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
- $\Longrightarrow$ 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)

Since in the original SkyBERT paper many parameters were taken over from the original BERT paper and these were not examined more closely for their influence on the new model, in the course of this work the influence of one of these parameters is investigated more closely. To be precise, this parameter is the batch size. On the one hand, the batch size has an influence on the efficiency and especially on the graphics card memory requirements of the model, on the other hand, it can be responsible for a faster convergence or even a non-convergence. The extremes, for example the convergence that does not come about, do not occur in general. Nevertheless, for example the reduction of the batch size can lead to the fact that a model, which was not trainable on the given hardware before because of video memory capacities, now suddenly can be trained.

State of the art

2.1 BERT

BERT is a modern approach to create a model that can take into account the context of a word in both directions. Thus, the model would be better able to interpret words through their context. Based on this architecture, a model is then pretrained on a corpus using masklm and nextsentence tasks. This base model can then be used for new tasks with relatively little effort.

The Masklm task consists of masking random words in a sentence and having the model guess what the masked word actually was. This task is commonly referred to as masked ML task or MLM for short. For training the original BERT model, about 15% of all input tokens are masked and by predicting the missing words, BERT is supposed to learn an understanding of natural language.

The Next Sentence Prediction task consists of inferring relationships between two sentences. Because this task is not covered by MLM. The Next Sentence Prediction, or NSP for short, is intended to teach the model of a given sentence following a previously given one by means of a binary classification task. Especially QA and NLI are supposed to benefit from this training.

- BERT as revolution
- pretrained-models
- usefull even without finetuning
- ⇒ 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]
Based on this prior knowledge, it was hypothesized that a model that takes these particularities into account should perform significantly better than general models in tasks of this specific domain. Based on this hypothesis, BioBERT was created. A model that should be better adapted to the biomedical domain than BERT.
Nevertheless, BioBERT itself is only 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. For the tokenizer WordPiece was used to handle the problem with unknown words. It was also considered to create a new vocabulary, but this idea was discarded in order to preserve the previously mentioned advantage of being able to utilize the pre-trained BERT model to save resources. As a result, it was observed that BioBERT performed better than BERT in version 1.0 that was based on PubMed abstracts and the full text articles of PubMed Central biomedical landscape with very few exceptions. Although the actual measured improvements sometimes vary quite significantly, it can be safely concluded that even continued training of BERT on a biomedical corpus can lead to significant improvements on tasks from this domain.

2.3 S2ORC-BERT

S2ORC, known as the Semantic Scholar Open Research Corpus, is a corpus that contains a large number of papers as well as metadata and the references associated with the papers. According to the authors, the full text portion of the corpus alone is the largest structured academic text corpus available in April 2020. This corpus is based on semantic scholar papers and thus comes from several different origins. Although the main part of the paper is the acquisition and processing of papers and the resulting construction of the S2ORC corpus, the authors also use that corpus to continue training BERT. This is relatively reminiscent to the basic idea and the approach, which is also behind BioBERT, but in this case it is closer to the approach used for SciBERT.

We will take a closer look at SciBERT at a later point in time. S2ORC-BERT, as the model trained here is called, unlike BioBERT, is trained from scratch and is therefore not based on the pretrained BERT model. Some important features are that the loss is calculated by cross entropy and the optimizer is Adam. The learning rate as well as the number of epochs per task are derived from two predefined sets. Here the combination is chosen that gives the best result for the respective task according to the development set. Since the S2ORC-BERT is rather used for validation of the corpus and clearly less focused on a new architecture for BERT, many parameters are identical to those of other papers, in particular the values reported for SciBERT were used. It should be mentioned that the difference between this model and BERT and SciBERT is almost exclusively due to the underlying corpus and the associated vocabulary.

2.4 AOG-BERT

Another approach that does not only aim at matching the corpus on which BERT is trained better to the later domain, but at the same time tries to teach the model a more extensive understanding of the texts is AOG-BERT. With this further goal, however, the training strategy underlying BERT needs to be adapted as well, because as we saw earlier, BERT's pre-training is based only on a Masklm task and a Nextsentence classification. Although these two tasks together ensure that we get a decent initial model, the question remains whether specific tasks can be solved better by a more specific training in the pre-training process or by embedding additional information. Based on this, one could argue that a model with domain entity knowledge could perform better than a model that does not use this information. For example, Liu et al. argue that specific institutes may be more focused on specific scientific domains and that this knowledge could help the model to assign a paper to a research domain. Although it should be reasonable to assume that the actual text should be sufficient to assign a paper to a research area, this knowledge might prove beneficial for the model.

Compared to existing approaches, OAG-BERT attempts to use not only homogeneous knowledge but also non-homogeneous knowledge from the Open Academic Graph. The information types used include authors, fields of study, venues and affiliation.

To integrate this information into the model, three crucial changes are made compared to BERT.

The first adaptation is the heterogeneous entity

type embedding. Here, the token type embeddings are replaced with entity type embeddings, providing unique labels for the different entity types, so that it is possible to identify which input belongs to which category. The next modification is the entity-aware 2D-positional encoding. Just like BERT, OAG-BERT needs positional embeddings to be aware of the sequence order. However, BERT does not have the ability to distinguish between two directly adjacent entities and would interpret them as one entity. To solve this problem, the positional embeddings in OAG-BERT are two dimensional and encode in the first dimension the so-called inter-entity sequence order and in the second dimension the intra-entity sequence order. (Final result is the sum of the two dimensions?)

The last change is the span-aware entity masking. Although the masking strategy is not changed for either the abstract or the actual text, special masking is proposed for the new entity types. This is supposed to help OAG-BERT to learn complex entities especially if they consist of many tokens. This means that an entity consisting of four or less tokens is completely masked, but as soon as it is longer only a part of the entity is masked. Here the length of the mask is drawn from a geometric distribution.

- als kontrahend zu SciBERT [4]

2.5 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.

While the architecture used in the SciBERT model is described in great detail, as well as the changes made for specific tasks, there are some missing pieces of information that underlie certain decisions, and some decisions were made based on previous models and are therefore not necessarily transferable to this new model without some restrictions.

The first thing to mention here is that the corpus on which the model is trained is not specified precisely enough or that the actual data used cannot be traced. Only the source of the data was described, but not the selected data, and this information does not seem to be reproducible at all.

Another example would be the architecture for the tasks themselves. This is determined without considering other possibilities that could potentially produce similar or even better results. Likewise, the optimizer, dropout, and loss function are referenced from another paper without considering the possibility that an alternative might be more appropriate. A similar situation can be observed for the batch size and the choice concerning the number of epochs and the learning rates being considered. While several options are mentioned for epochs and learning rates, this is followed by settling for the best epoch and learn rate in each case. Furthermore, the decisions regarding learn rate and epoch are not discussed further and thus the information on the exact influence of the learning rate and the number of epochs is missing. Another aspect would be that of tasks and grades. Here, some frequently used tasks were used without further explanation. In addition, the selected scores are based solely on other work and are not further justified or extended by other possible scores. Also, only one score is described here, which is supposed to be an average of several attempts, and it would have been more useful to include information about how much the scores vary.

In this paper, we will focus on the influence of the number of epochs on the score as well as on the change of the loss over time. In particular, we will look into how the model changes from epoch to epoch. At the same time we explore whether the range of two to four epochs considered in the original paper is sufficient or whether more epochs might be reasonable. To be more precise, we will look at the range of one to ten epochs and consider the changes in the loss and the score. This will then be extended using different

learning rates to show how these affect the speed of convergence of the model.

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
! $\approx$ 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 Simple tests and examples

To ensure that these individual areas function correctly and that errors have not already sneaked in here. There are several outputs that show the functionality in a small scale and can ideally catch some errors. These outputs can be found in the Jupyter Notebook under Tests. In this section, the first record and the corresponding label of the training dataset are displayed first. The same is repeated for the test data set. After these examples, a dry run of the tokenizer follows. This is to assure us that the tokenize function, which puts the data into a form that the transformer can read, is working correctly. This is shown with a sentence that leads to the same result both by hand and by the function. Next, the tokenization of special tokens is examined. This is to guarantee that the special tokens needed for the NER task are correctly recognized and translated. This is followed by some examples that test the functionality of the loss function and the embedding of the previously translated sentence. At this point, however, the verification of correctness becomes difficult, since the interpretation of the representation of the CLS token is no longer directly evaluable for humans. Although it can be shown that this process functions and presumably also operates correctly, this example no longer guarantees this. This is followed by the tests that refer to the evaluation, in other words to the F1 score.Here we first consider whether the formats have been converted correctly so that the Metric package which is used for the calculation can read them and then we briefly compare whether the example delivers the expected F1 score.

4.2 Hardware requirements

In this section, we will take a brief look at the usability of different hardware platforms for creating transformer models and 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 hardware assignment on the google-colab page, I cannot define in more detail which GPU was used on this platform. The Nvidia GPU that was used was a GeForce 940MX with about 2 GB of VRAM, and the AMD GPU on the other side was an RX580 with about 8 GB of VRAM.

At this point I would like to 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 created state. Unfortunately, due to instabilities in the ROCM stack after an update that must have broken some internal dependencies that the kernel and ROCM driver must have relied on, the Linux kernel could no longer use the GPU, and so the video output of the computer was unusable.

Even though this shows that an AMD GPU is quite capable of running the Transformer package and loading at least one defined model. Even though I cannot reveal 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 only potentially possible, and Julia describes AMD GPU support as Level 3, which is the lowest level of support.

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

This brings us to the GeForce 940MX. This graphics card unfortunately reaches its limits due to its memory size. 2GB of Vram, of which even a little bit less is usable, does not suffice for the used BERT model nor for SciBERT and thus results in an out of memory error of the Cuda runtime. Therefore, this hardware will not be considered further.

Thus only the google-colab environment remains.

This platform provides about 10 to 15 GB of VRAM. After the model has been moved to the graphics card, Julia shows an occupancy of just over 2 GB memory with the help of the memory indication of the Cuda driver. Here you can see why the 940MX was not able to load the model. The calculation of an epoch usually takes between 500 and 1000 seconds. These fluctuations could have several causes, but are most likely due to the google-colab environment and in particular the fact that several instances share the same graphics card. This is most likely the reason for the observed strong fluctuations from epoch to epoch. This assumption is particularly reasonable, since in the meantime almost every epoch lasted the same time, apart from relatively small fluctuations of less than 50 seconds. Interestingly, the entire video memory is used for caching, even though the model consumes just over 2 GB of memory and the entire original chemprot dataset is about 5MB in size when unpacked. Still, the runtime required in the google-cab environment is about as expected and significantly faster than training on a CPU. The CPU, which was a Xeon E3-1230 v3 with four cores, needed about as long for a single step as the graphics card in the google-colab eviremont needed for an epoch. A step here consisted of only one sentence and the corresponding label.

4.3 Data preprocessing

4.4 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)

4.5 Sequenz labeling tasks

Named Entity Recognition (NER)

Pretrained model -¿ linear classification layer with softmax output

PICO Extraction (PICO)

4.6 DEP

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

4.7 Frozen embeddings

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

4.8 Complications

In particular, the classic text classification was relatively unproblematic to implement. Nevertheless, there were some minor obstacles on the way to a finished reproduction of the original CLS. The order will follow that of the information flow in the model, starting with the input data and ending with the output.

After this sequence, the first problem is to transform the data, which are now already the two text areas with the special tokens "[<<]", "[>>]", "[[[ ]" and "[ ]]]" marked, the sentences now have to be processed by the tokenizer and by wordpiece, without changing the mentioned tokens. If the individual sentences would be processed without further precautions, then the special tokens would be modified. Through this alteration, however, these tokens would no longer be interpreted by the vocabulary as special tokens, but would be regarded as "normal" words. To avoid this, a wrapper was written, which divides the text into partitions and processes them individually by the Tokenizer and Wordpiece. This was particularly necessary due to the fact that the partitions marked by a pair of corresponding special tokens might not have the same length after processing. Due to this potential variation of the segment length, it is not directly possible to find the new position of the tokens from the old one and simply adding them manually. Instead, the individual sections separated by the special tokens are processed individually and subsequently the special tokens are placed between the corresponding segments. This procedure is based on the addition of the "[CLS]" and "[SEP]" tokens in the CLS, but extends it. The next problem follows directly the previously described one. The individual sentences of the data sets sometimes consist of characters that do not occupy only one "code unit" and so problems can occur if the end of a string is expected at the position of the length of the string.A 'code unit' describes here a fixed size in the memory which is needed in general for the coding of a single character. Fortunately, an error is generated if an attempt is made to create a view of a string that ends or begins in the middle of a character that spans multiple code units. Although the error is relatively vague, a closer look at the string quickly reveals that the error occurs in the vicinity of special characters. With this knowledge one can quickly find the explanation and the underlying problem in the corresponding documentation due to the different number of code units that are sometimes necessary to encode special char-

acters. The functions 'firstindex' and 'lastindex' or the 'split' function can help here. All these functions take into account the differences between position and code unit. Another, more general problem was that some packages provide very few examples of their use and at the same time are not sufficiently detailed in the specifications. This leaves open questions that make the use of the packages at first impossible. For example, the MLBase package provides many different metrics, but requires an object of type ROCNums for the calculation. This type can be generated by the MLBase package, but the documentation only says that a ground truth and the predictions of the model are required as input. However, the documentation for the actual ROCNum type says nothing about the format in which these two pieces of information must be provided, and there is nothing at all in the examples regarding the ROCNum type. However, if you are lucky, you may stumble across the documentation for the Confusion matrix, which includes an example on how to create a ROCNum object. Based on this example one can then derive the correct formats. However, missing information of this type can be found several times and requires either some luck to find a working example or will require trial and error to eventually figure out the correct format, hopefully.

Discussion

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

Further development

Nur als subsection?

Bibliography

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: 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.