Thực hiện Cognitive Graph cho Multi-Hop Reading Comprehension trong Wiki

Tran Hoang Anh[1,1*], Ngo Huynh Truong[1,1†] and Phong Lai Bao Minh[1,1†]

[1*]VNUHCM, University of Information Technology, VietNam.

*Corresponding author(s). E-mail(s): 20521079@gmail.com;
Contributing authors: 20522085@gmail.com; 20522217@gmail.com;
†These authors contributed equally to this work.

Tóm tắt nội dung

Với đề tài này, nhóm chúng em thực hiện lại CogQA framework cho hỏi đáp multi-hop trong cơ sở dữ liệu được trích xuất từ Wiki tiếng Anh. Được lấy ý tưởng từ quy trình kép trong khoa học nhận thức, framework này sẽ xây dựng một đồ thị nhận thức (Cognitive Graph) bằng cách phối hợp một module chiết suất ngầm (implicit extraction) hay còn gọi là System 1; và một modulel lý luận rõ ràng (explicit reasoning) hay còn gọi là System 2. Và ngoài việc trả lời cho câu hỏi được đặt ra, framework sẽ đưa ra được lý do vì sao chọn ra được đáp án đó, được gọi là reasoning path. Việc thực hiện sẽ dựa trên BERT và GNN (Graph Neural Network), được train và test trên bộ dữ liệu HotpotQA fullwiki, đạt được F1-score là 25.6

Keywords: GNN, BERT, multi-hop, QA, cognitive graph

1 Introduction

Deep Learning đã và đang thống trị trong lĩnh vực MRC (Machine Reading Comprehension). Tuy nhiên, để chạm và vượt qua được khả năng đọc hiểu giữa con người và máy móc, chúng ta có 3 thách thức chính: 1) Khả năng lý luận. Thông thường thì một model cho việc hỏi đáp sẽ có khuynh hướng đưa ra câu trả lời cho câu hỏi, mà không đưa ra được ngữ cảnh hay lý do vì sao nó đưa ra được câu trả lời đó. Và do đó, multi-hop đã và đang trở thành tiên phong trong vấn đề này. 2) Khả năng giải thích. Đường dẫn lý luận rõ ràng

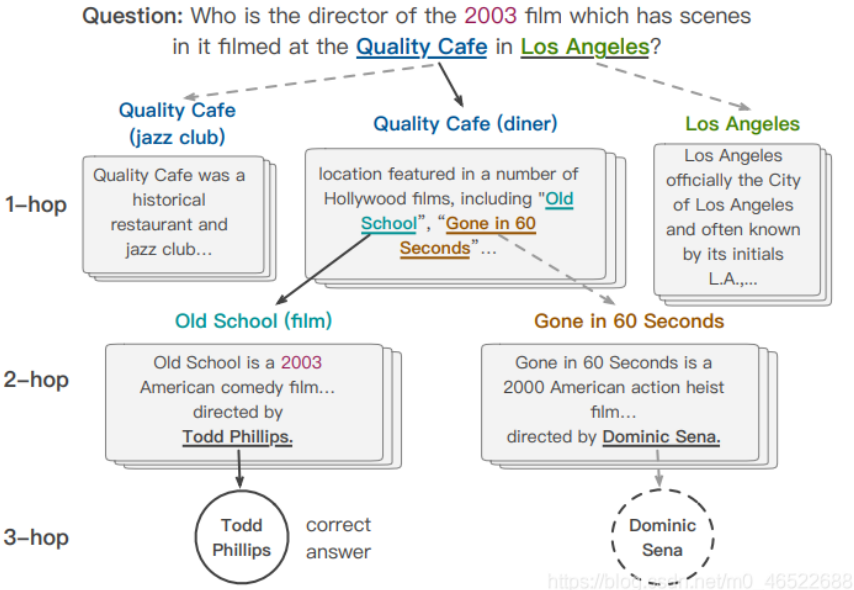
(explicit reasoning paths), cho phép xác minh được tính chặt chẽ trong logic, và điều đó rất quan trọng đối với độ tin cậy của hệ thống hỏi đáp. Bộ dữ liệu HotpotQA yêu cầu model phải đưa ra được supporting fact, có nghĩa là các giải thích *không có thứ tự* và ở *mức độ câu (sentence-level)*, mà con người có thể xem và diễn giải câu trả lời theo từng bước một, và chỉ ra được các giải thích *theo thứ tự* và ở *mức độ thực thể (entity-level)*. 3) Khả năng mở rộng. Cho nhiều mục đích thực tiễn hữu ích của một hệ thống hỏi đáp, khả năng mở rộng là điều rõ ràng và thiết yếu. Các hệ thống hỏi đáp dựa trên đọc hiểu máy học đa phần sử dụng retrieval - extraction framework in DrQA (Chen et al., 2017)[1], nhằm giảm phạm vi nguồn phải truy xuất xuống vài văn bản bằng cách truy xuất trước. Framework này đơn giản là sự kết hợp giữa hỏi đáp trong một văn bản và thông tin truy xuất có thể mở rộng, so với khả năng lý luận lướt qua với kiến thức lớn được lưu trữ của con người.

Từ những điều trên, insight của những giải pháp cho những thách thức này có thể rút ra được từ quá trình nhận thức của con người (cognitive process). Lý thuyết kép (Evans, 184, 2003, 2008; Sloman, 1996)[2] cho rằng bộ não của chúng ta đầu tiên sẽ truy xuất thông tin dựa vào sự chú ý thông qua một quá trình *tiềm ẩn, vô thức và trực quan*, hay còn gọi là **System 1**; Dựa trên một quá trình lập luận *rõ ràng, có ý thức và có thể kiểm soát*, hay còn gọi là **System 2**, được thực hiện sau đó. **System 1** có thể đưa ra được những tài nguyên, thông tin dựa vào những yêu cầu, trong khi đó **System 2** có khả năng đi sâu vào sự liên quan của những thông tin trước đó bằng cách thực hiện suy nghĩ có trình tự trong bộ nhớ làm việc, tuy chậm hơn nhưng giống với quá trình duy lý độc nhất của con người. Cho những lý luận phức tạp hơn, 2 hệ thống **(System 1 và 2)** sẽ cùng nhau điều phối để thực hiện *suy nghĩ nhanh và chậm* (Fast and Slow Thinking) được lặp lại nhiều lần.

Trong đề tài này, chúng em sẽ thực hiện framework Cognitive Graph QA (CogQA), được xây dựng để xử lí các thử thách được đề trên. Được lấy ý tưởng từ lý thuyết kép, framework này bao gồm 2 module khác nhau là **System 1** và **System 2**. **System 1** sẽ trích xuất những entity (thực thể) có liên quan đến câu hỏi và answer candidates (các câu trả lời có tiềm năng) từ đoạn văn bản trích xuất từ các entity đã được trích xuất từ câu hỏi; ngoài ra sẽ mã hóa các thông tin ngữ nghĩa của chúng. Những entity được trích xuất sẽ được tổ chức thành cognitive graph (đồ thị nhận thức) (Hình 1), biểu diễn bộ nhớ làm việc. System 2 sau đó sẽ thực hiện thủ tục luận lý dựa trên đồ thị nhận thức, và thu thập các *clues* (manh mối) để có thể chỉ dẫn **System 1** trích xuất next-hop entity (thực thể có thể "nhảy" đến một node mới trong đồ thị) tốt hơn cho lần sau. Quá trình trên được thực hiện lặp lại cho đến khi tất cả câu trả lời có thể có được tìm được, và câu trả lời cuối cùng được chọn dựa trên kết quả luận lý từ System 2. Việc thực hiện framework này dựa trên BERT (Devlin et al., 2018)[3] và Graph Neural Network (GNN) (Bettaglia et al., 2018)[4].

Tóm tắt:

- Giới thiệu framework CogQA cho multi-hop reading comprehension với web-scale document dựa trên nhận thức của con người.



Hình 1 Một ví dụ về Cognitive Graph cho multi-hopQA.

- Cho thấy kiến trúc của cognitive graph (đồ thị nhận thức) cung cấp khả năng giải thích có thứ tự và ở entity-level; phù hợp cho việc lý luận quan hệ.
- Việc thực hiện framework dựa trên BERT và GNN.

2 Bộ dữ liệu HotpotQA (HotpotQA Dataset)

Sử dụng cài đặt full-wiki của HotpotQA để tiến hành thực hiện. Bộ dữ liệu HotpotQA được lưu trữ ở định dạng JSON, với từng điểm dữ liệu ở train và dev set bao gồm:

- _id: (int) số id duy nhất cho từng điểm dữ liệu.
- question: (string) câu hỏi.
- answer: (string) câu trả lời. Tập test không bao gồm câu trả lời.
- supporting_facts : (list) mỗi phần tử của supporting_facts là một list gồm 2 thành phần [title, sent_id], với title được định nghĩa là tiêu đề của đoạn văn bản trong Wikipedia, và sent_id được định nghĩa là id của supporting_facts (bắt đầu từ 0) trong đoạn văn bản với tiêu đề là title. Tập test không bao gồm supporting_facts.
- context: (list) mỗi phần tử là một đoạn văn bản, được biểu diễn bởi 1 list với 2 thành phần [title, sentences], với title là tiêu đề của đoạn văn bản trong Wikipedia, và sentences là một list bao gồm các câu trong đoạn văn bản. Ngoài ra còn có một số key khác:
- type: (string) là "comparision" hoặc "brigde", chỉ ra kiểu câu hỏi.
- level: (string) "easy", "medium" hoặc "hard". Là mức độ của câu hỏi.

Bộ dữ liệu HotpotQA bao gồm các file:

- Training set
- Dev set với cài đặt distractor
- Dev set với cài đặt full wiki: Là dev set với cài đặt distractor không có gold paragraph (các văn bản hữu ích cho việc huấn luyện mô hình), nhưng thay vào đó là top 10 đoạn văn bản thu được sử dụng hệ thống truy xuất của tác giả HotpotQA. Có thể thay đổi hệ thống truy xuất khác bằng cách thay đổi các đoạn văn bản thành kết quả truy xuất riêng.
- Test set, với cài đặt fullwiki.

Cài đặt full wiki của bộ dữ liệu HotpotQA bao gồm 112.779 câu hỏi được thu thập bởi nguồn lực cộng đồng dựa trên đoạn văn bản đầu tiên trong Wikipedia (hay còn gọi là introductory paragraph), 84% của bộ dữ liệu yêu cầu luận lý multi-hop. Bộ dữ liệu được chia làm training set (90,564 câu hỏi), development set (7,405 câu hỏi) và test set (7,405 câu hỏi). Tất cả level của câu hỏi trong tập dev và test là các trường hard multi-hop. Trong training set, cho mỗi câu hỏi, 1 câu trả lời và đoạn văn bản của 2 gold (hữu ích) entity được cung cấp, với nhiều supporting facts, các câu chứa các thông tin từ khóa quan trọng cho việc luận lý, đánh dấu. Ngoài ra còn có 8 đoạn văn bản không hữu ích cho việc huấn luyện mô hình. Trong quá trình đánh giá, chỉ có câu hỏi được cung cấp và trong khi đó supporting facts yêu cầu cũng phải được cung cấp ngoài câu trả lời.

Để xây dựng đồ thị luận lý (cognitive graph) cho huấn luyện, các cạnh (edges, có thể hiểu là các đường dẫn, cũng là các entity, nằm trong các câu của đoạn văn bản) trong gold-only cognitive graph được suy ra từ supporting fact bằng so khớp mờ (fuzzy matching) (Levenshtein distance, Navarro, 2001). Cho mỗi supporting fact trong *para[x]*, nếu bất kì gold entity (next-hop entity) hoặc câu trả lời (đều được định nghĩa là y), được tìm thấy bởi fuzzy matching với 1 span trong supporting fact, edge(x,y) sẽ được thêm vào trong cognitive graph.

3 Cognitive Graph QA Framework

Khả năng luận lý của con người dựa vào cấu trúc quan hệ của thông tin. Theo trực giác, chúng ta áp dụng cấu trúc đồ thị có hướng để suy luận và khám phá từng bước trong quá trình nhận thức của multi-hop QA. Mỗi node trong cognitive graph **G** tương ứng với một entity hoặc một câu hỏi có tiềm năng **x**, cũng có thể định nghĩa là node **x**. Module trích xuất **System 1** đọc introductory paragraph **para[x]** (đoạn văn bản giới thiệu của một article ứng với entity trong Wiki) của entity **x** và trích xuất ra các *câu trả lời tiềm năng (answer candidates)* và *next-hop entity* hữu ích trong đoạn văn bản của entity **x**. *G* sau đó được mở rộng với những node đó, cung cấp cấu trúc rõ ràng cho module luận lý, **System 2**. Trong framework này, giả định **System 2** tiến hành thực hiện deep learning thay vì sử dụng luận lý rule-based bằng cách tính toán hidden representations **X** (là list tổng hợp các vector ngữ nghĩa của tất cả các node) của các node trong cognitive graph. Do đó, **System 1** cũng được yêu cầu tổng hợp **para[x]** thành 1 vector ngữ nghĩa (semantic vector),

như là khởi tạo cho hidden representiation khi trích xuất spans trong đoạn văn bản của entity **x**. Sau đó **System 2** sẽ cập nhật **X** dựa vào cấu trúc của cognitive graph như là kết quả luận lý cho down-stream prediction.

Khả năng giải thích được thể hiện ở đường luận lý rõ ràng trong cognitive graph. Ngoài khả năng đưa ra đường suy luận đơn giản thì cognitive graph còn đưa ra được những tiến trình luận lý chung (joint) hoặc lặp (loopy), nơi mà những node trước mới (predecessors node) sẽ lấy ra được *clues (manh mối)* mới về câu trả lời. *Clues* trong CogQA là một khái niệm linh hoạt về mặt hình thức, liên quan đến thông tin từ node trước đó để hướng dẫn **System 1** trích xuất spans một cách tốt hơn. Ngoài các node mới đã được thêm vào, những node với các edge mới cũng cần được xem lại do các *clues* mới được tìm thấy sau việc cập nhật ở từng vòng lặp. Những node mới nhất được thêm vào được định nghĩa là *frontier nodes*.

Khả năng mở rộng có nghĩa là thời gian tiêu thụ của việc hỏi đáp sẽ không tăng một cách đáng kể theo số lượng đoạn văn bản. Việc quản lí cơ sở dữ liệu các đoạn văn văn bản của Wiki sử dụng *Redis* là một cơ sở dữ liệu NoSQL, và việc truy cập thông tin của một article đơn giản chỉ cần đến chỉ mục (index) được định nghĩa là tiêu đề (title) của article đó. Framework CogQA có thể mở rộng vì thao tác duy nhất để có thể truy cập đến tất cả đoạn văn bản là truy cập đoạn văn bản cụ thể theo chỉ mục tiêu đề của chúng. Với câu hỏi dạng multi-hop, framework retrieval - extraction thông thường có thể sẽ hi sinh đi tiềm năng của những mô hình tiếp nối, bởi vì những đoạn văn bản mà được hop (nhảy) nhiều ra xa khỏi câu hỏi thì sẽ ít mang được những từ liên quan và quan hệ ngữ nghĩa với câu hỏi, dẫn đến việc truy xuất thông tin thất bại. Tuy nhiên thì những đoạn văn bản liên quan đến câu hỏi có thể tìm thấy được bằng cách mở rộng nhiều lần cùng với *clues* trong CogQA framework.

Giải thuật 1 miêu tả quy trình của CogQA framework. Sau khi khởi tạo, một quy trình lặp cho việc mở rộng cognitive graph và luận lý được diễn ra. Mỗi vòng lặp, chúng ta sẽ đi qua 1 frontier node **x**, và **System 1** đọc *para[x]* (đoạn văn bản của node **x**) dưới sự hướng dẫn của *clues* và câu hỏi **Q**, trích xuất spans và sinh ra vector ngữ nghĩa *sem[x, Q, clues]*. Trong khi đó, **System 2** cập nhật hidden representation **X** và chuẩn bị *clues[y, G]* cho bất kỳ node **y** (được sinh ra từ việc trích xuất span từ *para[x]*) nào thành công được tìm ra. Kết quả dự đoán cuối cùng sẽ được thực hiện dựa trên hidden representation **X**.

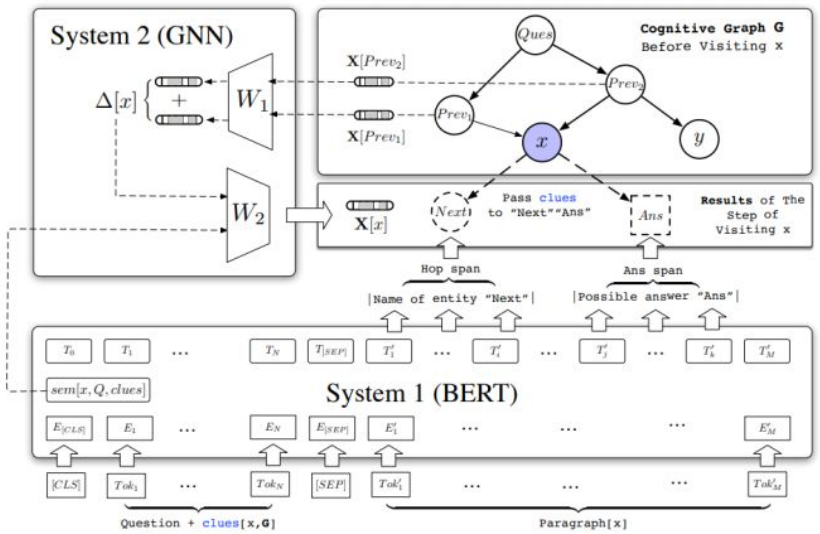
4 Triển khai thực hiện CogQA Framework

Công việc chính của việc thực hiện framework CogQA là để xác định những mô hình cụ thể (System 1 và System 2) và định dạng chính xác của *clues*.

Sử dụng BERT là **System 1** và GNN là **System 2**. Trong khi đó, *clues[x,G]* là những câu nằm trong đoạn văn bản của node trước **x** đang được xét đến. Việc sử dụng trực tiếp văn bản gốc cho *clues*, thay vì bất kì các dạng hidden state khác, là để giúp cho việc huấn luyện mô hình dễ dàng hơn

cho **System 1**. Hidden representation **X** được cập nhật mỗi bước forward propagation của GNN.

Tổng quan cho mô hình được trực quan ở Hình 2.



Hình 2 Tổng quan về cách triển khai CogQA

4.1 System 1

Khả năng trích xuất của **System 1** là nền tảng của việc xây dựng cognitive graph, do đó việc áp dụng một mô hình mạnh mẽ là cần thiết. Đến hiện tại, BERT vẫn là một mô hình biểu diễn ngôn ngữ (language representation) thành công và mạnh mẽ nhất với nhiều tác vụ NLP. Các câu đầu vào của mô hình bao gồm hai bộ phận với chức năng khác nhau A và B.

Framework sử dụng BERT là **System 1**, và đầu vào khi đi tới node **x** được biểu diễn như sau:

$$\underbrace{[CLS]Question[SEP]clues[x,G][SEP]}_{\text{Sentence A}}\ \underbrace{Para[x]}_{\text{Sentence B}}$$

với *clues[x,G]* là những câu được lấy từ những node trước đó (predecessor nodes). Và đầu ra của BERT được định nghĩa là $\mathbf{T} \in \mathbb{R}^{L \times H}$, với L là chiều dài của chuỗi đầu vào và H là số chiều của hidden representation.

4.1.1 Trích xuất Span

Câu trả lời và next-hop entity có những thuộc tính khác nhau. Việc trích xuất câu trả lời phụ thuộc vào các ký tự được chỉ ra bởi câu hỏi. Ví dụ "New York City" thì có khả năng là câu trả lời cho câu hỏi "where" hơn là "2019", trong khi đó thì next-hop entity thưởng có xu hướng là những diễn tả mà giống hoặc khớp với các câu trong câu hỏi. Vì thế mà việc dự đoán span cho câu trả lời và span cho next-hop entity được dự đoán riêng biệt nhằm tăng độ chính xác

Sử dụng "pointer vectors" $S_{hop}, E_{hop}, S_{ans}, E_{ans}$ cho việc đánh dấu vị trí bắt đầu và kết thúc của 1 span được dự đoán, ứng với các tham số có thể học được.Từng pointer vector này là một sequence có số chiều là độ dài lớn nhất có thể của một câu (maxL). Xác suất của token đầu vào thứ i là pointer bắt đầu của một span câu trả lời là $\mathcal{P}_{ans}^{start}[i]$ được tính với công thức như sau:

$$\mathcal{P}_{ans}^{start}[i] = \frac{e^{S_{ans}.T_i}}{\sum_j e^{S_{ans}.T_j}} (1)$$

Với $\mathcal{P}_{ans}^{end}[i]$ là xác xuất của token đầu vào thứ i là pointer kết thúc của 1 span câu trả lời, thì công thức tính toán cũng tương tự như công thức nêu trên. Và chỉ tập trung vào top K các xác xuất của $start_k$. Cho mỗi k, thì xác suất của vị trí end_k là:

$$end_k = \underset{start_k \leq j \leq start_k + maxL}{argmax} P_{ans}^{end}[j] (2)$$

Với maxL là độ dài lớn nhất có thể của span

Để xác định được các đoạn văn bản có liên quan đến câu hỏi, chúng ta tận dụng các mẫu âm (negative sampling) được nhắc đến trong phần Huấn luyện, để huấn luyện cho **System 1** sinh ra ra một *ngưỡng âm (negative threshold)*. Trong top K span, thì token nào có xác suất bắt đầu của 1 span thấp hơn ngưỡng âm thì sẽ bị loại bỏ. Bởi vì token thứ 0 là [CLS] được tiền huấn luyện để tổng hợp các token đầu vào cho tác vụ Dự đoán câu tiếp theo (Next sentence prediction), $\mathcal{P}_{end}^{start}[0]$ sẽ là ngưỡng (threshold).

4.1.2 Sinh vector ngữ nghĩa

Như đề cập ở trên, đầu ra của BERT tại position 0 có khả năng sử dụng để tổng hợp một chuỗi từ. Vì vậy, phương pháp đơn giản nhất đó là sử dụng trực tiếp T0 (T_0) (với T là đầu ra của BERT) cho *sem[x,Q,clues]*. Tuy nhiên, những layer cuối của BERT chủ yếu phụ trách cho việc biến đổi hidden represetation cho tác vụ dự đoán span. Trong những thử nghiệm, thì việc sử dụng lớp thứ 3 đến lớp cuối cùng của đầu ra tại position 0 cho *sem[x,Q,clues]* cho ra kết quả tốt nhất.

4.2 System 2

Hàm đầu tiên của **System 2** dùng để chuẩn bị *clues[x,G]* cho các frontier node (được triển khai bằng cách thu thập các câu gốc của các node trước đó của x, với câu của các node đó đề cập đến x)

Hàm thứ hai dùng để cập nhật hidden representation **X**, là cốt lõi của **System 2**. Hidden representation $\mathbf{X} \in \mathbb{R}^{n \times H}$ đại diện cho việc hiểu tất cả n entity trong *G*. GNN đã được đề xuất sử dụng để thực hiện deep learning trong đồ thị (Kipf and Welling, 2017)[5], đặc biệt là suy luận quan hệ do khuynh hướng quy nạp của cấu trúc đồ thị (Battaglie et al., 2018)[4].

Trong quá trình thực hiện, một biến thể của GNN được thiết kế để làm **System 2**. Với mỗi node x, chúng ta khởi tạo hidden representation $\mathbf{X}[x] \in \mathbb{R}^H$ là vector ngữ nghĩa *sem[x,Q,clues]* từ **System 1**. Gọi **X'** là hidden representation mới sau khi thực hiện bước propagation của GNN, và $\Delta \in \mathbb{R}^{n \times H}$ là vector tổng hợp được truyền từ các node gần kề trong quá trình propagation. Công thức cập nhật cho **X** như sau:

$$\Delta = \sigma((AD^{-1})^T \sigma(\mathbf{X}W_1)) \quad (3)$$

$$\mathbf{X'} = \sigma(\mathbf{X}W_2 + \Delta) \quad (4)$$

Với σ là hàm kích hoạt và $W_1, W_2 \in \mathbb{R}^{H \times H}$ là các ma trận tham số. A là ma trận kề (Adjacency matrix) của G, được chuẩn hóa theo cột thành AD^{-1} với $D_{jj} = \sum_i A_{ij}$ .Vector ẩn được biến đổi là $\sigma(\mathbf{X}W_1)$ được nhân trái với $(AD^{-1})^T$.

Trong các bước lặp khi đi đến frontier node x, hidden representation **X**[x] được cập nhật theo công thức (3)(4).

5 Mô hình dự đoán (Predictor)

Những câu hỏi trong bộ dữ liệu HotpotQA đa phần sẽ rơi vào 3 phân loại: câu hỏi đặc biệt (special question), câu hỏi thay thế (alternative question) và câu hỏi chung (general question), được coi là 3 tác vụ dự đoán khác nhau cùng lấy **X** là đầu vào.

Câu hỏi đặc biệt là trường hợp thường gặp nhất, yêu cầu tìm span ví dụ như vị trí, ngày hoặc tên entity trong các đoạn văn bản. Sử dụng FCN 2 lớp (fully connected network) để thực hiện quá trình dự đoán cho $\mathcal{F}$:

$$answer = \underset{answer\ node\ x}{\arg max}\ \mathcal{F}(\mathbf{X}[x]) \quad (5)$$

Câu hỏi thay thế và câu hỏi chung đều nhằm mục đích so sánh một thuộc tính nhất định của entity x và entity y trong bộ dữ liệu HotpotQA , tương ứng trả lời với tên trực thể và "yes hoặc no". Những câu hỏi này có thể xem

là phân loại nhị phân với đầu vào là **X**[x] - **X**[y] và được giải quyết bằng hai FCN giống nhau khác.

6 Huấn luyện (Training)

Việc huấn luyện CogQA framework không giống như những mô hình thông thường (việc lan truyền và truyền ngược để cập nhật tham số cùng chung một computation graph), bởi vì framework là sự kết hợp của nhiều mô hình khác nhau với những đầu vào khác nhau. Tuy nhiên thì nhìn chung, mô hình được huấn luyện dưới một **mô hình được giám sát (supervised paradigm)** với **mẫu âm (negative sampling)**. Trong bộ training, span của next-hop và câu trả lời được trích xuất trước (pre-extracted) trong đoạn văn bản, việc trích xuất trước để cho việc huấn luyện được nhanh chóng và chính xác hơn nhờ các context có liên quan ít nhiều đến câu trả lời. Sau đó, cognitve graph được xây dựng cùng với các node âm, và việc truy xuất dữ liệu dựa vào cơ sở dữ liệu của Wiki, tạo ra một **bộ training mới (refined)** phù hợp với mô hình. Đầu vào cho việc huấn luyện riêng cho từng mô hình cho các tác vụ khác nhau sẽ được trích xuất từ cognitive graph trong bộ training. Chính xác hơn thì cho mỗi *para[x]* liên quan đến câu hỏi **Q**, chúng ta sẽ có span là:

$$\mathcal{D}[x, Q] = (y_1, start_1, end_1), \ldots, (y_n, start_n, end_n)$$

Với span bắt đầu từ $start_i$ đến end_i trong *para[x]* được so khớp mờ (fuzzy matching) với tên của một entity hoặc câu trả lời y_i.

6.1 Trích xuất span.

Dữ liệu thực tế của $P_{ans}^{start}, P_{ans}^{end}, P_{hop}^{start}, P_{hop}^{end}$ được xây dựng dựa trên $\mathcal{D}[x, Q]$. Có nhiều nhất một span câu trả lời (y, start, end) trong mỗi đoạn văn bản, do đó $\mathbf{gt}_{end}^{start}$ (ground-truth) là một vector one-hot với $\mathbf{gt}_{end}^{start}[start] = 1/k$ với k là số lượng của next-hop span.

Để tăng khả năng phân biệt các đoạn văn bản không liên quan, thì các hop node âm không liên quan được thêm vào **G**. Được nhắc đến ở phía trên, đầu ra của *[CLS]*, T_0, được dùng để sinh ra *ngưỡng âm (negative threshold)*.

Và loss function được sử dụng để huấn luyện **System 1** trong tác vụ trích xuất span đó là Cross entropy loss.

$$\mathcal{L}_{ans}^{start} = -\sum_i gt_{ans}^{start}[i] \cdot logP_{ans}^{start}[i] (6)$$

Với các loss của $E_{hop}, S_{hop}, E_{hop}$ tương tự với công thức trên. Sau đó, loss cuối cùng được tổng hợp bằng cách lấy trung bình từng loss của hop và answer loss, và cộng chúng lại với nhau.

6.2 Dự đoán node câu trả lời.

Để có thể điều khiển được khả năng luận lý, thì model phải học được cách xác định node câu trả lời một cách chính xác từ cognitive graph. Cho mỗi câu hỏi trong bộ train, xây dựng một mẫu huấn luyện cho tác vụ này. Mỗi mẫu huấn luyện là một thành phần của đồ thị gold-only (là kết hợp của tất cả các đường luận lý chính xác và các node âm). Node âm bao gồm hop node âm dùng trong Tác vụ #1 và hai node câu trả lời âm khác. Một node trả lời âm được xây dựng từ việc trích xuất span ngẫu nhiên tại một hop node ngẫu nhiên nào đó.

Để thực hiện bước lan truyền, thì **System 2** vẫn phải thông qua **System 1** nhằm trích xuất ra được các vector ngữ nghĩa.

Với câu hỏi đặc biệt, tính toán *xác xuất* của câu trả lời cuối cùng cho mỗi node bằng cách sử dụng *softmax* cho đầu ra của $\mathcal{F}$. Loss $\mathcal{L}$ được định nghĩa là cross entropy giữa các xác suất và one-hot vector của node câu trả lời ans.

$$\mathcal{L} = -log(softmax(\mathcal{F}(\mathbf{X}))[ans]) (7)$$

Câu hỏi thay thế và chung được tối ưu bằng binary cross entropy với cách tương tự. Loss cuối cùng được tính bằng cách tính tổng của answer, hop loss và loss của một trong các loss của từng loại câu hỏi nhân với hệ số $\boldsymbol{\alpha}$ (là hệ số cân bằng của 2 system). Losses của tác vụ này không chỉ back-propagation (lan truyền ngược) để tối ưu dự đoán của **System 2**, mà còn fine-tune cho cả **System 1** thông qua vector ngữ nghĩa *sem[x,Q,clues]*

7 Chi tiết thử nghiệm (Experimental Details)

Vì gặp khó khăn trong việc huấn luyện mô hình, cụ thể hơn là việc phần cứng của nhóm không đủ, cũng như giới hạn về thời gian. Thì nhóm đi đến quyết định sẽ sử dụng 20.000 mẫu dữ liệu để huấn luyện mô hình, thay vì sử dụng hơn 90.000 mẫu trong bộ HotpotQA. Ngoài ra như đã đề cập ở phần Bộ dữ liệu HotpotQA (HotpotQA Dataset), thì tập test của sẽ không bao gồm câu trả lời, nên nhóm tạm thời đi đến quyết định sử dụng tập Dev để đánh giá thay cho tập test.

Sử dụng mô hình pre-trained BERT-base cho **System 1**, với hidden size H là 768, không có sự thay đổi trong node vector của GNN và các mô hình dự đoán. Tất cả các hàm kích hoạt trong thử nghiệm đều là *gelu* (Hendrycks and Gimpel, 2016)[6]. Huấn luyện các mô hình trong Tác vụ #1 cho 1 epoch và sau đó kết hợp huấn luyện Tác vụ #1 và Tác vụ #2 cho 1 epoch. Các siêu tham số trong lúc huấn luyện được thể hiện ở bảng dưới:

Mô hình	Tác vụ	batch size	learning rate	weight decay
BERT	#1,#2	10	$10^{-4}, 4 \times 10^{-5}$	0.01
GNN	#2	graph	10^{-4}	0

Bảng 1 Kết quả thử nghiệm

BERT và GNN được tối ưu hóa bằng 2 bộ tối ưu Adam khác nhau, với $\beta_1 = 0.9, \beta_2 = 0.999$. Mô hình dự đoán cùng chia sẻ chung bộ tối ưu hóa với GNN. Learning rate (tốc độ học) cho các tham số trong BERT sẽ được giữ nguyên trong 10% số bước lặp đầu tiên, sau đó thì sẽ giảm dần tuyến tính về 0.

Để chọn ra được các supporting fact, thì ta xem như lấy các câu trong clues của bất kì node nào trong cognitive graph, là các supporting fact. Trong lúc khởi tạo đồ thị **G**, những span của 1-hop (những hop đầu tiên) đã tồn tại trong câu hỏi và có thể tìm thấy bằng fuzzy matching với các supporting facts trong bộ train. Việc trích xuất 1-hop entity bằng CogQA framework có thể cải thiện pha truy xuất của các mô hình khác.

8 Kết quả (Results)

Việc đánh giá cho câu trả lời và supporting fact bao gồm hai thang đo: Extract Match (EM) và F_1 score. EM chung đạt giá trị 1 chỉ khi chuỗi câu trả lời và supporting fact phải cùng giống với mẫu thực tế. Độ chính xác và độ phủ chung được tính toán dựa trên Ans (Answer) và Sup (supporting facts), và sau đó độ đo F_1 được tính toán dựa trên độ phủ và độ chính xác. Các kết quả của các thang đo được lấy trung bình trên bộ test (với Dev là kết quả của toàn bộ dữ liệu gốc và Test là 20000 mẫu dữ liệu từ dữ liệu gốc).

Data	Ans				Sup				Joint			
	EM	F_1	Prec	Recall	EM	F_1	Prec	Recall	EM	F_1	Prec	Recall
Dev	37.6	49.4	52.2	49.9	23.1	58.5	64.3	59.7	12.2	35.3	40.3	36.5
Test	27.8	38.6	41.1	39.3	15.4	52.9	54.3	59.5	6.3	25.6	27.8	29.0

Bảng 2 Kết quả tổng thể

Kết quả tổng thể : CogQA framework cho ra kết quả khá tốt, nếu so sánh với các phương pháp retrieval-extraction khác. Bởi vì các phương pháp đó khi lấy các đoạn văn bản được multi-hop ra xa câu hỏi sẽ càng có ít các từ chung, thậm chí là ít có liên quan về ngữ nghĩa với câu hỏi hơn, từ đó thất bại trong việc tìm đoạn văn bản chính xác cho câu trả lời. Tuy nhiên với CogQA, thì việc tìm kiếm được các entity có liên quan sẽ chính xác hơn bằng cách sử dụng *clues*.

So với bài báo gốc được huấn luyện trên toàn bộ dữ liệu thì có thể thấy rằng hiệu suất bị sụt giảm đi khoảng 10% đến 20%. Tuy nhiên thì đây vẫn là một con số có thể chấp nhận được khi mà số lượng dữ liệu bị giảm hơn 4 lần.

9 Tổng kết (Conclusion)

Chúng em đã thực hiện lại CogQA framework để xử lí tác vụ multi-hop machine reading trong cơ sở dữ liệu Wiki tiếng Anh[7]. Quá trình luận lý được tổ chức thành đồ thị nhận thức (cognitive graph), nhằm cố gắng thực hiện lại quá trình luận lý của con người khi tìm kiếm câu trả lời cho một câu hỏi và đưa ra được giải thích cho câu trả lời đó. Quá trình thực hiện dựa trên BERT

và GNN cho bộ dữ liệu HotpotQA. Nhờ ưu thế của cấu trúc luận lý rõ ràng trong cognitive graph, **System 2** có khả năng tận dụng các kỹ thuật logic thần kinh để cải thiện độ tin cậy của hệ thống

Tuy nhiên thì với đề tài này thì chúng em chỉ sử dụng Baseline của CogQA framework, bao gồm cả hệ thống truy xuất được sử dụng (fuzzy matching). Phương hướng phát triển sắp tới thì chúng em dự định sẽ áp dụng một hệ thống truy xuất mới, thử nghiệm các siêu tham số mới cũng như xây dựng lại framework cho bộ dữ liệu tiếng Việt, nhằm đáp ứng nhu cầu sử dụng cũng như nghiên cứu trong tương lai.

Tài liệu

[1] Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051 (2017)

[2] Evans, J.S.B.: Heuristic and analytic processes in reasoning. British Journal of Psychology **75**(4), 451–468 (1984)

[3] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[4] Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)

[5] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

[6] Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)

[7] Ding, M., Zhou, C., Chen, Q., Yang, H., Tang, J.: Cognitive graph for multi-hop reading comprehension at scale. arXiv preprint arXiv:1905.05460 (2019)