Zero-shot classifier distillation

Author: @joeddav

This script provides a way to improve the speed and memory performance of a zero-shot classifier by training a more efficient student model from the zero-shot teacher's predictions over an unlabeled dataset.

The zero-shot classification pipeline uses a model pre-trained on natural language inference (NLI) to determine the compatibility of a set of candidate class names with a given sequence. This serves as a convenient out-of-the-box classifier without the need for labeled training data. However, for a given sequence, the method requires each possible label to be fed through the large NLI model separately. Thus for `N` sequences and `K` classes, a total of `N*K` forward passes through the model are required. This requirement slows inference considerably, particularly as `K` grows.

Given (1) an unlabeled corpus and (2) a set of candidate class names, the provided script trains a student model with a standard classification head with `K` output dimensions. The resulting student model can then be used for classifying novel text instances with a significant boost in speed and memory performance while retaining similar classification performance to the original zero-shot model

Usage

A teacher NLI model can be distilled to a more efficient student model by running `distill_classifier.py`:

```
python distill_classifier.py \
--data_file <unlabeled_data.txt> \
--class_names_file <class_names.txt> \
--output_dir <output_dir>
```

`<unlabeled_data.txt>` should be a text file with a single unlabeled example per line. `<class_names.txt>` is a text file with one class name per line.

Other optional arguments include:

- `--teacher_name_or_path` (default: `roberta-large-mnli`): The name or path of the NLI teacher model.
- `--student_name_or_path` (default: `distillbert-base-uncased`): The name or path of the student model which will be fine-tuned to copy the teacher predictions.
- `--hypothesis_template` (default `"This example is {}."`): The template used to turn each label into an NLI-style hypothesis when generating teacher predictions. This template must include a `{}` or similar syntax for the candidate label to be inserted into the template. For example, the default template is `"This example is {}."` With the candidate label `sports`, this would be fed into the model like `[CLS] sequence to classify [SEP] This example is sports . [SEP]`.
- `--multi_class`: Whether or not multiple candidate labels can be true. By default, the scores are normalized such that the sum of the label likelihoods for each sequence is 1. If `--multi_class` is passed, the labels are considered independent and probabilities are normalized for each candidate by doing a softmax of the entailment score vs. the contradiction score. This is sometimes called "multi-class multi-label" classification.
- `--temperature` (default: `1.0`): The temperature applied to the softmax of the teacher model predictions. A higher temperature results in a student with smoother (lower confidence) predictions than the teacher while a value `<1` resultings in a higher-confidence, peaked distribution. The default `1.0` is equivalent to no smoothing.

- `--teacher_batch_size` (default: `32`): The batch size used for generating a single set of teacher predictions. Does not affect training. Use `--per_device_train_batch_size` to change the training batch size.

Any of the arguments in the 🤗 Trainer's `TrainingArguments` can also be modified, such as `--learning_rate`, `--fp16`, `--no_cuda`, `--warmup_steps`, etc. Run `python distill_classifier.py -h` for a full list of available arguments or consult the Trainer documentation.

> ***Note**: Distributed and TPU training are not currently supported. Single-node multi-GPU is supported, however, and will run automatically if multiple GPUs are available.*

Example: Topic classification

> *A full colab demo notebook of this example can be found here.*

Let's say we're interested in classifying news articles into one of four topic categories: "the world", "sports", "business", or "science/tech". We have an unlabeled dataset, AG's News, which corresponds to this problem (in reality AG's News is annotated, but we will pretend it is not for the sake of example).

We can use an NLI model like `roberta-large-mnli` for zero-shot classification like so:

```
>>> class_names = ["the world", "sports", "business", "science/tech"]
>>> hypothesis_template = "This text is about {}."
>>> sequence = "A new moon has been discovered in Jupiter's orbit"

>>> zero_shot_classifier = pipeline("zero-shot-classification", model="roberta-
large-mnli")
>>> zero_shot_classifier(sequence, class_names,
hypothesis_template=hypothesis_template)
{'sequence': "A new moon has been discovered in Jupiter's orbit",
 'labels': ['science/tech', 'the world', 'business', 'sports'],
 'scores': [0.7035840153694153, 0.18744826316833496, 0.06027870625257492,
0.04868902638554573]}
```

Unfortunately, inference is slow since each of our 4 class names must be fed through the large model for every sequence to be classified. But with our unlabeled data we can distill the model to a small distilbert classifier to make future inference much faster.

To run the script, we will need to put each training example (text only) from AG's News on its own line in `agnews/train_unlabeled.txt`, and each of the four class names in the newline-separated `agnews/class_names.txt`. Then we can run distillation with the following command:

```
python distill_classifier.py \
--data_file ./agnews/unlabeled.txt \
--class_names_files ./agnews/class_names.txt \
--teacher_name_or_path roberta-large-mnli \
--hypothesis_template "This text is about {}." \
--output_dir ./agnews/distilled
```

The script will generate a set of soft zero-shot predictions from `roberta-large-mnli` for each example in `agnews/unlabeled.txt`. It will then train a student distilbert classifier on the teacher predictions and save the resulting model in `./agnews/distilled`.

The resulting model can then be loaded and used like any other pre-trained classifier:

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer
model = AutoModelForSequenceClassification.from_pretrained("./agnews/distilled")
tokenizer = AutoTokenizer.from_pretrained("./agnews/distilled")
```

and even used trivially with a `TextClassificationPipeline`:

```
>>> distilled_classifier = TextClassificationPipeline(model=model,
tokenizer=tokenizer, return_all_scores=True)
>>> distilled_classifier(sequence)
[[{'label': 'the world', 'score': 0.14899294078350067},
  {'label': 'sports', 'score': 0.03205857425928116},
  {'label': 'business', 'score': 0.05943061783909798},
  {'label': 'science/tech', 'score': 0.7595179080963135}]]
```

> *Tip: pass `device=0` when constructing a pipeline to run on a GPU*

As we can see, the results of the student closely resemble that of the trainer despite never having seen this example during training. Now let's do a quick & dirty speed comparison simulating 16K examples with a batch size of 16:

```
for _ in range(1000):
    zero_shot_classifier([sequence] * 16, class_names)
# runs in 1m 23s on a single V100 GPU
```

```
%%time
for _ in range(1000):
    distilled_classifier([sequence] * 16)
# runs in 10.3s on a single V100 GPU
```

As we can see, the distilled student model runs an order of magnitude faster than its teacher NLI model. This is also a seeting where we only have `K=4` possible labels. The higher the number of classes for a given task, the more drastic the speedup will be, since the zero-shot teacher's complexity scales linearly with the number of classes.

Since we secretly have access to ground truth labels for AG's news, we can evaluate the accuracy of each model. The original zero-shot model `roberta-large-mnli` gets an accuracy of 69.3% on the held-out test set. After training a student on the unlabeled training set, the distilled model gets a similar score of 70.4%.

Lastly, you can share the distilled model with the community and/or use it with our inference API by uploading it to the 🤗 Hub. We've uploaded the distilled model from this example at joeddav/distilbert-base-uncased-agnews-student.