Extractive question answering with RoBERTa

Eamonn Tweedy, PhD

Extractive Q & A task

One important task for language models is *extractive question answering*:

- The user provides a question and a *context* paragraph which (hopefully) contains the answer to the question
- The model model extracts the answer from the context by predicting the most likely range of token positions containing the answer
- Popular for training and benchmarking: SQuAD[1]. SQuAD v2 contains:
  - 100,000 (question,context,answer) triples
  - 50,000 (question,context) pairs which are "unanswerable", i.e. for which the context doesn't contain the answer to the question

[1] Stanford Question Answering Dataset

BERT models

The BERT[2] model was proposed in 2018 by a team at Google AI Language[3].

- BERT is an encoder-only model based on the Transformer[4] architecture
- Pre-trained on masked language modeling and next-sentence prediction
- Can then be fine-tuned for a variety of tasks, such as
  - sentence classification (e.g. sentiment analysis)
  - word classification (e.g. named entity recognition)
  - extractive question answering

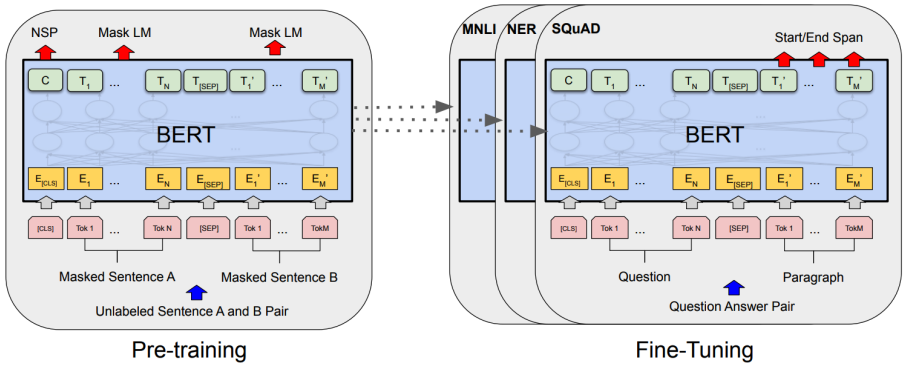

Figure: BERT model diagram from the original paper[3]

[2] Bidirectional Encoder Representations from Transformers

[3] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019)

[4] Attention is All you Need (Vaswani et al., NIPS 2017)

A sample from SQuAD

SQuAD consists of questions posed by crowdworkers on a set of Wikipedia articles, e.g.

Context: "The 1973 oil crisis began in October 1973 when the members of the Organization of Arab Petroleum Exporting Countries (OAPEC, consisting of the Arab members of OPEC plus Egypt and Syria) proclaimed an oil embargo. By the end of the embargo in March 1974, the price of oil had risen from US$3 per barrel to nearly $12 globally; US prices were significantly higher..."

Question: "What was the price of oil in March 1974?"

Ground truth answer: "$12"

Longer context samples are cut into overlapping pieces to create several features

- analogous to feature engineering
- context chunks overlap (length 384, stride 128)

Feature sets from a sample

⟨s⟩According to Apple, how fast can the Thunderbolt port transfer data?⟨/s⟩⟨/s⟩The current Mac product family uses Intel x86-64 processors. Apple introduced an emulator during the transition from PowerPC chips (called Rosetta), much as it did during the transition from Motorola 68000 architecture a decade earlier. The Macintosh is the only mainstream computer platform to have successfully transitioned to a new CPU architecture, and has done so twice. All current Mac models ship with at least 8 GB of RAM as standard⟨/s⟩

⟨s⟩ According to Apple, how fast can the Thunderbolt port transfer data?⟨/s⟩⟨/s⟩ and has done so twice. All current Mac models ship with at least 8 GB of RAM as standard other than the 1.4 GHz Mac Mini, MacBook Pro (without Retina Display), and MacBook Air. Current Mac computers use ATI Radeon or nVidia GeForce graphics cards as well as Intel graphics built into the main CPU. All current Macs (except for the MacBook Pro without Retina Display) do not⟨/s⟩

⟨s⟩According to Apple, how fast can the Thunderbolt port transfer data?⟨/s⟩⟨/s⟩ main CPU. All current Macs (except for the MacBook Pro without Retina Display) do not ship with an optical media drive that includes a dual-function DVD/CD burner. Apple refers to this as a SuperDrive. Current Macs include two standard data transfer ports: USB and Thunderbolt (except for the MacBook (2015 version), which only has a USB-C port and headphone port). MacBook Pro,⟨/s⟩

⟨s⟩According to Apple, how fast can the Thunderbolt port transfer data?⟨/s⟩⟨/s⟩ MacBook (2015 version), which only has a USB-C port and headphone port). MacBook Pro, iMac, MacBook Air, and Mac Mini computers now also feature the "Thunderbolt" port, which Apple says can transfer data at speeds up to 10 gigabits per second. USB was introduced in the 1998 iMac G3 and is ubiquitous today, while FireWire is mainly reserved for high-performance devices⟨/s⟩

- each feature is assigned target label $(\text{start position}, \text{end position})$ indicating the position span of the answer
- if the context chunk doesn't contain the answer, the target label assigned is $(0, 0)$
- inference involves predicting the most likely span pair for the answer

Fine-tuned RoBERTa model

I used a pre-trained variant of BERT called RoBERTa, proposed in 2019 by a team at Facebook AI[5], which has several improvements:

- significantly expanded training corpus (10x more than BERT)
- optimized architecture and hyperparameters

I used the `pytorch`, `datasets`, and `transformers` libraries to do the following:

- download, tokenize, and pre-process the SQuAD v2 dataset
- fine-tune a `roberta-base` (110 million parameters) model for 3 epochs:
  - AdamW optimizer
    - adaptive learning rates for individual weights
    - weight decay for regularization
  - linear learning rate scheduler with base rate of 3×10^{-5}

[5] A Robustly Optimized BERT Pre-training Approach with Post-training (Zhuang et al., CCL 2021)

Evaluation of the fine-tuned model

Question answering performance is often evaluated using two per-sample metrics comparing the tokens of the predicted and ground truth answers:

- Exact match (EM) score, which is equal to 1 if the tokens match exactly and 0 otherwise
- F_1-score, which is the harmonic mean of precision and recall scores and is typically more forgiving:
  - $\text{recall} = \dfrac{\#\text{ of tokens shared by predicted and ground truth answers}}{\text{token length of ground truth}}$
  - $\text{precision} = \dfrac{\#\text{ of tokens shared by predicted and ground truth answers}}{\text{token length of predicted answer}}$

How about answers with no tokens?

- If the ground truth (resp. predicted) has no tokens, we set recall (resp. precision) equal to EM.

	EM_total	F1_total	EM_has_ans	F1_has_ans	EM_no_ans
scores	80.459867	83.525435	78.694332	84.834259	82.220353

RoBERTa Q&A app with Wiki tools

`https://huggingface.co/spaces/etweedy/roberta-wiki`

An app which interacts with the model in three ways:

- basic extractive question answering
- question-answering with a user-guided Wikipedia search tool
  - user queries Wikipedia API and chooses article(s)
  - article(s) chunked and fed to RoBERTa as context
- question-answering with an automated Wikipedia search tool
  - user's question parsed to generate search query
  - article chunks ranked by Okapi BM25 similarity
  - most relevant chunks fed to RoBERTa as context

Thanks! Questions?