IR Based Chatbot

Final Presentation

Group 15
Yash Khasbage - CS17BTECH11044
Puneet Mangla - CS17BTECH11029
Mentor: Suvodip Dey

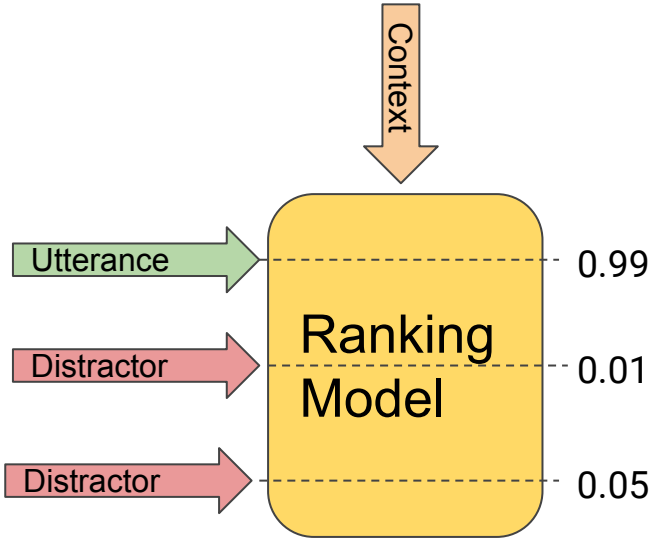
Problem Statement

The task is to build a chatbot.

- *Context:* Conversation upto this point
- *Utterance:* Correct response to context.
- *Distractor:* Incorrect response to context

Develop a ranking model

- to assign highest score to true utterance
- lower scores to distractors



Dataset description: Ubuntu Dialogue Corpus

- Consists of chats from Ubuntu-related chat from Freenode IRC for various technical support.

Train: 50% positive and 50% negative samples

Context	*Response*	*Flag (0 or 1)*

Test and Validation:

Context	*True Response*	*False Response 1*	….	*False Response 9*

examples:

Split	Train	Validation	Test
# examples	1,000,000	19,561	18,921

url: https://github.com/rkadlec/ubuntu-ranking-dataset-creator

Dataset Snapshots

5 samples from

Basic Processing like Stemming, Lemmatization, Stop-word removal are already done

Train:

Context	Utterance	Label
i think we could import the old comment via rs...	yes . same binari packag . __eou__	1
i 'm not suggest all - onli the one you modifi...	ok let me tri that .. thank man __eou__	0
afternoon all __eou__ not entir relat to warti...	http : //www.ubuntu.com/download/ -- ani mirro...	0
interest __eou__ grub-instal work with / be ex...	i fulli endors this suggest < /quimbi > __eou_...	1
and becaus python give mark a woodi __eou__ __...	ok if your defaultdepth 24 __eou__	0

Validation:

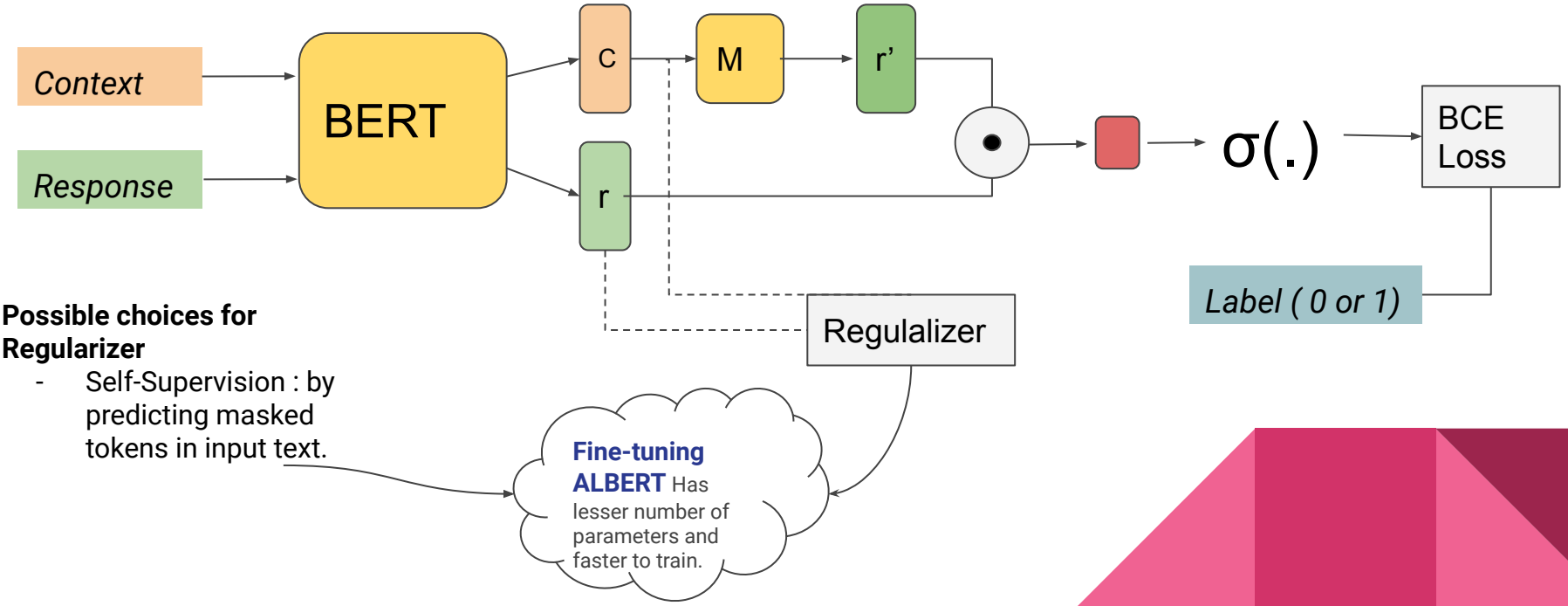
Context	...	Distractor_8
ani idea on how lts will be releas ? __eou__ _...	...	i be want to extract a split .rar file . i be ...
how much hdd use ubuntu default instal ? __eou...	...	umm . you mean tempfil be program and tmpfile ...
in my countri it near the 27th __eou__ when wi...	...	ah ok. it work now . u be right , i do a mista...
it 's not out __eou__ __eot__ they probabali b...	...	it look like that articl tell you how to setup...
be the ext4 driver stabl ? __eou__ __eot__ i b...	...	jeah iam in the bumblebe room ;) __eou__ coul...

Test:

Context	...	Distractor_8
anyon know whi my stock oneir export env var u...	...	there be a variabl path (run `` echo $ path '...
i set up my hd such that i have to type a pass...	...	first place i look __eou__
im tri to use ubuntu on my macbook pro retina ...	...	backport be a repo you can enabl in your packa...
no suggest ? __eou__ link ? __eou__ how can i ...	...	you could re-encod it with mencod or ffmpeg __...
i just ad a second usb printer but not sure wh...	...	if that be the case , they whi on earth would ...

Overview of Approach: Tuning BERT Architecture

Posing it as a binary classification task



Possible choices for Regularizer

- Self-Supervision : by predicting masked tokens in input text.

Implementation details

Transformers

Frameworks and Libraries

- PyTorch
- Hugging Face - Transformers Library
- NLTK

Training Details

- Single GTX 1080 GPU - 16 GB
- *Batch Size* - 64
- *Input Length* - 87 tokens for context and 17 for response (padded/truncated if needed)
- *# Training samples:* 50K and 500K
- *# Evaluation samples:* 5K
- *Learning rate :* 2e-5

BERT Config

- *Activation:* GELU
- *# hidden layers :* 12, hidden size: 768
- *Attention Heads:* 12
- *Vocabulary:*
  - Initially Size is 30522
  - Added some Linux related vocab
  - Added tokens like “__eou__”, “__eot__”, “__dialog_end__”
  - *Final size* - 50155

Evaluation Metric: Top-K metric

Measures if utterance lies among the top-k relevant documents retrieved by the learning algorithm.

Let $[r_1, r_2 \ldots, r_{10}]$ where r_1 *is utterance* and $r_2, r_3 \ldots r_{10}$ *are distractors*

Π(r) gives the relevance index of document *r* in ordered list (descending order of relevance) obtained using learning algorithm.

Top(k) Score = $\sum (\Pi(r_1) \leq k) / N$

Baselines and Variants

Baselines:

1. *Random Baseline* selects the utterance among all responses given
2. *Dual-Encoder LSTM* trained on 1M examples
3. *Dual Encoder RNN* trained on 1M examples
4. *TF-IDF model* learned on 1M points

Approach Variants:

1. *Ours-BERT (50K)* is BERT model trained using our approach on 50K examples.
2. *Ours-BERT (500K)* is BERT model trained using our approach on 500K examples.
3. *Ours-ALBERT (50K)* is ALBERT model trained using our approach on 50K examples.

Results

Method	**Training Samples**	**Top1**	**Top2**	**Top5**
Random	-	0.1	0.2	0.5
LSTM (1M)	1M	**0.49**	**0.68**	**0.91**
RNN (1M)	1M	0.38	0.56	0.83
TF-IDF (1M)	1M	**0.49**	0.587	0.76
Ours-BERT	50K	0.235	0.371	0.71
Ours-BERT	500K	0.43	0.61	0.84
Ours-ALBERT	50K	0.22	0.4680	0.68

Results

Method	Training Samples	Top1	Top2	Top5
Random	-	0.1	0.2	0.5
LSTM (1M)	1M	**0.49**	**0.68**	**0.91**
RNN (1M)	1M	0.38	0.56	0.83
TF-IDF (1M)	1M	**0.49**	0.587	0.76
Ours-BERT	50K	0.235	0.371	0.71
Ours-BERT	500K	0.43	0.61	0.84
Ours-ALBERT	50K	0.22	0.4680	0.68

With just half training examples (500K) our approach is :

- better than *Random, Dual-Encoder RNN and TF-IDF* model is most cases.
- Comparable to *Dual-Encoder LSTM* which gives best performance

Results

Method	Training Samples	Top1	Top2	Top5
Random	-	0.1	0.2	0.5
LSTM (1M)	1M	**0.49**	**0.68**	**0.91**
RNN (1M)	1M	0.38	0.56	0.83
TF-IDF (1M)	1M	**0.49**	0.587	0.76
Ours-BERT	50K	0.235	0.371	0.71
Ours-BERT	500K	0.43	0.61	0.84
Ours-ALBERT	50K	0.22	0.4680	0.68

Results

Method	Training Samples	Top1	Top2	Top5
Random	-	0.1	0.2	0.5
LSTM (1M)	1M	**0.49**	**0.68**	**0.91**
RNN (1M)	1M	0.38	0.56	0.83
TF-IDF (1M)	1M	**0.49**	0.587	0.76
Ours-BERT	50K	0.235	0.371	0.71
Ours-BERT	500K	0.43	0.61	0.84
Ours-ALBERT	50K	0.22	0.4680	0.68

Comparing ours-BERT and ours-ALBERT

- Top-2 performance of ours-ALBERT is fairly better that ours-BERT. Top1 and Top5 are comparable.
- We observe that convergence of Ours-ALBERT was slow - possibly because of additional SSA task.
- While BERT achieved 90% in just 15 epochs, ALBERT only achieved 60%
- More epochs and data might help.

Example: Correct in Top-1

Context & Responses	hello guys . i have a problem instal ubuntu with wubi . At the end this error occurs - > http : //shrani.si/f/1s/4o/1n9nru75/capture.png My only idea here be you run wubi as admin ? Hmm what window do you use ? Help shoot me in the foot I'm out of idea I tried with cd boot and installation gets terminated by signal 15 [9028]
1	**Try re-download ?**
2	Try the nvidia one
3	I've never done it myself
4	I find /usr/share/x11/xerrordb through the manpage
5	It is not valgrind by any means but it help sometimes

Example: Correct in Top-3

Context & Responses	Can someone help me out? I'm having a problem with dualboot a sony vaio in raid 0 How did you create the raid 0 ? It's hardware raid - two 256 GB SSDs so which raid control do you use then ?
1	Whatever process can run the video. So it may be vlc or the browser for flash. A co-worker said that legacy hardware will usually generate a warning upon OS installation. I get no such warning.
2	You noticed that's an IPv6 port only? Do you expect an IPv4 port too? Indeed. Does the vpn host operate a firewall independent of the guest ?
3	**Um I'm not entirely sure... That's why I can use some help. I tried dualboot ubuntu, but after installing 12.04 the Windows 7 loader didn't appear in grub.**
4	Space is space, is all I mean.
5	NTPD won't work if your clock be too far out.

Creating BERT vocabulary

```
# Initialize an empty BERT tokenizer
tokenizer = BertWordPieceTokenizer(
  clean_text=True,
  handle_chinese_chars=False,
  strip_accents=False,
  lowercase=True,
)

# prepare text files to train vocab on them
files = ['input.txt']

# train BERT tokenizer
tokenizer.train(
  files,
  vocab_size=30000,
  min_frequency=10,
  show_progress=True,
  special_tokens=['[PAD]', '[UNK]', '[CLS]', '[SEP]', '[MASK]'],
  limit_alphabet=1000,
  wordpieces_prefix="##"
)


tokenizer.save('./vocab.txt')

f = open('./vocab.txt')
jf = json.load(f)
vocab = list(jf['model']['vocab'].keys())
vocab = '\n'.join(vocab)
print(vocab)
```

Initialize the word-piece tokenizer which will learn top tokens

Input.txt contains all text from training data

Define Vocabulary size, as special tokens

Save the vocabulary

Data Loaders

load_data.py | vocab.py | a.txt | vocabulary.txt | main.py | eval.py | io_utils.py

```
37
38      def __getitem__(self, index):
39          if self.split == 'train':
40              context, response, label = self.dataset.iloc[index]
41              context, response  = word_tokenize(context), word_tokenize(response)
42
43              # print(len(context))
44              # print(self.tokenizer.tokenize(context))
45              context = self.tokenizer.encode_plus(
46                              context,
47                              add_special_tokens=True,                    # Sentence to encode.
48                              max_length = self.max_length,           # Pad & truncate all sentences.
49                              pad_to_max_length = True,
50                              return_attention_mask = True,   # Construct attn. masks.
51                              truncation=True,
52                              return_tensors = 'pt',     # Return pytorch tensors.
53                      )
54
55              # print(context['input_ids'].shape)
56              # print(tokenizer.convert_ids_to_string(context['input_ids']))
57
58              response = self.tokenizer.encode_plus(
59                              response,
60                              add_special_tokens=True,                        # Sentence to encode.
61                              max_length = self.max_length,           # Pad & truncate all sentences.
62                              pad_to_max_length = True,
63                              return_attention_mask = True,   # Construct attn. masks.
64                              truncation=True,
65                              return_tensors = 'pt',     # Return pytorch tensors.
66                      )
67
68              label = int(label)
69              return context['input_ids'].squeeze(0), context['attention_mask'].squeeze(0), \
70              response['input_ids'].squeeze(0), response['attention_mask'].squeeze(0), label
```

Tokenize using NLTK

BERT-Tokenizer

- Encode using vocabulary
- Padd/Truncate to max length (87 for context, 17 for response)
- Return PyTorch Tensors
- Add tokens like [CLS], [UKN] etc.

Return input *encodings* and respective *attention masks* and *label*

Loading Models

```
if __name__ == '__main__':

    args = parse_args()

    # create tokenizer
    tokenizer = BertTokenizer(vocab_file='bert_vocab.txt', do_lower_case=True)
    # get data loader
    if args.dataset.startswith('udc'):
        dataset = UDC(root=args.dataset_root, split=args.split, tokenizer=tokenizer)
        data_loader = torch.utils.data.DataLoader(dataset,
            batch_size=args.batch_size,
            shuffle=True,
            num_workers=2
        )
        num_labels = 2
    else:
        raise Exception("Unknown dataset: {}".format(args.dataset))

    # get bert config
    config = BertConfig.from_pretrained(args.config_name if args.config_name else args.model_name_or_path,
                                        num_labels=num_labels, finetuning_task="ir")
    print('bert configs:')
    print(config)
    projection =  nn.Sequential(nn.Linear(config.hidden_size, config.hidden_size),
                                nn.LeakyReLU(),
                                nn.Linear(config.hidden_size, config.hidden_size),
                                nn.LeakyReLU(),
                                nn.Linear(config.hidden_size, config.hidden_size))
    # get bert(pretrained)
    model = BertModel(config=config)

    model.cuda()
    projection.cuda()

    model = nn.DataParallel(model)
    projection = nn.DataParallel(projection)
```

Initializing a tokenizer using our custom Vocabulary

Loading dataset. Passing batchsize, num workers as parameters

Configuration parameters of BERT model

Multi layered perceptron as Projection network

Loadings models on multiple GPUs

Training Script

load_data.py | vocab.py | a.txt | vocabulary.py | main.py | eval.py | io_utils.py | about.md

```python
def train(train_dataloader, model,projection, classifier, args):
    # Prepare optimizer and schedule (linear warmup and decay)
    criterion = torch.nn.BCELoss()
    no_decay = ['bias', 'LayerNorm.weight']

    optimizer = AdamW([{'params':model.parameters()},{'params':projection.parameters()},{'params':classifier.parameters()}], \
                      lr=args.learning_rate, eps=args.adam_epsilon)
    t_total = len(train_dataloader)*args.num_train_epochs
    scheduler = get_linear_schedule_with_warmup(optimizer, args.warmup_steps, t_total)

    for epoch in range(args.start_epoch,args.num_train_epochs):
        correct,total = 0,0
        avg_loss = 0.
        for i, (context_in, context_mask, response_in, response_mask, label) in enumerate(train_dataloader):
            model.train()
            # print(context_in.shape, context_mask.shape, response_in.shape, response_mask.shape)
            context_feature  = model(input_ids=context_in.cuda(), attention_mask=context_mask.cuda())[0]
            response_feature  = model(input_ids=response_in.cuda(), attention_mask=response_mask.cuda())[0]
            context_feature, response_feature = torch.mean(context_feature,1), torch.mean(response_feature,1)
            logit = F.sigmoid(torch.sum(projection(context_feature)*response_feature,1))
            # logit = F.sigmoid(classifier(concat_feature)).squeeze(1)

            pred =  (logit>0.5).long()
            correct += (pred == label.cuda()).sum().item()
            total += pred.size(0)
            loss = criterion(logit,label.cuda().float())
            loss = loss.mean()
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            avg_loss += loss.data.item()
            scheduler.step()

            if i % 10 == 0:
                print('Epoch {:d} | Batch {:d}/{:d} | Loss {:3f} | Accuracy {:3f}'.format(epoch, i, len(train_dataloader), \
                    avg_loss/(i+1), (100.*correct)/total))

        if (epoch % args.save_freq == 0) or (epoch == args.num_train_epochs - 1):
            outfile = os.path.join(args.checkpoint_dir, '{:d}.tar'.format(epoch))
            state_dict = {}
            state_dict['epoch'] = epoch
            state_dict['feature'] = model.state_dict()
            state_dict['projection'] =  projection.state_dict()
            torch.save(state_dict, outfile)
```

Binary Cross Entropy Loss

AdamW Optimizer with Learning rate: 2e-5

Linear Learning rate Scheduler

Sample a batch

Get BERT embeddings. We take mean of all hidden embeddings

Pass context embedding to projection, take dot product and sigmoid.

Optimize Loss

Save checkpoint

Evaluation Script

load_data.py | vocab.py | a.txt | vocabulary.txt | main.py | eval.py

```
18
19  def evaluate(dataloader, model, projection, args):
20      model.eval()
21      projection.eval()
22
23      ranking_list = list()
24      for batch in dataloader:
25          # bs x dim
26          context_text = batch[0].cuda()
27          # bs x dim
28          context_mask = batch[1].cuda()
29          # bs x 10 x dim
30          response_text = batch[2].cuda()
31          # bs x 10 x dim
32          response_mask = batch[3].cuda()
33          # (bs * 10) x dim
34          response_text = response_text.view(-1, response_text.size(2))
35          # (bs * 10) x dim
36          response_mask = response_mask.view(-1, response_mask.size(2))
37          with torch.no_grad():
38              # bs x dim
39              context_features = model(input_ids=context_text, attention_mask=context_mask)[0]
40              # (bs * 10) x dim
41              response_features = model(input_ids=response_text, attention_mask=response_mask)[0]
42
43              context_features = torch.mean(context_features,1)
44              response_features = torch.mean(response_features,1)
45
46              # bx x 10 x dim
47              response_features = response_features.view(-1, 10, response_features.size(1))
48              context_projection = projection(context_features)
49              # bs x 10 x dim
50              response_projection = response_features
51
52              # bs x 1 x dim
53              context_projection = context_projection.unsqueeze(1).repeat(1,10,1,)
54              similarities = torch.sum(context_projection * response_projection, dim=2)
55              # bs x 10
56              ranking = torch.argsort(similarities, dim=1, descending=True)
57              # print(ranking)
58
59              ranking_list.append(ranking.detach().cpu())
60
61      rankings = torch.cat(ranking_list)
62
63      return recall(rankings, topk=(1, 2, 3, 4, 5))
```

Calculate context and responses embeddings

Project context embedding

Take dot product and rank responses according to it.

Calculate top 1,2,3,4,5 Recall values from rankings

References

- **The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems**, Lowe, Ryan and Pow, Nissan and Serban, Iulian and Pineau, Joelle, 2015, *Association for Computational Linguistics*, https://www.aclweb.org/anthology/W15-4640
- **Pipeline:** http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/
- **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding :** https://arxiv.org/abs/1810.04805
- **ALBERT: A Lite BERT for Self-supervised Learning of Language Representations:** https://arxiv.org/abs/1909.11942

IR Based Chatbot

Progress I

Group 15
Yash Khasbage - CS17BTECH11044
Puneet Mangla - CS17BTECH11029
Mentor: Suvodip Dey

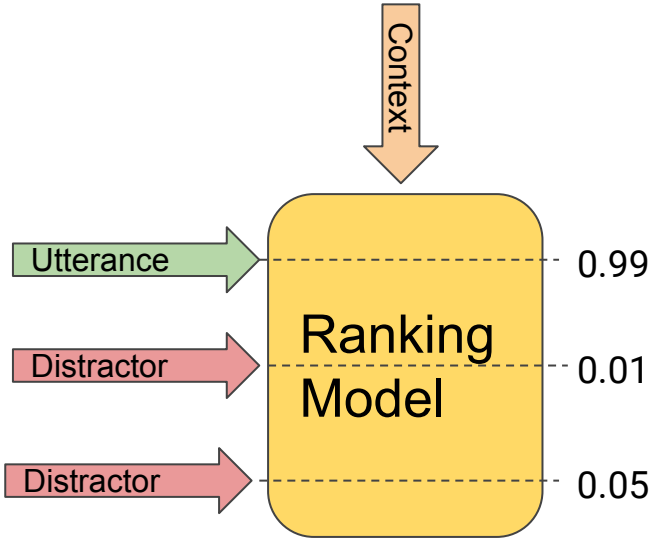
Recall: Problem Statement

The task is to build a chatbot.

- *Context:* Conversation upto this point
- *Utterance:* Correct response to context.
- *Distractor:* Incorrect response to context

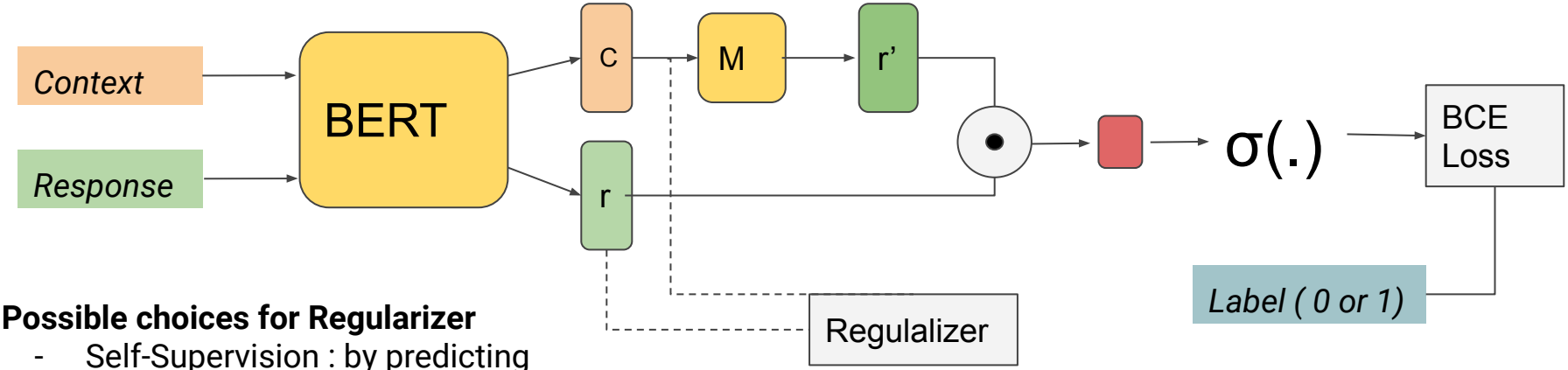
Develop a ranking model

- to assign highest score to true utterance
- lower scores to distractors



Overview of Approach

Posing it as a binary classification task



Possible choices for Regularizer

- Self-Supervision : by predicting masked tokens in input text.

BERT is fine tuned

Implementation details

Transformers

Frameworks and Libraries

- PyTorch
- Hugging Face - Transformers Library
- NLTK

Training Details

- Single GTX 1080 GPU - 16 GB
- *Batch Size* - 64
- *Input Length* - 128 tokens (padded/truncated if needed)
- *# Training samples:* 50K
- *# Evaluation samples:* 5K
- *Learning rate :* 2e-5

BERT Config

- *Activation:* GELU
- *# hidden layers :* 12, hidden size: 768
- *Attention Heads:* 12
- *Vocabulary:*
  - Initially Size is 30522
  - Added some Linux related vocab
  - Added tokens like “__eou__”, “__eot__”, “__dialog_end__”
  - *Final size* - 50155

Data Loaders

load_data.py | vocab.py | a.txt | vocabulary.txt | main.py | eval.py | io_utils.py

```
37
38      def __getitem__(self, index):
39          if self.split == 'train':
40              context, response, label = self.dataset.iloc[index]
41              context, response  = word_tokenize(context), word_tokenize(response)
42
43              # print(len(context))
44              # print(self.tokenizer.tokenize(context))
45              context = self.tokenizer.encode_plus(
46                              context,
47                              add_special_tokens=True,                    # Sentence to encode.
48                              max_length = self.max_length,           # Pad & truncate all sentences.
49                              pad_to_max_length = True,
50                              return_attention_mask = True,   # Construct attn. masks.
51                              truncation=True,
52                              return_tensors = 'pt',     # Return pytorch tensors.
53                      )
54
55              # print(context['input_ids'].shape)
56              # print(tokenizer.convert_ids_to_string(context['input_ids']))
57
58              response = self.tokenizer.encode_plus(
59                              response,
60                              add_special_tokens=True,                        # Sentence to encode.
61                              max_length = self.max_length,           # Pad & truncate all sentences.
62                              pad_to_max_length = True,
63                              return_attention_mask = True,   # Construct attn. masks.
64                              truncation=True,
65                              return_tensors = 'pt',     # Return pytorch tensors.
66                      )
67
68              label = int(label)
69              return context['input_ids'].squeeze(0), context['attention_mask'].squeeze(0), \
70              response['input_ids'].squeeze(0), response['attention_mask'].squeeze(0), label
```

Tokenize using NLTK

BERT-Tokenizer

- Encode using vocabulary
- Padd/Truncate to 128 length
- Return PyTorch Tensors
- Add tokens like [CLS], [UKN] etc.

Return input *encodings* and respective *attention masks* and *label*

Training Script

```python
def train(train_dataloader, model,projection, classifier, args):
    # Prepare optimizer and schedule (linear warmup and decay)
    criterion = torch.nn.BCELoss()
    no_decay = ['bias', 'LayerNorm.weight']

    optimizer = AdamW([{'params':model.parameters()},{'params':projection.parameters()},{'params':classifier.parameters()}], \
                      lr=args.learning_rate, eps=args.adam_epsilon)
    t_total = len(train_dataloader)*args.num_train_epochs
    scheduler = get_linear_schedule_with_warmup(optimizer, args.warmup_steps, t_total)

    for epoch in range(args.start_epoch,args.num_train_epochs):
        correct,total = 0,0
        avg_loss = 0.
        for i, (context_in, context_mask, response_in, response_mask, label) in enumerate(train_dataloader):
            model.train()
            # print(context_in.shape, context_mask.shape, response_in.shape, response_mask.shape)
            context_feature  = model(input_ids=context_in.cuda(), attention_mask=context_mask.cuda())[0]
            response_feature  = model(input_ids=response_in.cuda(), attention_mask=response_mask.cuda())[0]
            context_feature, response_feature = torch.mean(context_feature,1), torch.mean(response_feature,1)
            logit = F.sigmoid(torch.sum(projection(context_feature)*response_feature,1))
            # logit = F.sigmoid(classifier(concat_feature)).squeeze(1)

            pred =  (logit>0.5).long()
            correct += (pred == label.cuda()).sum().item()
            total += pred.size(0)
            loss = criterion(logit,label.cuda().float())
            loss = loss.mean()
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            avg_loss += loss.data.item()
            scheduler.step()

            if i % 10 == 0:
                print('Epoch {:d} | Batch {:d}/{:d} | Loss {:3f} | Accuracy {:3f}'.format(epoch, i, len(train_dataloader), \
                    avg_loss/(i+1), (100.*correct)/total))

        if (epoch % args.save_freq == 0) or (epoch == args.num_train_epochs - 1):
            outfile = os.path.join(args.checkpoint_dir, '{:d}.tar'.format(epoch))
            state_dict = {}
            state_dict['epoch'] = epoch
            state_dict['feature'] = model.state_dict()
            state_dict['projection'] =  projection.state_dict()
            torch.save(state_dict, outfile)
```

Binary Cross Entropy Loss

AdamW Optimizer with Learning rate: 2e-5

Linear Learning rate Scheduler

Sample a batch

Get BERT embeddings. We take mean of all hidden embeddings

Pass context embedding to projection, take dot product and sigmoid.

Optimize Loss

Save checkpoint

Evaluation Script

load_data.py | vocab.py | a.txt | vocabulary.txt | main.py | eval.py

```python
def evaluate(dataloader, model, projection, args):
    model.eval()
    projection.eval()

    ranking_list = list()
    for batch in dataloader:
        # bs x dim
        context_text = batch[0].cuda()
        # bs x dim
        context_mask = batch[1].cuda()
        # bs x 10 x dim
        response_text = batch[2].cuda()
        # bs x 10 x dim
        response_mask = batch[3].cuda()
        # (bs * 10) x dim
        response_text = response_text.view(-1, response_text.size(2))
        # (bs * 10) x dim
        response_mask = response_mask.view(-1, response_mask.size(2))
        with torch.no_grad():
            # bs x dim
            context_features = model(input_ids=context_text, attention_mask=context_mask)[0]
            # (bs * 10) x dim
            response_features = model(input_ids=response_text, attention_mask=response_mask)[0]

            context_features = torch.mean(context_features,1)
            response_features = torch.mean(response_features,1)

            # bx x 10 x dim
            response_features = response_features.view(-1, 10, response_features.size(1))
            context_projection = projection(context_features)
            # bs x 10 x dim
            response_projection = response_features

            # bs x 1 x dim
            context_projection = context_projection.unsqueeze(1).repeat(1,10,1,)
            similarities = torch.sum(context_projection * response_projection, dim=2)
            # bs x 10
            ranking = torch.argsort(similarities, dim=1, descending=True)
            # print(ranking)

            ranking_list.append(ranking.detach().cpu())

    rankings = torch.cat(ranking_list)

    return recall(rankings, topk=(1, 2, 3, 4, 5))
```

Calculate context and responses embeddings

Project context embedding

Take dot product and rank responses according to it.

Calculate top 1,2,3,4,5 Recall values from rankings

Current Results

Baselines:

1. Randomly selects the utterance among all responses given
2. Dual-Encoder LSTM

Results : our model is able to perform better than random baseline

Recall	Random	LSTM	Ours	
Top1	0.1	0.49	0.235	0.27
Top2	0.2	0.68	0.371	0.41
Top5	0.5	0.91	0.703	0.74

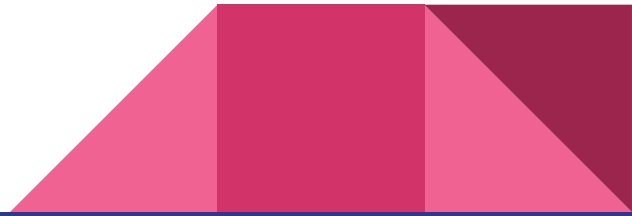
Conclusions/Explanations :

- Original baseline was trained on 1M training points. Ours was trained on 50K
- BERT has limited linux related vocabulary.

Next Steps

Our next steps would be..

- Scrape linux related vocabulary and concat with BERT one.
- Train on more training samples 100K-200K
- Use self-supervision regularizers



IR Based Chatbot

Group 15
Yash Khasbage - CS17BTECH11044
Puneet Mangla - CS17BTECH11029
Mentor: Suvodip Dey

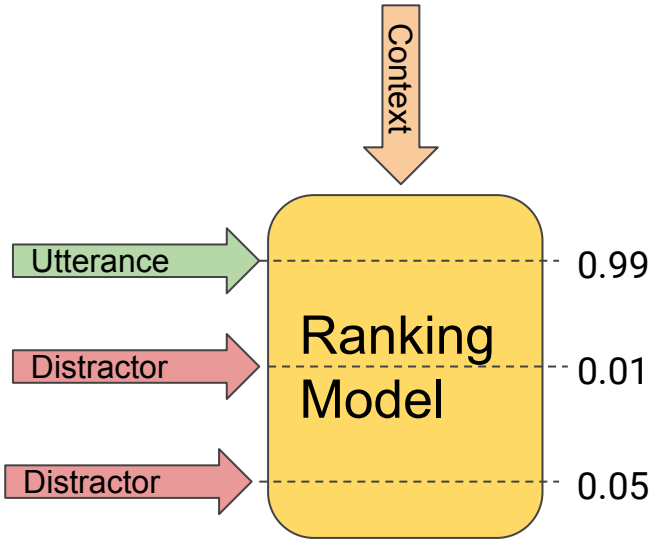
Problem Statement

The task is to build a chatbot.

- *Context:* Conversation upto this point
- *Utterance:* Correct response to context.
- *Distractor:* Incorrect response to context

Develop a ranking model

- to assign highest score to true utterance
- lower scores to distractors



Dataset description: Ubuntu Dialogue Corpus

- Consists of chats from Ubuntu-related chat from Freenode IRC for various technical support.

Train: 50% positive and 50% negative samples

Context	*Response*	*Flag (0 or 1)*

Test and Validation:

Context	*True Response*	*False Response 1*	….	*False Response 9*

examples:

Split	Train	Validation	Test
# examples	1,000,000	19,561	18,921

url: https://github.com/rkadlec/ubuntu-ranking-dataset-creator

Dataset Snapshots

5 samples from

Basic Processing like Stemming, Lemmatization, Stop-word removal are already done

Train:

Context	Utterance	Label
i think we could import the old comment via rs...	yes . same binari packag . __eou__	1
i 'm not suggest all - onli the one you modifi...	ok let me tri that .. thank man __eou__	0
afternoon all __eou__ not entir relat to warti...	http : //www.ubuntu.com/download/ -- ani mirro...	0
interest __eou__ grub-instal work with / be ex...	i fulli endors this suggest < /quimbi > __eou_...	1
and becaus python give mark a woodi __eou__ __...	ok if your defaultdepth 24 __eou__	0

Validation:

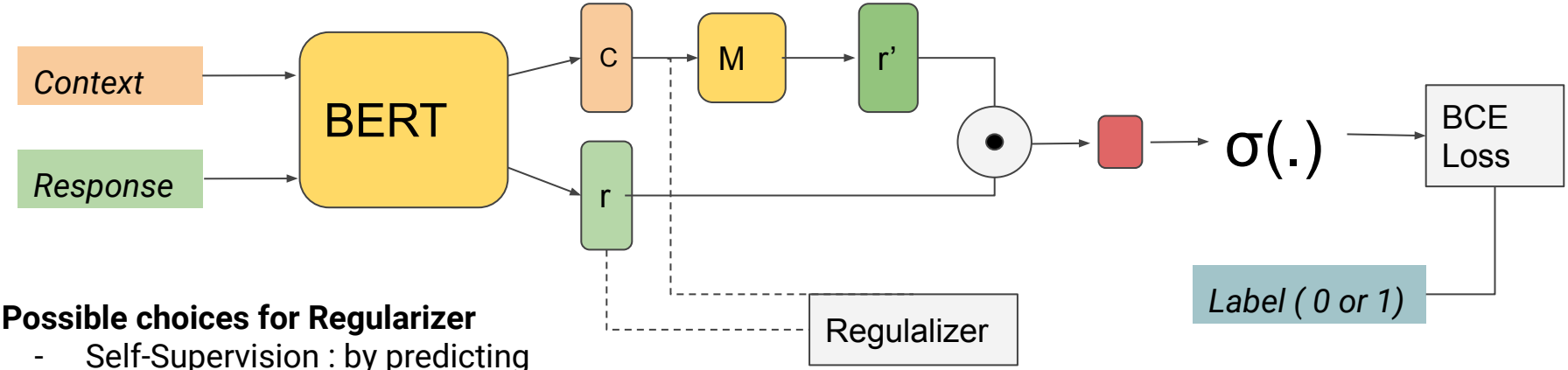
Context	...	Distractor_8
ani idea on how lts will be releas ? __eou__ _...	...	i be want to extract a split .rar file . i be ...
how much hdd use ubuntu default instal ? __eou...	...	umm . you mean tempfil be program and tmpfile ...
in my countri it near the 27th __eou__ when wi...	...	ah ok. it work now . u be right , i do a mista...
it 's not out __eou__ __eot__ they probabali b...	...	it look like that articl tell you how to setup...
be the ext4 driver stabl ? __eou__ __eot__ i b...	...	jeah iam in the bumblebe room ;) __eou__ coul...

Test:

Context	...	Distractor_8
anyon know whi my stock oneir export env var u...	...	there be a variabl path (run `` echo $ path '...
i set up my hd such that i have to type a pass...	...	first place i look __eou__
im tri to use ubuntu on my macbook pro retina ...	...	backport be a repo you can enabl in your packa...
no suggest ? __eou__ link ? __eou__ how can i ...	...	you could re-encod it with mencod or ffmpeg __...
i just ad a second usb printer but not sure wh...	...	if that be the case , they whi on earth would ...

Approach

Posing it as a binary classification task



Possible choices for Regularizer

- Self-Supervision : by predicting masked tokens in input text.

BERT is fine tuned

Use SVMs instead of logistic regression

References

- **The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems**, Lowe, Ryan and Pow, Nissan and Serban, Iulian and Pineau, Joelle, 2015, *Association for Computational Linguistics*, https://www.aclweb.org/anthology/W15-4640
- **Pipeline:** http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/