RELATED

Exploring Text Generation With Python and GPT-4 (/articles/exploring-text-generation-with-python-and-gpt-4)

Code Search Using Retrieval Augmented Generation (/articles/code-search-using-retrieval-augmented-generation)

How Artificial Intelligence (AI) Is Transforming the Mortgage Industry (/articles/how-artificial-intelligence-is-transforming-the-mo)

Unlocking the Secrets of Data Privacy: Navigating the World of Data Anonymization: Part 2 (/articles/unlocking-the-secrets-of-data-privacy-navigating-t-2)

How to Fine-Tune BERT Transformer With spaCy v3.0

A step-by-step guide on how to fine-tune BERT for NER on spaCy v3.0 to successfully predict various entities, such as job experience and education on resumes.

By Walid Amamou (/users/4598876/mesterniz.html) · Aug. 09, 21 · Tutorial

Since the seminal paper "Attention Is All You Need (https://arxiv.org/abs/1706.03762)" of Vaswani et al, transformer models have become by far the state of the art in NLP technology. With applications ranging from NER, text classification, question answering, or text generation, the applications of this amazing technology are limitless.

More specifically, BERT — which stands for Bidirectional Encoder Representations from Transformers — leverages the transformer architecture in a novel way. For example, BERT analyses both sides of the sentence with a randomly masked word to make a prediction. In addition to predicting the masked token, BERT predicts the sequence of the sentences by adding a classification token [CLS] at the beginning of the first sentence and tries to predict if the second sentence follows the first one by adding a separation token [SEP] between the two sentences.

BERT Architecture

In this tutorial, I will show you how to fine-tune a BERT model to predict entities such as skills, diploma, diploma major, and experience in software job descriptions.

Fine-tuning transformers requires a powerful GPU with parallel processing. For this, we use Google Colab since it provides freely available servers with GPUs.

For this tutorial, we will use the newly released spaCy v3.0 library (https://spacy.io/usage/v3) to fine-tune our transformer. Below is a step-by-step guide on how to fine-tune the BERT model on spaCy v3.0. The code along with the necessary files are available in the Github repo (https://github.com/UBIAI/Fine_tune_BERT_with_spacy3). (https://github.com/UBIAI/Fine_tune_BERT_with_spacy3)

To fine-tune BERT using spaCy v3.0, we need to provide training and dev data in the spaCy v3.0 JSON format (see here (https://spacy.io/api/data-formats)) which will be then converted to a `.spacy` binary file. We will provide the data in IOB format contained in a TSV file then convert it to spaCy JSON format.

I have only labeled 120 job descriptions with entities such as skills, diploma, diploma major, and experience for the training dataset and about 70 job descriptions for the dev dataset.

In this tutorial, I used the UBIAI (https://ubiai.tools/) annotation tool because it comes with extensive features such as:

- ML auto-annotation
- Dictionary, regex, and rule-based auto-annotation
- Team collaboration to share annotation tasks
- Direct annotation export to IOB format

- **Direct annotation export to IOB format**

Using the regular expression feature in UBIAI, I have pre-annotated all the experience mentions that follow the pattern "\d.*\+.*" such as "5+ years of experience in C++." I then uploaded a CSV dictionary containing all the software languages and assigned the entity skills. The pre-annotation saves a lot of time and will help you minimize manual annotation.

For more information about UBIAI annotation tool, please visit the documentation (https://ubiai.tools/Docs) page.

The exported annotation will look like this:

Python

```
MS B-DIPLOMA
in O
electrical B-DIPLOMA_MAJOR
engineering I-DIPLOMA_MAJOR
or O
computer B-DIPLOMA_MAJOR
engineering I-DIPLOMA_MAJOR
. O
5+ B-EXPERIENCE
years I-EXPERIENCE
of I-EXPERIENCE
industry I-EXPERIENCE
experience I-EXPERIENCE
. I-EXPERIENCE
Familiar O
with O
storage B-SKILLS
server I-SKILLS
architectures I-SKILLS
with O
HDD B-SKILLS
```

In order to convert from IOB to JSON (see documentation here (https://spacy.io/api/cli#convert)), we use spaCy v3.0 command:

Python

```
!python -m spacy convert drive/MyDrive/train_set_bert.tsv ./ -t json -n 1 -c iob
!python -m spacy convert drive/MyDrive/dev_set_bert.tsv ./ -t json -n 1 -c iob
```

After conversion to spaCy v3.0 JSON, we need to convert both the training and dev JSON files to `.spacy` binary file using this command (update the file path with your own):

Python

```
!python -m spacy convert drive/MyDrive/train_set_bert.json ./ -t spacy!python -m spacy convert drive/MyDrive/dev_set
```

Model Training

Open a new Google Colab project and make sure to select GPU as hardware accelerator in the notebook settings.

In order to accelerate the training process, we need to run parallel processing on our GPU. To this end, we install the NVIDIA 9.2 CUDA library:

Python

```
!wget https://developer.nvidia.com/compute/cuda/9.2/Prod/local_installers/cuda-repo-ubuntu1604-9-2-local_9.2.88-1_am
```

To check the correct CUDA compiler is installed, run: `!nvcc --version`

Install the spacy library and spacy transformer pipeline:

Python

```
pip install -U spacy
!python -m spacy download en_core_web_trf
```

Next, we install the PyTorch machine learning library that is configured for CUDA 9.2:

Python

```
pip install torch==1.7.1+cu92 torchvision==0.8.2+cu92 torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_st
```

After PyTorch install, we need to install spaCy transformers tuned for CUDA 9.2 and change the `CUDA_PATH` and `LD_LIBRARY_PATH` as below. Finally, install the CuPy library, which is the equivalent of NumPy library but for GPU:

Python

```
!pip install -U spacy[cuda92,transformers]
!export CUDA_PATH=”/usr/local/cuda-9.2"
!export LD_LIBRARY_PATH=$CUDA_PATH/lib64:$LD_LIBRARY_PATH
!pip install cupy
```

SpaCy v3.0 uses a config file `config.cfg` that contains all the model training components to train the model. On the spaCy training page (https://spacy.io/usage/training), you can select the language of the model (English in this tutorial), the component (NER), and hardware (GPU) to use and download the config file template.

The only thing we need to do is to fill out the path for the train and dev `.spacy` files. Once done, we upload the file to Google Colab.

Now we need to auto-fill the config file with the rest of the parameters that the BERT model will need; all you have to do is run this command:

Python

```
!python -m spacy init fill-config drive/MyDrive/config.cfg drive/MyDrive/config_spacy.cfg

```

I suggest debugging your config file in case there is an error:

Python

```
!python -m spacy debug data drive/MyDrive/config.cfg

```

We are finally ready to train the BERT model! Just run this command and the training should start:

Python

```
!python -m spacy train -g 0 drive/MyDrive/config.cfg — output ./

```

Note: If you get the error `cupy_backends.cuda.api.driver.CUDADriverError: CUDA_ERROR_INVALID_PTX:` then a PTX JIT compilation failed. Just uninstall `cupy` and install it again and it should fix the issue.

If everything went correctly, you should start seeing the model scores and losses being updated.

At the end of the training, the model will be saved under the folder `model-best`. The model scores are located in meta.json file inside the model-best folder:

Python

```
"performance":{"ents_per_type":{"DIPLOMA":{"p":0.5584415584,"r":0.6417910448,"f":0.5972222222},"SKILLS":{"p":0.67968
```

The scores are certainly well below a production model level because of the limited training dataset, but it's worth checking its performance on a sample job description.

Entity Extraction with Transformers

To test the model on a sample text, we need to load the model and run it on our text:

Python

```
nlp = spacy.load("./model-best")
text = ['''Qualifications- A thorough understanding of C# and .NET Core- Knowledge of good database design and usage
```

Below are the entities extracted from our sample job description:

Python

```
[("C", "SKILLS"),("#", "SKILLS"),(".NET Core", "SKILLS"),("database design", "SKILLS"),("usage", "SKILLS"),("NoSQL",
```

Pretty impressive for only using 120 training documents! We were able to extract most of the skills, diplomas, diploma majors, and experiences correctly.

With more training data, the model would certainly improve further and yield higher scores.

Conclusion

With only a few lines of code, we have successfully trained a functional NER transformer model thanks to the amazing spaCy v3.0 library. Go ahead and try it out on your use case and please share your results. Note, you can use UBIAI (https://ubiai.tools/) annotation tool to label your data.

As always, if you have any comments, please leave a note below or email at admin@ubiai.tools!

Machine Learning Python (Language)

Published at DZone with permission of Walid Amamou. See the original article here. (https://towardsdatascience.com/how-to-fine-tune-bert-transformer-with-spacy-3-6a90bfe57647)
Opinions expressed by DZone contributors are their own.

RELATED

Exploring Text Generation With Python and GPT-4

Code Search Using Retrieval Augmented Generation

How Artificial Intelligence (AI) Is Transforming the Mortgage Industry

Unlocking the Secrets of Data Privacy: Navigating the World of Data Anonymization: Part 2