CS221 Final Report

Xử lý ngôn ngữ tự nhiên (Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh)

Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Khoa Khoa học máy tính

Môn học: Xử lý ngôn ngữ tự nhiên

Tiêu đề: Efficient Document Retrieval by End-to-End Refining and Quantizing BERT Embedding with Contrastive Product Quantization

Nhóm 7

Giảng viên:
Ts. Nguyễn Thị Quý

1 Giới thiệu

1.1 Bài toán truy vấn tài liệu (Document Retrieval)

Truy xuất tài liệu được định nghĩa là việc so khớp một số truy vấn của người dùng đã nêu với một tập hợp các bản ghi văn bản tự do. Những bản ghi này có thể là bất kỳ loại văn bản phi cấu trúc nào, chẳng hạn như các bài báo, hồ sơ bất động sản hoặc các đoạn văn trong sổ tay. Truy vấn của người dùng có thể bao gồm từ mô tả đầy đủ nhiều câu về nhu cầu thông tin cho đến một vài từ.

1.2 Dataset

NYT là dataset chứa những bài báo được xuất bản bởi The New York Times. Dataset chứa 3 tập train, val, test trong đó tập train chứa 9221 bài báo, tập val chứa 1152 bài báo và tập test chứa 1154 bài báo.
Dataset là file **csv** với 2 cột: cột thứ 1 là label (category của bài báo) và cột thứ 2 là nội dung bài báo.
Các bài báo được chia vào 26 categories: dance, imigration, gun_control, abortion, economy, tennis, football, basketball, movies, federal_budget, international_business, military, stocks_and_bonds, law_enforcement, golf, baseball, environment, energy_companies, hockey, television, cosmos, soccer, surveillance, the_affordable_care_act, music, gay_rights.

index	7	cleveland — nothing else is in rookie linebacker barkevious mingo's way but some blockers.the browns signed the no. 6 overall pick in this year's nfl draf
0	22	manchester — manchester united striker wayne rooney is in good shape mentally and physically as the premier league champions prepare to face bitter rivals liverpool this wee he was with me at everton as a young boy and his career has gone exactly the way that people expected it to go chelsea's public pursuit of rooney has cooled off since monday
1	7	miami gardens, fla. — rex ryan is coming back next year. the miami dolphins are sitting out another postseason the new york jets eliminated the dolphins from playoff contention said. but, that's average. and i don't look around this locker room and see average. inconsistency has been our issue. and here it is again showing its ugly head today the dolphi and we didn't come up with the plays we needed, wallace said. tannehill had passer ratings of 6 and 1 in the final two games.
2	22	berlin — germany's storied bayern munich soccer club is sticking by its president, despite a state court's announcement on monday that he will stand trial on charges of tax evas the team to build its financial reserves to a level rarely seen in european club soccer. his decision to go public with his story triggered a national discussion over social justice an
3	8	waco, texas — odyssey sims scored 29 points and had 13 assists to become the baylor career leader in that category as the no. 9 lady bears beat san jose state 113-73 on tues cowboys' stadium in quite a doubleheader night after the baylor and kentucky women finish their game at the site of the men's ncaa final four next april, the men's teams from the
4	7	east rutherford, n.j. — the talk all week by both teams was of desperation, a must-win showdown between the miami dolphins and new york jets one of the afc east rivals brough round pick to sit? or, should the jets let him try to work himself out of his funk? with the playoffs an unlikely scenario, new york might opt to find out if smith could be the quarterb the jets couldn't get anything going all game.leaky defense: the jets' defense couldn't get off the field in the first half, and things weren't much better in the last two quarters.they
5	7	the n.f.l. loves to trumpet its calendar, from draft day to kickoff weekend to super bowl sunday. then there is black monday, the day after the regular season ends, when a dozen competitive environment, there's a race to get things done because there is the same group of candidates out there, said mike tannenbaum, the former general manager of the j ran an article that said that the day after the regular season ends traditionally is called "black monday" in the coaching profession. though when this so-called tradition started wa
6	14	centennial, colo. — the ghostly self portraits, taken with a cellphone hours before he opened fire at an aurora movie theater, show james e. holmes in the final sprint of his deadl there were never going to be enough ambulances, enough police officers to get everyone out of there fast enough.she added: he didn't care who he killed and how many he kille
7	5	washington — americans spent more in may as their income rose, encouraging signs after a slow start to the year.the commerce department said on thursday that consumer sp
8	16	corey kluber pitched eight shutout innings, and the cleveland indians spoiled stephen strasburg's return with a 2-0 win over the visiting washington nationals on sunday.kluber es visiting milwaukee . marlins, cardinals 2 ricky nolasco allowed one run and three hits in seven innings, and host miami won to take two of three games from st . louis.the cardinal
9	1	the poetically named shantala shivalingappa was born in india, but raised in paris, where she trained in kuchipudi, a south indian classical dance style, with her mother, the well

Figure 1: Minh hoạ dataset

1.3 Độ đo đánh giá

Với mỗi query từ tập test, tìm trong tập train top 100 document phù hợp nhất với query. Sau đó, độ chính xác truy xuất được tính bằng tỷ lệ của các tài liệu liên quan. Các tài liệu được tính là liên quan nếu có cùng label với label của query.

2 Quy trình thực hiện

2.1 Install and import dependencies

Model yêu cầu 2 thư viện chính là **torch** và **transformers**.

```
pip install -r requirements.txt
```

Ngoài ra ta cần import một số thư viện khác:

```
import os
import argparse
import re
import pandas as pd
import numpy as np
```

2.2 Tạo Data Loader và xử lý dữ liệu

Trước tiên, ta cần đọc dữ liệu từ file **.csv**, tiền xử lý dữ liệu, sau đó tạo Data loader để load dữ liệu theo batch size.

```
class LabeledDocuments(Data):
    def __init__(self, hparams):
        super().__init__(hparams=hparams)

    def load_datasets(self):
        self.X_train, self.Y_train = self.read_data(
            f"{self.file_path}/train.csv", max_length=self.
    max_length)
        self.X_val, self.Y_val = self.read_data(
            f"{self.file_path}/val.csv", max_length=self.
    max_length)
        self.X_test, self.Y_test = self.read_data(
            f"{self.file_path}/test.csv", max_length=self.
    max_length)

        print(
```

```
            f"train size: {len(self.X_train)}, validation
    size: {len(self.X_val)}, test size: {len(self.X_test)}")

    def read_data(self, path, max_length):
        def label_fn(x):
            return x - 1

        rows = pd.read_csv(
            path,
            sep=",",
            on_bad_lines='skip',
            header=None,
            skiprows=None,
            quoting=0,
            keep_default_na=False,
            encoding="utf-8",
        )

        label_fn = label_fn if label_fn is not None else (
    lambda x: x)
        labels = rows[0].apply(lambda x: label_fn(x))
        sentences = rows[1]
        sentences = sentences.apply(
            lambda x: clean_tokenize_truncate(x, max_length))
        return sentences.tolist(), labels.tolist()

    def get_loaders(self, batch_size, num_workers,
                    shuffle_train=False,
                    get_test=True):
        train_dataset = My_Dataset(
                                self.X_train,
                                self.Y_train,
                                self.max_length)
        database_dataset = My_Dataset(
            self.X_train, self.Y_train, self.max_length)
        val_dataset = My_Dataset(self.X_val, self.Y_val, self
    .max_length)
        test_dataset = My_Dataset(self.X_test, self.Y_test,
    self.max_length)

        # DataLoader
        train_loader = DataLoader(
                            dataset=train_dataset,
                            batch_size=batch_size,
                            shuffle=shuffle_train,
```

```
                                num_workers=num_workers)

        database_loader = DataLoader(
                                dataset=database_dataset,
                                batch_size=512,
                                shuffle=False,
                                num_workers=num_workers)

        val_loader = DataLoader(
                                dataset=val_dataset,
                                batch_size=512,
                                shuffle=False,
                                num_workers=num_workers)

        test_loader = DataLoader(
                                dataset=test_dataset,
                                batch_size=512,
                                shuffle=False,
                                num_workers=num_workers) if
    get_test else None
        return train_loader, database_loader, val_loader,
    test_loader,
```

Một số hàm hỗ trợ tiền xử lý dữ liệu.

```
def clean_string(string):
    """
    Tokenization/string cleaning for yelp data set
    Based on https://github.com/yoonkim/CNN_sentence/blob/
    master/process_data.py
    """
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`\"]", " ", string)
    string = re.sub(r"\'s", " 's", string)
    string = re.sub(r"\'ve", " 've", string)
    string = re.sub(r"n\'t", " n't", string)
    string = re.sub(r"\'re", " 're", string)
    string = re.sub(r"\'d", " 'd", string)
    string = re.sub(r"\'ll", " 'll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \( ", string)
    string = re.sub(r"\"\"", ' " ', string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)
    return string.lower().strip()
```

```
def clean_tokenize_truncate(x, max_length):
    x = clean_string(x)
    x = x.split(" ")
    x = x[:max_length]
    return x
```

2.3 Xây dựng model

- Xây dựng Product Quantization:

```
class PQ_head(nn.Module):
    """
    The pq implementaion refers to SPQ(https://github.com/youngkyunJang/SPQ).
    """
    def __init__(self, N_words, N_books, L_word, gumbel_temp, dist_metric, sample_method = "gumbel_softmax"):
        # N_words = K; number of codewords
        # N_books = M; number of codebooks
        # L_word = Dimension of the codewords

        super(PQ_head, self).__init__()

        self.C = nn.Parameter(torch.randn(N_words, N_books * L_word))

        self.N_books = N_books
        self.L_word = L_word
        self.gumbel_temp = gumbel_temp
        self.dist_metric = dist_metric

        self.sample_method = sample_method

    def sample_gumbel(self, shape, eps = 1e-20):
        U = torch.rand(shape)
        U = U.cuda()
        return -torch.log(-torch.log(U + eps) + eps)

    def gumbel_softmax_sample(self,logits, temperature = 1):
        y = logits + self.sample_gumbel(logits.size())
        return F.softmax(y / temperature, dim = -1)

    def gumbel_softmax(self, logits, temperature = 1, hard = True):
        y = self.gumbel_softmax_sample(logits, temperature)
```

```
32
33          if not hard:
34              return y
35          raise NotImplementedError()
36
37
38      def defined_sim(self, x, c):
39          if self.dist_metric == "euclidean":
40              diff = x.unsqueeze(1) - c.unsqueeze(0)
41              return  - torch.sum(diff * diff, - 1)
42          else:
43              raise NotImplementedError
44
45
46
47      def forward(self, X):
48          # input shape: (bsz, N_books * L_word)
49          x = torch.split(X, self.L_word, dim = 1) # tuple; ele
        of the tuple has the shape like (bath_size, L_word)
    N_books element
50          c = torch.split(self.C, self.L_word, dim = 1) # tuple
    ; ele of the tuple has the shape like (N_words, L_word)
        N_books elements.
51          prob_list = []
52          for i in range(self.N_books):
53              logits = self.defined_sim(x[i], c[i])
54
55              if self.sample_method == "softmax":
56                  soft_prob = F.softmax(logits / self.
    gumbel_temp, dim = 1)
57                  prob_list.append(soft_prob)
58              elif self.sample_method == "gumbel_softmax":
59                  logits = F.softmax(logits, dim = 1)
60                  logits = torch.log(logits + 1e-9)
61                  soft_prob = self.gumbel_softmax(logits, self.
    gumbel_temp, False)
62                  prob_list.append(soft_prob)
63
64              if i == 0:
65                  Q = torch.mm(soft_prob, c[i])
66              else:
67                  Q = torch.cat((Q, torch.mm(soft_prob, c[i])),
        dim = 1)
68          return Q, prob_list
```

- Xây dựng Mutual Information Improved Contrastive Product Quantization

(MICPQ) model:
+ Khởi tạo tham số:

```
def define_parameters(self):
    bert_config = AutoConfig.from_pretrained('bert-base-uncased')
    bert_config.attention_probs_dropout_prob = self.hparams.dropout_rate
    bert_config.hidden_dropout_prob = self.hparams.dropout_rate
    self.bert_model = AutoModel.from_pretrained('bert-base-uncased', config = bert_config)
    for param in self.bert_model.parameters():
        param.requires_grad = False

    assert self.hparams.encode_length == self.hparams.N_books * int(np.log2(self.hparams.N_words))

    self.pq_head = PQ_head(
        N_words = self.hparams.N_words,
        N_books = self.hparams.N_books,
        L_word = self.hparams.L_word,
        gumbel_temp = self.hparams.gumbel_temperature,
        dist_metric = self.hparams.dist_metric,
        sample_method = self.hparams.sample_method
        )

    self.pro_layer = nn.Sequential(
        nn.Linear(768, 768),
        nn.ReLU(),
        nn.Linear(768, int(self.hparams.L_word * self.hparams.N_books)),
    )

    self.criterion = NtXentLoss(self.hparams.temperature)
```

+ Tinh chỉnh BERT embedding:

```
def get_embeddings(self, inputs, pooling="mean"):
    bert_output = self.bert_model(**inputs)[0] # (batch_size, sequence_length, hidden_size) Sequence of hidden-states at the output of the last layer of the model.

    if pooling == 'mean':
        attention_mask = inputs['attention_mask'].unsqueeze(-1) # (batch_size, sequence_length, 1)
```

```
        output = torch.sum(bert_output*attention_mask, dim = 1) / torch.sum(attention_mask, dim = 1)
    elif pooling == 'cls':
        output = bert_output[:,0]
    else:
        raise NotImplementedError()
    return output
```

+ Hàm Mutual Information Loss:

```
def compute_prob_loss(self, prob0, prob1):
    def get_entropy(probs):
            q_ent = -(probs.mean(0) * torch.log(probs.mean(0) + 1e-12)).sum()
            return q_ent

    def get_cond_entropy(probs):
        q_cond_ent = - (probs * torch.log(probs + 1e-12)).sum(1).mean()
        return q_cond_ent

    q_ent0 = get_entropy(prob0)
    q_cond_ent0 = get_cond_entropy(prob0)

    q_ent1 = get_entropy(prob1)
    q_cond_ent1 = get_cond_entropy(prob1)

    wim_0 = q_ent0 - self.hparams.cond_ent_weight * q_cond_ent0
    wim_1 = q_ent1 - self.hparams.cond_ent_weight * q_cond_ent1

    wim = (wim_0 + wim_1) / 2.
    prob_loss = - wim
    return prob_loss
```

+ Hàm forward để tính tất cả các hàm loss:

```
def forward(self, inputs):
    embd_0 = self.pro_layer(self.get_embeddings(inputs, pooling=self.hparams.pooler_type))
    embd_1 = self.pro_layer(self.get_embeddings(inputs, pooling=self.hparams.pooler_type))

    Q_0, prob0_list = self.pq_head(embd_0)
    Q_1, prob1_list = self.pq_head(embd_1)

```

```
    codebooks_losses = []
    for i in range(len(prob0_list)):
        prob0 = prob0_list[i]
        prob1 = prob1_list[i]
        codebooks_losses.append(self.compute_prob_loss(prob0, prob1))

    prob_loss = sum(codebooks_losses)
    contra_loss = self.criterion(Q_0, Q_1)

    loss = self.hparams.code_weight * contra_loss + self.hparams.prob_weight * prob_loss

    return {'loss': loss, 'contra_loss': contra_loss, 'prob_loss': prob_loss}
```

+ Hàm run_training_session để train model. Tham số **bad_epochs** để model có thể dừng train nếu số lượng **bad_epochs** lớn hơn số lượng bad epochs được quy định trước (**num_bad_epochs**).

```
def run_training_session(self, run_num, logger):
    self.train()

    # Scramble hyperparameters if number of runs is greater than 1.
    if self.hparams.num_runs > 1:
        logger.log('RANDOM RUN: %d/%d' % (run_num, self.hparams.num_runs))
        for hparam, values in self.get_hparams_grid().items():
            assert hasattr(self.hparams, hparam)
            self.hparams.__dict__[hparam] = random.choice(values)

    random.seed(self.hparams.seed)
    torch.manual_seed(self.hparams.seed)

    self.define_parameters()
    # logger.log(str(self))
    logger.log('%d params' % sum([p.numel() for p in self.parameters()]))
    logger.log('hparams: %s' % self.flag_hparams())

    device = torch.device('cuda' if self.hparams.cuda else 'cpu')
        self.to(device)
```

```
21
22      optimizer = self.configure_optimizers()
23      gradient_clippers = self.configure_gradient_clippers()
24      train_loader, database_loader, val_loader, _ = self.data.
    get_loaders(
25          self.hparams.batch_size, self.hparams.num_workers,
26          shuffle_train=True, get_test=False
27          )
28      best_val_perf = float('-inf')
29      best_state_dict = None
30      bad_epochs = 0
31
32      # logger.log('Time: %s' % str(timedelta(seconds=round(
    timer() - start))))
33      try:
34
35          for epoch in range(1, self.hparams.epochs + 1):
36              epoch_start = timer()
37
38              forward_sum = {}
39              num_steps = 0
40              for batch_num, batch in enumerate(train_loader):
41                  optimizer.zero_grad()
42
43                  inputs = squeeze_dim(
44                      move_to_device(batch[0], device), dim=1)
45                  forward = self.forward(inputs)
46
47                  for key in forward:
48                      if key in forward_sum:
49                          forward_sum[key] += forward[key]
50                      else:
51                          forward_sum[key] = forward[key]
52                  num_steps += 1
53
54                  if math.isnan(forward_sum['loss']):
55                      logger.log('Stopping epoch because loss
    is NaN')
56                          break
57
58                  forward['loss'].backward()
59                  for params, clip in gradient_clippers:
60                      nn.utils.clip_grad_norm_(params, clip)
61                  optimizer.step()
62
```

```
            if math.isnan(forward_sum['loss']):
                logger.log('Stopping training session because loss is NaN')
                break

            val_perf = self.evaluate(
                database_loader, val_loader, device, is_val=True)
                logger.log('End of epoch {:3d}'.format(epoch), False)
            logger.log(' '.join([' | {:s} {:8.2f}'.format(
                key, forward_sum[key] / num_steps)
                for key in forward_sum]), False)
            logger.log(' | val perf {:8.2f}'.format(val_perf), False)
            logger.log(' | time : %s' %
                       str(timedelta(seconds=round(timer() - epoch_start))))

            if val_perf > best_val_perf:
                best_val_perf = val_perf
                bad_epochs = 0
                logger.log('\t *Best model so far, deep copying*')
                    best_state_dict = deepcopy(self.state_dict())
            else:
                bad_epochs += 1
                logger.log('\t Bad epoch %d' % bad_epochs)

            if bad_epochs > self.hparams.num_bad_epochs:
                    break
    except KeyboardInterrupt:
        logger.log('-' * 89)
        logger.log('Exiting from training early')

    return best_state_dict, best_val_perf
```

+ Để load các tham số từ pretrained model, ta sử dụng hàm **load()**:

```
def load(self):
    device = torch.device('cuda' if self.hparams.cuda else 'cpu')
    checkpoint = torch.load(self.hparams.model_path) if self.hparams.cuda \
        else torch.load(self.hparams.model_path,
```

```
                                map_location=torch.device('cpu'))
    if checkpoint['hparams'].cuda and not self.hparams.cuda:
        checkpoint['hparams'].cuda = False
    self.hparams = checkpoint['hparams']
    self.define_parameters()
    self.load_state_dict(checkpoint['state_dict'])
    self.to(device)
```

+ Hàm **run_test** để chạy model, trả về kết quả top 100 documents và độ chính xác

```
def run_test(self):
    device = torch.device('cuda' if self.hparams.cuda else '
    cpu')
    _, database_loader, val_loader, test_loader = self.data.
    get_loaders(
        self.hparams.batch_size, self.hparams.num_workers,
        shuffle_train=True, get_test=True)
    val_perf, val_res = self.evaluate(
        database_loader, val_loader, device, is_val=True)
    test_perf, test_res = self.evaluate(
        database_loader, test_loader, device, is_val=False)
    return val_perf, val_res, test_perf, test_res
```

2.4 Hàm đánh giá

- Hàm evaluate để đánh giá hiệu quả của model:

```
def evaluate(self, database_loader, eval_loader, device,
    is_val=True):
    self.eval()
    with torch.no_grad():
        perf, top_k = compute_retrieval_precision(
    database_loader, eval_loader,
        device, self.encode_continuous,
        self.pq_head.C,
        self.hparams.N_books,
        self.hparams.num_retrieve)
    self.train()
    return perf, top_k
```

Trong đó, precision được tính bằng tỷ lệ số document liên quan đến query trên top 100 documents tìm kiếm được.

```
def compute_retrieval_precision(train_loader, eval_loader,
                                device, encode_continuous,
```

```
                                Codebooks, N_books,
                                num_retrieve=100):
    def extract_target(loader):
        encoding_chunks = []
        label_chunks = []
        for (docs, labels) in loader:
            docs = squeeze_dim(move_to_device(docs, device),
    dim=1)
            encoding_chunks.append(docs if encode_continuous
    is None else
                encode_continuous(docs))
            label_chunks.append(labels)

        encoding_mat = torch.cat(encoding_chunks, 0)
        label_mat = torch.cat(label_chunks, 0)
        label_lists = [[label_mat[i].item()] for i in range(
    label_mat.size(0))]
        return encoding_mat, label_lists

    def extract_database_code(loader, Codebooks):
        code_chunks = []
        label_chunks = []
        for (docs, labels) in loader:
            docs = squeeze_dim(move_to_device(docs, device),
    dim=1)
            encodings = encode_continuous(docs)
            codes = Indexing(Codebooks, N_books, encodings)
            code_chunks.append(codes)

            label_chunks.append(labels)

        code_mat = torch.cat(code_chunks, 0).type(torch.int)
        label_mat = torch.cat(label_chunks, 0)
        label_lists = [[label_mat[i].item()] for i in range(
    label_mat.size(0))]
        return code_mat, label_lists

    src_codes, src_label_lists = extract_database_code(
    train_loader, Codebooks)

    import time
    start = time.time()
    tgt_encodings, tgt_label_lists = extract_target(
    eval_loader)
    prec, top_k = compute_topK_average_precision(
```

```
                                tgt_encodings, tgt_label_lists,
                                src_codes, src_label_lists,
                                Codebooks, N_books,
                                num_retrieve)
    return prec, top_k
```

3 Thực nghiệm

3.1 Thông tin các hyperparameter

- Số chiều của codeword $\frac{D}{M}$ trong codebook nhỏ là C^m cố định ở 24.
- Số lượng codeword K trong mỗi codebook C^m cố định ở 16.
- Số lượng codebook nhỏ M có thể là {4, 8, 16, 32}. Trong phần thực nghiệm này, số lượng được chọn là 4.
- Các codeword trong các codebook C có thể biểu diễn dưới dạng {16, 32, 64, 128} bit theo công thức $B = M \log_2 K$. Do đã chọn $M = 4$ và $K = 16$ nên số bit là 16.
- Learning rate được chọn là 0.001.
- Drop rate p_{drop} để tạo các cặp positives cho contrastive learning được chọn là 0.3.
- Gumbel-softmax temperature τ được chọn là 10 cho 16 bit và 5 cho code dài hơn.
- Trade-off coefficient α được chọn là 0.1.
- Coeficient được chọn trong {0.1, 0.2, 0.3} tuỳ theo hiệu năng trên tập validation.

3.2 Kết quả

Sau khi train với 100 epochs (tuy nhiên bị dừng sau epoch thứ 24 do số lương bad epoch vượt quá số lượng bad epoch cho phép) và đánh giá trên tập test ta được kết quả :

	Validation	Test
Precision	81.29	80.39

So với kết quả do tác giả cung cấp trong paper, precision của thực nghiệm thấp hơn khoảng 2%. Nguyên nhân có thể là do quá trình train bị dừng sớm ở epoch 24. Tuy nhiên, kết quả thực nghiệm vẫn cao hơn đa số các phương pháp cũ.