Problem Overview

Motivation
We tackle Natural Questions (NQ), a question answering (QA) challenge released by Google. QA is an important natural language processing task where a system, given a question and a context document, returns the correct answer to the question.

Task
Given a question and a relevant Wikipedia page, our task is to:
1. Identify the long answer (if any) from the document
2. Identify the short answer span (if any) from the document

Overview
We focus on predicting long and short answers using a BERT Bi-GRU model that is computationally more efficient.

Related Work
- DecAtt+DocReader Baseline (Google)
  - Long Answer Selection: Decomposable Attention
  - Short Answer Select: Document Reader (DrQA)
- BERT Based Baseline (Google)
  - Sliding context window generating multiple instances with overlapping text
  - Application of pre-trained BERT model

Dataset & Output

Dataset
The NQ dataset contains real queries issued to the Google search engine and a corresponding Wikipedia article. An example takes the form (question, wikipedia page, long answer candidates, annotation).

The NQ dataset totals 42GB larger than existing popular QA datasets (SQuAD 2.0 is 44MB). We aggressively downsampled:
- Training Data
  - 115K training instances (2 per training example)
- Test Data
  - Used 2007830 development examples as dev set
  - Used full 7830 development examples as test set

Output
We output the start, end location and answer type.

Model Architecture

Layers
1. TF-IDF Candidates Retrieval
   Selects top candidates by cosine similarity of tf-idf values of query and candidates
   \[ f(t, d) = 0.5 + 0.5 \cdot \frac{f(t)}{\max [f(d): t \in d]} \]
   \[ \text{idf}(t, D) = \log \left( \frac{|d \in D : t \in d|}{N} \right) \]

2. BERT Layer
   We use the BERT-base-uncased model. BERT uses Transformer Architecture which has a "Multi-Head Attention" block. The Multi-Head attention block computes multiple attention weighted sums; attention is calculated by:
   \[ \text{Attention}(h_{i}) = \sum_{j} \text{softmax}(W_{h}h_{i}W_{h}h_{j}) \]

3. Bi-GRU Output Layer
   We sequentially predict the start location, end location and answer type in 3 different output layers. Each output layer consists of a bi-GRU layer followed by a feed forward layer.

Loss Function
The loss of our model is defined as follows:
\[ L = H(l_s, s) + H(l_e, e) + H(l_t, t) \]
where \( H(x, y) = -x \cdot \log(y) + \log(\sum z \exp(z)) \) is the cross entropy loss between x and class, \( l_s, l_e, l_t \in \mathbb{R}^{D} \) is the output of the linear layer corresponding to the start/end position, and \( s, e, t \) are the ground truth of start/end position from the training example annotation.

Results & Discussion

Loss Function
The loss of our model is defined as follows:
\[ L = H(l_s, s) + H(l_e, e) + H(l_t, t) \]
where \( H(x, y) = -x \cdot \log(y) + \log(\sum z \exp(z)) \) is the cross entropy loss between x and class, \( l_s, l_e, l_t \in \mathbb{R}^{D} \) is the output of the linear layer corresponding to the start/end position, and \( s, e, t \) are the ground truth of start/end position from the training example annotation.

Reference
[1] Yiming Chen, Rena Sha
{yilingc, renasha}@stanford.edu

Dataset & Output

Output
We output the start, end location and answer type.

Ablation Analysis

Loss Function
The loss of our model is defined as follows:
\[ L = H(l_s, s) + H(l_e, e) + H(l_t, t) \]
where \( H(x, y) = -x \cdot \log(y) + \log(\sum z \exp(z)) \) is the cross entropy loss between x and class, \( l_s, l_e, l_t \in \mathbb{R}^{D} \) is the output of the linear layer corresponding to the start/end position, and \( s, e, t \) are the ground truth of start/end position from the training example annotation.

Contribution
- A BERT based Bi-GRU model that is computationally more efficient than current work and outperforms the DecAtt+DocReader baseline in long answer prediction.
- Experimentation and analysis of the model

Lessons Learned
- Information retrieval techniques such as tf-idf can be used to reduce scope in question answering
- Bi-GRU’s ability to retain features across sequential input allows for better performance

Future Work
- Test out sliding context window data pre-processing.
- Train network with more data and training time.

Evaluation
- Outperform the Google DecAtt+DocReader Model in the long answer prediction task
- Not performing as well on the short answer prediction task
- Evaluates much faster! 2hr on M60 GPU vs. 5hr on P100 GPU