トランスフォーマー向けインテル® エクステンションを利用した効率的な自然言語埋め込みモデル

検索拡張生成の効率化

Yuwen Zhou、Zhenzhong Xu、Xin He、Bo Dong、Wenxin Zhang、および Haihao Shen
インテル コーポレーション

自然言語埋め込みの概要

自然言語埋め込みは、自然言語処理（NLP）に不可欠なもので、テキストを意味情報をキャプチャーするベクトルとして表します。これらの埋め込みは計算操作に必要な数値入力を提供するため、さまざまな下流の NLP タスクにとって重要です。

数ある埋め込みモデルの中でも、BGE（BAAI General Embedding）（英語）は効率が非常に優れています。small（英語）および base（英語）バージョンは、速度と効率のバランスが良く、テキスト埋め込みの理想的な選択肢となっています。BGE はベクトル・データベースとシームレスに統合され、テキスト埋め込み以外でも、その可能性をさらに拡大しています。

この記事では、BGE small モデルの速度と精度を大幅に向上するオープンソース・ツールのトランスフォーマー向けインテル® エクステンション（英語）を利用して、埋め込みモデルのパフォーマンスを向上させる革新的なアプローチを紹介します。

INT8 静的なトレーニング後の量子化

静的なトレーニング後の量子化（PTQ）は、追加のトレーニング段階を量子化する効果的なアプローチです。モデルの量子化パラメーター（スケール、ゼロ点など）を決定するには、代表的なデータセットを使用したキャリブレーションが必要です。bge-small-en-v1.5（英語）に精度を考慮した自動チューニングを備えた PTQ を適用して、最適な量子化モデルを生成します。以下のコード例は、トレーニング後の量子化を活用して BGE small モデルを最適化する方法を示しています。CQADupStack（英語）データセットをキャリブレーションに使用し、MTEB（英語）STS タスクを評価ベンチマークとして使用します。完全なコードはこちらから（英語）入手できます（ドキュメントは、readme（英語）を参照してください）。

```
from intel_extension_for_transformers.transformers import metrics, objectives,
QuantizationConfig
from intel_extension_for_transformers.transformers.trainer import NLPTrainer
# Replace transformers.Trainer with NLPTrainer
# trainer = transformers.Trainer(.....)
trainer = NLPTrainer(.....)
metric = metrics.Metric(
    name="eval_accuracy", is_relative=True, criterion=0.01
)
objective = objectives.performance
q_config = QuantizationConfig(
    approach="PostTrainingStatic",
    metrics=[metric],
    objectives=[objective]
)
model = trainer.quantize(quant_config=q_config, eval_func=mteb_sts_eval)
```

BGE で高速な NLP 推論のパフォーマンスを引き出す

このセクションでは、精度を損なうことなくこれらの BGE モデルの推論を高速化するように設計されたハイパフォーマンス NLP バックエンドを紹介します。軽量のベアメタル推論バックエンドの Apple Neural Engine*(ANE) を活用して、圧縮 NLP モデルのパフォーマンスを引き出します。ハードウェアとソフトウェアの両方の最適化を活用して、パフォーマンスを最大化します。

開発のプロセスを合理化するため、トランスフォーマー向けインテル® エクステンションで、Hugging Face の使い慣れたトランスフォーマー API と使いやすいモデル圧縮ツールを拡張します。このシームレスな統合により、ユーザーは ANE の機能を活用し、NLP モデルを最適化して推論を高速化し、生産性を向上させることができます。

開始方法を次に示します。

```
from transformers import AutoTokenizer
from intel_extension_for_transformers.transformers import AutoModel

sentences_batch = ['sentence-1', 'sentence-2', 'sentence-3', 'sentence-4']
tokenizer = AutoTokenizer.from_pretrained('BAAI/bge-small-en-v1.5')
encoded_input = tokenizer(sentences_batch,
                          padding=True,
                          truncation=True,
                          max_length=512,
                          return_tensors="np")

engine_input = [encoded_input['input_ids'], encoded_input['token_type_ids'],
                encoded_input['attention_mask']]

model = AutoModel.from_pretrained('./model_and_tokenizer/int8-model.onnx',
                                  use_embedding_runtime=True)
sentence_embeddings = model.generate(engine_input)['last_hidden_state:0']
print("Sentence embeddings:", sentence_embeddings)
```

パフォーマンスの測定

MTEB STS 上で最適な量子化 BGE モデルを測定しました。すべてのモデルの精度相対損失は 1% 以内です(**表 1**)。

埋め込みレイテンシーとして、1 ソケット、24 コア / インスタンス、シーケンス長 = 512 の 1 つのインスタンス、バッチサイズ = 1 を使用して 1 つの文をエンコードする平均ミリ秒も測定しました（**図 1**)。

モデル	MTEB STS 平均 (10 データセット)	
	PyTorch* FP32	ITREX INT8
BAAI/bge-small-en-v1.5	81.59	81.43
BAAI/bge-base-en-v1.5	82.39	82.19
BAAI/bge-large-en-v1.5	83.11	82.82

表 1. 埋め込みモデルの FP32 と INT8 の精度

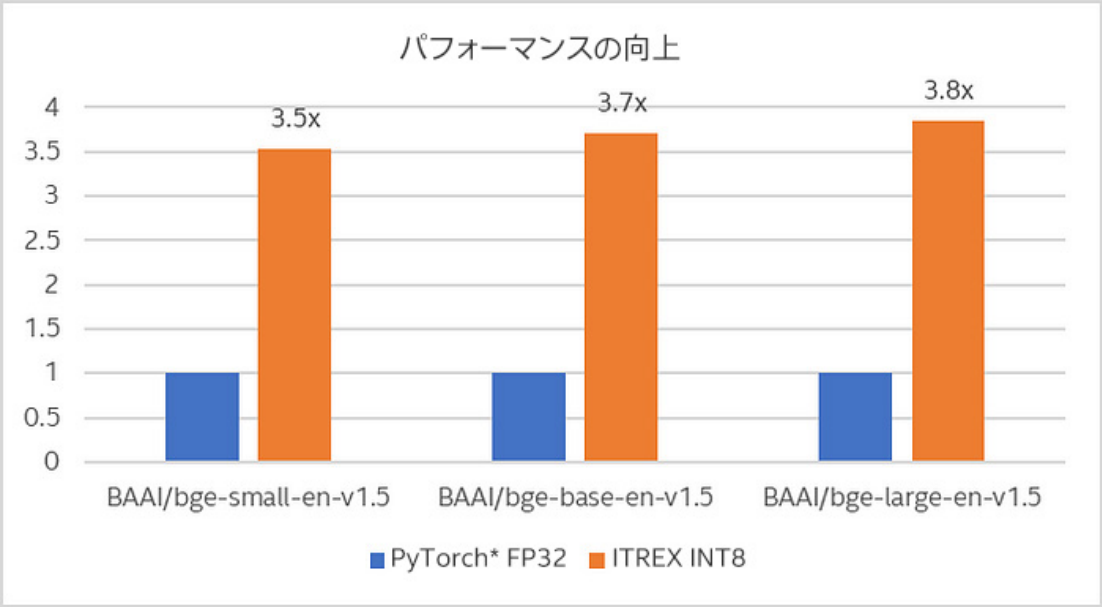

図 1. INT8 埋め込みモデルのパフォーマンスの向上

ハードウェア構成：インテル® Xeon® Platinum 8480+ プロセッサー、2 ソケット（ソケットあたり 56 コア）、2048GB RAM（16 スロット /128GB/4800MHz）、インテル® ハイパースレッディング・テクノロジー有効。OS : Ubuntu* 22.04.2LTS。**ソフトウェア構成：**Python* 3.9、NumPy* 1.26.3、ONNX* Runtime 1.13.1、ONNX* 1.13.1、Torch 2.1.0+cpu、Transformers 4.36.2。**テスト実施日：**2024 年 1 月 26 日。

最適化された BGE モデルを使用したチャットボットの構築

トランスフォーマー向けインテル® エクステンションで LangChain 埋め込み API を拡張し、次に示すようにユーザーが量子化された BGE モデルをロードできるようにしました。

```
from intel_extension_for_transformers.langchain.embeddings import
HuggingFaceBgeEmbeddings
embed_model = HuggingFaceBgeEmbeddings(model_name="/path/to/quantized/bge/model"
```

さらに、トランスフォーマー向けインテル® エクステンションの一部である NeuralChat と呼ばれるカスタマイズ可能なチャットボット・フレームワークを導入しました。このフレームワークを使用すると、複数のアーキテクチャー（インテル® Xeon® スケーラブル・プロセッサーやインテル® Gaudi® AI アクセラレーターなど）でチャットボットを迅速に構築できます。以下の例は、量子化された BGE small モデルを使用して知識検索向けのチャットボット・アプリケーションを開発する方法を示しています。

```
from intel_extension_for_transformers.neural_chat import plugins
from intel_extension_for_transformers.neural_chat import build_chatbot
from intel_extension_for_transformers.neural_chat import PipelineConfig

plugins.retrieval.enable = True
plugins.retrieval.args["input_path"] = "/path/to/docs"
plugins.retrieval.args["embedding_model"] = "/path/to/quantized/bge/model"
pipeline_config = PipelineConfig(model_name_or_path="facebook/opt-125m",
plugins=plugins)
chatbot = build_chatbot(pipeline_config)
response = chatbot.predict(query="What is Intel extension for transformers?")
```

まとめ

優れたパフォーマンスを達成するため、トランスフォーマー向けインテル® エクステンションを使用した埋め込みモデルの量子化と最適化の有効性を説明しました。品質を犠牲にすることなく推論の効率をさらに向上するため、ほかのモデル圧縮技術（プルーニングなど）も検討しています。ほかのインテル® AI ツールも試してみてください。最新の最適化に関する情報を受け取りたい場合は、トランスフォーマー向けインテル® エクステンションのリポジトリーに星を追加してください。プルリクエストを作成したり、リポジトリーに問題を送ることもできます。ご質問がある場合はお気軽にお問い合わせください。