INTRODUCTION

Google released Bert in 2018 and this transformer model has quickly become a standard for NLP tasks. Bert (and bert *based*) models allow for easy finetuning with only a single layer and not much task dependent architecture changes. One can import these massive models and fine tune on a small dataset and get state of the art performance. Due to this robustness (and Bert's massive availability via TFHub and Hugging Face) for many NLP tasks such as text classification, textual entailment, and Q/A Bert-*like* models have become a standard recipe.

As Bert models do not require massive changes, the recipes for fine tuning these models have become pretty standard. Two popular python packages allow users to do a variety of tasks with minimal code (10 lines). In this review we compare the packages FastBert and simpletransformer which provide convenient pipelines to train these models. We will compare the features, advantages, drawbacks and support of standard NLP tasks for these packages.

THE PROMISE

Both FastBert and simpletransformers offer similar services which is ease of training models with minimal code, so we compare the main features of the packages on three important stages of transfer learning on Bert(like) models namely:

1. Dataset Preparation
2. Model Instantiation
    a. Supported Tasks
    b. Hyper-parameter search
3. Model Saving/Deployment

1. Dataset Preparation

simpletransformers:
The package accepts pandas dataframes. The entire dataset needs to be loaded in memory and parallelisation requires as many copies as the number of workers. The package also performs the tokenization step at model instantiation stage and does not

have any API which saves the dataset which means that the tokenization step needs to be performed every rerun.

FastBert:
The dataset step has its own function and API (BertDataBunch). The BertDataBunch only accepts csv objects but it tokenizes and prepares data before model instantiation. It also has loaders for parallelization and in practice has a lower memory footprint than simpletransformers and nearly as flexible.

2. Model Instantiation

```
from fast_bert.learner_cls import BertLearner
from fast_bert.metrics import accuracy
import logging

logger = logging.getLogger()
device_cuda = torch.device("cuda")
metrics = [{'name': 'accuracy', 'function': accuracy}]

learner = BertLearner.from_pretrained_model(
                                            databunch,
                                            pretrained_path='bert-base-uncased',
                                            metrics=metrics,
                                            device=device_cuda,
                                            logger=logger,
                                            output_dir=OUTPUT_DIR,
                                            finetuned_wgts_path=None,
                                            warmup_steps=500,
                                            multi_gpu=True,
                                            is_fp16=True,
                                            multi_label=False,
                                            logging_steps=50)
```

```
from simpletransformers.classification import ClassificationModel, ClassificationArgs


model_args = ClassificationArgs(sliding_window=True)

model = ClassificationModel(
    "roberta",
    "roberta-base",
    args=model_args,
)
```

	FastBert	simpletransformer
Supported Models	BERT, RoBERTa, DistilBERT and XLNet	BERT, ALBERT, RoBERTa, DistilBERT, ELECTRA, XLM-RoBERTa, XLNet
Supported Tasks	Classification, Named Entity Recognition(beta)	Classification, QA, NER, Seq2Seq models, BART, Language Generation, Conversational AI
Hyper-parameter search	PyTorch learning rate finder	Not Supported
CustomLayer	Not Supported	Not Supported
MultiGPU	Supported	Supported (but error prone)
Serving	SageMake support	Not Supported
Custom Model PreTraining	Supported	Supported
Visualization Support	Not Supported	Supported (wandB)
Tensorboard Support	Supported(Beta)	Supported by default
Model Export (reusing with different libraries)	Not Supported	Not Suppported

Overall simpletransformer provides an easier interface for model instantiation. Moreover as dataset tokenization happens at model instantiation step it is easy to train multiple models with varying embedding sizes reusing the same pipeline.
FastTransformer has many necessary functions which are in development or Beta. The main differentiator apart from the set of tasks supported for the two packages is the support for visualisation platforms

DRAWBACKS

Both the packages provide a lot of convenience functions at the cost of flexibility. As CustomLayer is not supported in either of the packages we have to rely on the standard

implementations. Moreover converting models during inference to ONNX runtime for serving is a standard recipe for production deployment of Pytorch models but is not supported.
Overall the packages have really good support to start working on such models for baselines and initial explorations but currently support for involved pipelines does not seem possible.

THE VERDICT

Due to minimal architecture changes required for Bert-*like* models in supporting a variety of NLP tasks, a lot of the pipeline steps can be automated. This automation provides easy and fast implementations for these state of the art models at the cost of reduced flexibility. Between the two packages compared simpletransformers comes out as the clear winner as except it's data processing techniques, it supports more features, models and convenience functions.

While it is currently not possible to support large scale production ML pipelines entirely on these packages, it is still possible to explore and train baseline models very quickly.