**Developer’s Manual for SensiMix, Ver. 1.0**

2021.08.03

Tairen Piao (piaotairen@snu.ac.kr)

U Kang (ukang@snu.ac.kr)

**매뉴얼 구성**

1. SensiMix 코드 개요
2. 코드 실행 과정
3. 세부 코드 모듈 별 설명

**1. SensiMix 코드 개요**

**文本

描述已自动生成**

Figure 1. SensiMix 코드의 구조도

SensiMix 코드 구조는 그림 1에서 확인할 수 있듯이, 크게 두 가지의 분류로 볼 수 있다. models 폴더에는 SensiMix 모델과 학습 과정에서 필요한 함수들을 구현한 코드이고, example 폴더에는 학습 및 추론을 진행하는 코드이다. models 폴더에 있는 configuration\_auto.py는 configuration을 로드하는 class가 구현되 있고, configuration\_mpqbert.py는 configuration(e.g., layer 개수)을 저장하는 파일이라고 보면 된다. modeling\_mpqbert.py와 modeling\_mpqbert\_infer.py는 거이 같은데, 그중 후자는 inference에 맞게 1/16비트로 추론할 수 있게 수정한 코드이다. optimization.py에는 모델을 학습할 때 필요한 Adam optimizer를 구현한 코드이다. quantized\_modules.py는 quantization module들을 구현한 코드이다. 예를 들어, 8-bit Index quantized embedding 레이어, 1-bit Linear 레이어 등이 구현되 있다. example 폴더에 있는 run\_gule\_new.py와 run\_glue\_inference.py도 거의 비슷한데, run\_glue\_new.py는 학습할 때만 사용되고, run\_glue\_inference.py는 inference할 때만 사용된다. scripts 폴더에는 쉽게 실행할 수 있는 shell script들이다. 기존에 있는 .sh파일들을 직접 사용할 수도 있고 custom한 shell script들도 만들 수 있다.

**2. 코드 실행 과정**

* 실행 환경
  + Ubuntu 16.0.4 LTS
  + Python 3.7
  + PyTorch 1.7.1 (CUDA 10.1)
  + transformers 2.5.1
* 실행 방법
  + 1. Github repository를 clone 해서 local에 저장한다
    - git clone https://github.com/snudm-starlab/SensiMix.git
    - cd SensiMix
  + 2. 필요한 패키지들을 설치한다
    - pip install -r requirements.txt
      * 다른 package들이 없으면 pip install로 설치한다
    - xnor 패키지를 설치한다
      * cd src/xnor/cuda
      * python setup.py install
  + 3. 간단한 demo를 실행한다
    - bash demo.sh

**3. 세부 코드 모듈 별 설명**

1. **configuration\_auto.py**: 모델configuration을 받아오는 구현체
   1. class AutoConfig  
      AutoConfig는 generic configuration class이다. 이 class는 sensimix혹은 다른 모델의 configuration을 받는 class이다.
2. **configuration\_mpqbert.py**: SensiMix configuration의 구현체
   1. class MPQBertConfig(PretrainedConfig)SensiMix의 configuration을 저장하는 class (예를 들어, vocab\_size, hidden\_size, num\_hidden\_layers등)
3. **modeling\_mpqbert.py**: SensiMix 모델의 구현체
   1. class MPQBertEmbeddings(nn.Module)  
      BERT embedding layer의 PyTorch구현체 (주로 quantized\_modules.py에 있는qEmbedding을 사용해서 구현)
   2. class MPQBertSelfAttention(nn.Module)  
      Self-Attention중의 Q, K, V와 Q, K, V 연산의 구현체
   3. class MPQBertSelfOutput(nn.Module)  
      W\_o의 구현체 (간단한 Linear 레이어)
   4. class MPQBertAttention(nn.Module)  
      Self-Attention 레이어의 구현체 (class MPQBertSelfAttention과 class MPQBertSelfOutput의 조합)
   5. class MPQBertIntermediate(nn.Module)  
      Transformer encoder의 feed-forward network의 첫 레이어의 구현체
   6. class MPQBertOutput(nn.Module):  
      Transformer encoder의 feed-forward network의 두번째 레이어의 구현체
   7. class MPQBertLayer\_MP\_encoder(nn.Module)  
      Transformer encoder의 PyTorch구현체, self-attention 모듈과 feed-forward network 레이어 모듈을 조합
   8. class MPQBertModel(nn.Module)SensiMix모델의 구현체, embedding layer, encoder and pooler
4. **modeling\_mpqbert\_infer.py**: SensiMix모델 inference 구현체
   1. class MPQBertEmbeddings(nn.Module)  
      BERT embedding layer의 PyTorch구현체 (주로 quantized\_modules.py에 있는qEmbedding을 사용해서 구현)
   2. class MPQBertSelfAttention(nn.Module)  
      Self-Attention중의 Q, K, V와 Q, K, V 연산의 구현체
   3. class MPQBertSelfOutput(nn.Module)  
      W\_o의 구현체 (간단한 Linear 레이어)
   4. class MPQBertAttention(nn.Module)  
      Self-Attention 레이어의 구현체 (class MPQBertSelfAttention과 class MPQBertSelfOutput의 조합)
   5. class MPQBertIntermediate(nn.Module)  
      Transformer encoder의 feed-forward network의 첫 레이어의 구현체
   6. class MPQBertOutput(nn.Module):  
      Transformer encoder의 feed-forward network의 두번째 레이어의 구현체
   7. class MPQBertModel(nn.Module)  
      SensiMix모델의 구현체, embedding layer, encoder and pooler
5. **optimiation.py:** optimizer의 구현체
   1. def get\_linear\_schedule\_with\_warmup(optimizer, num\_warmup\_steps, num\_training\_steps, last\_epoch)   
      주어진 optimizer로부터 PyTorch schedule의 구현체, learning rate를 linear하게 decrease 하는 함수
   2. class AdamW(Optimizer)  
      Adam algorithm의 구현체 (weight decay를 추가한 버전)
6. **quantized\_modules.py:** quantization에 필요한 함수들의 구현체
   1. def Binarize(tensor)  
      input tensor를 binarize하는 함수
   2. def xnor\_linear\_inference(input, weight, bias)  
      xnor gemm의 구현체, input과 weight를 xnor gemm으로 곱하는 함수
   3. class BinarizeLinear\_inference(nn.Module)  
      xnor\_linear\_inference를 사용한 PyTorch Linear 레이어의 구현체
   4. def quantization (input, bits)  
      input tensor를 8-bit index quantization 및 dequantization을 진행하는 함수
   5. class q\_Linear(nn.Linear)  
      8-bit index quantization 을 적용한 PyTorch Linear 레이어의 구현체
   6. class mix\_Linear(nn.Module)  
      setting에 따라서 BinarizeLinear혹은 full-precision Linear 레이어를 선택할 수 있는 PyTorch Linear레이어 함수의 구현체
7. **run\_glue\_new.py:** 모델 학습 구현체
   1. def train(args, train\_dataset, model,tokenizer)  
      주어진 dataset, 모델과, tokenizer로부터 모델을 학습시키는 함수
   2. def evaluate(args, model, tokenizer)  
      주어진 모델과 tokenizer로부터 모델을 evaluate하는 함수
   3. def load\_and\_cache\_examples(args, task, tokenizer)  
      주어진 테스크 와 tokenizer로부터 모델에 직접 입력가능한dataset을 return하는 함수
8. **run\_glue\_inference.py:** 모델 추론 구현체
   1. def train(args, train\_dataset, model,tokenizer)  
      주어진 dataset, 모델과, tokenizer로부터 모델을 학습시키는 함수 (실제로 안 쓰는데 구현체는 run\_glue\_new.py와 같음)
   2. def evaluate(args, model, tokenizer)  
      주어진 모델과 tokenizer로부터 모델을 evaluate하는 함수
   3. def load\_and\_cache\_examples(args, task, tokenizer)  
      주어진 테스크 와 tokenizer로부터 모델에 직접 입력가능한dataset을 return하는 함수