Institut für Informatik

Reproducibility of SciBert

Jan-Niklas Weder

Modul : Data Mining

Contents

Introduction

- Als pretrained model hängt bert stark von dem korpus ab

- Genauso ist das vocabular sehr wichtig

- Andere arbeiten zeigten den einfluss eines erweiterten trainings/besser passenden korpus? referenz suchen

- $\implies$ Roberta zum Beispiel zeigte, dass allein weiteres training die ergebnisse verbessern kann

- Wissenschaftlöiche texte unterschieden sich allgemein sehr stark von "normalen"

- $\Rightarrow$ Somit ist ein auf ähnliche weise trainiertes modell als bessere grundlagen für NLP aufgaben im wissenschaftlichen bereich sinnvoll

- Gab es in dieser form noch nicht

- $\Rightarrow$ besitzt daher potential (annahme das BERT gut sei)

- insbesondere als grundbaustein für unterschiedlichste aufgaben im !!! wiss. bereich

- Allgemein stellt sich das Problem von Datensätzen insbesondere da diese annotiert werden müssen (im wiss. bereich teuer da hochqualifizierte experten notwendig sind)

State of the art

2.1 BERT

- BERT as revolution

- pretrained-models

- usefull even without finetuning

- $\Rightarrow$ unexpected precision

- nowadays used for many different NLP tasks

- Architecture of BERT

  - explain corpora

  - explain vocabulary

  - explain tokenizer

- extensions of BERT like roBERTa

2.2 BioBERT

BioBERT is one of the extensions of BERT which has emerged because BERT by itself did not yield the desired results in the biomedical landscape. This observation has often been related to the differences in word distributions between a general domain such as Wikipedia, on which BERT was trained and the highly specialized words that are used often or with this meaning only in the corresponding domain. This difference in the underlying corpora not only produces differences in the architecture of the model itself but implies a possible need for adjustments to the used vocabulary and the tokenizer. [3]

BioBERT itself is a further trained version of BERT. This means the original BERT model was used as the basis and further trained on either PubMed abstracts, PubMed central full-text articles, or on both. Therefore the authors chose to keep the original BERT vocabulary to be able to use the pretrained version of BERT as the basis. This had the advantage that the original model only needed to be further trained on the new corpus and the needed training time could be reduced.

2.3 AOG-BERT

- als kontrahend zu SciBERT [4]

2.4 Datasets

- zum Beispiel NCBI-Disease (versuch einen goldstandard für corpora zu erstellen)

- sehr günstig um darauf entsprechende modelle zu trainieren [2]

- SciERC /sciie im repo [5]

Due to the availability of the Datasets used by the original authors, we will use their prepared datasets, which are already prepared in a way that it is easier to use them for training and still only vary slightly from the original datasets. The datasets which we will use are directly retrieved from the SciBERT GitHub page and made available through the DataDeps package which provides an easy way to retrieve data that

may or may not be locally available. If it is not already stored locally, it will be cached in the local Julia path and inside Julia, DataDeps provides the corresponding paths to the data and retrieves it from the defined source if needed. Furthermore, a hash can be defined as well to ensure that the provided data is identical to the expected one.[7]

In the following paragraph, we will take a closer look at the original data and the individual changes that have been made to use those Datasets for the training process.

Chemprot

Chemprot is in a JSON lines file format provided. More precisely every line consists of a text and the corresponding label. A field for metadata exists as well but is most of the time not used. In its original format, the Chemprot corpus consists of a develop, test, and train set of which the develop, test, and train folder correspond to the identically named files inside the chemprot folder provided on the GitHub site of SciBERT. The difference arises from a database-like structure in which the Chemprot corpus is originally provided, in contrast to those subdivided information sets where for example the text itself is in another file than the positions and annotations. Those divided pieces of information were combined and are provided in a single file in the already mentioned format. [1, 6]

SciBert

SciBERT uses the original BERT architecture and thus includes customizability and universal applicability. SciBERT is a version of BERT that is trained on a different corpus. This is supposed to make SciBERT better at understanding natural language which comes from its designated domain. More precisely in this instance the biomedical and computer science domain. In the original paper, the authors presented four different versions of SciBERT two which utilize the standard BERT Vocabulary, and two which use a specialized vocabulary to represent the vocabulary of the corpus better. It should be noted that the two different vocabularies overlap only by approximately 42%. This shows how different the two underlying corpora are and thus makes the assumption that the SciBERT versions which utilise this adjusted vocabulary might have only through this better to the domain adapted vocabulary already an advantage over the standard BERT model and the SciBERT model which uses BERTs vocabulary.

3.1 Experiments to be reproduced

In this section, we will take a brief look at the experiments performed in the original paper and briefly discuss what these experiments are about. To do this, we will split the tasks into two different parts. First, we will take a look at the so-called classification tasks and what this category entails, and then we will move on to the so-called labeling tasks. How these tasks are implemented in detail we will discuss later in chapter 4.

Classification tasks

Classification tasks, as the name suggests, describe how we "classify" some input data. This includes tasks such as choosing the right label for a given text or word. In the world of BERT, this means that BERT must determine which label best matches a given sequence of tokens or a single token.

Text Classification (CLS)

A text classification task corresponds to the example given earlier. Thus, the model must predict the correct label for the given text. This is the same problem definition as in the SciBERT work.

Relation Classification (REL)

The relation classification task can be considered as another type of classification problem. Here, the model must choose the correct label for two given parts of a sentence. The resulting label can then be interpreted as the predicted relation between the two given parts of the sentence.

Sequenz labeling tasks

Unlike classification tasks, a single sentence can have multiple labels. This makes the model, in general, more complex but provides us with the flexibility to define a given sentence with many properties which are expressed by the labels. Thus, we can obtain any given combination of labels.

Named Entity Recognition (NER)

PICO Extraction (PICO)

DEP ?

dependency tag and arc embedding of size 100 and
biaffine matrix attention
(maybe why it wasn't considered here any further ?
[abhängig davon ob noch implementiert])

3.2   Corpus

Comparison

3.3   Vocabulary

BaseVocab vs. SCIVocab
! ≈ 42% overlap

SentencePiece

Experiments

- kurze einführung in die test fälle mit einer erklärung, was f1 scores sind

- Alles NLP Aufgaben bei denen Bert "überraschend" gut abschneided

- Jetzt mit der erweiterung zu scibert erneut betrachtet

- Einföuss des vocabulars und des corpus genau gegenübergestellt

- all got dropout of 0.1

- loss cross entropy

- optemizer adam

- finetuning for 2 to 5 epochs

In the following, we will take a closer look at how the tasks already described conceptually are implemented. Since some tasks use the same architecture for the model, more precisely the identical architecture for the last layer as the all other always stay the same. We will first examine the classification tasks and then the labeling tasks in more detail. Thus, the order in which the tasks will follow will remain the same as before.

Nevertheless, both task types will use a dropout of 0.1, cross-entropy for loss, and Adam as the optimizer, all following the instructions from the SciBERT paper.

4.1 Classification tasks

Both classification tasks use the same architectural structure. This means that the final BERT or SciBERT layer is followed by a dense or fully connected linear layer. This dense layer then acts as the linear classification layer described in the original work.

Text Classification (CLS)

Classic sequence classification tasks or text classification tasks, as they are called in the original paper, can be implemented in Julia without any particular challenges. Especially since the data is already available in a format that mostly only needs to be loaded and the task is only a transformation from a text to a single label. Together with the architecture for classification tasks already described, this makes the implementation of this task very easy. We just need to combine Bert and the classification layer and provide this model with the text and the label in the correct formats. This means, on the one hand, adding the "[CLS]" and "[SEP]" tokens to the text after preparing the text with the tokenizer and wordpiece and, on the other hand, preparing the labels in a format that can be read by the model, which transforms them into a "OneHotMatrix" for "Flux", corresponding to a one out of k representation per sequence.

Relation Classification (REL)

4.2 Sequenz labeling tasks

Named Entity Recognition (NER)

Pretrained model -¿ linear classification layer with softmax output

PICO Extraction (PICO)

4.3 DEP

dependency tag and arc embedding of size 100 and biaffine matrix attention

4.4 Frozen embeddings

Vorherige experimente unter dem "festhalten" von BERT selbst und nur der anpassung der letzten ebene

4.5 Influences of different platforms

In this section, we will take a brief look at the usability of different hardware platforms for creating transformer models and when training or testing them. More specifically, we will compare the google-colab environment with an Nvidia GPU and an AMD GPU. Due to the randomness of the assignment of hardware on the google-colab side, I cannot further define the GPU that was used on this platform. The Nvidia GPU used was a GeForce 940MX with about 2GB of VRAM, and the AMD GPU on the other side was an RX580 with about 8GB of VRAM. At this point I will briefly describe the extent to which AMD's ROCM stack is usable, because surprisingly I was able to define the model and make predictions with it in a newly generated state. Unfortunately, due to instability in the ROCM stack, the Linux kernel was no longer able to use the GPU after an update that must have broken some internal dependencies that the kernel and ROCM driver rely on, and thus the video output of the computer was also unusable. Even though this shows that an AMD GPU is indeed capable of running the Transformer package and loading at least a defined model. Even though I can't disclose whether the model could be trained or otherwise used further. This fact in itself is surprising, since AMD itself describes the support status of the RX580 as "may or may not work" and Julia describes AMD GPU support as Level 3, which is the lowest level of support. (verweis zu der aussage)[belege und verweis zu ROCM]

However, I would advise anyone against installing the ROCM stack on a production system, as it is still unstable and I would therefore recommend experimenting only in some form of virtualized environment. Of course, this warning only applies to systems that rely on working video output.

Due to the failure with the ROCM stack, only the MX940 and the google-colab environment will be considered in the following part. These two platforms will not only be compared, but also how they behave in the face of two different implementations written in Julia and Python.

Discussion

präsentation und kritische auseinandersetzung mit
eigenen Daten
gefolgt mit dem vergleich zu den orginal reporteten
Daten.

Further development

Nur als subsection?

Appendix

Data availability

Code availability

Bibliography

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciB-ERT: A Pretrained Language Model for Scientific Text". In: *EMNLP 2019* (Mar. 26, 2019). arXiv: `1903.10676 [cs.CL]`.

[2] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. "NCBI disease corpus: A resource for disease name recognition and concept normalization". In: *Journal of Biomedical Informatics* 47 (Feb. 2014), pp. 1–10. DOI: `10.1016/j.jbi.2013.12.006`.

[3] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* (Sept. 2019). Ed. by Jonathan Wren. DOI: `10.1093/bioinformatics/btz682`.

[4] Xiao Liu et al. "OAG-BERT: Pre-train Heterogeneous Entity-augmented Academic Language Model". In: (Mar. 3, 2021). arXiv: `2103.02410 [cs.CL]`.

[5] Yi Luan et al. "Multi-Task Identification of Entities, Relations, and Coreferencefor Scientific Knowledge Graph Construction". In: *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*. 2018.

[6] Qinghua Wang et al. "Overview of the interactive task in BioCreative V". In: *Database* 2016 (2016), baw119. DOI: `10.1093/database/baw119`.

[7] Lyndon White et al. "DataDeps.jl: Repeatable Data Setup for Reproducible Data Science". In: *Journal of Open Research Software* 7 (2019). DOI: `10.5334/jors.244`.