B”H

Named Entity Recognition on Alice in wonderland

Dmitry Kremiansky, Yossi Gisser

Students for data science and engineering at The Technion

Abstract

It is known that transformers are in the state-of-art of NLP tasks. Bert models usually fine-tuned on specific datasets that were made for specific tasks. In our work we want to apply Bert model after it was fine-tuned for NER (Named Entity Recognition) task and in addition was fine-tuned for specific domain. We saw the big difference between the performance of our model after just a general fine-tuning for NER task and the performance after the second fine-tuning for our specific domain. The code can be founded in this link: https://github.com/gisser770/fine-tuning-Bert-for-NER-task.

1 Introduction

In natural language processing, Named Entity Recognition (NER) is the problem of recognizing and extracting specific types of entities in text. Such as people or place names. In fact, any concrete “thing” that has a name. At any level of specificity. Job titles, public school names, sport names, music album names, musician names, music genres, … You get the idea. NER is both an interesting problem in NLP and also has many applications.

Named Entity Recognition (NER) seeks to locate and classify named entities from sentences into predefined classes (Yadav and Bethard, 2019). Humans can immediately recognize new entity types given just one or a few examples (Lake et al., 2015).

The Transformer NLP model introduced an ‘attention’ mechanism that takes into account the relationship between all the words in the sentence. It creates differential weightings indicating which other elements in the sentence are most critical to the interpretation of a problem word. In this way ambiguous elements can be resolved quickly and efficiently.

Transformer networks pre-trained with this approach are able to provide top performance in standard NLP benchmarks. Compared to the sequential models that had gone before, they deliver better results while making more efficient use of the available processing power. Transformer architecture also allows the model to take advantage of the powerful parallel processing routines available in the GPUs increasingly used for NLP training applications.

BERT is an NLP model, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications. As shown in the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, that pre-trained representations reduce the need for many heavily-engineered task specific architectures. BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.

Alice in wonderland is a well-known book. We wanted to see how well Bert model deals with NER task on this book. After fine-tuning the model both on conll2003 dataset and a few paragraphs from the book (that we tagged manually) we got pretty good results.

B"H

2 Related work

Many studies have focused on the extension of NER task, using pre-trained models. For example in Learning from Miscellaneous Other-Class Words for Few-shot Named Entity Recognition they describe a novel Few-Shot NER model, which called Mining Undefined Classes from Other-class (MUCO), that can automatically induce different undefined classes from the other class to improve few-shot NER.

Another related paper (Exploring Cross-sentence Contexts for Named Entity Recognition with BERT), discussed the NER task in long texts which is similar to our approach that we are looking on paragraph instead of looking at sentence. In this paper they presented a systematic study exploring the use of cross-sentence information for NER using BERT models in five languages. They find that adding context as additional sentences to BERT input systematically increases NER performance.

In BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision the writers prepose a two-stage training algorithm: In the first stage, they adapt the pre-trained language model to the NER tasks using the distant labels, which can significantly improve the recall and precision; In the second stage, they drop the distant labels, and propose a self-training approach to further improve the model performance.

Our method combines base ideas from all these three papers we mention. We use for the second fine-tuning just small train set, like Few-Shot NER, we decide to work on paragraphs instead of sentences and we divide our algorithm into 2 stages. The first one, general fine-tuning for NER task and the second one, specific fine-tuning for our domain.

3 Methods

3.1 Datasets

In our work we used two datasets. The first one is CoNLL-2003 (that we upload using hugging-face), already existed dataset that we use for fine-tuning Bert model to NER task.

The shared task of CoNLL-2003 concerns language-independent named entity recognition. We will concentrate on four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups.

The CoNLL-2003 shared task data files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag and the fourth the named entity tag. The chunk tags and the named entity tags have the format I-TYPE which means that the word is inside a phrase of type TYPE. Only if two phrases of the same type immediately follow each other, the first word of the second phrase will have tag B-TYPE to show that it starts a new phrase. A word with tag O is not part of a phrase. Note the dataset uses IOB2 tagging scheme, whereas the original dataset uses IOB1.

The second one, is created by us from Alice in wonderland book, that we use for additional fine-tuning and for testing our model.

Pre-process that we apply on the book is split the book into paragraphs. Then we split every paragraph for tokens (using text.split() method).

We have 793 paragraphs in the book, so we have 793 rows in our dataset.

We manually tag the first 5 paragraphs and use them for fine-tuning and training the model. In the end our dataset consists of 2 dictionaries train and test, and each one of them have 3 columns: id of the paragraph (string type), tokens (list of the tokens) and ner_tags (list of tags for the paragraph).

The list of the ner_tags that we use:

```
"O",      # Outside of a named entity
"B-PER",  # Beginning of a person's name right after another person's name
"I-PER",  # Person's name
"B-ORG",  # Beginning of an organisation right after another organisation
"I-ORG",  # Organisation
"B-LOC",  # Beginning of a location right after another location
"I-LOC",  # Location
"B-MISC", # Beginning of a miscellaneous entity right after another miscellaneous entity
"I-MISC"  # Miscellaneous entity
```

For using the tags in bert model, we map each tag to a number in range [0,7].

3.2 Models

As our base model we use the bert-base-uncased model, that we take from hugging-face.

Here is a description of the model:

BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:

- Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.
- Next sentence prediction (NSP): the models concatenate two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

This way, the model learns an inner representation of the English language that can then be used to extract features useful for downstream tasks: if you have a dataset of labeled sentences for instance, you can train a standard classifier using the features produced by the BERT model as inputs.

Then, we first fine-tubed the model for NER task using the conll2003 dataset. The parameters that were used for the fine-tuning are:

```
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=5,
    weight_decay=0.01,
    do_train=True
)
```

After received good performance on the conll2003 test set (as we will show in results section) we add to the fine-tuned model another fine-tuning on our train set (the first 5 paragraphs from Alice in wonderland book) with the same parameters, except of the number of epochs that was changed to 50.

The final model is the one we use for tagging the whole book.

4 Results

We will divide our results into 2 parts. The first results are those that we got after the fine-tuning the model on conll2003 and evaluate the model on the test set. The second results aren’t numerically accurate, but more of a general estimation.

The first results:

Although the results are very good (as expected)

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	0.214600	0.113388	0.885857	0.882388	0.884119	0.974294
2	0.049300	0.120573	0.891411	0.904106	0.897714	0.975830
3	0.025200	0.128727	0.890478	0.901258	0.895836	0.975744
4	0.015400	0.139971	0.894953	0.906955	0.900914	0.976400
5	0.010300	0.149178	0.893560	0.905649	0.899564	0.976366

when we apply this model without additional fine-tuning on the Alice in wonderland book, the tagging wasn’t logical at all.

The second results, which are the tagging we got after the additional fine-tuning are very logical (according to our manually tagging of the train set). For example: orange-marmalade was tagged as B-MISC, Australia was tagged as B-LOC, and New-Zealand was tagged as B-LOC and I-LOC respectively, Dinah was tagged as B-PER and butterfly as B-MISC.

There are some tags that aren’t logical (specially tags of B-ORG), and we will address it in discussion section.

5 Discussion

After getting our final results we can say that our model has learned well the data we tagged. The problems in this approach are: The lack in tagged data which causes that we don’t have all the tags in our train set and as a result our model have mistaken in B-ORG tags because he didn’t learn it in the second fine-tuning. Another problem in that. Is that we don’t have enough data for split the tagged data into train and test and have a quantitative estimation to the performance of the model.

Also, we have a problem that the tagging of the train set was done just by us.

A possible solution that we try is to tag a large number of paragraphs using NLTK or Spacy and treat that as ground truth. In this way we have solution for both of our problems, we can have a large train set and the tags are more stable and not rely just on our opinion. But when we try this approach, we find out that the tags from NLTK or

Spacy aren't logical at all (what was surprising for us).

Another possible solution that we haven't try because of lack in resources (but would like to try in future work) is use platform such as M-Turk or Prolific for tagging the train set. In this way we can have large train set tagged by few human judges (in case that they aren't agree on some tag we can choose the mode for example).

But even without those solution our model showed pretty good performance on the new domain. It's reminded us of the Few-Shot NER, that after just 5 paragraphs (out of 793 paragraphs) our model improve performance significantly.

In future work we would like to extend our model so that it will also perform co-reference resolution task (coref).