BOOK SUMMARIZATION PROBLEM (đợt 1- Unsupervised methods)

*_Nguyễn Việt Dũng_*

Link Github project: https://github.com/toilanvd/Book-SUMMER

I, Bài toán:

Tóm tắt văn bản dài (sách, chương, đoạn văn) theo kiểu trích rút (extractive: chọn một số câu trong văn bản gốc để ghép lại thành tóm tắt).

1, Ngôn ngữ: tiếng Anh

2, Dữ liệu: *BookSum* (https://github.com/salesforce/booksum)

Link download: https://huggingface.co/datasets/kmfoda/booksum (không chính thức, mới chỉ có dữ liệu ở mức chương).

- Kích thước:

+ Train set: 9600 sample

+ Validation set: 1484 sample

+ Test set: 1431 sample

- Loại tóm tắt mẫu: tóm lược (abstractive).

3, Cách đánh giá độ tốt:

- ROUGE-1: dựa vào 1-gram trùng lặp giữa tóm tắt sinh ra và tóm tắt mẫu.
- ROUGE-2: dựa vào 2-gram trùng lặp giữa tóm tắt sinh ra và tóm tắt mẫu.
- ROUGE-L: dựa vào dãy con chung dài nhất giữa tt sinh ra và tt mẫu.
- BertScore: GitHub - Tiiiger/bert_score: BERT score for text generation

Công thức tính điểm (đặt tham số rescale_with_baseline = True):

$$BertScoreF1(can) = avg_i(max_j BertScoreF1(ref\_sent_i, can\_sent_j))$$

Trong đó ref_sent_i là câu văn thứ i trong bản tóm tắt mẫu, can_sent_j là câu văn thứ j trong bản tóm tắt được mô hình sinh ra.

- SummaQA: GitHub - ThomasScialom/summa-qa

Công thức tính điểm:

$$SummaQAF1(can) = avg_i(max_j SummaQAF1(chap\_sent_i, can\_sent_j))$$

Trong đó $chap\_sent_i$ là câu văn thứ i trong văn bản gốc, can_sent_j là câu văn thứ j trong bản tóm tắt được mô hình sinh ra.

II, Cách cài đặt:

1, Cài đặt mô hình nói chung:

- Mô hình được viết trong các file notebook và được chia làm các phần: tải dữ liệu, chia tách câu, các hàm tính toán, chạy training set (sinh tóm tắt, tính điểm ROUGE, BertScore và SummaQA), chạy validation set, chạy test set. Các mô hình có hyperparameter α sẽ có thêm một bước duyệt chọn α tốt nhất dựa trên validation set, trước khi chính thức khởi chạy training - validation - test set. Khi test, có thể chạy trực tiếp test set mà không cần phải chạy qua training set và validation set.

- Một số mô hình sẽ in dữ liệu ra file và gọi chương trình C++ để chạy các thao tác mất nhiều thời gian trên Python. Cần phải biên dịch các file code C++ này ra file chạy trước rồi mới chạy được notebook. Câu lệnh biên dịch 1 file C++ ra file chạy trong terminal Linux có cú pháp như sau:

g++ -O2 -o tên-file-C++ tên-file-C++.cpp

VD: g++ -O2 -o MCS-One-and-Two-Words MCS-One-and-Two-Words.cpp

2, Cài đặt SummaQA:

Trước khi cài đặt package SummaQA bằng pip, phải bỏ 2 dòng lệnh thứ 19 và 20 trong file summaqa/setup.py vì lý do conflict dependency. Hai dòng đó là:

19: 'en_core_web_sm @ https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.0/en_core_web_sm-2.2.0.tar.gz',

20: 'transformers==4.30.0',

3, Lưu ý về độ dài input cho BertScore model và SummaQA model:

Hai mô hình này chỉ nhận input dài không quá 512 từ, vì thế ta mới phải tách câu để tính BertScoreF1 và SummaQAF1 như trong phần I-3. Trong paper BookSum các tác giả cũng làm như vậy.

?- Nếu không thêm tham số rescale_with_baseline = True cho BertScore model thì nó sẽ cho ra kết quả điểm rất lớn, không như trong paper BookSum. Tại sao lại thế?

III, Các mô hình đã thử nghiệm:

Các mô hình được cài đặt trên Linux, chạy bằng CPU (chưa thử GPU).

1, Hệ mô hình sử dụng ma trận điểm số giữa các câu văn:

**** Cấu trúc chung của các mô hình này gồm 3 bước:***

- Bước 1: Tách chương thành N câu văn.
- Bước 2: Sinh một ma trận điểm số giữa các cặp câu văn, kích thước N x N. Điểm số này có thể là:

+ BERT + cosine similarity: độ tương đồng ngữ nghĩa.

+ average ROUGE(1, 2, L): độ tương đồng từ vựng.

+ Next sentence prediction (NSP - classification 0/1): khả năng câu B có thể là câu đứng ngay sau câu A, dựa vào giả định rằng nếu câu A đứng trước câu B thì khả năng cao sự kiện trong câu A là nguyên nhân dẫn đến sự kiện trong câu B. Có 2 biến thể khác của dạng điểm số này là "khả năng câu A đứng trước câu B trong một đoạn văn" (ASP) và "khả năng câu A và câu B nằm cùng đoạn văn" (SPSP).

- Bước 3: Xử lý ma trận N x N ở bước 2 để chọn ra các câu văn phù hợp nhất tạo thành tóm tắt trích rút.

**** Các mô hình đã thử nghiệm:***

a, Greedy-BERT:

- Ở bước 2, ta xây dựng ma trận BERT + cosine similarity.
- Còn ở bước 3, ta tham lam chọn dần các câu vào bản tóm tắt, ưu tiên câu nào có độ tương đồng trung bình với các câu khác (chính là trung bình các số trên cùng hàng của ma trận) là lớn nhất thì chọn trước. Chọn cho đến khi tóm tắt đạt độ dài nhất định (khoảng 15% độ dài văn bản gốc, con số % này tính được từ thống kê train set) thì dừng.

b, TextRank-BERT:

- Ở bước 2, ta xây dựng ma trận BERT + cosine similarity.
- Ở bước 3, ta dùng thuật toán TextRank. Thuật toán TextRank này (https://aclanthology.org/W04-3252.pdf) dựa trên tư tưởng của thuật toán PageRank (https://en.wikipedia.org/wiki/PageRank). Thuật toán PageRank mô phỏng lại việc một người lướt web, trong đó có hai khả năng là người đó sẽ click vào một liên kết có trong trang web hiện tại (với xác suất d), hoặc tự nhập liên kết của một trang web bất kỳ (với xác suất 1-d). Khi cho trước xác suất các khả năng click chuột khi người dùng đang ở một trang web, thuật toán PageRank sẽ cho biết xác suất để người lướt web dừng lại ở một trang web nào đó vào thời điểm họ dừng

việc lướt web, giả định rằng người lướt web lướt đủ lâu. Trang web nào có xác suất dừng lại cao hơn thì được đánh giá là quan trọng hơn, bởi người lướt web khả năng cao sẽ truy cập trang web đó trong quá trình lướt. Từ đó thuật toán TextRank cũng làm tương tự PageRank nhưng với đối tượng là các câu văn, nhằm xác định xem câu văn nào là quan trọng trong văn bản.

Khi đã có xác suất TextRank cho từng câu văn, ta chọn các câu theo thứ tự xác suất nhỏ dần để đưa vào bản tóm tắt.

c, MCS-BERT:

- Ở bước 2, ta xây dựng ma trận BERT + cosine similarity.
- Ở bước 3, ta đặt ra một hyperparameter α (sẽ tune bằng validation set). Gọi ma trận BERT + cosine similarity ở bước 2 là S, thì ta xây dựng một ma trận A cùng kích thước với ma trận S, và gán $A[i, j] = 0$ nếu $S[i, j] < \alpha$, gán $A[i, j] = 1$ nếu $S[i, j] \geq \alpha$. Ma trận có ý nghĩa rằng nếu $A[i, j] = 1$ thì khi đưa câu văn i vào tóm tắt ta cũng sẽ bao hàm được nội dung của câu văn j, đồng nghĩa với việc bao hàm được tất cả các từ trong câu j. Như vậy mục tiêu bây giờ là chọn ra các câu văn làm thành tóm tắt sao cho chúng bao hàm được nhiều từ nhất có thể. Với bài toán mới này, một cách làm dễ nghĩ đến nhất là tham lam chọn từng câu văn một vào tóm tắt, ưu tiên chọn câu bao hàm được nhiều từ nhất hiện tại. Sau khi chọn xong một câu, ta đánh dấu tất cả các câu văn mà nó bao hàm được là "đã được bao hàm", rồi lặp lại quá trình chọn câu cho đến khi tóm tắt đạt độ dài nhất định.

Từ bài toán mới kể trên, ta có thêm 2 mô hình biến thể để giải quyết:

+ **MCS-SA-BERT:** thay vì tham lam chọn dần các câu văn, ta tìm tổ hợp câu văn tốt nhất bằng giải thuật luyện kim (Simulated Annealing - SA). Cách sinh kết quả hàng xóm của kết quả hiện tại s: Thêm 1 câu, xóa 1 câu, đổi 1 câu lấy 1 câu khác chưa chọn. Chọn ngẫu nhiên 1 kết quả trong số K hàng xóm tốt nhất đã được sinh ra từ s để so sánh với s. Xác suất chấp nhận kết quả tệ hơn là T/Tmax. Nhiệt độ giảm theo công thức T = T - Tdec.

+ **MCS-GA-BERT:** thay vì dùng giải thuật luyện kim, ta dùng giải thuật di truyền (Genetic Algorithm - GA). Cách sinh cá thể cho quần thể mới: crossover (với mỗi cá thể trong quần thể cũ, chọn một cá thể trong quần thể cũ với xác suất ưu tiên các cá thể tốt hơn để phối ngẫu. Khi phối ngẫu, ta chọn ngẫu nhiên các câu có trong đáp án cha và mẹ để tạo đáp án con), mutation (thêm 1 câu, xóa 1 câu, đổi

1 câu lấy 1 câu khác chưa chọn). Quần thể mới chỉ giữ lại POPULATION_SIZE cá thể tốt nhất được sinh ra. Nếu sau G_BEST thế hệ mà cá thể tốt nhất không thay đổi và sau G_SUM_PROFIT thế hệ mà tổng độ tốt của quần thể không tăng lên thì ta dừng giải thuật di truyền.

d, SOBP-BERT:

- Ở bước 2, ta xây dựng ma trận BERT + cosine similarity. Gọi ma trận này là S.

- Giả định ở đây là mỗi câu văn được chọn sẽ bao hàm ý nghĩa của một số câu văn xung quanh nó. Với giả định này, thì ở bước 3, bài toán đặt ra là nếu mỗi câu văn i bao hàm được các câu văn từ l_i đến r_i ($l_i \leq i \leq r_i$), ta phải chọn các câu văn làm tóm tắt sao cho chúng bao hàm được nhiều từ nhất. Để có được đoạn bao phủ (l_i, r_i), ta đặt ra một hyperparameter α (sẽ tune bằng validation set). Khi đó:

l_i = câu xa nhất về phía bên trái câu i thỏa mãn $\forall l_i \leq j \leq i$: $S[i, j] \geq \alpha$

r_i = câu xa nhất về phía bên phải câu i thỏa mãn $\forall r_i \geq j \geq i$: $S[i, j] \geq \alpha$

Bài toán bao phủ kể trên sẽ được giải quyết trọn vẹn bằng tối ưu hàm quy hoạch động f[i, c], với i là câu văn đang xét hiện tại, và c là tổng độ dài các câu văn đã được chọn vào tóm tắt. Hàm này tính được trong thời gian đa thức.

+ **SOBP-MA-BERT:** biến thể này khác với SOBP-BERT ở cách tính (l_i, r_i). Ở đây trung bình cộng sẽ được sử dụng nhằm làm mượt, cụ thể là:

l_i = câu xa nhất bên trái câu i mà $\forall l_i \leq j \leq i$: $\frac{\Sigma S[i, j->i]}{i-j+1} \geq \alpha$

r_i = câu xa nhất bên phải câu i mà $\forall r_i \geq j \geq i$: $\frac{\Sigma S[i, i->j]}{j-i+1} \geq \alpha$

e, GA-BERT:

- Ở bước 2, ta xây dựng ma trận BERT + cosine similarity, gọi là S.

- Ở bước 3, sử dụng giải thuật di truyền chọn ra một tổ hợp các câu văn tối ưu hóa hàm mục tiêu SummaryScore sau:

$$SummaryScore = \sum_j max_i(S[i, j])$$

Về các thao tác di truyền, ý tưởng giống với GA trong MCS-GA-BERT.

2, Hệ mô hình sử dụng số lần xuất hiện của 1-gram và 2-gram:

Hệ mô hình này sinh ra dựa vào cách tính điểm ROUGE-1 và ROUGE-2. Ý tưởng ở đây là các 1-gram hoặc 2-gram xuất hiện nhiều lần trong văn bản gốc cũng sẽ có khả năng cao xuất hiện trong văn bản tóm tắt mẫu. Vì thế, độ tốt của một câu văn có thể được tính bằng tổng số lần xuất hiện trong văn bản của các 1-gram và 2-gram được chứa trong câu văn đó (dựa theo tư tưởng của tf-idf, nhưng chỉ dùng tf, chưa dùng idf). Cụ thể, công thức tính độ tốt của câu văn là:

$$Score(sent_i) = \sum_{x \in sent_i} (c(x) - 1) + \sum_{y \in sent_i} 2(c(y) - 1)$$

Trong đó $sent_i$ là câu văn thứ i của văn bản gốc, x là 1-gram có trong $sent_i$, y là 2-gram có trong $sent_i$, c(x) là số lần xuất hiện của x trong văn bản gốc. Có hai kiểu biến thể chính:

a, One-and-Two-Words:

Trực tiếp tham lam chọn dần các câu có Score(sent) lớn nhất vào tóm tắt.

b, MCS-One-and-Two-Words:

Cũng tham lam giống như mô hình One-and-Two-Words, nhưng giả định rằng mỗi một 1-gram hoặc 2-gram được chứa trong nhiều câu văn của bản tóm tắt thì chỉ được tính điểm 1 lần. Vì thế sau khi chọn câu văn i vào tóm tắt, ta phải đặt lại c(x) = 0 với mọi 1-gram và 2-gram x có trong câu văn i.

+ **MCS-GA-One-and-Two-Words:** Thay việc tham lam chọn câu đơn thuần như trên bằng giải thuật di truyền để tối ưu hóa bộ câu văn được chọn. Đây đang là mô hình có kết quả test tốt nhất.

+ **MCS-GA-One-and-Two-Words-noStopword:** Mặc định các unigram và bigram x có c(x) = 0 nếu x có chứa một stopword.

+ Ngoài các kiểu biến thể này, ta có thể lấy công thức tính điểm chia cho idf để có kiểu biến thể thứ ba gồm hai mô hình được chọn là **MCS-GA-UniBigram-tfidf** và **MCS-GA-UniBigram-tfidf-noStopword**.

+ **2profits:** Mỗi một solution sẽ có 2 thông số để đánh giá, đó là tổng điểm Score(sent) với mỗi gram chỉ tính 1 lần (profit) và Score(sent) với mỗi gram được tính nhiều lần không giới hạn (original profit). Ưu tiên profit trước original profit.

3, Extractive Oracle:

Extractive Oracle là mô hình dùng để sinh ra bản tóm tắt trích rút tối ưu, nhằm làm phương tiện đánh giá độ tốt của các mô hình tóm tắt trích rút.

a, Greedy-Extractive Oracle:

Dựa theo tư tưởng của Extractive Oracle trong paper BookSum, tuy nhiên vì không có dữ liệu ở mức đoạn văn nên ta phải cải biên thuật toán gốc, dẫn đến kết quả không được tốt.

Với mỗi đoạn văn trong văn bản gốc, chọn ra 5 câu có avg ROUGE(1, 2, L) là lớn nhất so với bản tóm tắt mẫu. Tiếp theo, chọn ra 1 tổ hợp không quá 3 câu văn trong 5 câu văn này mà có avg ROUGE(1, 2, L) so với bản tóm tắt mẫu là lớn nhất để làm thành tóm tắt cho đoạn văn. Sau đó ta tham lam chọn dần các tóm tắt đoạn văn có avg ROUGE(1, 2, L) so với tóm tắt mẫu là lớn nhất để làm bản tóm tắt của mô hình. Khác biệt là ở đây là mô hình gốc dùng tóm tắt mẫu ở mức đoạn văn, còn mô hình này dùng tóm tắt mẫu ở mức chương, vì thế kết quả không tốt.

b, GA-ExtractiveOracle:

Sử dụng giải thuật di truyền để giải trực tiếp bài toán chọn tổ hợp câu tốt nhất từ văn bản gốc để làm tóm tắt, chứ không cần phải tham lam từng đoạn văn như với Greedy-ExtractiveOracle. Kết quả tốt vượt trội mọi mô hình, đúng như yêu cầu cho một extractive oracle. Đồng thời, kết quả đó cũng cho thấy tóm tắt mà các mô hình đã thử nghiệm sinh ra còn cách rất xa so với kết quả tối ưu.

IV, Kết quả thử nghiệm:

*** Với bộ dữ liệu BookSum:**

Vì 2 metric BertScore và SummaQA mất rất nhiều thời gian chạy nên tạm thời chưa được tính.

Hệ mô hình	Mô hình	ROUGE-1	ROUGE-2	ROUGE-L
Dùng ma trận điểm số giữa các câu văn	Greedy-BERT	30.66	4.67	13.32
	TextRank-BERT (d = 0.85)	30.62	4.66	13.31
	MCS-BERT (α = 0.5)	31.46	4.67	13.60
	MCS-SA-BERT (α = 0.5)	32.07	4.77	13.66
	MCS-GA-BERT (α = 0.5)	31.94	4.74	13.55
	SOBP-BERT (α = 0.2)	31.40	4.41	13.40
	SOBP-MA-BERT (α = 0.5)	31.55	4.50	13.61
	GA-BERT	31.68	4.41	13.29
Embed doc & summary	GA-BERT-SumCosine	31.51	4.88	13.27
Dùng số lần xuất hiện 1-gram và 2-gram	One-and-Two-Words	31.13	4.86	13.87
	MCS-One-and-Two-Words	32.03	5.04	13.69
	MCS-GA-One-and-Two-Words (fix summary length = 10%)	**33.15** (33.68)	5.57 (5.24)	13.84 (14.13)
	MCS-GA-UniBigram-2profits (fix summary length = 10%)	33.14 **(33.75)**	**5.61** **(5.26)**	13.90 (14.16)
	MCS-GA-One-and-Two-Words -noStopword (fix summary length = 10%)	32.66 (32.90)	5.28 (5.05)	13.95 (14.23)
	MCS-GA-UniBigram-2profits -noStopword (fix summary length = 10%)	32.72 (33.06)	5.33 (5.05)	**14.01** **(14.30)**
	MCS-GA-UniBigram-tfidf	32.96	5.70	14.11
	MCS-GA-UniBigram-tfidf -noStopword	32.39	5.28	13.90
Best in paper	**T5_fine-tuned**	**37.38**	**8.42**	**16.77**
Extractive Oracle	Greedy-ExtractiveOracle	30.50	5.09	13.59
	GA-ExtractiveOracle	**49.74**	**14.03**	**25.25**

*** Với các bộ dữ liệu khác:**

Bộ dữ liệu	Mô hình	ROUGE-1	ROUGE-2	ROUGE-L
CNN / Dailymail	Greedy-BERT	32.37	11.22	20.29
	MCS-GA-One-and-Two-Words (fix summary length = 10%)	35.91 (35.85)	14.34 (14.36)	22.03 (22.00)
	MCS-GA-UniBigram-2profits (fix summary length = 10%)	35.82 (35.82)	14.32 (14.34)	21.97 (21.95)
	MCS-GA-One-and-Two-Words -noStopword (fix summary length = 10%)	**37.17** **(37.10)**	**15.38** **(15.37)**	**23.10** **(23.04)**
	MCS-GA-UniBigram-2profits -noStopword (fix summary length = 10%)	37.00 (36.92)	15.33 (15.30)	22.98 (22.97)
	MCS-GA-UniBigram-tfidf	36.63	15.34	22.96
	MCS-GA-UniBigram-tfidf -noStopword	36.95	15.38	23.11
	GA-BERT-SumCosine	33.55	11.45	20.22
	HAHSum	**44.68**	**21.30**	**40.75**
	GA-ExtractiveOracle	**56.20**	**33.30**	**42.92**

V, Công việc tiếp theo:

1, Áp dụng thử mô hình MCS-GA-One-and-Two-Words vào các bộ dữ liệu khác để xem xét hiệu quả, nếu đủ tốt có thể viết paper.

2, Dùng Docker, upload code lên server để chạy.
-> Chạy thử các mô hình thuộc hệ 1 dùng ma trận avg ROUGE(1, 2, L).

3, Thử fine-tune next sentence prediction model bằng BookSum data -> lắp vào các mô hình thuộc hệ mô hình sử dụng điểm số giữa các câu văn để chạy test.

- Làm sao để predict được nhiều cặp câu trong 1 lần gọi model để tăng tốc độ chạy.

- Để ý một số trường hợp có thể gây nhiễu khi làm fine-tune data (ví dụ như câu cuối đoạn văn và câu đầu đoạn tiếp theo…).

- Kiểm tra chất lượng fine-tune bằng cách cho chạy thử với validation data.

- Chạy với GPU.

4, Xây dựng mô hình Deep Learning end-to-end supervised để giải bài toán.

- Làm giàu dữ liệu bằng GA.

- Bài toán sequence labeling có layer quy hoạch động (Viterbi/CRF/HMM…) để tính điểm giá trị tạo ra bởi các câu được chọn vào bản tóm tắt? Làm như thế nào?

- Học về Reinforcement Learning.

5, Download bộ dữ liệu đầy đủ về để chạy.