tra để đánh giá liệu máy tính có thể bắt chước phản ứng của con người trong các điều kiện cụ thể. Phép thử được thực hiện với Một thiết bị do con người trực tiếp điều khiển đóng vai trò như kẻ thẩm vấn. thiết bị này thực hiện đối thoại với 2 kẻ bị thẩm vấn bằng ngôn ngữ tự nhiên. Trong đó có 1 máy tính được điều khiển bằng con người và 1 máy tính được thực hiện bằng thuật toán. Nhiệm vụ của kẻ thẩm vấn là xác định đâu là người thật, đâu là máy tính còn nhiệm vụ của 2 kẻ bị thẩm vấn là cùng cố gắng chứng tỏ mình là con người.

Năm 1951 chương trìnhtrí tuệ nhân tạo dựa trên lập trình đầu tiên được viết trên cỗ máy Ferranti Mark 1 tại đại học Manchester. Thay vì được viết phục vụ cho robot công nghiệp, chương trình này chơi cờ đam - một trò chơi chiến lược đối kháng giữa người và máy. Hai bên lần lượt di chuyển các quân giống hệt nhau theo đường chéo và bắt quân đối phương bằng cách nhảy qua quân đó.

Đến năm 1955, chương trình do Arthur Samuel cải thiện đã biết tự học chơi cờ. Thuật ngữ artificial intelligence (AI - trí tuệ nhân tạo) ra đời trong năm 1955.

Tới năm 1965, chatbot đầu tiên được phát minh có tên là Eliza. Là một chương trình của Joseph Weizenbaum ([Viện Công nghệ Massachusetts](https://vi.wikipedia.org/wiki/Vi%E1%BB%87n_C%C3%B4ng_ngh%E1%BB%87_Massachusetts), Mỹ), ELIZA đã đạt được những thành tích đáng kể và được coi là thành tựu đỉnh cao về trí thông minh nhân tạo vào thời điểm đó. Bằng cách nhận ra các từ và cụm từ chính từ truy vấn của người dùng và đưa những câu trả lời tương ứng bằng cách sử dụng các tập lệnh viết sẵn. Kịch bản mà nhà sản xuất đưa ra có tên là “DOCTOR”. Trong kịch bản này, ELIZA sẽ đóng vai trò như một nhà tâm lý học hay một bác sĩ tâm thần nhằm mục đích hỗ trợ điều trị bệnh nhân tâm thần phân liệt. Cũng bởi công bố này mà khái niệm chatbot được ra đời.



Hình 1.3 Chatbot Eliza (nguồn ELIZE – Acomputer Program for the Study of Natural Language Communication, https://medium.com//)

Sau đó năm 1974 là năm mà phương tiện giao thông tự động đầu tiên được tạo ra.

Đến năm 1997, Máy tính chơi cờ vua đã đánh bại nhà vô địch cờ vua thế giới Garry Kasparov. Máy tính đã dành chiến thắng có tên gọi Deep Blue do hàng IBM thiết kế và có khả năng xử lý 100 triệu phép tính trên giây. Trận đầu được tổ chức tại Philadelphia, Hoa kỳ. Trong đó, Deep Blue đã đánh thắng ván đầu tiên trong chuỗi 6 ván đấu. Nhà vô địch cờ vua người Nga Kasparov đã đánh thắng 3 trận và có 2 trận hòa. Mặc dù chung cuộc Deep Blue thua với tỷ số 1-3 nhưng nó vẫn là một dấu mốc mang tính biểu tượng trong ngành trí tuệ nhân tạo

Năm 2017, Ả Rập Xê út là quốc gia đầu tiên đã cấp quyền công dân cho một robot mang tên Sophia. Sophia là một robot mang hình dạng con người được lấy cảm hứng từ minh tinh người anh Audrey Hepburn được phát triển bởi công ty công nghệ Mỹ Hanson Robotic. Robot ngoài có khả năng giao tiếp tự nhiên với con người, nó còn có thể biểu hiện 62 sắc thái trên khuôn mặt. Năm 2021, Sophia còn bất ngờ đưa ra tuyên bố muốn có một đứa con và xây dựng gia đình.

### Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ - công cụ hoàn hảo nhất của tư duy và giao tiếp.

Một số ứng dụng tiêu biểu của ngành xử lý ngôn ngữ tự nhiên thường gặp như:

Nhận dạng tiếng nói: Nhận dạng tiếng nói rồi chuyển chúng thành văn bản tương ứng. Nhận dạng tiếng nói thường được áp dụng trong các chương trình điều khiển bằng tiếng nói với một dạng nâng cao là nhà thông mình, giúp thao tác của con người trên các thiết bị nhanh hơn và đơn giản hơn, chẳng hạn thay vì gõ một tài liệu nào đó bạn đọc nó lên và trình soạn thảo sẽ tự ghi nó ra. Đồng thời, nó được ứng dụng trong việc hỗ trợ tương tác người, máy, hỗ trợ người khiếm thị.

Tổng hợp tiếng nói: Là bài toán ngược lại của nhận dạng tiếng nói, thay vì chuyển tiếng nói thành ngôn ngữ mà máy có thể hiều được, tổng hợp tiếng nói là việc chuyển ngôn ngữ tự nhiên ở dạng văn bản hoặc ngôn ngữ máy thành tiếng nói. Đây thường là nhiệm vụ cuối cùng trong tương tác người, máy. Cũng với ứng dụng như nhận dạng tiếng nói, tổng hợp tiêng nói thường được phát triển nhằm hỗ trợ người khiếm thị.

Dịch tự động (machine translate): Như tên gọi đây là chương trình dịch tự động văn bản mà không cần có sự can thiệp của con người. Dịch tự động thường phả đối mặt với một thách thức lớn là việc ý nghĩa của một câu văn thường không thể chỉ được hiểu bởi sự kết hợp của cấu trúc, từ ngữ và ngữ pháp mà còn phải được hiểu theo văn cảnh. Trong khi nhiệm vụ dịch máy thường không chỉ phải đối diện với từng câu riêng lẻ mà thường là cả một văn bản dài. Đồng thời việc dịch máy không phải là việc xử lý ngôn ngữ tự nhiên theo một chiều mà ngoài việc đọc hiểu đoạn văn, nó còn phải bố trí văn bản sinh ra sao cho phù hợp. Trên thực tế dịch thuật là một công việc khá khó mà con người cũng không thể thực hiện chính xác 100% được.

Truy xuất thông tin (information retrieval): Nội dung chính của nhiệm vụ này là tìm kiếm thông tin trong các tài liệu không có cấu trúc chính quy (thường là văn bản). Tiêu biểu cho ứng dụng truy xuất thông tin là các công cụ tìm kiếm như Google search, Yahoo, Bing search, …

Tóm tắt văn bản: Công việc này là trích rút thông tin quan trọng từ một văn bản để tạo ra một phiên bản ngắn gọn hơn nhưng vẫn giữ được đầy đủ ý nghĩa cần thiết. Nó cùng đối diện với một vấn đề như trong dịch máy là tóm tắt văn bản cần phải sinh ngôn ngữ, do đó cần đảm bảo về ngữ pháp và văn phong. Tóm tắt văn bản thường được ứng dụng trong giám sát truyền thông, nghiên cứu tài chính, tổng hợp sách, tài liệu, ….

Việc xử lý ngôn ngữ tự nhiên hiện đại thường sử dụng học sâu nhằm giảm thời gian trích xuất thông tin và tăng độ chính xác, tuy nhiên phương pháp này đòi hỏi nhiều dữ liệu và tính chính xác của dữ liệu phải thực sự cao.

### Chatbot

Chatbot hay còn gọi là hệ thống trả lời tự động là một phần mềm ứng dụng được xây dựng trên nền tảng trí tuệ nhân tạo với nòng cốt chủ yếu là lĩnh vực xử lý ngôn ngữ tự nhiên nhằm thay thế con người trò chuyện và tương tác với con người. Chatbot thường được sử dụng phổ biến cho các tác vụ như chăm sóc khách hàng, quảng cáo sản phầm, trợ lý ảo, … ở Việt Nam, chatbot được phát triển mạnh mẽ trong thời gian gần đây dưới động lực là xây dựng chăm sóc khách hàng tự động cho các sàn thương mại điện tử, các quảng cáo trên mạng xã hội.

Lịch sử phát triển chatbot:

Ở Việt Nam, thuật ngữ chatbot mới được phổ biến gần đây tuy nhiên nó đã có lịch sử phát triển từ 50 năm về trước. Sau công bố về phép thử turing (1.2.1), chatbot đã lần đầu tiên được công nhận vào năm 1966 với tên ELIZA (1.2.1),

Năm 1972, Bác sĩ Kenneth Colby đã phát triển chatbot PARRY dựa trên ELIZA. Trong khi kịch bản của ELIZA được xây dựng để trở thành một bác sĩ thì PARRY lại đóng vai trò là một bệnh nhân tâm thần phân liệt hoang tưởng. Đây là chatbot đã thắng được thử nghiệm turning khi các bác sĩ tham gia thử nghiệm đã không thể phân biệt được sự khác biệt giữa người và máy.

Năm 1992: Dr. SBAITSO: Được tạo ra bởi Creative Labs vào đầu những năm 1990, Dr SBAITSO là từ viết tắt của Sound Blaster Artificial Intelligent Text to Speech Operator). Đây là chatbot đầu tiên hồi đáp người dùng bằng âm thanh thay vì chữ viết. Tuy nhiên nó chưa thể tạo ra các tương tác phức tạp mà thường hỏi người dùng với câu hỏi “bạn cảm thấy thế nào.” Và thường trả lời câu “Đó không phải là vấn đề của tôi” khi không thể hiều được yêu cầu từ người.

Năm 1995: Chatbot ALICE ra đời lần đầu tiên sử dụng một ngôn ngữ đánh dấu trí thông minh nhân tạo AIML (Artificial Intelligence Markup Language).

Khoảng thời gian từ năm 2010 đến năm 2016 là thời kỳ bùng nổ các trợ lý ảo. Trợ lý ảo là những chatbot thay vì tập trung vào những cuộc trò chuyện với con người, nó hiểu ngôn ngữ tự nhiên chủ yếu ở dạng giọng nói và thực hiện những tác vụ mà người dùng yêu cầu. Một sản phẩm tiêu biểu của thời kỳ này là trợ lý ảo cortana được tích hợp trên windows 10.

Từ năm 2016 đến nay là giai đoạn Chatbot Messenger bùng nổ. Với sự phổ biến của thiết bị di động và xu hướng mua sắm online, facebook đã trở thành 1 kênh quảng cáo tiện lợi và hiện đại mỗi doanh nghiệp đều có cho mình một trang cá nhân marketing riêng trên mạng xã hội này.

Chatbot có thể chia thành 2 loại dựa trên mục tiêu trả lời của chúng:

Loại miền mở (Open Domain): là mô hình trả lời tự động có thể trò chuyện về một lĩnh vực bất kỳ mà không được chỉ định cho bất kỳ 1 ý định nào trước đó từ các cuộc trò chuyện xã giao đến hỏi đáp về ý tế sức khỏe, khoa học, tư vấn bán hàng, … Do đó ưu điểm của loại chatbot này là có thể trò chuyện liên quan đến nhiều đề tài khác nhau, tuy nhiên nhược điểm của chúng là câu trả lời thường sơ sài, không có kiến thức chuyên sâu, đồng thời cần một lượng dữ liệu lớn, phong phú.

Loại chatbot miền đóng (Close Domain): là mô hình trả lời tự động tập trung cuộc trò chuyện vào một lĩnh vực cụ thể ví dụ như: Giáo dục, thể thao, mua sắm, thời tiết, … Những hệ thống này hướng tới mục tiêu trả lời câu hỏi với tính chính xác cao trong một lĩnh vực cụ thể thay cho việc trả lời với tính chính xác thấp như loại miền mở, do đó thường được áp dụng trong các vấn đề hỗ trợ kỹ thuật tư vấn mua hàng. Ưu điểm của loại này là câu trả lời mang tính cụ thể, tính tin cậy cao, tuy nhiên nhược điểm của nó là chỉ có thể trả lời trong 1 lĩnh vực nhất định.

Dựa trên cách thức xử lý của chatbot có thể chia làm 2 loại:

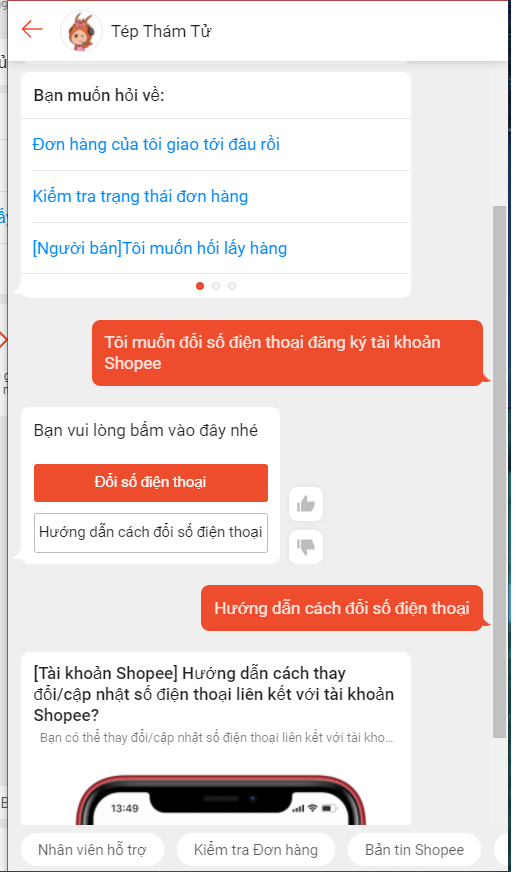
Chatbot theo kịch bản Chatbot theo kịch bản: Kiểu chatbot này thường được cài đặt như một cây quyết định, nó đưa cho người dùng các lựa chọn và khi người dùng chọn lựa chọn này, chatbot lại đưa cho người dùng những lựa chọn khác cho đến khi giải quyết được vấn đề. Chatbot này thường được tích hợp trên tin nhắn Facebook nhằm tránh việc người dùng phải đi đến những trang khác gây mất tự nhiên trong việc tương tác.

Chatbot nhận dạng từ khóa: Khác với các Chatbot kịch bản, các Chatbot dựa trên nhận dạng từ khóa có thể lắng nghe những câu nói của người dùng và trả lời một cách thích hợp. Những Chatbot sử dụng các từ khóa tùy biến và AI để xác định làm thế nào để đưa ra câu trả lời phù hợp nhất cho người dùng [3]. Người ta thường kết hợp chatbot kịch bản với chatbot nhận dạng từ khóa khi chatbot hiều yêu cầu của người dùng ở dạng ngôn ngữ tự nhiên và trả lời người dùng ở dạng kịch bản. Những dịch vụ hay sử dụng loại chatbot này là dịch vụ bán hàng, ship đồ ăn, ...vv.

1.2.3.4 Một số chatbot hiện được sử dụng ở Việt Nam

Chatbot bán hàng: Là công cụ hỗ trợ bán hàng hoạt động 24/7. Chatbot đóng vai trò thay thế một nhân viên bán hàng phải túc trực tại quầy để “chốt đơn”. Ưu điểm nổi bật là đơn giản, dễ sử dụng. Chatbot bán hàng không cần dùng các phần mềm xử lý ngôn ngữ tự nhiên mà thường chỉ được xây dụng cho các tương tác đơn giản để tương tác với khách hàng. Kịch bản trả lời cũng được xây dựng sẵn từ trước.

Chatbot chăm sóc khách hàng: Loại này thường được các trung tâm chăm sóc khách hàng lớn sử dụng để trả lời các câu hỏi thường gặp theo kịch bản hoặc dữ liệu có sẵn. Đối với những câu hỏi đơn giản, chatbot sẽ tự trả lời. Với các câu hỏi phức tạp hơn, chatbot sẽ chuyển đến nhân viên chăm sóc khách hàng để giải quyết. Trong quá trình hoạt động, chatbot sẽ tự học để đưa ra những câu trả lời chính xác và phù hợp với thực tế hơn. Một ví dụ tiêu biểu này là chatbot hỗ trợ sàn thương mại Shopee. Chatbot này áp dụng cả kỹ thuật chatbot theo kịch bản và chatbot nhận dạng từ khóa. Trong phần lớn trường hợp, chatbot không trả lời bằng ngôn ngữ tự nhiên mà đưa ra một số lựa chọn dựa trên việc nhận dạng từ khóa. Về bản chất, chatbot này khoanh vùng yêu cầu của người dùng vào 3 đến 5 lựa chọn nhằm hỗ trợ người dùng tìm kiếm phương thức thao tác trong một hệ thống lớn.



Hình 1.4 Chatbot shopee

## Phát biểu bài toán

Nhiệm vụ chính của chương trình là thiết kế mô hình tương tác giữa người với máy nhằm giải đáp những thắc mắc của người dùng khi sử dụng hệ thống IQ-TREE.Từ câu hỏi, thắc mắc của người dùng, hệ thống cần tìm ra những tài liệu liên quan hợp lý để trả lời người dùng một cách chính xác. Do IQ-TREE là một hệ thống mang tính đặc thù cao nên chatbot hỗ trợ hệ thống IQ-TREE cũng cần có những đặc điểm riêng biệt:

* Chatbot phát triển trên miền đóng, tập trung giải đáp thắc mắc liên quan đến hệ thống IQ-TREE, không được phát triển để giao tiếp xã giao, giải trí.
* Những câu hỏi mà hệ thống tập trung vào dựa trên những thắc mắc từ nhóm google của hệ thống IQ-TREE: <https://groups.google.com/g/iqtree?pli=1>.
* Câu trả lời được trích xuất từ tài liệu chính thức của nhóm phát triển IQ-TREE <http://www.iqtree.org/doc/>.
* Do phục vụ một hệ thống kỹ thuật, độ chính xác được ưu tiên hàng đầu trong việc đánh giá kết quả. thời gian phản hồi được ưu tiên sau. Do câu trả lời được trích xuất từ tài liệu chính thức của nhà phát triển nên không dùng sự mượt mà của câu trả lời làm thước đo đánh giá kết quả.

# CÁC CÔNG NGHỆ SỬ DỤNG

## Các kỹ thuật trong xử lý ngôn ngữ tự nhiên

### Vector hóa

Vector hóa hay nhúng từ (thuật ngữ tiếng anh word embedding) là phương pháp phổ biến hiện để giúp máy tính có thể hiểu được ngôn ngữ từ văn bản sang dạng số. Đó là việc ánh xạ một văn bản vào một không gian mới được gọi là không gian từ nhúng (embedding space). Không gian nhúng là một không gian vector dùng để biểu diễn dữ liệu có khả năng miêu tả được mối liên hệ, sự tương đồng về mặt ngữ nghĩa, văn cảnh của dữ liệu với đặc điểm những từ có cùng văn cảnh hoặc cùng ngữ nghĩa sẽ được ánh xạ vào hai vị trí gần nhau. 2 Cách tiếp cận chính thường thấy ở word embedding là Frequency base embedding và predictionbase embedding.

* Frequency base embedding là mô hình vector hóa dựa trên tần suất xuất hiện của từ trong đó phổ biến là TF-IDF, Co-occurrence matrix. Những mô hình này thường không đánh giá các từ về mặt nghĩa mà dựa trên lý thuyết những từ càng xuất hiện nhiều thì độ quan trọng của từ đó sẽ càng cao để xây dựng vector.
* Prediction base embedding là mô hình vector hóa dùng mạng nơ-ron với đại diện tiêu biểu là Word2Vec. Chúng là những mô hình học máy bán giám sát sử dụng mô hình skip-Gram hoặc CBOW là một mạng nơ-ron gồm 3 lớp: lớp đầu vào (input layer), lớp ẩn (hidden layer), lớp đầu ra (output layer). Việc vector hóa cận dựa trên ý tưởng một từ có thể được dự đoán bởi những từ xung quanh nó. Về cơ bản, Prediction base embedding là mô hình ánh xạ 1 từ thành 1 vector sao cho các từ gần nhau về ngữ nghĩa sẽ có khoảng cách về mặt vector như cosine, manhattan, ecluid gần nhau. Bởi vậy Prediction base embedding có thể phát hiện các từ đồng nghĩa hoặc gợi ý bổ sung các từ trong câu.

### TF-IDF và biến thể BM25:

TF-IDF là một kỹ thuật sử dụng trong khai phá dữ liệu văn bản nhằm đánh giá tầm quan trọng của 1 từ trong văn bản dựa trên tần suất xuất hiện của từ đó trong văn bản. Với ý tưởng một từ nếu xuất hiện càng nhiều lần trong 1 văn bản thì tính quan trọng càng cao, ngược lại 1 từ nếu xuất hiện trong càng nhiều văn bản thì tính quan trọng càng thấp. TF-IDF được đặt bằng tích của 2 chỉ số TF và IDF.

Trong đó, TF viết tắt của Term Frequency là tần suất xuất hiện của 1 từ trong văn bản. Nó được tính bằng số lần xuất hiện của từ trên tổng số từ của văn bản.

Với:

* + là giá trị TF của từ t trong văn bản d.
  + là số lần từ t xuấ hiện trong văn bản d.
  + là số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d.

IDF đặc trưng cho mức độ quan trọng của 1 từ, được tính bằng logarit của tổng số văn bản trên số văn bản có xuất hiện từ.

Với:

* + là giá trị IDF của từ t trong tập văn bản.
  + là số văn bản trong tập D.
  + là số văn bản trong tập D chứa từ t.

TF-IDF được tính bằng tích giữa TF và IDF. Nó chỉ ra những từ thường xuất hiện trong văn bản này nhưng ít xuất hiện trong những văn bản khác sẽ mang ý nghĩa lớn hơn.   
Biến thể TF-IDF trên BM25 Okapi là một biến thể của TF-IDF nhằm thực hiện sắp xếp thứ hạng văn bản dựa trên mức độ liên quan.

Vector hóa bằng TF-IDF:

Phương pháp TF-IDF cổ điển ánh xạ 1 chuỗi thành 1 vector bằng cách mỗi từ có xuất hiện trong toàn thể tập truy vấn và tập câu truy vấn sẽ tạo thành 1 chiều của vector. giá trị của chiều đó bằng giá trị TF-IDF của từ đó trong chuỗi, nếu từ đó không tồn tại thì TF-IDF được chọn bằng 0.

BM25 là một kỹ thuật dựa trên TF-IDF, tuy nhiên nó không tiến hành vector hóa mà tìm kiếm văn bản dựa trên từng từ của truy vấn xuất hiện trong tài liệu, không quan tâm đến mối quan hệ giữa các từ đó trong từng văn bản. Công thức đánh giá mức độ liên quan của một văn bản D trong một tập truy vấn Q (q1, q2, ...qn) sẽ là

trong đó:

là số lần mà từ xuất hiện trong văn bản D.

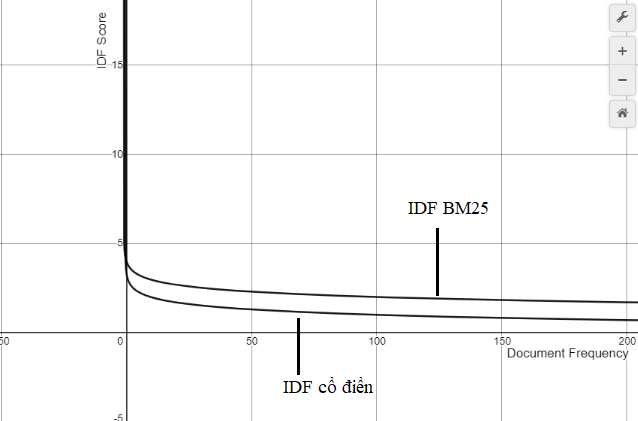
là số từ trong tất cả các tài liệu D.

là độ dài trung bình của văn bản trong tập Q

và là những hệ số được bổ sung thường được chọn bằng 1,2 hoặc 2,0. thường bằng 0,75

IDF được tính bằng

 Nhìn chung hình dạng đồ thị của TF-IDF cơ sở và BM25 khá giống nhau. Tuy nhiên, IDF trong BM25 có thể mang giá trị âm khi từ phổ biến, xuất hiện trong hơn một nửa số văn bản trong tập văn bản. Do đó giá trị IDF thường được quy định một ngưỡng thấp nhất espsilon. IDF chỉ được lấy khi nó lớn hơn giá trị này.



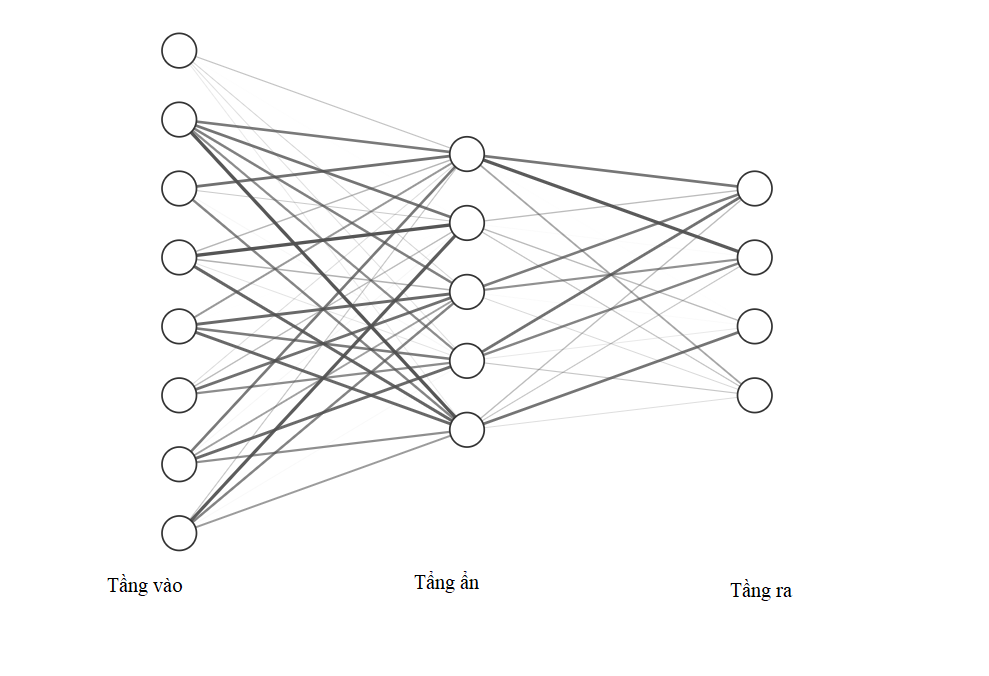
Hình 2.1 IDF với BM25 và IDF cổ điển

## Kiến trúc Transformer

Năm 2017 Google giới thiệu kiến trúc Transformer qua bài báo Attention is all you need là một kiến trúc mới dành cho các bài toán xử lý ngôn ngữ tự nhiên mà không có sự tham gia của mạng nơ-ron hồi quy hay mạng nơ-ron tích chập. Transfomer giải quyết được những vấn trước đó đối với RMM hay LSTM khi giúp quá trình huấn luyện nhanh hơn và quá trình huấn luyện cũng cho kết quả tốt hơn rất nhiều. Phần dưới đây trình bày quá trình phát triển từ mạng nơron nhân tạo tới kiến trúc Transformer.

### Mạng nơ-ron nhân tạo, mạng nơ-ron truyền thẳng

Lấy ý tưởng từ mạng noron tự nhiên cấu thành bộ não con người, mạng nơ-ron nhân tạo được tạo nên từ 1 lượng lớn các phần tử kết nối với nhau thông qua các liên kết) làm nên một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu, ...) thông qua một quá trình học từ tập các mẫu huấn luyện [2]. Việc học của một mạng nơ-ron chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơron với nhau. Mạng nơron truyền thẳng là một mạng nơron nhân tạo, trong đó các kết nối giữa các nút không tạo thành chu kỳ. Mô hình mạng nơron chuyển tiếp là hình thức đơn giản nhất của mạng nơron vì thông tin chỉ được xử lý theo một hướng.

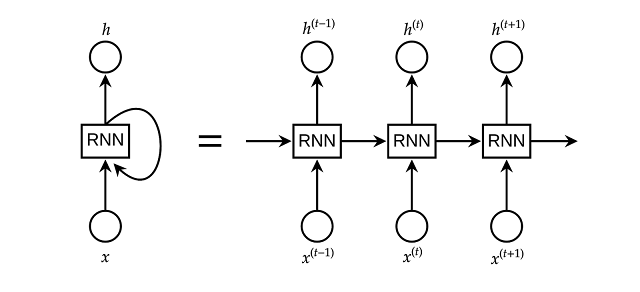


Hình 2.2 Mạng nơ-ron truyền thẳng

Có thể thấy các đối số đầu vào của 1 mạng nơron là hoàn toàn độc lập với nhau. Do đó mà nó hoàn toàn không phù hợp với các bài toán xử lý ngôn ngữ tự nhiên do các tù trong câu đầu vào thường có sự ảnh hưởng lẫn nhau và không có sự đồng đều về mức độ quan trọng. đồng thòi ý nghĩa của câu còn phụ thuộc vào vị trí của từ trong câu, phụ thuộc vào sự liên quan giữa các tù với nhau. Thay vào đó, mạng nơ-ron truyền thẳng phù hợp với những nhiệm vụ mang tính tuyến tính cao như phân lớp tuyến tính và xây dựng một số hàm số cơ bản.

### Mạng nơ-ron hồi quy và mạng nhớ dài hạn

Do những hạn chế của mạng nơron truyền thằng, người ta đã cho ra đời mạng nơron hồi quy (RNN). Một mạng nổn hồi quy là một mạng nơron chứa 1 vòng lặp bên trong nó.



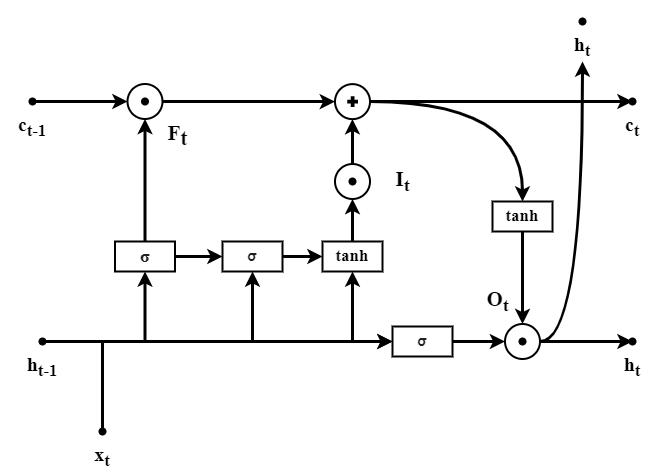
Hình 2.3 Mạng nơ-ron hồi quy

Trong hình trên là mạng neural hồi quy dạng vòng lặp và dạng duỗi thẳng. Ý tưởng của nó cho xử lý ngôn ngữ tự nhiên là tạo ra một vòng lặp mà từng từ được xử lý ở từng vòng lặp một cách tuần tự, mỗi từ được xử lý sẽ nhận thêm đầu vào là kết quả xử lý của từ trước và đồng thời sau khi được xử lý thành đầu ra ht sẽ được đưa trở lại vào mô hình để tham gia xử lý cho từ . Như vậy một mạng neural hồi quy là một chuỗi những mạng con giống hệt nhau, mỗi mạng sẽ truyền thông tin nó vừa xử lý cho mạng phía sau nó. Vì tính chất này, mạng neural hồi quy phù hợp cho những bài toán với dữ liệu đầu vào dưới dạng chuỗi với các sự kiện trong chuỗi có mối liên hệ với nhau. Vì vậy, mạng neural hồi quy có ứng dụng quan trọng trong các bài toán xử lý ngôn ngữ tự nhiên như: Dịch máy - Neural Machine Translation, Phân loại ngữ nghĩa - Semantic classification, Nhận dạng giọng nói: Speech Recognition.

RNN đã giải quyết được 1 vài hạn chế của mạng nơ-ron truyền thẳng. Tuy nhiên, có thể thấy rõ rằng với 2 từ liền kề nhau thì sự liên hệ giữa 2 từ với nhau được thể hiện rõ ràng bởi đầu ra của từ trước được truyền trực tiếp tới từ sau, nhưng khi 1 câu quá dài, thông tin từ những từ đầu tiên sẽ không thể đến được với những xử lý cuối cùng của mạng do trải qua nhiều vòng lặp, thông tin đã bị mất mát đi nhiều. Đặc biệt trong ngôn ngữ tự nhiên, một số từ thường được thay thế bằng các đại từ. Nếu một tù đọc được không được lưu giữ đủ thông tin mang sang câu khác thì đại tù nằm ở câu khác sẽ không có nghĩa.

Để khắc phục hạn chế của mạng RNN. người ta đã làm mềm mỏng hóa RNN và tạo nên mạng nhớ dài hạn. mạng nhớ dài hạn – thường được gọi là “LSTM”, là trường hợp đặc biệt của mạng RNN, có khả năng học với sự phụ thuộc lâu dài của các nơron. Mô hình này được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được cải tiến lại bởi Ayako Mikami (2016). Mục tiêu chính của LSTM là quyết định thông tin nào được lưu lại và loại bỏ tại mỗi nơron của RNN.

Trong Khi mạng RNN cứng nhắc mang toàn bộ trạng thái từ vòng lặp trước tới vòng lặp sau dẫn đến việc trạng thái của vòng lặp khi đi xa sẽ bị lu mờ quá múc, LSTM tăng cường thêm 1 đường truyền dữ liệu đi xa gọi là băng chuyền (C) đồng thời mềm mỏng việc truyền dữ liệu từ cell trước sang cell sau. Băng chuyền C làm được chức năng này là bởi nó chạy dọc xuyên suốt mạng trong khi ít có tương tác với các mắt xích. Băng chuyền của 1 vòng lặp được điều chỉnh bởi 3 cổng , :



Hình 2.4 Mạng nhớ dài hạn

* Cổng quên (): Cổng này là cổng tương tác đầu tiên tới băng chuyền, nó quyết định xem thông tin nào trong bộ nhớ hiện tại được giữ và thông tin nào bị bỏ lại. Đầu vào của cổng này chính là trạng thái của cell trước vào đầu vào của cell hiện tại, sau khi xử lý, kết quả của cổng này là kết quả của hàm sigmoid với một con số trong khoảng từ 0 đến 1 cho biết bao nhiêu phần của dữ liệu được truyền đi.
* Cổng vào (): Cổng này dùng để cập nhật bộ nhớ với các thông tin mới. Bên trong cổng này chứa một hàm sigmoid và một hàm tanh, trong đó hàm tanh có nhiệu vụ tạo ra vector mới nhằm ghi lại vào trạng thái của cell trong khi hàm sigmoid có tác dụng tương tự như với cổng quên tức là nó quyết định thông tin nào sẽ được nhớ lại.
* Cổng ra (): Kết quả từ cổng này chính là đầu ra của mỗi cell. Nó nhận đầu vào từ kết quả của cổng vào và đối tượng đang được xử lý và thông tin mà băng chuyền mang đến. Sau khi xử lý thông tin ở cổng này, kết quả được sao chép thành 2 phần, 1 phần là đầu ra cần thiết và một phần sẽ được đưa lại vào vòng lặp như đối với RNN.

### Kiến trúc Transformer

RNN là một kiến trúc tính toán tuần tự trong khi trái với CPU có các nhân mạnh mẽ, GPU lại có nhiều nhân nhưng yếu hơn. Việc RNN không thể tính toán song song không thể tận dụng hết sức mạnh của GPU. Giải quyết vấn đề không thể tính toán song song và không thể ghi nhớ dài lâu của RNN, Năm 2017, Google công bố bài báo “Attention is all you need” là một trong những bài báo có tầm ảnh hưởng nhất trong những năm gần đây, đặc biệt là trong lĩnh vực dịch máy. Bài báo đã đề xuất cơ chế Self-Attention mà sau này trở thành cốt lõi của Transformer - kiến trúc xử lý ngôn ngữ tự nhiên phổ biến nhất ngày này. Có thể tưởng tượng Self-Attention có thể được hiểu như một cơ chế tìm kiếm. Với một từ cho trước, cơ chế này sẽ cho phép mô hình tìm kiếm trong các từ còn lại để xác định từ nào liên quan để sau đó thông tin sẽ được mã hóa dựa trên tất cả các từ trên tạo thành một vector gọi là Self- Attention.

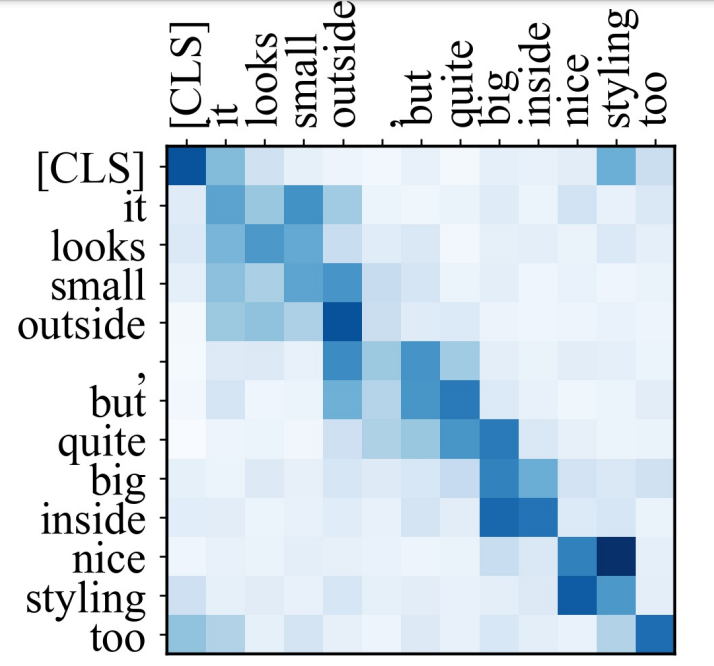
Self- Attention là kết quả của việc tính toán trên 3 vector query, key, value:

• query vector là vector dùng để chứa thông tin của từ được tìm kiếm, so sánh.

• key vector là vector dùng để biểu diễn thông tin các từ được so sánh với từ

cần tìm kiếm ở trên.

• value vector là vector biểu diễn nội dung, ý nghĩa của các từ.

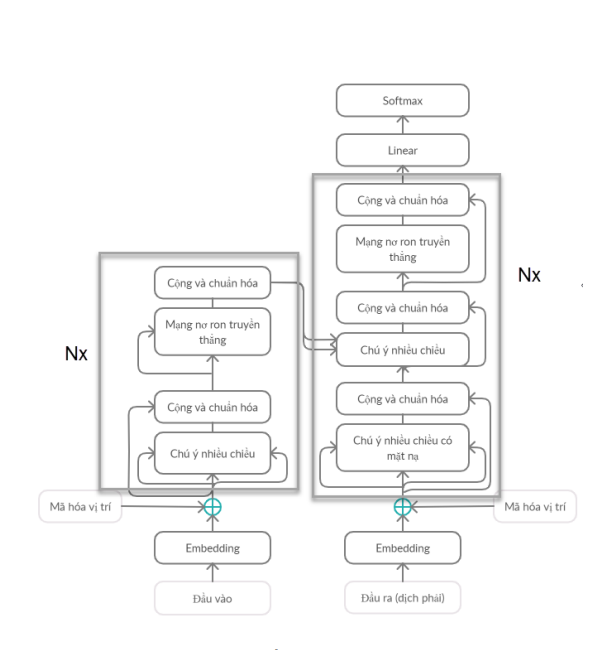


Hình 2.5 Trực quan hóa Self- Attention (nguồn Fine-tune BERT with Sparse Self- Attention Mechanism [9])

Bằng việc thay đổi cách tạo ra query vector, key vector và value vector, người ta sẽ thu được các Self- Attention khác nhau và khi lặp lại nhiều lần rồi ghép các Self- Attention lại với nhau, người ta thu được Multi-head Attention.

Về cấu trúc, Transformer là một mô hình có thể gồm 2 phần: bộ mã hóa (encoder) và bộ giải mã (decoder). Bộ mã hóa biểu diễn ngôn ngữ nguồn thành các vector, bộ giải mã sẽ nhận các vector biểu diễn này và dịch nó sang ngôn ngữ đích.

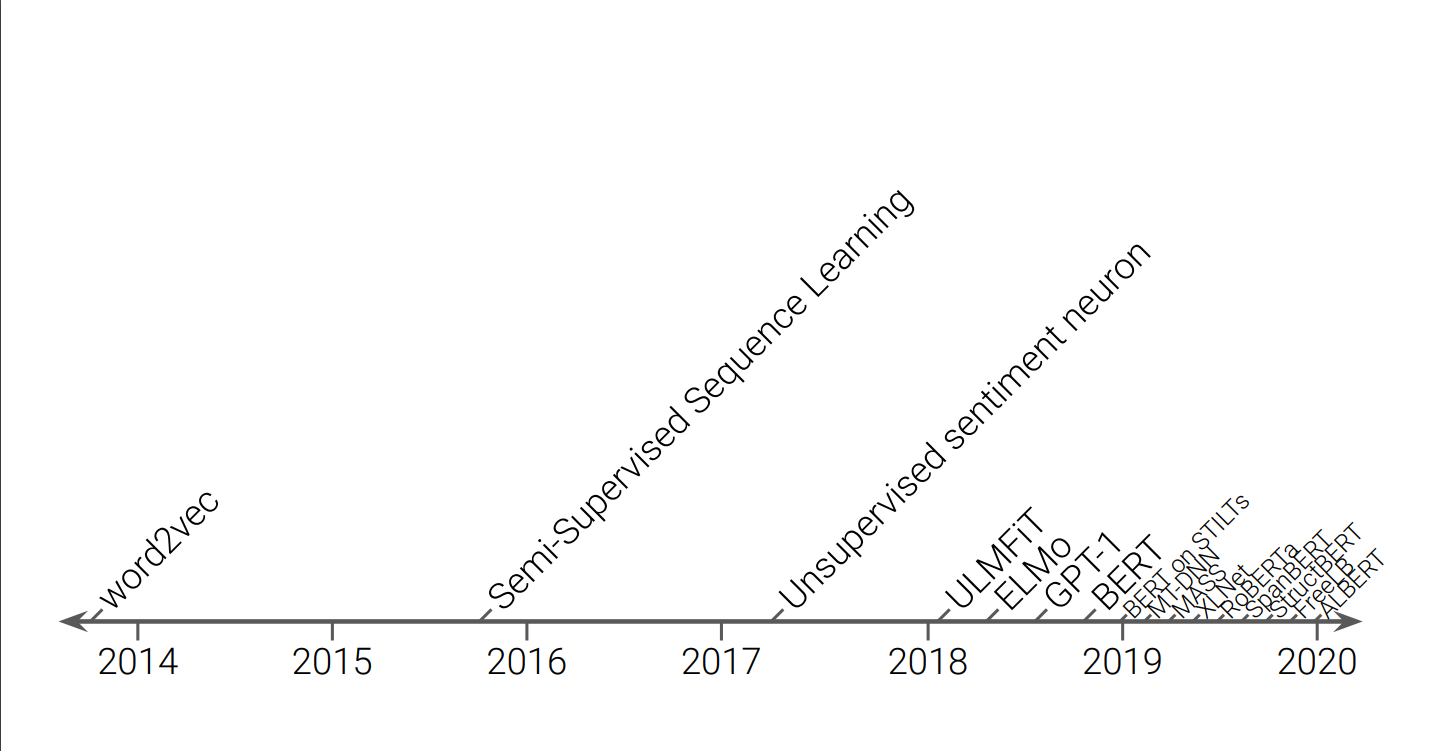
Bộ mã hóa của mô hình Transformer là một dạng mạng lan truyền (feed forward) với nhiều các tầng (encoder layer) chồng lên nhau. Thành phần chính của mỗi tầng là Multi-head Attention và feedforward network. Mỗi tầng sẽ xử lý đồng thời các từ và đưa cho tầng khác cho đến khi câu đi khỏi encoder. Encoder trong Transformer có thể được hiểu là thành phần đọc hiểu ngôn ngữ, nó biểu diễn ngôn ngữ tự nhiên dưới dạng một vector mà decoder có thể hiểu được. Trong khi đó decoder thực hiện chức năng giải mã câu nguồn thành câu đích. Kiến trúc của decoder khá giống với encoder, ngoại trừ có thêm một Multi-head Attention nằm ở giữa dùng để học mối liên quan giữa các từ trong câu giải mã với các từ trong câu nguồn.



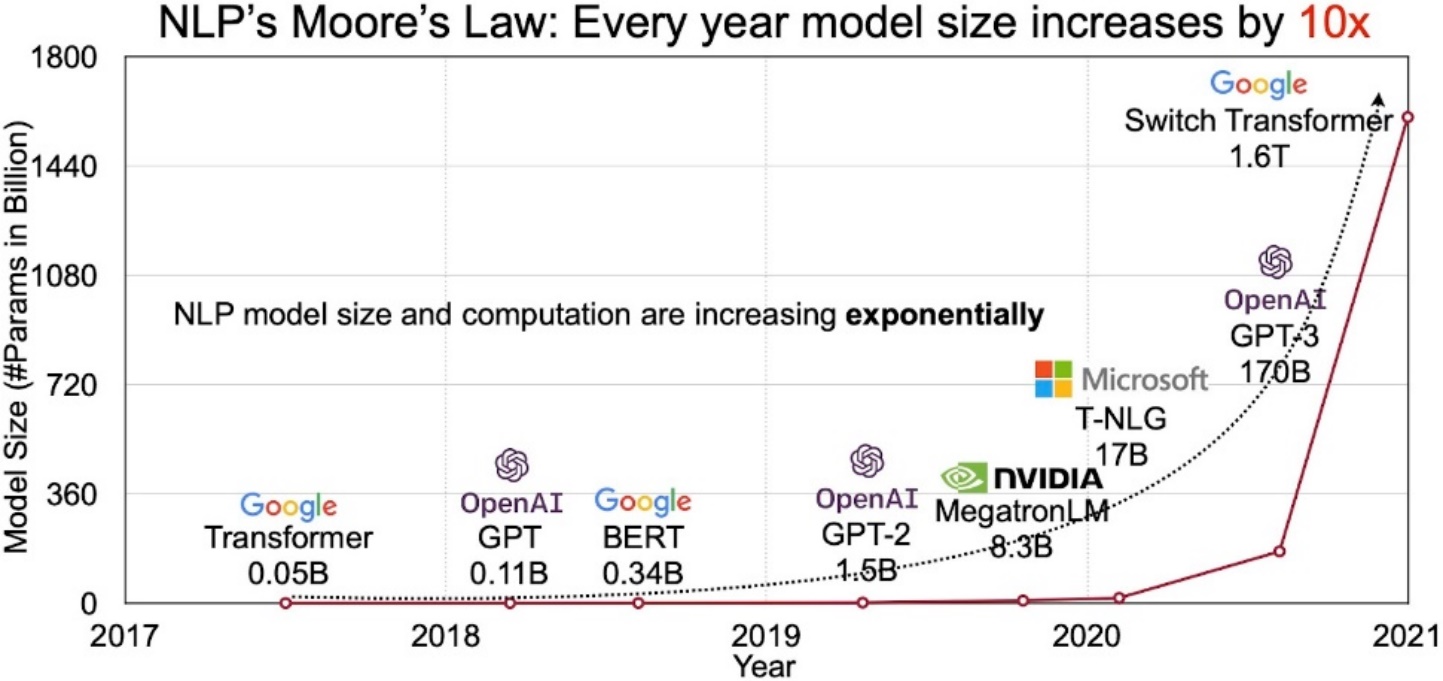
Hình 2.6 Kiến trúc mô hình Transformer (nguồn Nghiên cứu dịch máy Trung-Việt dựa vào mô hình Transformer [2])

Tầng cuối cùng của kiến trúc Transformer (Final Fully Connected Layer) thường là một mạng hồi quy tuyến tính nhằm chuyển kích thước vector từ decoder thành kích thước từ điển, sau đó hàm softmax có nhiệm vụ tính toán xác suất của từ xuất hiện của từng từ trong từ điển qua đó dự đoán từ tiếp theo sẽ xuất hiện.

Có thể thấy rõ được Transformer là một động lực mạnh mẽ đóng góp vào sự phát triển của ngành xử lý ngôn ngữ tự nhiên bởi ngay sau khi Transformer ra đời năm 2018, hàng loạt những mô hình xử lý ngôn ngữ tự nhiên ra đời và liên tục đạt SOTA trong những tác vụ thường gặp. Lượng tham số của những mô hình này cũng tăng liên tục và không ngừng với thống kê cho thấy lượng tham số được gấp 10 lần mỗi năm theo hàm số mũ.



Hình 2.7 Sự phát triển của các mô hình xử lý ngôn ngữ tự nhiên



Hình 2.8 Sự phát triển của kiến trúc Transformer về lượng tham số (nguồn Turing-NLG: A 17-billion-parameter language model by Microsoft, microsoft.com)

## Mạng nơ-ron kép

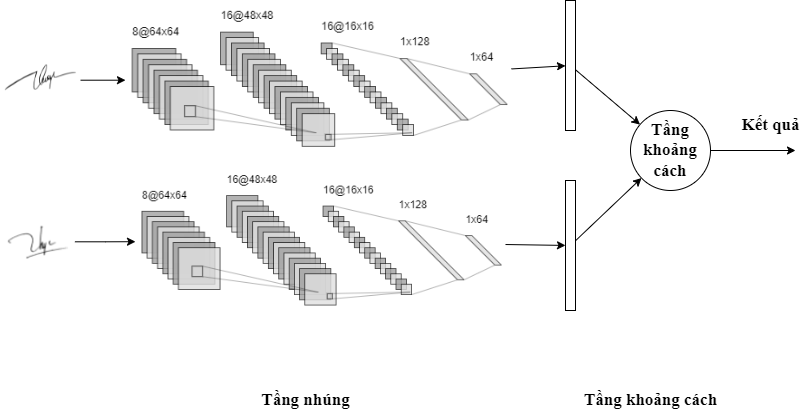
Mạng nơ-ron kép (SNN) là một mạng nơ-ron nhân tạo được tạo thành từ 2 mạng nơ-ron giống hệt nhau. Mạng này nhận đầu vào là 2 thực thể thuộc cùng một trường tính chất và đánh giá mức độ giống nhau của chúng dựa trên lý thuyết về việc so sánh 2 vector là mã hóa của 2 thực thể. Ứng dụng phổ biến của SNN là cho những bài toán nhận dạng và tìm kiếm với ít dữ liệu như nhận dạng chữ ký, nhận dạng khuôn mặt, tìm kiếm ảnh. Bởi vậy người ta gọi kỹ thuật này là One-shot learinng hay few-shot learning.

Mạng nơ-ron kép cho khả năng dự đoán chính xác cao với một lượng nhỏ dữ liệu bởi vì nó không cần trích xuất đặc trưng cho mẫu thử mà chỉ cần trích xuất tính giống và khác nhau giữa các mẫu thử. Do đó cấu trúc mạng có thể đơn giản. Dữ liệu để huấn luyện SNN chỉ cần trong khoảng 1 đến 5 mẫu thử cho một lớp. Cũng bởi việc trích xuất tính giống nhau giữa các mẫu mà mạng nơ-ron kép có tính mở rộng cao vì trái với một mạng nơ-ron thông thường, khi thêm một lớp vào mạng ta cần phải thay đổi cấu trúc mạng và huấn luyện lại mạng, trong khi đối với mạng SNN chỉ cần huấn luyện thêm mà không cần phải huấn luyện lại từ đầu.

Tuy nhiên SNN mang một số nhược điểm như thời gian huấn luyện sẽ lâu do việc huấn luyện SNN là huấn luyện từng cặp dữ liệu một và SNN không thể hiện sác xuất của mỗi lớp trong kết quả mà chỉ thể hiện sự giống nhau của 2 mẫu.

Khi huấn luyện SNN, Người ta sử dụng hàm mất mát Triple loss hoặc constrastive loss.

* Với triple loss là hàm mát mát với đầu vào là 3 vector A, P, N, A là vector cần dữ đoán, P là một vector gẵn nhãn cùng lớp A và N là vector được gắn nhãn khác với A. Trong đó A sẽ được tối ưu để có khoảng cách gần P nhất và xa N nhất.
* Constrative loss là hàm mất mát với chỉ 2 vector. Vector A là vector cần dữ đoán và 1 vector có thể là P hoặc N. Hàm mất mát này tối ưu A về gần vector kia hơn nếu nó là P và xa hơn nếu nó là N.



Hình 2.9 Mạng nơ-ron kép trong nhận dạng chữ ký với constrative loss

## Những mô hình và thư viện được sử dụng

### Beautifulsoup

Beautifulsoup là một thư viện python được phát triển cho phép khai phá dữ liệu html (web crawling) một cách nhanh và hiệu quả nhất. Nó cung cấp các cách để điều hướng, tìm kiếm và chỉnh sửa trong cây html, giúp lấy nội dung bạn mong muốn, xóa bớt những thông tin không cần thiết khi lấy HTML từ trang web nào đó.

### Pytorch

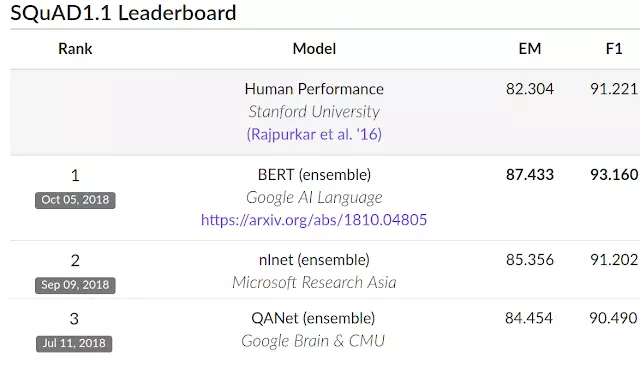
PyTorch là một framework được xây dựng dựa trên python cung cấp nền tảng tính toán khoa học phục vụ lĩnh vực Deep learning. Pytorch tập trung vào 2 khả năng chính: 1 là thay thế thư viện numpy để tận dụng sức mạnh tính toán của GPU, 2 là thực hiện những nhiệm vụ deep learning phục phụ nghiên cứu với sự linh hoạt và tốc độ cao. Ưu điểm của pytorch là Mang lại khả năng gỡ lỗi(debug) và trực quan hóa(visualize) dễ dàng hơn theo hướng tương tác, rất nhiều nhà nghiên cứu và kỹ sư đã dùng cả pytorch và tensorflow đều đánh giá cáo pytorch hơn trong vấn đề gỡ lỗi và trực quan hóa.

### Hugging Face

Nếu npmj là cộng đồng cung cấp hầu hết các thư viện thường dùng cho những ứng dụng dựa trên nodejs thì hugging face là cộng đồng cung cấp hầu hết những mô hình xử lý ngôn ngữ tự nhiên thời kỳ Transformer. Ngoài những mô hình được những tập đoàn công nghệ lớn côn bố như BERT, GPT, T5, RoBERTa, ... hugging face còn có một cộng đồng xử lý ngôn ngữ tự nhiên lớn, họ đem đến những mô hình được tinh chỉnh và những biến thể dựa trên những mô hình trên. Hugging Face đồng thời cung cấp thư viện python Transformer, nó đóng gói những nhiệm vụ trong xử lý ngôn ngữ tự nhiên từ những mô hình trên như FillMark Pipeline, Summarization Pipeline, Text Classification Pipeline, Text2Text Generation Pipeline, …. Hugging face còn đóng gói việc huấn luyện những mô hình xử lý ngôn ngữ tự nhiên giúp cách dùng của những mô hình khác nhau trở nên gần nhau hơn không đối với cả pytorch và tensorflow. làm cho việc huấn luyện trở nên dễ dàng hơn

### Mô hình BERT

BERT là viết tắt của Bidirectional Encoder Representations from Transformers. Được xây dựng trên kiến trúc Transformer với chỉ Encoder mà không có decoder, BERT tập trung vào những nhiệm vụ đọc hiểu ngôn ngữ tự nhiên như phân lớp văn bản, hỏi đáp, tóm tắt, ...Mô hình BERT có thể xử lý đa tác vụ, tuy nhiên với mỗi tác vụ ta cần thay đổi tầng cuối cùng của mô hình (fully\_connected) cho phù hợp với nhiệm vụ. Một phát triển của BERT so với những mô hình trước đó là nó học các vector đại diện theo ngữ cảnh 2 chiều (trái sang phải và phải sang trái) của từ. Trong phương pháp này, thay vì dùng những từ trước đó để dự đoán những từ sau như đối với GPT hoặc ELMo, Người huấn luyện BERT che bớt đi 15% từ trong chuỗi và yêu cầu nó dự đoán những từ bị che đi đó. Điều này làm BERT có khả năng đọc văn bản theo cả 2 chiều và mang lại hiệu quả cao hơn trong những tác vụ không phải là sinh văn bản hồi quy. Vào thời điểm mà BERT ra đời, nó đã đạt thành tích xuất sắc trong nhiều nhiệm vụ. Tiêu biểu là thành tích đứng đầu trong SQuaD V1.1.

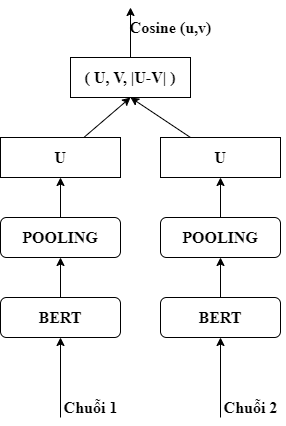


Hình 2.10 Kết quả SQuAD 1.1 năm 2019

Với khả năng đọc hiểu lớn, ngoài việc sử dụng như một mô hình deep learning, người ta thường lấy kết quả từ những tầng ẩn của BERT thay cho việc vector hóa.

### Mô hình SBERT

SBERT là một biến thể của mô hình BERT. Được sentence Transformer giới thiệu năm 2020 Nó dựa trên kiến trúc Transformer với chỉ encoder của BERT, tuy nhiên nó đã được tinh chỉnh để phù hợp hơn với mục đích vector hóa chuỗi thành vector. Đồng thời, SBERT được huấn luyện và áp dụng dựa trên kiến trúc của một mạng nơ-ron kép để những chuỗi tương tự nhau sẽ có khoảng cách cosine gần nhau hơn. Do sử dụng Transformer nên mô hình này tỏ ra hiệu quả hơn những phương pháp vector hóa trước đây như Word2vec và Glove do có khả năng đọc hiểu ngữ cảnh. SBERT có kiến trúc dựa trên một mạng nơron kép, được xây dựng để vector hóa các đoạn văn, do đã được huấn luyện trên tập dữ liệu lớn nên nên nó hầu như không cần huấn luyện lại mà có thể áp dụng được ngay. Ngoài ra, SBERT còn có thể được dùng cho một số bài toán như phân cụm, khai phá đoạn văn, tóm tắt đoạn văn, phát hiện trùng lặp, tìm kiến theo ngữ cảnh.



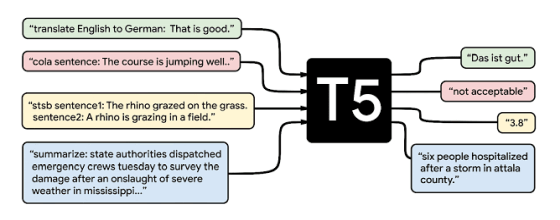
### Mô hình T5

T5 là một mô hình xử lý ngôn ngữ tự nhiên được Google công bố năm 2020. Mô hình T5 được Google thiết kế dựa trên kiến trúc Transformer với đầy đủ encoder và decoder. Mô hình T5 cũng được huấn luyện theo phương pháp dự đoán từ bị che khuất như BERT nên cũng có khả năng đọc hiểu văn bản 2 chiều. T5 đặc biệt hơn so với những mô hình Transformer trước đây bởi những đặc điểm sau.

 T5 là một mô hình xử lý đa tác vụ: Dù được thiết kế chủ yếu cho những nhiệm vụ sinh ngôn ngữ tự nhiên như dịch máy, tóm tắt văn bản. T5 cũng đồng thời được huấn luyện và đạt được nhiều kết quả xuất sắc trong những tác vụ xử lý ngôn ngữ của GLUE như Cola, SST-2, STSB, MNLI, …

Do có đầy đủ encoder và decoder, T5 chủ yếu được thiết kế cho nhiệm vụ dự đoán ngôn ngữ tự nhiên. T5 xử lý tất cả những tác vụ đều dựa trên cơ chế Sequence-to-Sequence tức là xử lý dữ liệu đầu vào và đầu ra đều ở định dạng văn bản.  Bằng việc thêm một từ chỉ định nhiệm vụ mà mô hình cần thực hiện vào đầu của chuỗi đầu vào. T5 có thể hiểu được nhiệm vụ mà nó cần thực hiện và đưa ra kết quả tương ứng. Kết quả mà mô hình xuất ra cũng là một chuỗi phù hợp với nhiệm vụ mà nó thực hiện. Ví dụ khi sử dụng cho nhiệm vụ dịch máy Anh-Pháp, đầu vào được thêm vào tiền tố translate English to French. kết quả sẽ là câu tiếng Pháp. Khi sử dụng cho nhiệm vụ đánh giá ngữ pháp tiếng Anh, tác vụ được chỉ định là cola sentence. Thay vì true hoặc false mô hình T5 trả về chuỗi unacceptable hoặc acceptable.

Mô hình T5 được google cung cấp 5 lựa chọn với 5 kích thước từ 60 triệu tham số đến 11 tỷ tham số T5-small, T5-base, T5-large, T5-3b, T5-11b. Được huấn luyện với bộ dữ liệu đồ sộ khai phá từ phần lớn website C4, hầu hết những mô hình của t5 đã đạt được nhiều thành tích xuất sắc đứng đầu (SOTA) đối với những tác vụ xử lý ngôn ngữ tự nhiên của GLUE



Hình 2.11 Đa tác vụ với mô hình T5 (nguồn Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [8])

2.3.6 Reactjs:

Reactjs là một thư viện javascript đang nổi lên trong những năm gần đây dưới nhu cầu của xu hướng Single Page Application. Được Facebook thiết kế riêng cho việc xây dựng giao diện, reactjs có thể xây dựng lên giao diện dựa trên những thành phần có tính tương tác cao và có tính tái sử dụng lớn. Reactjs trở nên phát triển bởi 2 ưu điển chính là:

* ReactJS giúp cho việc viết các đoạn code Javascript sẽ trở nên dễ dàng hơn vì nó sử dụng một cú pháp đặc biệt đó chính là cú pháp JSX. Thông qua JSX cho phép nhúng code HTML và Javascript.
* Cho phép người lập trình cắt nhỏ những giao diện phức tạp thành nhứng thành phần độc lập (component) giúp giảm độ phức tạp của giao diện và tăng tính tái sử dụng của mã nguồn.

# CÁCH TIẾP CẬN CHUNG CHO CHATBOT

Một chatbot thường được cấu tạo từ 3 thành phần chính:

* Thành phần nhiều ngôn ngữ tự nhiên (NLU): Có nhiệm vụ xác định ý định của người dùng và trích chọn thông tin.
* Thành phần quản lý hội thoại (DM): Thực hiện nhiệm vụ quản lý ngữ cảnh của cuộc hội thoại, từ đó xác định hành động tiếp theo dựa trên hành động trước đó. Đồng thời thành phần này chịu trách nhiệm truy xuất dữ liệu từ hệ thống hay thực hiện tác vụ từ người dùng.
* Thành phần sinh ngôn ngữ (NLG): Đảm nhận việc sinh ra câu trả lời phản hồi người dùng.  Câu trả lời thường được sinh ra từ những mẫu sẵn có và những luật được đặt trước

## Hiểu ngôn ngữ tự nhiên (NLU)

NLU có thể được coi như là thành phần quan trọng nhất của chatbot bởi chatbot có thông minh hay không thì đây là thành phần quyết định. Chatbot thông thường thường xây dựng NLU cho nhiệm vụ trích xuất thông tin từ yêu cầu của người dung thông qua việc trích xuất 2 thành phần chính là phân loại ý định và nhận diện thực thể. Mỗi cặp ý định và thực thể sẽ cho ra một ngữ nghĩa cụ thể. Ví dụ như người dùng yêu cầu “Bằng cách nào tôi có thể thực hiện phân tích bootstrap?”. Ý định trong câu hỏi là người dùng muốn cần hướng dẫn sử dụng và thực thể là “phân tích bootstrap”. Từ đó mà chatbot có thể xác định được câu trả lời cần là một hướng dẫn sử dụng và hướng dẫn này nói về phân tích bootstrap. Tuy nhiên trong một số trường hợp tùy thuộc vào định nghĩa của nhà phát triển, có thể coi cả thực thể và ý định là 1 ý định như ví dụ trên có thể coi ý định của người dùng là tìm hướng dẫn phân tích bootstrap.

Với nhiệm vụ này, Các chatbot thường được định nghĩa trước một danh sách các ý định mà người dung có thể yêu cầu, Một mô hình phân lớp được xây dựng bằng cách dự đoán xác suất xảy ra các ý định này. Việc phân lớp ý định sau đó được thực hiện bằng cách chọn ra ý định có xác xuất xảy ra cao nhất.Việc dự đoán xác xuất thường được tiếp cận qua 2 cách chính là phân lớp dựa trên mô hình học máy (thường sử dụng phương pháp Naive Bayes, Cây quyết định, SVM, mạng nơron) và phân lớp dựa trên khoảng cách số học (thường sử dụng khoảng cách euclid hoặc khoảng cách cosine).  
Phương pháp dựa trên mô hình học máy được thực hiện bằng cách xây dựng một máy học nhận đầu vào là vector các đặc trưng của yêu cầu từ người dung và đầu ra là một vector với số chiều bằng số ý định. Mỗi trọng số ở đầu ra của mô hình sẽ là sác xuất xảy ra ý định tương ứng. Ưu điểm của phương pháp này là hiệu năng cao do chỉ sử dụng 1 lần dự đoán, mạng dự đoán lưu trữ đặc điểm chính xác của mẫu dự đoán. Nhược điểm của phương pháp này là do cần trích xuất được đặc điểm chính xác của mẫu dự đoán nên độ phức tạp cao, cần nhiều dữ liệu.

Phương pháp phân lớp dựa trên so sánh mẫu là phương pháp so sánh mẫu cần dữ đoán với tập mẫu có gắn nhãn rồi dựa vào sự tương quan giống và khác nhau mà kết luận về mẫu dự đoán. Phương pháp này thường được thực hiện bằng cách ánh xạ truy vấn của người dùng vào 1 không gian vector qua đó so sánh 2 mẫu qua khoảng cách cosine hoặc ecluid, hoặc có thể xây dựng một mô hình với đầu ra là một số thực cho biết khoảng cách của 2 mẫu. Ưu điểm của phương pháp này là do không trích xuất được đặc điểm của mẫu nên không yêu cầu độ phức tạp cao nhưng vẫn cho độ chính xác cao. Nhược điểm của nó là cần nhiều dự đoán cho 1 mẫu, không trích xuất được đặc điểm của mẫu.

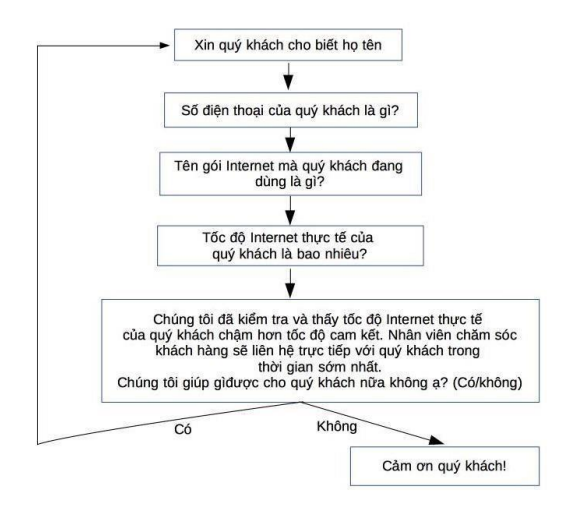
Hầu hết các Chatbot hiện tại đều áp dụng deep learning vào phần lớp ý định.

## Quản lý hội thoại

Trong các cuộc trò chuyện dài, Chatbot sẽ cần phải ghi nhớ được thông tin về ngữ cảnh hoặc quản lý các trạng thái hội thoại. Việc quản lý hội thoại sẽ giúp cho việc giao tiếp giữa người dùng và chatbot được liên tục và thông suốt. Việc này đặc biệt quan trọng trong trường hợp chatbot có câu hỏi cho người dùng. Khi đó, câu trả lời của người dùng có thể không có nghĩa khi đứng độc lập mà để hiểu câu trả lời của người dùng, chatbot cần liên hệ câu trả lời này với những phần hội thoại trước đó. Những mô hình Chatbot hiện nay thường dùng máy trạng thái hữu hạn (FSA) và mô hình Frame-base để xây dựng thành phần quản lý hội thoại.

### Mô hình máy trạng thái hữu hạn

Mô hình FSA là mô hình quản lý hội thoại cơ bản nhất. Trong mô hình này, chatbot định hướng người dùng bằng cách đưa ra các câu hỏi để người dùng nhập câu trả lời, cứ sau mỗi câu hỏi và trả lời thành công, máy chuyển đến trạng thái tiếp theo. Ưu điểm của mô hình FSA là nó rất đơn giản vì những câu hỏi được chatbot đưa ra một cách dự tính trước và có trình tự sẵn. Tuy nhiên nhược điểm của nó là cứng nhắc, không thể cùng một lúc chuyển 2 trạng thái. Hình dưới đây ví dụ về mô hình FSA phục vụ khách hàng phản ánh về chất lượng internet.



Hình 3.1 Mô hình FSA cho phản ánh chất lượng internet

### Mô hình Frame-base

Mô hình Frame-base giải quyết những hạn chế của mô hình FSA.Với mô hình này, chatbot được xây dựng một khung câu hỏi sẵn có và bảng kết quả tương ứng. Dù chatbot không hỏi nhưng một khi người dùng đã cung cấp thông tin thì kế quả sẽ được ghi lại vào khung và mô hình sẽ kết thúc khi toàn bộ khung được điền đẩy đủ. Trong những chatbot phức tạp có thể có nhiều khung khác nhau.

Bảng 3.1 Minh họa mô hình Frame-base

|  |  |  |
| --- | --- | --- |
| Nội dung | Câu hỏi | Trả lời |
| Họ tên | Xin quý khác vui lòng cho biết họ tên | ? |
| Số điện thoại | Số điện thoại của quý khách là gì ạ? | ? |
| Tên gói internet | Tên gói Intenet mà quý khách đang dùng là gì ạ | ? |
| Tốc dộ intenet | Tốc độ thực tết của quý khách là bao nhiêu? | ? |

## Mô hình sinh ngôn ngữ (NLG)

NLG là thành phần phụ trách việc sinh ra câu trả lời cho chatbot. Nó dựa vào việc ánh xạ hành động của quản lý hội thoại vào ngôn ngữ tự nhiên để trả lời người dùng. Có 2 phương pháp ánh xạ chính thường được dùng là dựa trên mẫu và sử dụng trí tuệ nhân tạo.

### Sinh ngôn ngữ dựa trên khuôn mẫu

Phương pháp này ánh xạ câu trả lời dựa trên những mẫu câu sẵn có. Phương pháp này thường được ứng dụng trên các chatbot dạng truy vấn ví dụ như tra cứu tài khoản, tra cứu đơn hàng, khi người dùng yêu cầu tìm kiếm một thứ gì đó, chatbot sẽ tìm trong cơ sở dữ liệu và ghép kết quả tìm được với một mẫu sẵn có để trả lời người dùng.

Ví dụ:

Bảng 3.2 Minh họa sinh ngôn ngữ dựa trên luật sinh mẫu

|  |  |
| --- | --- |
| Hành vi | Luật sinh |
| Confirm () | “Please tell me more about the product you are looking for” |
| Confirm (area = $V) | “Do you want somewhere in the $V” |
| Confirm (food = $V) | “Do you want a $V restaurant” |
| Confirm (food = $V, area = $W) | “Do you want a $V restaurant in the $W” |

Ưu điểm của phương pháp này là đơn giản, tính chính xác cao, dễ dàng kiểm soát. Nhược điểm của phương pháp này là cần định nghĩa các luật thủ công, câu trả lời có thể thiếu tự nhiên, khó có thể phát triển trên hệ thống lớn.

### Sinh ngôn ngữ dựa trên trí tuệ nhân tạo

Người thiết kế huấn luyện các mô hình máy học nhằm sinh văn bản dựa trên từ khóa, chủ đề hoặc từ những câu trước đó. Nó thường được ứng dụng cho những câu hỏi phức tạp câu trả lời khó được định trước ví dụ như sinh mã nguồn. Gần đây với thành công của Transformer mà việc sinh ngôn ngữ ứng dụng trí tuệ nhân tạo cũng đạt được nhiều thành công. Tiêu biểu với việc sinh mã nguồn ứng dụng mô hình GPT-3.

Ưu điểm của nó là câu sinh ra mượt mà, không khô cứng nhàm chán Nhược điểm của nó là khó kiểm soát.

Trên thực tế việc sinh ngôn ngữ bằng những mô hình trí tuệ nhân tạo còn nhiều khó khăn do đòi hỏi nhiều về tài nguyên. Sinh ngôn ngữ ràng buộc (Common Gen) hay Conditional Gen là 2 tác vụ sinh ngôn ngữ cốt lõi trong xử lý ngôn ngữ tự nhiên. Theo bài báo Common Gen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning của nhóm tác giả tại đại học công nghệ máy tính Washington, “Gần đây, những mô hình ngôn ngữ được đào tạo sẵn (pre-trained) đã chứng minh hiệu suất ấn tượng trên tập dữ liệu benchmark. Tuy nhiên nhiệm vụ sinh ngôn ngữ ràng buộc vẫn luôn là một thách thức lớn.”  bài báo cho biết hiệu suất của việc sinh ngôn ngữ ràng buộc chỉ đạt 31,6% so với con người 63,5%. Hơn nữa những nhiệm vụ sinh ngôn ngữ đòi hỏi lượng tài nguyên lớn cả về mặt dữ liệu cũng như phần cứng. Trên thực tế, tài nguyên mà máy chủ huấn luyện học máy phổ biến Google Colab giới hạn tài nguyên chỉ đủ cho một mô hình cỡ GPT-2(1,5 tỷ tham số) hay T5-large huấn luyện hiệu quả trong khoảng 0,5 giờ đồng hồ

# GIẢI PHÁP XÂY DỰNG CHATBOT CHO PHẦN MỀM IQ-TREE

Ứng dụng có tính năng như một chatbot cơ bản, được cài đặt cho nhiệm vụ hỏi đáp. Về cơ bản, người dùng có thể giao tiếp với chatbot thông qua hộp thoại bằng văn bản. Với mỗi truy vấn của người dùng, hệ thống sẽ tìm ra lời giải thích hợp lý và đưa ra cho người dùng. Ngoài ra, hệ thống sẽ có những đặc điểm riêng như đã trình bày ở mục 1.3.

Bởi những đặc điểm đó mà khóa luận này chọn cách tiếp cận chatbot theo hướng hiểu ngôn ngữ tự nhiên tiếp đến truy vấn dữ liệu trong cơ sở dữ liệu và đưa ra câu trả lời tương ứng. Phần tới đây trình bày chi tiết pha thiết kế và phân tích từng cách tiếp cận cho từng thành phần của hệ thống Chatbot.

## Dữ liệu và khai phá dữ liệu

Dữ liệu cho bài toán Xây dựng chatbot cho IQ-TREE được lấy từ 2 nguồn.

Nguồn thứ nhất là dữ liệu hỏi và đáp thực tế giữa đội ngũ phát triển và người dùng tại google group <https://groups.google.com/g/iqtree?pli=1>. Dữ liệu gồm khoảng 1200 đoạn đối thoại hỏi và đáp về cách sử dụng chương trình cũng như về lỗi, về phương pháp cải thiện hoạt động. Tuy nhiên dữ liệu này hoàn toàn chưa được phân lớp và gắn nhãn rõ ràng nên chưa thể dùng làm dữ liệu huấn luyện chương trình mà thay vào đó được dùng để làm dữ liệu kiểm thử đánh giá chương trình.

Nguồn thứ hai là tài liệu chính thức của nhà phát triển. <http://www.iqtree.org/doc/>. Dữ liệu này đã được chia sẵn thành 16 cụm tương đương với 16 chủ đề đáng quan tâm khi sử dụng hệ thống. Mỗi cụm được chia thành từng vấn đề con (từ đây gọi là nhãn) một cách rõ ràng. Nhìn chung dữ liệu đã được phân cụm và phân nhãn theo mô hình cây phân cấp, tuy nhiên lượng dữ liệu thu được còn rất ít, có khả năng không đáp ứng được câu hỏi về mặt nội dung và cách hành văn. Đồng thời dữ liệu thu được không mang tính chất là một tài liệu trình bày, không phải là tập câu hỏi. Dữ liệu này được đem làm dữ liệu tham chiếu, huấn luyện mô hình đồng thời vì có hình thức chính quy, nội dung nhất quán nên được lấy làm câu trả lời.

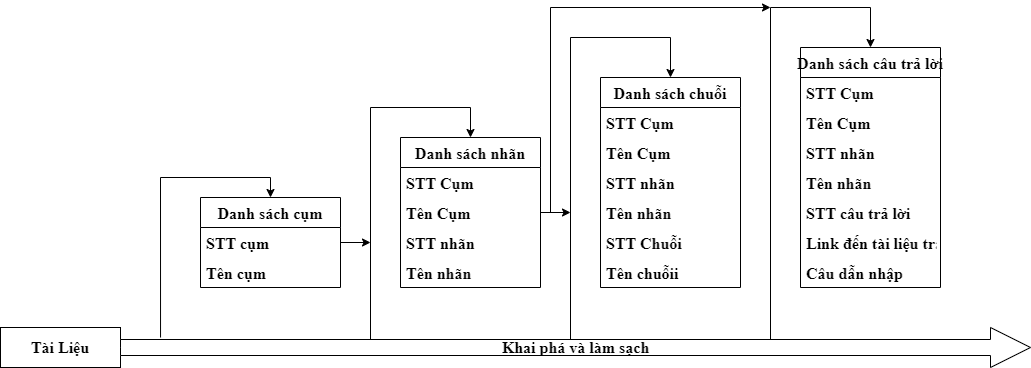
Dữ liệu được khai phá và làm sạch, loại bỏ những phần code, thẻ thừa nhờ thư viện Beautifulsoup đánh số cụm, đánh số nhãn, đánh số câu và lưu dưới dạng csv với các bảng sau.

* Bảng danh sách các cụm với số thứ tự cụm, tên cụm
* Bảng danh sách các nhãn với số thứ tự cụm, tên cụm, số thứ tự nhãn, tên nhãn
* Bảng danh sách các chuỗi với số thứ tự cụm, tên cụm, số thứ tự nhãn, tên nhãn, số thứ tự chuỗi, nội dung chuỗi.

Với chuỗi là nội dung dạng văn bản của một thẻ html là thẻ con của thẻ nhãn, hoặc là từng hàng của danh sách nếu thẻ đó là 1 danh sách.

* Bảng danh sách câu trả lời với số thứ tự cụm, tên cụm, số thứ tự nhãn, tên nhãn, số thứ tự câu trả lời, link đến tài liệu trả lời, câu dẫn nhập.

Với mỗi câu trả lời ứng với mỗi nhãn, câu dẫn nhập là câu chủ đề của nhãn tương ứng với thẻ <p> đầu tiên của nhãn.



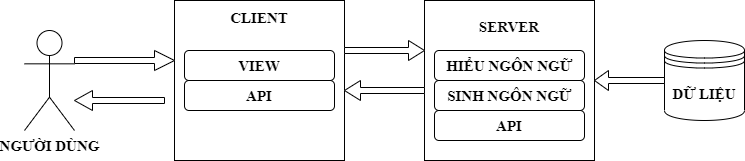
Hình 4.1 Khai phá dữ liệu

Bộ dữ liệu thu được gồm có: 16 cụm, 113 nhãn, 923 chuỗi và 113 câu trả lời.

Sau đó, dữ liệu được tách lấy một vùng gồm có 2 cụm, 21 nhãn với 209 chuỗi. Phần dữ liệu này được dùng cho các thử nghiệm tới đây nhằm giảm thời gian thử nghiêm.

Dữ liệu sau đó được đưa vào huấn luyện mô hình tùy theo từng cách tiếp cận sẽ được trình bày trong phần sau.

## Chi tiết hệ thống



Hình 4.2 Kiến trúc mô hình

Hệ thống được thiết kế dựa trên mô hình Client-Server. Phần dưới đây trình bày kiến trúc hệ thống từ khái quát đến chi tiết.

### Server

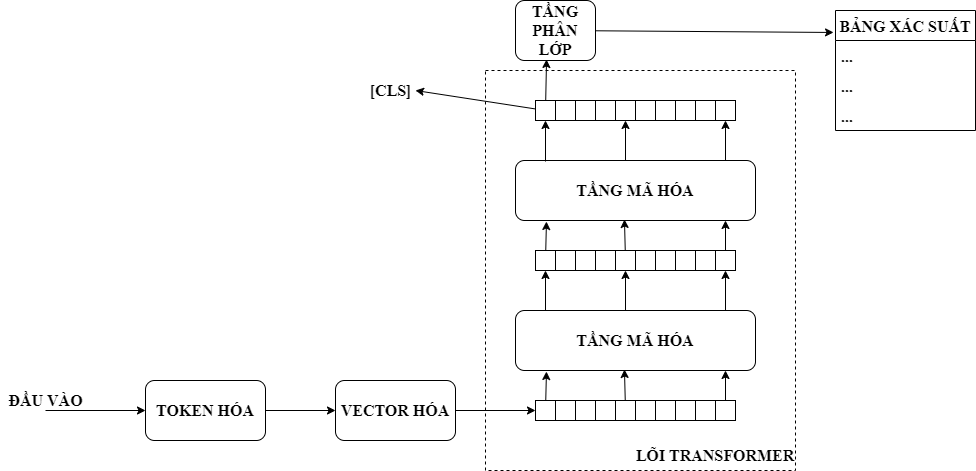
Server là lõi của chatbot, nó thực hiện hầu hết nhiệm vụ bên trong chatbot. Kiến trúc của 1 server chatbot thường gồm 3 thành phần: thành phần hiểu ngôn ngữ (NLU), thành phần quản lý hội thoại (DM), thành phần sinh ngôn ngữ (NLG) như đã trình bày ở chương 3. Tuy nhiên hệ thống chatbot hỗ trợ hệ thống IQ-TREE tối giản phần quản lý hội thoại và thành phần sinh ngôn ngữ mà tập trung xây dựng phần hiểu ngôn ngữ.

**4.2.1.1 Thành phần nhiều ngôn ngữ**

Được thiết kế để giải đáp những vấn đề mang tính kỹ thuật, tính chính xác của hệ thống chatbot được ưu tiên hàng đầu. Dựa vào việc quan sát các câu hỏi thực tế tại google group, khác với những chatbot thông thường (lấy chatbot bán hàng làm ví dụ) câu hỏi thường mang một ý định mà người phát triển có thể lường trước (ví dụ như hỏi giá tiền sản phầm, hỏi xem sản phẩm còn hay không, …) Câu hỏi mà người dùng đặt trên google group thường mang ý định khó có thể tính trước và lượng ý định nếu có thể tính trước cũng rất lớn. Khóa luận này chọn phương pháp coi cả ý định và thực thể là một thực thể và bỏ đi việc nhận diện thực thể (3.1.1). Tập ý định được xây dựng bằng cách ánh xạ một nhãn khai phá được thành một ý đinh. Thành phần NLU của chatbot sẽ hoàn toàn tập trung vào việc phân lớp ý định

Để đảm bảo có tương quan so sánh, khóa luận này thử tiếp cận bài toán theo cả 2 phương pháp, trong đó phương pháp dự đoán xác suất sử dụng mô hình BERT, phương pháp đo khoảng cách số học sử dụng 2 nhánh chính là dựa trên nhiệm vụ phân đánh giá độ tương quan của văn bản (STS-B) được đào tạo sẵn trên mô hình T5 và dựa trên khoảng cách cosine với phương pháp vector hóa là SBERT và BM25. Dưới đây trình bài chi tiết từng cách tiếp cận.

a) Dự đoán xác suất bằng mô hình BERT: Từ mô hình BERT base với 12 tầng con, mỗi tầng là 1 encoder với đầu vào là 1 vector 768 chiều, đầu ra cũng là 1 vector 768 chiều nối tiếp. Tầng cuối cùng của mô hình là tầng hồi quy tuyến tính với kích thước đầu ra là kích thước từ điển. Tuy nhiên tầng này được lược bỏ, thay vào đó là tầng phân lớp tuyến tính với kích thước đầu ra bằng số nhãn lớp. Hàm softmax tính xác suất xuất hiện của từng nhãn, qua đó chọn được nhãn phân lớp.



Hình 4.3 Kiến trúc mô hình BERT

Kích thước tập dữ liệu:

Bổ sung dữ liệu: Ngoài việc khai thác 1 thẻ html thành 1 chuỗi, tăng cường dữ liệu cho huấn luyện, phương pháp này đòi hỏi kết hợp 3 câu văn thông thường thành 1 chuỗi nhằm tăng cường dữ liệu huấn luyện. Vì lý do dữ liệu quá ít, phương pháp này không áp dung việc chia dữ liệu thành tập test và train với tỷ lệ 0.2/0.8 mà toàn bộ dữ liệu được tham gia vào quá trình huấn luyện.

Dữ liệu cho bài toàn phân nhãn gồm có 21 nhãn và mỗi nhãn với tổng 287 chuỗi. Phân phối số lượng chuỗi cho các nhãn được thể hiện ở biểu đồ dưới đây.

Biểu đồ 4.1 Phân phối mẫu cho từng nhãn với phương pháp sử dụng BERT

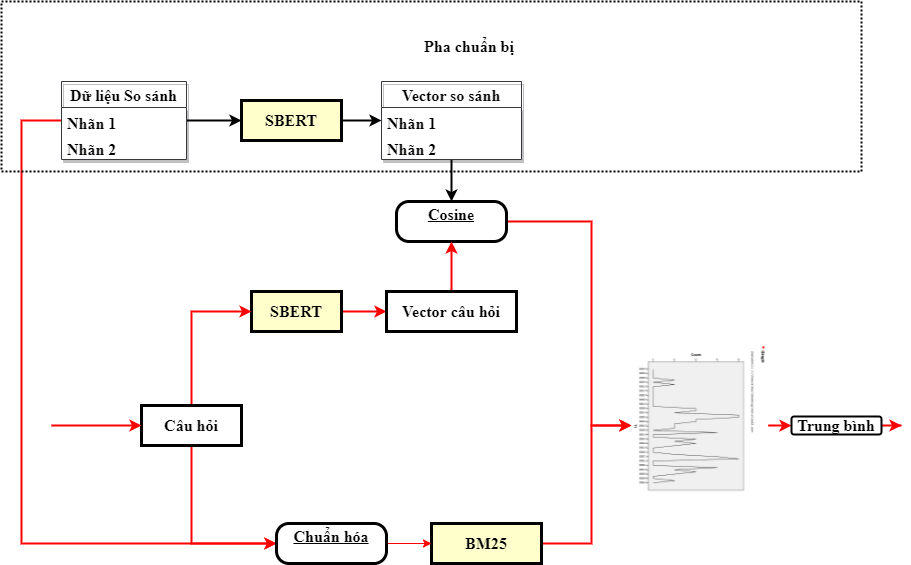
Tối ưu hóa siêu tham số

|  |  |  |
| --- | --- | --- |
| Siêu tham số | Giá trị | Lý do |
| Batch\_size | 16 | Giá trị tối đa cho phép của phần cứng |
| num\_train\_epochs | 10 | Dựa trên quan sát hội tụ |
| pf16 | True | Sử dụng pf16 nhằm giảm kích thước bộ nhớ chiếm dụng, có thể huấn luyện nhiều mẫu hơn trong 1 lô |
| learning\_rate | 1e-4 | learning rate thấp trong khoảng 1e-4 giúp giảm hiện tượng quên cục bộ trên mô hình |
| freeze layer | 1,2,3,4,5,6,7,8 | Dựa theo bài báo “How many layers of my BERT model should I freeze?” Tại Raphaël B.'s blog |
| optimizer | adamw | Tối ưu hóa mới và hiệu quả hiện nay |

b) Phương pháp khoảng cách cosine và khoảng cách TF-IDF

Phương pháp này dựa trên ý tưởng chuyển 2 cặp văn bản thành 2 vectơ rồi từ đó tính khoảng cách giữa 2 vectơ trong không gian:

Trong khóa luận này trình bày phương pháp vector hóa là sử dụng mô hình SBERT - một biến thể của mô hình BERT và một kỹ thuật khác là tính khoảng cách dựa trên công thức BM25.



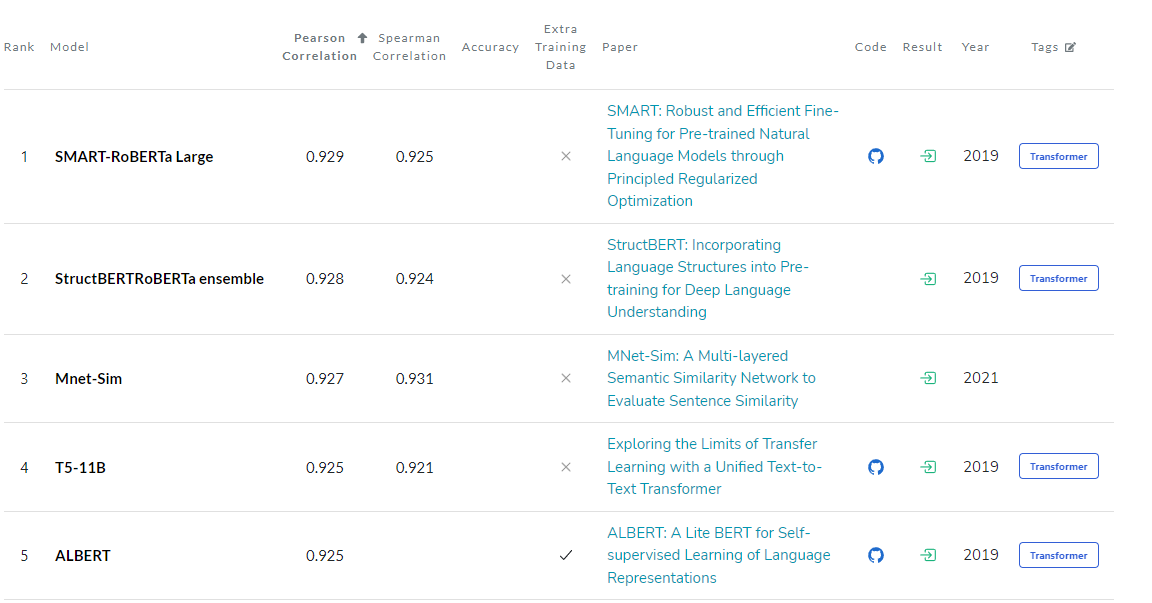
Hình 4.4 Phân lớp ý định bằng cosine và BM25

Về tổng quan: Dữ liệu sau khi khai phá, danh sách các nhãn được dùng làm dữ liệu so sánh. Truy vấn của người dùng sẽ được kết hợp với dữ liệu so sánh. Nếu dùng BM25 dữ liệu được chuẩn hóa bằng cách loại bỏ những từ không có nghĩa, chuyển động từ về dạng thường, loại bỏ tiền tố và hậu tố ở các từ. Sau đó sử dụng phương pháp tính điểm BM25 như trong chương 2 để tính khoảng cách giữa các từ. Cuối cùng, nhãn được chọn là nhãn có khoảng cách trung bình nhỏ nhất hoặc trung bình độ tương tự cao nhất.

Nếu dùng phương pháp vector hóa bậc cao SBERT, trước hết tập dữ liệu so sánh sẽ được mã hóa thành vector nhờ SBERT trước khi người dùng yêu cầu câu hỏi. Câu hỏi được vector hóa nhờ SBERT rồi ghép thành từng cặp với dữ liệu so sánh. Mỗi cặp sẽ được dùng phương pháp tính cosine để tính khoảng cách giữa 2 vector.

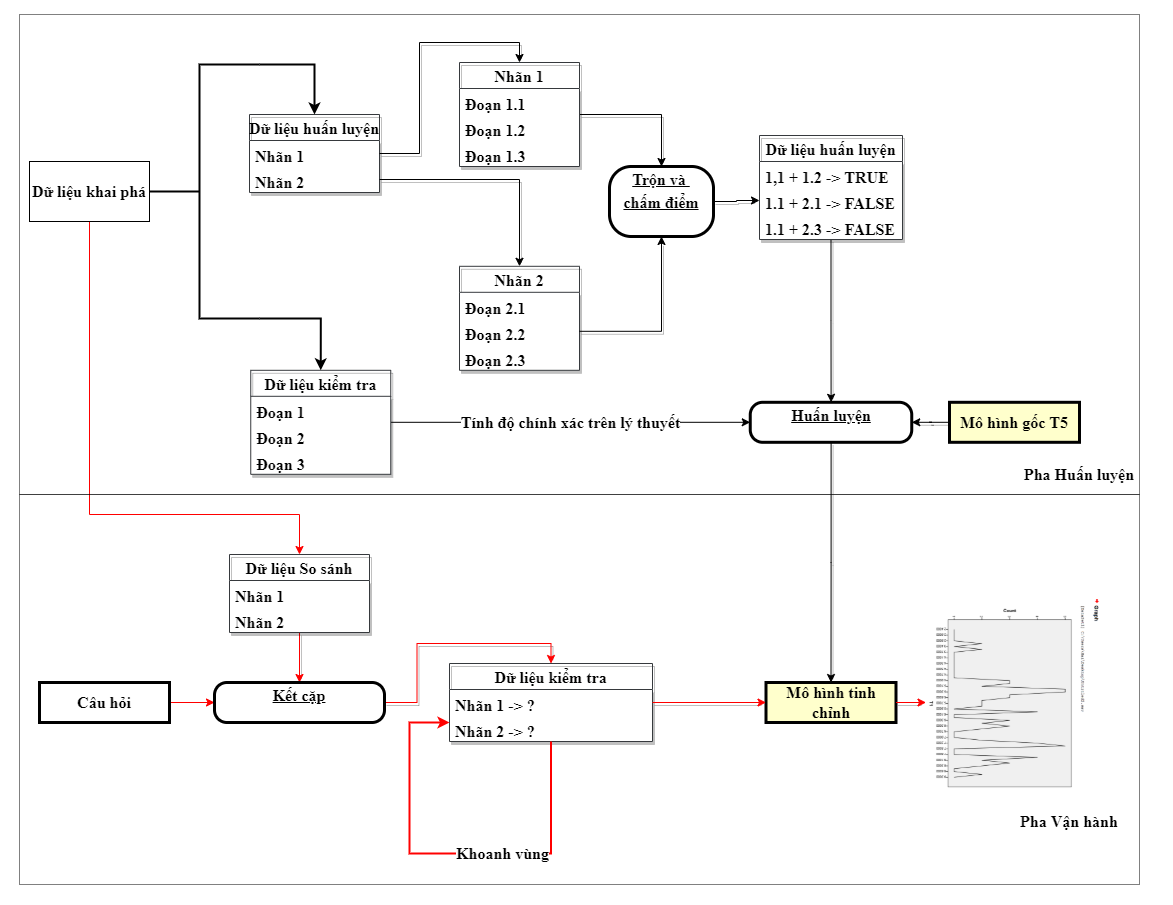
c) Phương pháp đối sánh dựa trên nhiệm vụ đánh giá độ tương tự STS-B

STS-B là tác vụ xử lý ngôn ngữ tự nhiên cơ bản được quy định trong kho dữ liệu xử lý ngôn ngữ tự nhiên của benchmark. Nhiệm vụ này có nhiệm vụ xác định mức độ liên quan của 2 câu văn hoặc 2 văn bản để thu được một kết quả trên thang đo 5.0 cho biết mức độ tương tự của 2 chuỗi. Nhiệm vụ này đã được huấn luyện sẵn trên mô hình T5 trong tất cả các phiên bản với thành tích xếp thứ 4 trong bảng thuộc về một đại diện của T5 đạt 0.925 đối với T5-11B. những phiên bản khác cũng giữ những thứ hạng cao trong nhiệm vụ này với kết quả thấp nhất là T5-small đứng thứ 21 với 0.856 điểm. Do phương pháp này cần nhiều thời gian huấn luyện và tài nguyên, khóa luận này chọn mô hình T5-small để thử nghiệm.



Hình 4.5 Kết quả STSB năm 2021

Do STS-B cho biết mức độ tương tự của 2 văn bản nên nó cũng mang chức năng cho biết khoảng cách của 2 văn bản tương tự như cosine. Tuy nhiên ở phương pháp này, văn bản cần tính khoảng cách và tập dữ liệu so sánh phải được đưa vào mô hình cùng lúc do đặc điểm của mô hình T5 thay vì có thể tính trước embedding của các vector so sánh như đối với BERT. Phần dưới đây trình bày chi tiết cách áp dụng mô hình T5 vào phân lớp ý định.



Hình 4.6 Phân lớp ý định dựa trên STS-B với T5

Xây dựng dữ liệu huấn luyện: Dữ liệu từ tài liệu của nhà phát triển được khai phá như mục 4.1. Toàn bộ dữ liệu được sao chép sang 1 vùng gọi là dữ liệu so sánh. dữ liệu này là 1 bảng chứa đoạn văn và cho biết đoạn văn đó thuộc nhãn nào, cụm nào. Đồng thời, toàn bộ dữ liệu gốc được chia theo tỉ lệ 0.8/0.2 thành dữ liệu huấn luyện và dữ liệu kiểm tra.

Sau đó, dữ liệu huấn luyện từ cùng 1 nhãn và từ các nhãn khác nhau được bắt cặp từng đôi một với nhau và được thêm tiền tố “stsb” cho tác vụ STS-B thành 1 đoạn huấn luyện. Nếu dữ liệu thuộc cùng 1 nhãn sẽ được chấm điểm liên quan cao.  dữ liệu khác nhãn sẽ được chấm điểm liên quan thấp. Cụ thể, ở phương pháp này, khóa luận đưa ra 2 cách thức cho điểm.

* Cách thức tích cực: Những đoạn huấn luyện được tạo thành từ 2 chuỗi cùng nhãn được cho điểm 5.0(tối đa), những đoạn huấn luyện từ 2 chuỗi khác nhãn và 2 nhãn không kề nhau nhưng cùng cụm được cho điểm 1.0, ngược lại nếu hai nhãn không kề nhau được 0.5 điểm, Mỗi chuỗi ghép với tên nhãn hoặc tên cụm của nó được 3,8 điểm. Còn lại được 0 điểm với 50% lượng chuỗi cho ra 0 điểm được loại bỏ ngẫu nhiên trong mỗi vòng lặp của pha huấn luyện nhằm cân bằng dữ liệu.
* Cách thức tiêu cực: Những đoạn huấn luyện được tạo thành từ 2 chuỗi cùng nhãn được cho điểm 5.0(tối đa), Mỗi chuỗi ghép với tên nhãn của nó được 5.0 điểm còn lại được 0.0 điểm với 50% lượng chuỗi cho ra 0 điểm được loại bỏ ngẫu nhiên trong mỗi vòng lặp của pha huấn luyện nhằm cân bằng dữ liệu.

Dữ liệu cho mỗi cách thức tính điểm gồm có:

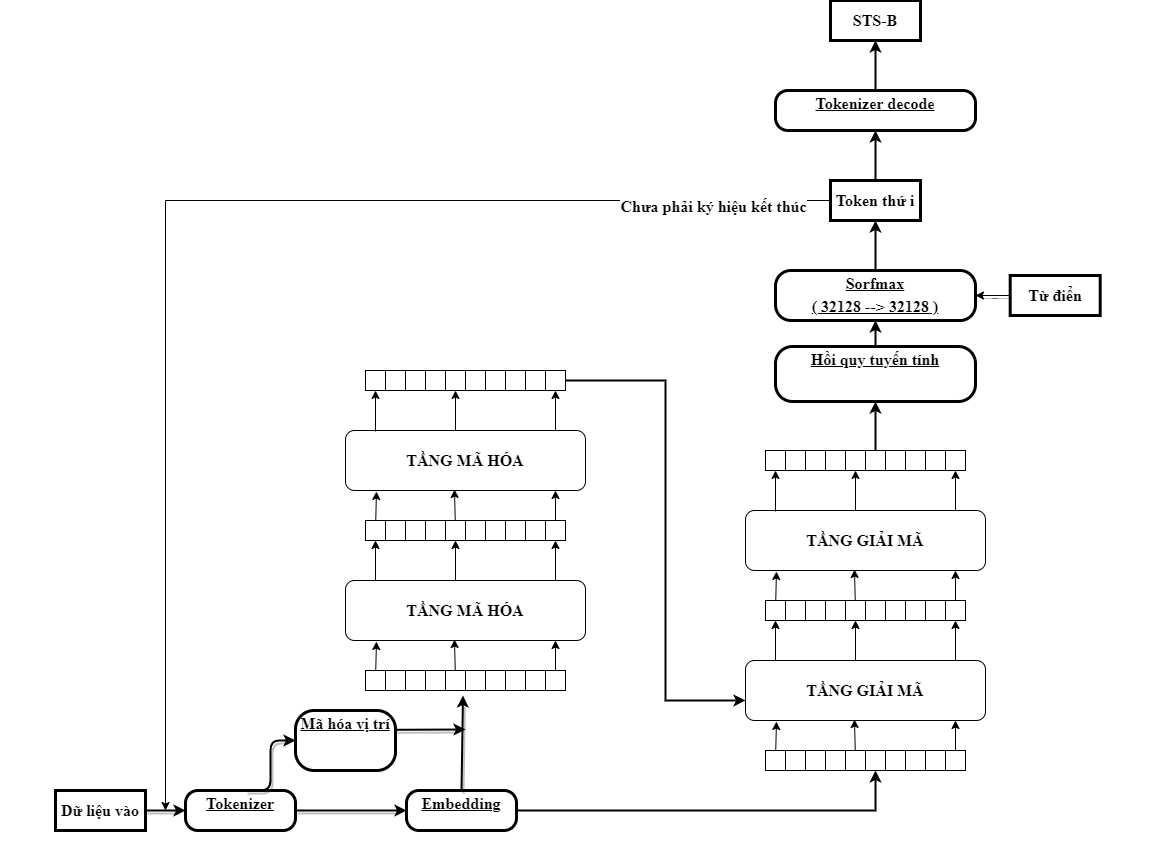
Biểu đồ 4.2 Thống kê số lượng mẫu huấn luyện cho từng điểm số với STS-B

Từ truy vấn mà người dùng yêu cầu, loại bỏ ký tự đặc biệt, html để trở thành câu hỏi. Khi người dùng gửi câu hỏi tới chatbot, câu hỏi được ghép cặp với từng đoạn trong dữ liệu so sánh. Do thời gian thực hiện của T5 tương đối chậm nên dữ liệu này được khoanh vùng bằng phương pháp BM25 nhưng không thực hiện phân lớp để tìm ra nhãn mà thực hiện phân lớp để tìm ra các cụm liên quan nhất. Sau đó được chuyển đổi sang định dạng cho tác vụ STS-B và lần lượt được đưa qua mô hình để đánh giá mức độ liên quan

Sau khi có bảng mức độ liên quan được gắn thêm vào bảng dữ liệu so sánh. bảng này được nhóm theo nhãn. mức độ liên quan của mỗi nhãn được tính bằng trung bình mức độ liên quan của từng đoạn trong nhãn. những nhãn liên quan nhất sẽ được chọn làm câu trả lời.

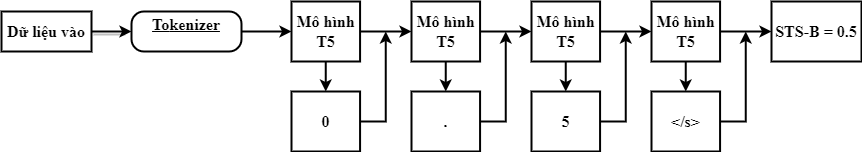
Về chi tiết hoạt động của mô hình T5, mỗi chuỗi được gẵn tiền tố cho tác vụ, sau đó được đưa vào mô hình T5 sẽ được đưa qua tuần tự các bước xử lý sau:

* Dữ liệu được đưa vào mô hình trước hết được mã hóa thành token. Mỗi token gồm 2 thành phần, thành phần lưu giữ nội dung của dữ liệu và thành phần cho biết chiều nào của chuỗi token thực sự chứa thông tin.
* Dữ liệu sau khi mã hóa sẽ được đi qua tầng embedding và position encoding để chuyển thành vector 512 chiều. Embedding layer ánh xạ token vào không gian vector để thu được embedding còn position encoding lưu giữ thông tin về vị trí của từ.
* Dữ liệu sau khi embedding được đưa qua tuần tự các encoder. Các encoder lần lượt đọc hiểu dữ liệu và trích xuất ra các đặc trưng tương ứng và phân hóa các đặc trưng này.
* decoder nhận đầu vào là vector từ tầng embedding và vector từ encoder. Đầu ra của encoder cũng là 1 vector 512 chiều.



Hình 4.7 Kiến trúc mô hình T5

* Cuối cùng từ vector thu được từ decoder sẽ được đưa qua tầng liên kết dày đặc (fully Connected) có bản chất là một mạng hồi quy tuyến tính có nhiệm vụ chuyển vector sau decoder (512) thành vector có số chiều bằng với kích thước từ điển (32128). Sau đó softmax tính ra xác suất xuất hiện của mỗi từ trong từ điển. từ đó chọn ra từ có khả năng xuất hiện cao nhất. Việc sinh ra giá trị STS-B của mô hình T5 được thực hiện thông qua nhiệm vụ sinh văn bản Sequen2Sequen, tức là nó sinh lần lượt từng thành phần của giá trị STS-B. (Giá trị STS-B 0.5 được coi là 1 chuỗi với ký tự “0”, “.”, “5”, “</s>” với “</s>” là ký tự kết thúc chuỗi.) Các kí tự được sinh liên tục và nếu đó không phải ký tự kết thúc câu thì tiếp tục giả mã thêm 1 lần nữa và thêm từ giả mã được vào câu đầu ra.  Nếu đó là ký tự kết thúc câu thì dừng lại.



Hình 4.8 Nhiệm vụ STS-B đối với mô hình T5

Tối ưu hóa Siêu tham số.

Bảng 4.1 Các tham số huấn luyện mô hình T5

|  |  |  |
| --- | --- | --- |
| Siêu tham số | Giá trị | Lý do |
| Batch\_size | 4 | Giá trị tối đa cho phép của phần cứng |
| num\_train epochs | 2 | Dựa trên quan sát hội tụ |
| pf16 | False | pf16 không hoạt động với mô hình T5 |
| learning\_rate | 1e-3 | nhà phát hành khuyến nghị tốc độ học cao cho tất cả các nhiệm vụ đối với mô hình T5 |
| freeze layer | 1,2,3, 5 | ở phần dưới |
| optimizer | AdamW | Tối ưu hóa mới và hiệu quả hiện nay |

Về việc đóng băng các tầng thuộc mô hình, T5 là một mô hình có đầy đủ encoder và decoder, trong đó decoder có nhiệm vụ sinh câu trả lời nên không được đóng băng, encoder có nhiệm vụ hiểu ngôn ngữ, việc đóng băng được thực hiện để tránh việc encoder sau khi tinh chỉnh chỉ giải quyết được những dữ liệu trong tập huấn luyện mà quên những dữ liệu đã được huấn luyện trước. Với các encoder tầng thấp sẽ hiểu được những ngôn ngữ phổ thông hơn các encoder tầng cao, các encoder tầng cao tập trung hiểu những ngôn ngữ hướng nhiệm vụ. Khóa luận này đã thực hiện đóng băng tất cả các tổ hợp có thể xảy ra và chọn được phương án đóng băng tốt nhất là đóng băng các tầng 1,2,3,5. Chi tiết kết quả được trình bày ở chương 5.

**4.2.1.2 Thành phần quản lý hội thoại**

Thành phần này nhận đầu vào là điểm đánh giá các nhãn của thành phần hiểu ngôn ngữ từ đó trích xuất câu trả lời từ tài liệu dựa trên dữ liệu thu được từ thành phần hiểu ngôn ngữ. Do đặc điểm khó có thể nhận dạng thực thể từ câu hỏi, thành phần quản lý hội thoại không đảm nhận chức năng lưu trữ những câu trò chuyện trước đó mà chỉ thực hiện một câu trò chuyện cho một đoạn hội thoại.

**4.2.1.3 Thành phần sinh ngôn ngữ**

Do tài liệu chính thức của nhà phát triển mang tính chính xác cao, câu văn hoàn chỉnh xúc tích nên việc sinh ngôn ngữ từ tài liệu trên dựa trên trí tuệ nhân tạo hoàn toàn không mang lại hiệu quả, thay vào đó, câu trả lời được sinh ra bằng luật duy nhất bằng cách ghép liên tiếp 3 yếu tố: liên kết đến tài liệu trả lời ở dạng liên kết và gạch chân 300 từ đầu tiên của câu dẫn nhập và 300 từ đầu tiên của chuỗi có độ liên quan lớn nhất trong nhãn được trả lời.

### Client

Client thực hiện nhiệm vụ chính là hiển thị nội dung hội thoại và gửi yêu cầu tới server để xử lý. Client đảm bảo cung cấp cho người dùng chức năng nhập yêu cầu và xem kết quả. Đồng thời người dùng có thể chọn phương pháp hiểu ngôn ngữ và chọn số lượng kết quả thu được. Client và server thực hiện giao tiếp với nhau qua RESTful API. Giao thức như bên dưới

Phương thức: GET.

Tham số:

params: {

question: yêu cầu từ người dùng,

top\_k: số lượng câu trả lời cần thiết (1-10),

left\_method: phương thức phân lớp ý định sử dụng trong khung trái (T5, cosine),

right\_method: phương thức phân lớp ý định sử dụng trong khung phải (T5, cosine),

left\_parameter: tham số cho phương thức bên trái (Số cụm nếu sử dụng T5, phương thức vector hóa nếu dùng BM25, SBERT).,

right\_parameter: tham số cho phương thức bên phải (Số cụm nếu sử dụng T5, phương thức vector hóa nếu dùng BM25, SBERT),

request\_hash: mã yêu cầu (sha1 ngẫu nhiên),

}

Phản hồi từ Server:

response: {

request\_hash: mã yêu cầu (sha1 ngẫu nhiên),

answer: {

left\_answer: [

label: nhãn câu trả lời,

answer: link đến tài liệu,

first: câu mở đầu,

highlight: đoạn văn liên quan nhất

],

right\_answer: [

label: nhãn câu trả lời,

answer: link đến tài liệu,

first: câu mở đầu,

highlight: đoạn văn liên quan nhất

]

}

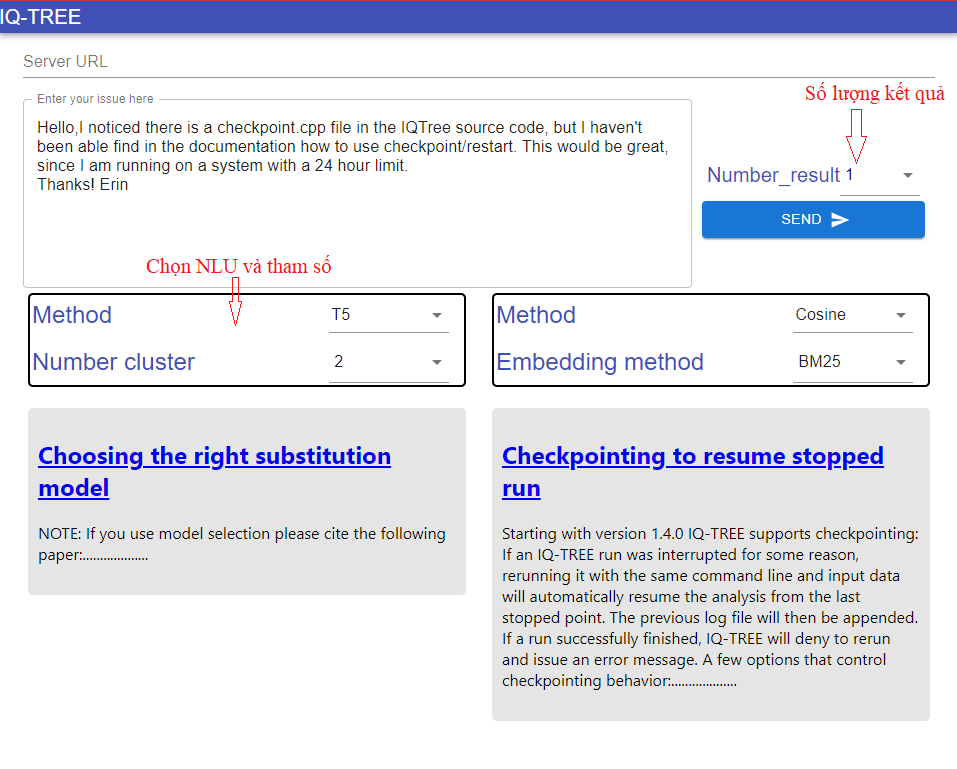
}

# HỆ THỐNG CHATBOT THỬ NGHIỆM

## Chương trình thử nghiệm

Ứng dụng phía client được thiết kế dưới dạng trang đơn (single page), những thao tác như hình dưới đây. Giao diện gồm các thành phần:

* Thanh nhập URL tới server cho các trường hợp dùng server đặc biệt.
* Khung nhập câu hỏi.
* Khu vực cho các lựa chọn bổ sung (chọn số lượng kết quả, chọn phương pháp hiểu ngôn ngữ, chọn loại vector hóa.)
* Khu vực khung trả lời trái và khung trả lời phải.



Hình 5.1 Ứng dụng thử nghiệm

## Thiết kế đánh giá hệ thống

### Đánh giá trong kịch bản

Đánh giá trong kịch bản được dùng để đánh giá trên lý thuyết các mô hình với nhau và chọn ra mô hình tốt nhất. Khóa luận này thực hiện 2 đánh giá trong kịch bản. Một là đánh giá độ chính xác khi thay đổi số tầng được đóng băng đối với mô hình T5, hai là đánh giá sự thay đổi khi thay đổi các mô hình. 2 đánh giá này dùng chung một thước đo.

Phương pháp đánh giá trong kịch bản được thiết kế dựa trên phương pháp đánh giá chuẩn của học máy. từ dữ liệu khai phá gồm 235 chuỗi, 20% số chuỗi tức là 47 chuỗi được đem làm dữ liệu đánh giá.

Kết quả đánh giá được quan sát thông qua chỉ số chính xác tuyệt đối (accuracy). Do lượng mẫu dữ liệu trong các nhãn tương đối đều nhau nên khóa luận không áp dụng độ chính xác trung bình điều hòa làm thước đo.

Với accuracy được tính bằng công thức

Với T là số mẫu dự đoán đúng.

N là tổng số mẫu.

accuracy được đánh giá gồm 3 thành phần

* accuracy top1 với việc đưa ra 1 câu trả lời và là câu trả lời đúng.
* accuracy top2 với việc đưa ra 2 câu trả lời trong đó có câu trả lời đúng.
* accuracy top 3 với việc đưa ra 3 câu trả lời trong đó có câu trả lời đúng.

### Đánh giá ngoài kịch bản

Dữ liệu đánh giá ngoài kịch bản được chọn từ những câu hỏi thực tế trên diễn đàn google group với dữ liệu gồm 8 câu hỏi và 5 câu hỏi tự tạo ra. Tính chính xác được quan sát bằng mắt thường. Kết quả được số hóa bằng chỉ số accuracy top 1, 2, 3 đã trình bày ở mục trên.

## Kết quả

### Kết quả đánh giá trong kịch bản

a) Đánh giá sự thay đổi độ chính xác khi thay đổi việc đóng băng các tầng trong mô hình T5.

Đối với chiến lược đóng băng tư trái sang phải, mô hình được lần lượt đóng băng các tầng liên tục từ tầng khởi đầu đến tầng được chọn từ 0 đến 6. Kết quả được cho ở đồ thị dưới.

Biểu đồ 5.1 Độ chính xác mô hình T5 khi đóng băng từ trái sang phải

Đối với việc đóng băng theo chiến lược đóng tất cả các tổ hợp có thể xảy ra: Mô hình T5 có 6 tầng encoder, mỗi tầng được ký hiệu là 0 nếu không đóng băng hoặc 1 nếu được đóng băng theo thứ tự từ trái sang phải. Từ đó mỗi mô hình được tạo nên có thể được ký hiệu bằng 1 chuỗi 6bit và chuỗi này được chuyển đổi thành 1 số nguyên làm ký hiệu mô hình. Kết quả việc đóng băng các mô hình như biểu đồ bên dưới.

Biểu đồ 5.2 Độ chính xác mô hình T5 khi đóng băng tất cả các tổ hợp

Qua đây có thể chọn được mô hình có độ chính xác cao nhất là mô hình đóng băng các tầng thứ 1,2,3,5. Với độ chính xác tuyệt đối 91.74%.

b) Đánh giá độ chính xác giữa các mô hình.

Bảng dưới đây thống kê số mẫu dự đoán chính xác của từng mô hình trên tổng số 47 mẫu thử.

Bảng 5.1 Kết quả thử nghiệm các mô hình trên lý thuyết

|  |  |  |  |
| --- | --- | --- | --- |
| Mô hình | accuracy top1 | accuracy top2 | accuracy top 3 |
| SBERT-Cosine | 36 | 40 | 43 |
| TF-IDF | 36 | 41 | 45 |
| T5-STSB | 43 | 45 | 46 |

Biểu đồ 5.3 Độ chính xác cho từng mô hình tính trên tập thử nghiệm lý thuyết

### Kết quả đánh giá ngoài kịch bản

Bảng dưới đây ghi lại kết quả thử nghiệm chương trình trên câu hỏi thực tế. Những kết quả được in đậm là kết quả mà mô hình dự đoán chính xác.

Bảng 5.2 Kết quả thử nghiệm các mô hình trên thực nghiệm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Câu hỏi | | | | | | |
| Hi,  Is the order of the support values always like this (aLRT%/UFBoot%), or does the order change depending on the order you add the commands in the command line argument? | | | | | | |
| kỳ vọng | Assessing branch supports with ultrafast bootstrap approximation/ Ultrafast bootstrap parameters/ Assessing branch supports with single branch tests | | | | | |
| BERT | 1 | Checkpointing to resume stopped run | 2 | Computing Robinson-Foulds distance | 3 | **Constructing consensus tree** |
| SBERT-Cosi ne | Assessing branch supports with standard nonparametric bootstrap | **Assessing branch supports with ultrafast bootstrap approximation** | reduccing impact with UFBoot |
| BM25 | **Assessing branch supports with ultrafast bootstrap approximation** | **Assessing branch supports with single branch tests** | reduccing impact with UFBoot |
| T5-STSB | **Assessing branch supports with single branch tests** | Using codon models | Checkpointing to resume stopped run |
| Câu hỏi | | | | | | |
| Hello, I noticed there is a checkpoint.cpp file in the IQTree source code, but I haven't been able find in the documentation how to use checkpoint/restart. This would be great, since I am running on a system with a 24hour limit.  Thanks! Erin | | | | | | |
| kỳ vọng | Checkpointing to resume stopped run | | | | | |
| BERT | 1 | **Constructing consensus tree** | 2 | **Miscellaneous options** | 3 | **Using codon models** |
| SBERT-Cosine | **Checkpointing to resume stopped run** | reduccing impact with UFBoot | Constructing consensus tree |
| BM25 | **Checkpointing to resume stopped run** | Utilizing multi-core CPUs | Using codon models |
| T5-STSB | Assessing branch supports with single branch tests | **Checkpointing to resume stopped run** | Utilizing multi-core CPUs |
| Câu hỏi | | | | | | |
| I am running IQ-TREE in a dataset of gene families of very closely relates species. I noticed that when IQ-TREE resolves branches from identical sequences (zero length branches), these are not associated with any support value (SH-aLRT/UFBoot). Do these branches lack support values because these methods cannot or should not be calculated in these zero branches? My guess is that lack of support value in a branch doesn't mean 0 or 100 because I saw 0 and 100 as support values in other branches with lengths higher than 0. I was reading on the documentation but I'm still having a hard time trying to understand this issue, so, I would appreciate if anybody can explain. | | | | | | |
| kỳ vọng | Ultrafast bootstrap approximation/ Assessing branch supports with single branch tests/ Single branch tests/ **Ultrafast bootstrap parameters/ Miscellaneous options** | | | | | |
| BERT | 1 | **Miscellaneous options** | 2 | **Ultrafast bootstrap parameters** | 3 | **Reducing impact of severe model violations with UFBoot** |
| SBERT-Cosine | **Assessing branch supports with single branch tests** | **Single branch tests** | Assessing branch supports with standard nonparametric bootstrap |
| BM25 | **Ultrafast bootstrap approximation** | **Assessing branch supports with single branch tests** | **Single branch tests** |
| T5-STSB | Rate heterogeneity | **Reducing impact of severe model violations with UFBoot** | Ancestral sequence reconstruction |
| Câu hỏi | | | | | | |
| Hi group,  I performed a search for the best partition scheme and selection of substitution models in conjunction with ultrafast boostrap approach. What I want to do next is to perform a standard boostrap analysis using the results from the previous run. So my question is if it is correct to fed the search with the best\_scheme.nex and best\_model.nex outputs, with the -p and -m flags, respectively. Like:  Or I may be missing something with that command? | | | | | | |
| kỳ vọng | Ultrafast bootstrap parameters/ Partition model options/ Nonparametric bootstrap/ **Specifying substitution models** | | | | | |
| BERT | 1 | **Specifying substitution models** | 2 | **Single branch tests** | 3 | **Input data** |
| SBERT-Cosine | **Ultrafast bootstrap parameters** | **Partition model options** | **Nonparametric bootstrap** |
| BM25 | **Partition model options** | Automatic model selection | **Ultrafast bootstrap parameters** |
| T5-STSB | **Partition model options** | Utilizing multi-core CPUs | **Nonparametric bootstrap** |
| Câu hỏi | | | | | | |
| Dear iqtree users,  I have a multiple alignment of 3000 bacterial genomes. Full alignment is about 5 Mb long, but only 50k sites are polymorphic (about 20k are AGCT only). Should I use full alignment or polymorphic sites only to determine the best model?  Generally speaking, does it make any difference if I keep monomorphic sites or not? | | | | | | |
| kỳ vọng | Choosing the right substitution model | | | | | |
| BERT | 1 | **Utilizing multi-core CPUs** | 2 | **Assessing branch supports with standard nonparametric bootstrap** | 3 | **Checkpointing to resume stopped run** |
| SBERT-Cosine | Using codon models | Partition model options | Binary, morphological and SNP data |
| BM25 | Using codon models | Site-specific frequency model options | Input data |
| T5-STSB | Checkpointing to resume stopped run | Assessing branch supports with single branch tests | Nonparametric bootstrap |
| Câu hỏi | | | | | | |
| Hi IQ-tree users and developers,  I am confused with the options in this command:  iqtree -s example.phy -sp example.nexus -m GTR+I+G  Will 'GTR+I+G' be applied to all gene partiotions? If that;s  Thanks in advance for your answer!  Kind Regards,  Elzed | | | | | | |
| kỳ vọng | Partition model options | | | | | |
| BERT | 1 | **Tree search parameters** | 2 | **Utilizing multi-core CPUs** | 3 | **Checkpointing to resume stopped run** |
| SBERT-Cosine | Using codon models | **Partition model options** | Choosing the right substitution model |
| BM25 | General options | reduccing impact with UFBoot | **Partition model options** |
|  |  |  |  |
| T5-STSB | Input data | Constructing consensus tree | General options |
| Câu hỏi | | | | | | |
| Dear IQTree developers,  I've been using IQTree v1.3.5 with a dataset of 51 taxa and > 2M bp. When running a partitioned analysis with "-spp" option for a coding DNA alignment (which according to what I understood from the manual should be linked partitions with proportional rates across them) in which I chose GTR+I+G for all 3 partitions, the program runs fine, but it keeps reporting rates of 1.0 for all 3 partitions. Is it a logging issue, or is the program actually not finding the partition-specific rates?  Thank you in advance. | | | | | | |
| kỳ vọng | Partition model options | | | | | |
| BERT | 1 | **Input data** | 2 | **Binary, morphological and SNP data** | 3 | **Nonparametric bootstrap** |
| SBERT-Cosine | **Partition model options** | Using codon models | Binary, morphological and SNP data |
| BM25 | **Partition model options** | Automatic model selection | Tree topology tests |
| T5-STSB | Constructing consensus tree | General options | Input data |
| Câu hỏi | | | | | | |
| Dear developers,  I am running currently a large (partitioned) dataset. I was wondering, in case the walltimes I have available (and max. 20 cores) are not sufficient, if I can simply restart the ML tree search: does it continues at the stage it crashed or is that not possible (so in other words: is is checkpointed?) dataset: 119 taxa, 931450 aa sites, LG4X included  (I now use random starting tree + gmedian but it takes remarkable longer than in a previous  test dataset (nearly this size) with an older IQtree version (version 1.3.10) where I used nearly same  settings - why is this?  Best Karen | | | | | | |
| kỳ vọng | Tree search parameters | | | | | |
| BERT | 1 | **Assessing branch supports with single branch tests** | 2 | **Generating random trees** | 3 | **Miscellaneous options** |
| SBERT-Cosine | **Tree search parameters** | Choosing the right substitution model | Single branch tests |
| BM25 | Generating random trees | **Tree search parameters** | Utilizing multi-core CPUs |
| T5-STSB | I can’t understand | I can’t understand | I can’t understand |
| Câu hỏi | | | | | | |
| Dear developers,  I am running currently a large (partitioned) dataset. I was wondering, in case the walltimes I have available (and max. 20 cores) are not sufficient, if I can simply restart the ML tree search: does it continues at the stage it crashed or is that not possible (so in other words: is is checkpointed?) dataset: 119 taxa, 931450 aa sites, LG4X included  (I now use random starting tree + gmedian but it takes remarkable longer than in a previous  test dataset (nearly this size) with an older IQtree version (version 1.3.10) where I used nearly same  settings - why is this? | | | | | | |
| kỳ vọng | Partition model options | | | | | |
| BERT | 1 | **Tree topology tests** | 2 | **Ultrafast bootstrap parameters** | 3 | **Likelihood mapping analysis** |
| SBERT-Cosine | Tree search parameters | Checkpointing to resume stopped run | Tree topology tests |
| BM25 | Generating random trees | Tree search parameters | Utilizing multi-core CPUs |
| T5-STSB | Tree topology tests | Utilizing multi-core CPUs | Rate heterogeneity |
| Câu hỏi | | | | | | |
| How can I perform bootstrap analysis? | | | | | | |
| kỳ vọng | Nonparametric bootstrap/ Assessing branch supports with standard nonparametric bootstrap | | | | | |
| BERT | 1 | **Tree topology tests** | 2 | **Reducing impact of severe model violations with UFBoot** | 3 | **Assessing branch supports with single branch tests** |
| SBERT-Cosine | **Nonparametric bootstrap** | Ultrafast bootstrap parameters | **Assessing branch supports with standard nonparametric bootstrap** |
| BM25 | **Nonparametric bootstrap** | Ultrafast bootstrap parameters | Single branch tests |
| T5-STSB | **Assessing branch supports with standard nonparametric bootstrap** | Assessing branch supports with single branch tests | Binary, morphological and SNP data |
| Câu hỏi | | | | | | |
| What are models for morphological data? | | | | | | |
| kỳ vọng | Binary, morphological and SNP data | | | | | |
| SBERT-Cosine | top1 | **Binary, morphological and SNP data** | top2 | Partition model options | top3 | Tree search parameters |
| TF-IDF | **Binary, morphological and SNP data** | Partition model options | Specifying substitution models |
| BERT | **Input data** | **Assessing branch supports with single branch tests** | **Rate heterogeneity** |
| T5-STSB | **Binary, morphological and SNP data** | Assessing branch supports with single branch tests | Partition model options |
| Câu hỏi | | | | | | |
| Show me how to adjust tree search parameters. | | | | | | |
| kỳ vọng | Tree search parameters | | | | | |
| BERT | 1 | **Assessing branch supports with single branch tests** | 2 | **Checkpointing to resume stopped run** | 3 | **Miscellaneous options** |
| SBERT-Cosine | **Tree search parameters** | Constructing consensus tree | Generating random trees |
| BM25 | **Tree search parameters** | Ultrafast bootstrap parameters | Constructing consensus tree |
| T5-STSB | Tree topology tests | Generating random trees | **Tree search parameters** |
| Câu hỏi | | | | | | |
| Can I run bootstrap along with branch test? | | | | | | |
| kỳ vọng | Single branch tests/ Assessing branch supports with single branch tests | | | | | |
| BERT | 1 | **Reducing impact of severe model violations with UFBoot** | 2 | **Assessing branch supports with ultrafast bootstrap approximation** | 3 | **Partition model options** |
| SBERT-Cosine | **Single branch tests** | Nonparametric bootstrap | Assessing branch supports with single branch tests |
| BM25 | **Single branch tests** | Nonparametric bootstrap | Assessing branch supports with single branch tests |
| T5-STSB | Site-specific frequency model options | **Assessing branch supports with single branch tests** | Using codon models |
| Câu hỏi | | | | | | |
| How can I run with multiple CPU cores? | | | | | | |
| kỳ vọng | Utilizing multi-core CPUs | | | | | |
| BERT | 1 | Tree search parameters | 2 | **Input data** | 3 | **Choosing the right substitution model** |
| SBERT-Cosine | **Utilizing multi-core CPUs** | Constructing consensus tree | General options |
| BM25 | **Utilizing multi-core CPUs** | Checkpointing to resume stopped run | Partition model options |
| T5-STSB | **Utilizing multi-core CPUs** | Site-specific frequency model options | Checkpointing to resume stopped run |

Thống kê kết quả các thuật toán từ bảng trên.

Biểu đồ 5.4 Thống kê kết quả các phương pháp phân lớp ý định trên thực nghiệm

Kết luận:

* Phương pháp nhận diện trực tiếp bằng mô hình BERT hầu như không có dự đoán đúng, các dự đoán đúng chỉ mang tính xác suất do tập dữ liệu huấn luyện quá nhỏ, không có khả năng bao phủ tập kiểm tra.
* Phương pháp Cosine sử dụng SBERT cho kết quả thực tế rất tốt do nó được huấn luyện trên phương pháp huấn luyện của một mạng nơ-ron kép, kết quả là một số có tính lớn nhỏ và hàm mất mát (loss function) khi huấn luyện tối ưu sự lớn nhỏ này. Đồng thời nó được huấn luyện trên tập dữ liệu rộng lớn, có tính bao phủ cao.
* Phương pháp nhận diện thực thể sử dụng BM25 mặc dù đã xuất hiện từ lâu và không có khả năng đọc hiểu văn bản trong văn cảnh và ngữ nghĩa nhưng lại cho kết quả vượt trội so với những phương pháp phân lớp bằng BERT và phương pháp khoảng cách T5. Kết quả này dẫn đến bởi lý do câu hỏi khá phức tạp, trình bày dài dòng nhưng nội dung hỏi lại ít. Do việc trình bày dài dòng mà từ khóa được lặp đi lặp lại nhiều lần nên phương pháp TF-IDF dễ nhận ra các từ khóa.
* Mô hình T5 có kết quả dự đoán trên tập lý thuyết rất cao song thực tế lại không cho kết quả cao bởi 2 lý do chính: Do dữ liệu huấn luyện còn khả nhỏ so với mô hình Transformer và do mô hình T5 đưa tất cả những tác vụ về tác vụ sinh văn bản hồi quy dẫn đến việc độ tương tự mà nó dữ đoán không phải là số nguyên, không mang tính đánh giá lớn nhỏ.

# CHƯƠNG 6. KẾT LUẬN

Sau thời gian nghiên cứu, dưới sự hướng dẫn tận tình của Tiến sĩ Hoàng Thị Điệp, tôi đã hoàn thành khóa luận “Xây dựng ứng dụng chatbot cho mô hình IQ-TREE” với kết quả chính như sau

* Tìm hiểu và nắm được những hệ thống kiến thức liên quan
  + Tổng quan về chatbot và những phương pháp xây dựng chatbot hiện nay.
  + Lược sử phát triển ngành xử lý ngôn ngữ tự nhiên
  + Kiến trúc mô hình Transformer
* Cài đặt thử nghiệm, huấn luyện và tối ưu hóa tham số cho mô hình Transformer nhằm trích xuất mức độ liên quan của văn bản
* so sánh việc sử dụng Transformer vào phân tích ý định người dùng với việc sử dụng phương pháp TF-IDF trong điều kiện dữ liệu hạn chế.

Hạn chế

Trong quá trình hoàn thành khóa luận và ứng dụng, mặc dù đã đạt được một số kết quả nhất định trong bài toán phân lớp ý định ứng dụng đối với hệ thống IQ-TREE tuy nhiên khóa luận vẫn còn một số hạn chế nhất định.

* Chưa tận dụng tối đa được sức mạnh của kiến trúc mới Transformer, dẫn đến hệ thống chưa có khả năng đọc hiểu câu văn một cách thực sự.
* Do hệ thống ý định người dùng phức tạp trong khi dữ liệu còn ít, nên độ chính xác của hệ thống chưa được như mong muốn
* Do yêu cầu độ chính xác cao nên hệ thống chưa thể thay thế người dùng trong một số thao tác như sinh ra mã chạy chương trình mà chỉ có thể tìm kiếm tài liệu cho người dùng

Hướng phát triển

Với những kiến thức và kỹ năng đã có được từ khóa luận này., trong tương lai em sẽ khai thác thêm nhiều đặc trưng của ngôn ngữ nhằm tăng độ chính xác của hệ thống. Đồng thời cài đặt để chatbot có thể ghi nhớ cuộc hội thoại, từ đó tạo ra những cuộc hội thoại dài hơn, tăng tính tương tác với người dùng.

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1] Bùi Đức Anh, Xây dựng chatbot hỗ trợ học tiếng anh, Luận văn thạc sĩ, Đại học Công nghệ, Đại học Quốc Gia Hà Nội, 2017, tr.11-32.

[2] Phạm Minh Nguyên, Nghiên cứu dịch máy Trung-Việt dựa vào mô hình Transformer, Luận văn thạc sĩ, Đại học Công nghệ, Đại học Quốc Gia Hà Nội, 2020, tr.18-41.

[3] Đỗ Viết Mạnh, Xây dựng chatbot bán hàng dựa trên mô hình sinh, Luận văn thạc sĩ, Viện hàn lầm khoa học và công nghệ Việt Nam, 2020, tr.5-44.

**Tiếng Anh**

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

[6] Baiyun Cui, Yingming Li∗, Ming Chen, and Zhongfei Zhang, Fine-tune BERT with Sparse Self-Attention Mechanism, 2019.

[7] Mihir Kale, Abhinav Rastogi. Text-to-Text Pre-Training for Data-to-Text Tasks, 2019

[8] Google AI. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2020.

[9] Jianmo Ni, Gustavo Hernandez ´ Abrego, Noah Constant, Ji Ma, ´ Keith B. Hall, Daniel Cer, Yinfei Yang. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models, 2020.