Capstone Project Report - for ML/AI training track at SpringBoard

Experiments on Conversational Questioning and Answering (CQA)

Student: Wei Wesley Chen
Advisor: Dr. Greg D. Zhang

1. Introduction
    1.1. Disclaimer: this is a required capstone project for machine learning/AI track I signed up for. Due to the time and resource constraints, the depth and breath of experiments are not quite what I was hoping for. The lessons learned and experiences encountered hopefully may be helpful to others in a similar learning process on the subject, especially on conversational QA, BERT, Tensorflow 2, etc.
    1.2. CQA: conversational questioning and answering is a task to predict/generate the answer based on given context text and the previous questions and answers. This can be modeled as a combination of conversational response generation problem and reading comprehension problem.
    1.3. Pretrained language models:
        1.3.1. Most NLP (natural language processing) tasks can start with a pretrained language models such as word embeddings or of late with pre-trained language representation models because of the vastness and complexity of human language. These pretrained models are non task specific, and are trained based some intrinsic properties of the language as training target to derive a way to represent the language. Such as word2vec is based on the patterns that words appear in context of other words, and BERT is based fact the masked out words can be predicted by its sentence environment and next sentence can logically derived a lot times. Common word embeddings are word2vec, Glove, etc. Common pretrained language models are ELMO, BERT, GPT, XLNet etc. Here we use BERT as our pretrained language model with various parameters.
        1.3.2. BERT: BERT stands for Bi-directional Encoder Representation with Transformer. Google released various pre-trained models so we can use them to do fine tuning based transfer learning to target a specific task. Since TF2 release, TF2 compatible versions has been available too.

    1.4. Dataset: CoQA dataset is a large-scale dataset created by Stanford University for building Conversational Question Answering systems.
        1.4.1. Goals of CoQA dataset:

- To understand the succinct nature of questions in a human conversation because of conversation history
- To ensure the naturalness of answers in a conversation
- To enable building QA systems that perform robustly across domains

1.5. Linguistics concepts relevant to CoQA dataset:

1.5.1. Ways to derive an answer:

- lexical match: an answer can be found by matching lexical parts, words or tokens. (contains at least one content word that appears in the given)
- Paraphrasing: an answer is derived from the source by paraphrasing the source but not the same in a lexical sense.
- Pragmatics: an answer is derived by understanding.

1.5.2. Types of relationship between a question and its conversation history:

- No co-reference
- Explicit co-reference
- Implicit co-reference

1.5.3. CoQA answer types:

- Yes
- No
- Fluency:
- Counting
- Multiple Choices

2. Scope Definition and Approaches:

2.1. Task definition: For this capstone project, we will use CoQA dataset to explore various QA system related techniques in many aspects of deep learning, namely, data analysis, data preprocessing, learning model design, training considerations, deployment considerations, etc.

2.2. Approaches:

2.2.1. Dataset investigation, preprocess logic and featurization logic

- CoQA dataset is a strongly supervised dataset, it provides a gold answer plus supporting text span as rationale. This makes models into three categories:
  - Extractive only
  - Abstractive only
  - Combined Extractive and Abstractive.
- Current leaders on the scoreboard seems all extractive-only and has surpassed human performance, but I think the abstractive approaches using weak supervision would be more challenging and meaningful.
- Tokenization techniques: There are many ways to tokenize a piece of text, besides the simple split() function and regular

expressions, there are canned tools and packages (SpaCy, Gensim, NLTK) too. No matter how big the dictionary is, OOV (Out Of Vocabulary) is still a problem plus the size of the vocabulary can be a problem too. BPE (Byte Pair Encoding) and WordPiece are two commonly used techniques to alleviate this problem algorithmically, and to gain the flexibility of limiting dictionary size and reducing OOV. BERT uses WordPiece with a 30522 tokens dictionary.

- Input sequence size limitation problem: since we have a limit on the input sequence size, if we have a longer context text, in order to make IID (Identical Independent Distributed) training examples, A Sliding Window technique is used to solve the problem. This solution creates another problem. A token can appear in multiple windows, and a maximum context flag can be used to indicate this. A maximum context score of a token is defined as the *minimum* of its left and right context(token counts) (the *sum* of left and right context will always be the same).

2.2.2. Strategies/Architectures/modeling techniques considered:

- Direct supervised learning vs fine tuning style transfer learning

  The CoQA dataset is big enough, any applicable supervised learning algorithm can be used to tackle the task. The following are architectures considered.
- Abstractive: directly use seq2seq: question + context text + QA history → seq2seq models → gold answers
- Abstractive:BERT based seq2seq:

  question+context text + QA history → BERT + decoders → gold answers
- Extractive: BERT based span only: question+context text + QA history → BERT + Logistic Regression → Span text (ignore gold answers)
- Combined: BERT based using both span and gold answer as targets:
  - BERT → span start and end: span reconstruction loss
  - BERT→ Decoder → gold answer : answer text reconstruction loss
- Techniques considered or used in experiments:
  - Simple LSTM encoder/decoder
  - Bidirectional LSTM (encode) and LSTM (decoder)
  - Attention
  - Coverage
  - Pointer (copy over)
  - Feature Engineering: Rationale tagging

- Data augmentation: Yes/No/unknown as targets to be combined with span prediction
- Data augmentation: refine the span based on gold answer
- Adversarial and Virtual Adversarial Training
- Knowledge Distillation: teacher/student knowledge transfer

3. Experimental Evaluations

3.1. Methodology

3.1.1. Due to the difficulties encountered, all abstractive approach experiments did not yield any meaningful results. Here we only listed span based experiments.

3.1.2. CoQA uses macro-average F1 score of word overlap as its evaluation metric.

3.1.3. Base model (BERT based span only): CoQA has a series of Questions (Q) and Answers (A), and a context text (C), QA history can be expressed as QAs = [Q1,A1,...Qk-1,Ak-1], since we used BERT model, the input is in format [CLS]Qk[SEP]C,QAs[SEP]. Another way to format input would be [CLSQAs,Qk[SEP]C[SEP], but experiment shows it did help to produce better results. We use the BERT sequence out with FC layers to predict the span start and end.

3.1.4. Data Augmentation 1: Data analysis shows there are significant of answers are Yes, No or Unknown, for extractive approach, we need to make these answers available to extract. We implement three separate target classes to be combined with span locations as the training target, thus augment the data. We use the pooled output from BERT with FC layer to predict Yes/No/Unknown classes.

3.1.5. Data Augmentation 2: The training dataset given a text span as rationale to arrive the gold answer, since we are on extractive approach only, we can refine the text span to help improve the F1 score, so we use the best F1 span text as training target.

3.1.6. Feature engineering: Rationale Tagging - since rationale text span comes from input context text, so a token is or isn't in rationale can be valuable information. We can not just predict the boundary of the span we can also add a training target (task) which is to predict whether a token is in the rationale or not.

3.2. Results:

3.2.1. BERT layers: (6 layers, 2 epochs, with data augmentations)

BERT Layers	6	8	10	12
F1 score(in domain)	63.1	67.2	68.6	69.2

3.2.2. Training epochs: (BERT 8 layers with data augmentations)

Epoches	1	2	4	6
F1 score(in domain)	67.1	67.2	64.4	59.8

3.2.3. Ablation study on techniques: (BERT 6 layers, 2epochs)

Models	**F1 Score**
Base (BERT span only)	39.2
Base + Yes/No/Unknown	47.1
Base+Y/N/U+Rationale Tagging	47.4
Base+Y/N/U+Rationale Tagging+Span Rewrite	63.1

3.3. Discussion

From the experiment results, we can see that:

- The BERT transformer layers is a key parameter that can affect the BERT model's learning capability.
- The models are very powerful for classification tasks. And CoQA dataset is large enough, so increase training epochs can quickly lead to overfitting.
- Various techniques can improve experiment results. so far data augmentations demonstrate the most improvements.

I started with abstractive approaches, have gone through a few models, but I encountered various issues with TensorFlow 2.0 and training on GPU. Extractive approaches is getting very good results based on CoQA dataset, but I am not sure these extractive approaches has advanced the original intent of the CoQA team, which is to develop more natural answering models with better understanding of the context and questioning history. I would recommend ignore the span information provided in CoQA focus only on gold answers. This will force more effort on abstractive approaches and hopefully will have more impact to QA task solutions.

4. Lessons Learned (rant on TF2): TensorFlow 2.0 is great trying to bring Pythonic coding style into TensorFlow graph based coding by using eager mode, autograph and tf.function, maybe it is just me, but I still can not get any custom Keras layers with LSTM or Conv2D to train on my GPU. So maybe we have to wait for TF2 to catch up with the promise of making Tensorflow programming pythonic. And Keras API implementation feels like not really matured in my experience. One really needs to pay attention to data flow interface from dataset to training loop then to Keras layer Call. I really don't like the single input convention on layer call function.

5. Future Work:
   For me personally, my future work on this would be to get abstractive models to work. Generally speaking, I feel that DNC type of working memory idea may be helpful to gain more reasoning capability. Also reinforcement learning can be helpful in decoding in abstractive approach. For seq2seq models, reinforcement learning has been introduced to reduce the training and testing model difference and reduce exposure bias introduced by the feedback of prior time step prediction as input for the next time step.

6. A TF serving/Django/React/Redux tool is developed to demonstrate this work. See https://github.com/wweschen/coqa-client