ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC

BÁO CÁO CUỐI KỲ

Xây Dựng Đồ Thị Ngữ Nghĩa Trừu Tượng (AMR) Cho Tiếng Việt

Môn học: Nhập Môn Xử Lý Ngôn Ngữ Tự Nhiên

Sinh viên thực hiện:

Bùi Hồng Phúc 22120270

Nguyễn Lê Anh Phúc 22120276

Giáo viên hướng dẫn:

PGS. TS. ĐINH ĐIỀN

TS. NGUYỄN HỒNG BỬU LONG

Ngày 25 tháng 12 năm 2024

Mục lục

1 Giới Thiệu

1.1 Giới thiệu chung

Trong vài thập kỷ qua, sự phát triển mạnh mẽ của lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) đã thúc đẩy các nghiên cứu tập trung vào việc biểu diễn ý nghĩa của ngôn ngữ dưới dạng hình thức hóa. Một trong những mô hình biểu diễn ý nghĩa của ngôn ngữ dưới dạng hình thức hóa. Một trong Những mô hình biểu diễn ý nghĩa ngữ nghĩa ngữ nghĩa phổ biến và thành công nhất là Abstract Meaning Representation (AMR), được giới thiệu bởi Banarescu et al. (2013). AMR biểu diễn ý nghĩa của câu dưới dạn một đồ thị có gốc, được gắn nhãn và có hướng, trong đó:

- `Nút (node)`: Đại diện cho các thực thể, khái niệm, hoặc sự kiện.
- `Cạnh (edge)`: Đại diện cho các mối quan hệ giữa các thực thể.

Khác với các mô hình biểu diễn khác, AMR tập trung hoàn toàn vào ý nghĩa, loại bỏ các yếu tố cú pháp, giúp các câu cùng ý nghĩa nhưng khác cấu trúc sẽ có cùng một đồ thị. Điều này làm cho AMR trở thành một công cụ mạnh mẽ trong các bài toán NLP như dịch máy, tóm tắt văn bản, trích xuất thông tin và phân tích cảm xúc.
Hiện tại, AMR đã được ứng dụng rộng rãi trên nhiều ngôn ngữ như Tiếng Anh, Trung Quốc, Đức, Tây Ban Nha và Hàn Quốc. Tuy nhiên, đối với tiếng Việt - một ngôn ngữ giàu ngữ cảnh và cú pháp phức tạp, việc xây dựng tập dữ liệu và mô hình AMR vẫn còn nhiều thách thức.

1.2 Thách thức trong xử lý ngôn ngữ tiếng Việt

Tiếng Việt là ngôn ngữ đơn âm tiết, không có hình thái của chia động từ, và phụ thuộc nhiều vào ngữ cảnh. Một số đặc trưng nổi bật của tiếng việt bao gồm:

- `Từ đồng âm khác nghĩa`: ví dụ, từ "bàn" có thể là danh từ "cái bàn" hoặc động từ "bàn luận".
- `Ngữ pháp tự do`: Thứ tự từ trong câu có thể thay đổi mà không làm mất ý nghĩa.
- `Từ ghép và tổ hợp`: Các từ có thể được tạo ra bằng cách ghép hoặc biến đổi cấu trúc từ gốc. Những đặc điểm này đặt ra bài toán khó cho việc chuẩn hóa dữ liệu và trích xuất các đặc trưng ngữ nghĩa khi xây dựng AMR cho tiếng việt.

1.3 Mục tiêu

Đề tài này nhằm mục đích xây dựng một hệ thóng tự động sinh đồ thị AMR cho tiếng việt, bao gồm các bước:

1. `Bước 1:Tạo tập dữ liệu AMR`: dịch tập AMR tiếng Anh(**AMR (Abstract Meaning Representation) release v3.0**) sang tiếng Việt.
2. `Bước 2: Huấn luyện AMR cho tập AMR tiếng Việt đã được dịch ở Bước 1`: sử dụng mô hình **/mbart-large-50** được thiết kế để hỗ trợ 50 ngôn ngữ.
3. `Bước 3: Tạo tập ngữ liệu chưa gán nhãn cho tiếng Việt`: có thể tìm những bài văn, những câu truyện,..
4. `Bước 4: Dùng mô hình đã được huấn luyện ở bước 2 và ChatGPT để gán nhãn cho Bước 3`.
5. `Bước 5: Kiểm tra lại 4 thủ công`
6. `Bước 6: Đánh giá`: so sánh các đồ thị AMR sinh ra với đồ thị AMR chuẩn để đánh giá độ chính xác và so sánh các **loss** trên tập **train(80%), test(20%)**

1.4 Ý nghĩa của đồ án

Việc xây dựng một hệ thống AMR tự động cho tiếng Việt có tiềm năng lớn trong việc:

- Cải thiện chất lượng các bài toán NLP liên quan đến tiếng Việt như dịch máy, tóm tắt văn bản và truy xuất thông tin.
- Tạo ra nguồn tài nguyên dữ liệu nền tảng cho các nghiên cứu trong lĩnh vực xử lý ngôn ngữ tiếng Việt.
- Góp phần thu hẹp khoảng cách giữa tiếng Việt và các ngôn ngữ có nhiều tài nguyên (high-resource languages) như tiếng Anh.

2 Dữ liệu và tiền xử lý

2.1 Tập Dữ Liệu

Tập dữ liệu được sử dụng trong nghiên cứu là một tập `Abstract Meaning Representation (AMR)`, được dịch từ bộ dữ liệu tiếng Anh sang tiếng Việt. Cụ thể:

- `Nguồn gốc`: Tập dữ liệu gốc là AMR corpus, chứa các cặp câu và đồ thị AMR, lấy từ cuốn tiểu thuyết *The Little Prince* (Hoàng Tử Bé). Từ đây, các câu trong tập AMR được dịch thủ công sang tiếng Việt.
- `Mục tiêu`: Tạo một tập dữ liệu AMR song ngữ Việt - Anh với cấu trúc đồ thị được giữa nguyên từ bản tiếng Anh, giúp mô hình có thể sinh đồ thị AMR từ câu tiếng Việt:
- Tổng số lượng mẫu dữ liệu: 1572 mẫu, mỗi mẫu bao gồm:
  - `Câu (sentence)`: Là câu tiếng Việt dịch từ câu gốc tiếng Anh.
  - `Đồ thị AMR (target)`: Là độ thị biểu diễn ngữ nghĩa của câu dưới dạng chuỗi ký tự.
- Dữ liệu được chia thành 2 tập: **Training(80%) và Testing(20%)**

`Cấu trúc dữ liệu mẫu`

Dữ liệu được lưu trong file *amr-bank-struct-vietnamese.txt* với cấu trúc cụ thể như sau:

```
# ::id DF-170-181103-888_4526.3 ::amr-annotator SDL-AMR-09 ::preferred
# ::snt Một lần khi tôi sáu tuổi, tôi đã nhìn thấy một bức tranh tuyệt vời trong một cuốn s
(s / xem-01
      :ARG0 (t / tôi)
      :ARG1 (p / hình-ảnh
            :mod (t2 / tráng-lệ)
            :location (b / sách)))
```

`Trong đó:`

- Dòng `# ::snt`: chứa câu tiếng Việt.
- Phần sau `# ::snt`: chứa đồ thị AMR..

2.1.1 Ý nghĩa đồ thị AMR

AMR biểu diễn ý nghĩa câu bằng cách tổ chức các thành phần ngữ nghĩa dưới dạng đồ thị:

- Các nút (node): đại diện cho khái niệm, thực thể, hoặc sự kiện. Ví dụ: xem-01 (xem), tôi (người xem).
- Các cạnh (edge): biểu diễn quan hệ giữa các nút. Ví dụ: `ARG0` (chủ thể của hành động), `ARG1` (đối tượng của hành động).

2.2 Tiền Xử Lý

Để chuẩn bị dữ liệu đưa vào mô hình, quy trình tiền xử lý được thực hiện qua các bước sau:

Bước 1: Tách câu và đồ thị

Do dữ liệu ban đầu chứa cả câu và đồ thị, cần phải tách riêng câu tiếng Việt và đồ thị AMR để dễ dàng sử dụng trong huấn luyện. Quy trình được thực hiện thông qua hàm `load_file()`

- Đọc từng đoạn dữ liệu (`block`) trong file.
- Loại bỏ các dòng không cần thiết, chỉ giữa lại:
  - **Câu tiếng Việt**: Các dòng bắt đầu bằng `# ::snt`.
  - **Đồ thị AMR**: Các dòng không bắt đầu bằng `#`
- Tạo hai danh sách:
  - **sentences**: Chứa các câu tiếng Việt.
  - **amrs**: Chứa đồ thị AMR đã chuẩn hóa.

Mã Nguồn:

```
def load_file(file_path):
    with open(file_path, "r", encoding="utf-8") as f:
        data = f.read()
    amr_blocks = data.strip().split("\n\n")
    sentences = []
    amrs = []

```

```
    for block in amr_blocks:
        lines = block.strip().split("\n")
        sentence = ""
        amr = ""

        for line in lines:
            if line.startswith("# ::snt"):
                sentence = line.replace("# ::snt", "").strip()
            elif not line.startswith("#"):
                amr += line.strip() + " "
        if sentence and amr:
            sentences.append(sentence)
            amrs.append(amr.strip())
    return {"input_text": sentences, "target_text": amrs}
```

Thông qua đoạn hàm trên ta sẽ thua được 1572 corpus. ví dụ mẫu dữ liệu sẽ có dạng:

`Câu: "Một lần khi tôi sáu tuổi, tôi đã nhìn thấy một bức tranh tuyệt vời."`

`AMR: "(s / xem-01 :ARG0 (t / tôi) :ARG1 (p / hình-ảnh :mod (t2 / tráng-lệ)))"`

Bước 2: Chuẩn hóa đồ thị AMR

Đồ thị AMR cần được chuẩn hóa thành chuỗi ký tự một dòng (linearized AMR) để phù hợp với quá trình huấn luyện mô hình. Điều này được thực hiện qua hàm `linearize_amr()`:

- Loại bỏ các ký tự thừa, các khoảng trắng không cần thiết.
- Đưa đồ thị AMR về dạng chuỗi ký tự.

Mã Nguồn:

```
def linearize_amr(amr):
    amr = "\n".join(line for line in amr.splitlines() if not line.startswith("#"
    ))
    amr = re.sub(r"\s+", " ", amr)
    return amr.strip()
```

Thông qua hàm này ta thu được đồ thị đã được chuẩn hóa:

Trước khi chuẩn hóa:

```
(s / xem-01
      :ARG0 (t / tôi)
      :ARG1 (p / hình-ảnh
            :mod (t2 / tráng-lệ)))
```

Sau khi chuẩn hóa:

```
"(s / xem-01 :ARG0 (t / tôi) :ARG1 (p / hình-ảnh :mod (t2 / tráng-lệ)))"
```

Bước 3: Token hóa dữ liệu

Để đưa dữ liệu vào model `mbart-large-50` cần token hóa câu và đồ thị AMR:

- **Câu tiếng Việt:** Token hóa bằng `MBart50Tokenizer`, với độ dài tối đa 256 token.
- **Đồ thị AMR:** Token hóa tương tự, với độ dài tối đa 256 token.

Mã Nguồn:

```python
def preprocess_function(examples):
    inputs = examples['sentence']
    targets = examples['amr']

    model_inputs = tokenizer(
        inputs,
        max_length=256,
        padding="max_length",
        truncation=True,
        text_target=targets
    )

    return model_inputs
```

Bước 4: Thêm các từ trong *additions.txt* và *predicates.txt* vào tokenizer của mbart-large-50

Mã Nguồn:

```
file_add = "additions.txt"

with open(file_add,"r",encoding = "utf-8") as f:
  additions = [line.strip() for line in f.readlines() if line.strip()]


file_predicates  = "predicates.txt"

with open(file_predicates,"r",encoding = "utf-8") as f:
  predicates = [predicates.strip() for predicates in f.readlines() if predicates
   .strip()]

tokenizer.add_tokens(additions)
tokenizer.add_tokens(predicates)
```

additions.txt: Các thành phần trong file như `:ARG0, :ARG1, :mod, have-purpose-91` là nhãn hoặc ký hiệu kỹ thuật. Những ký hiệu này được sử dụng để biểu diễn quan hệ ngữ nghĩa hoặc cấu trúc cú pháp, chứ không phải từ vựng tự nhiên.

predicates.txt: chứa danh sách các predicate, tức là các từ hoặc cụm từ biểu thị hành động, trạng thái, hoặc quan hệ. Được dịch từ tiếng Anh sang tiếng Việt.

2.3 Kết Quả Tiền Xử Lý

1. Phân chia dữ liệu:
   - Tập huấn luyện (Train): 80% dữ liệu.
   - Tập kiểm tra (Test): 20% dữ liệu.
2. Ví dụ dữ liệu sau token hóa:
   - Input IDS (tokenized sentence):

```
[0, 3242, 34, 54, 1382, 262, 372, 783, 457, 2, 0, ...]

```

- Labels (tokenizer AMR):

```
[0, 3452, 15, 234, 400, 1231, 92, 543, 2, 0, ...]
```

3. Đánh giá chất lượng tiền xử lý:
   - Câu và đồ thị được chuẩn hóa và token hóa về đúng định dạng.
   - Giảm thiểu lỗi cú pháp trong quá trình xử lý dữ liệu thô.

3 Mô hình

3.1 Giới thiệu chung

MBart50 từ thư viện transformers của Hugging Face để tạo đồ thị AMR (Abstract Meaning Representation) từ các câu tiếng Việt,hỗ trợ nhiều ngôn ngữ và kiến trúc seq2seq (sequence-to-sequence). Trong phần này, mBART-50 được tinh chỉnh (fine-tune) để chuyển đổi giữa biểu diễn ngữ nghĩa AMR (Abstract Meaning Representation) và câu tiếng Việt.

3.2 Các bước thực hiện

1. **Khởi tạo mô hình**:
   - Sử dụng `MBart50Tokenizer` và `MBartForConditionalGeneration` từ thư viện `transformers` của Hugging Face.
   - Tải mô hình với checkpoint `facebook/mbart-large-50`.
   - Thêm các token từ hai file `additions.txt` và `predicates.txt` để mở rộng từ vựng của tokenizer.
2. **Chuẩn bị dữ liệu**:
   - Dữ liệu từ tệp `amr-bank-struct-vietnamese.txt` được xử lý để tạo cặp câu gốc và biểu diễn AMR (AMR graph) tương ứng.
   - Sử dụng hàm `linearize_amr` để làm phẳng đồ thị AMR, giúp chuẩn bị dữ liệu cho quá trình huấn luyện.

- Tách dữ liệu thành hai tập:
  - Tập huấn luyện: 80% dữ liệu.
  - Tập kiểm tra: 20% dữ liệu.

3. **Tinh chỉnh mô hình**:
   - Các tham số huấn luyện:
     - `Batch size`: 4 (cho cả huấn luyện và kiểm tra).
     - Số epoch: 3.
     - Learning rate: 5×10^{-5}.
4. **Lưu trữ mô hình đã tinh chỉnh:**
   Để giảm kích thước phần nộp bài chúng em lựa chọn lưu trữ trên huggingface
   - Sau khi lưu lại mô hình với tên **amr_new_model** chúng em lựa chọn đưa mô hình này lên huggingface (Link mô hình).
   - Sau khi đã lưu trữ ta cần gọi model đã được fine-tune trước đó để có thể sử dụng mà không cần phải train lại.Khi đó chỉ cần chạy mục 5 trong file **Code.ipynb**.
5. **Sinh AMR từ câu không nhãn**:
   - Sử dụng mô hình đã tinh chỉnh để sinh đồ thị AMR từ các câu tiếng Việt không nhãn.
   - Đồ thị AMR được định dạng lại bằng hàm `format_amr_tree` để dễ đọc.
6. **Lưu kết quả**:
   - Kết quả AMR của từng câu được lưu trong tệp `all_amrs.txt` theo định dạng AMR.

4 Kết Quả Và Đánh Giá

4.1 Kết Quả Huấn Luyện

- Quá trình huấn luyện được thực hiện trong 3 epoch.
- Giá trị **Training Loss** giảm đều qua từng epoch, từ 0.2640 ở epoch đầu tiên xuống 0.1055 ở epoch cuối cùng.

- Giá trị **Validation Loss** giảm liên tục qua các epoch, từ 0.234631 ở epoch đầu tiên xuống 0.192080 ở epoch cuối cùng.

Epoch	Training Loss	Validation Loss
1	0.264000	0.234631
2	0.175200	0.203502
3	0.105500	0.192080

Bảng 1: Training and Validation Loss for Each Epoch

4.2 Phân tích kết quả

- **Training Loss** giảm liên tục cho thấy mô hình đã học tốt từ tập huấn luyện.
- **Validation Loss** giảm đều qua từng epoch, không có dấu hiệu tăng trở lại, cho thấy mô hình không bị overfitting trong quá trình huấn luyện. Điều này thể hiện khả năng tổng quát hóa tốt của mô hình trên tập dữ liệu kiểm thử.

4.3 Đánh giá chất lượng gán nhãn

Khi thông qua mô hình ta thu được một cấu trúc AMR ta nhận thấy:

- Mô hình còn một vài vấn đề với những tên riêng.
- Còn xuất hiện một vài lỗi chính tả. Ví dụ: cánh-r ừng,..Do mô hình chưa học tốt.

Trong quá trình sử dụng ChatGPT để gán nhãn và sửa những tập AMR được tạo ra từ mô hình fine-tune.

- **Ưu điểm:**
  - Tăng tốc quá trình gán nhãn, giảm đáng kể thời gian so với gán nhãn thủ công.
  - ChatGPT có khả năng tạo nhãn phù hợp dựa trên ngữ cảnh của câu.
- **Hạn chế:**
  - Một số nhãn có thể không hoàn toàn chính xác.
  - Yêu cầu kiểm tra thủ công để đảm bảo tính chính xác.

4.4 Kết luận

- Mô hình đạt hiệu suất tốt trong quá trình huấn luyện với cả **Training Loss** và **Validation Loss** giảm đều đặn qua từng epoch.
- Không có dấu hiệu của overfitting, Validation Loss giảm liên tục và đạt giá trị tốt nhất ở epoch cuối cùng (0.192080).
- Chất lượng nhãn tự động từ ChatGPT được đánh giá là khá tốt nhưng cần kiểm tra thủ công để cải thiện độ chính xác.

5 Tổng kết

- Dịch tập dữ liệu AMR tiếng Anh sang tiếng Việt.
- Huấn luyện mô hình AMR tiếng Việt.
- Gán nhãn dữ liệu nhờ mô hình đã huấn luyện, ChatGPT và kiểm tra thủ công để đảm bảo chất lượng.
- Xây dựng đồ thị ngữ nghĩa.

6 Sử dụng tài liệu tham khảo

Tài liệu PAMR: Persian Abstract Meaning RepresentationCorpus[2]

Tài liệu ACL Anthology: 2024.cl-2.1[3]

Tài liệu ACL Anthology: W19-3317[1]

Tài liệu

[1] Hà Mỹ Linh and Nguyễn Thị Minh Huyền. A case study on meaning representation for vietnamese, 2019. Accessed: 2024-12-25.

[2] Author(s) Name. Pamr: Persian abstract meaning representation corpus, 2024. Accessed: 2024-12-25.

[3] Shira Wein and Nathan Schneider. Assessing the cross-linguistic utility of abstract meaning representation, 2024. Accessed: 2024-12-25.