Biomedical Relation Extraction with Deep Neural Networks

Nils Broman
F16, *Department of Computer Science, LTH*
Lund, Sweden
Email: ni8258br-s@student.lu.se

***Abstract*—With the vast amount of research publications, methods of automating the gathering of information to increase the ease of access are of great interest. In this paper, I describe the use of pre-trained BERT-based language models for extracting the relations between chemicals and proteins from medical literature specifically. I attempt to find ways to improve previous student models obtained by finetuning SciBERT, which is pre-trained on scientific litterature, with the ChemProt corpus as well as simple artificially constructed sentences. In the precious project, a top validation F_1-score of 0.65 was achieved from finetuning on an oversampled ChemProt for five epochs, and 0.56 when first trained on artificial data for five epochs, ending with a single epoch on the ChemProt corpus. The largest improvement was found after changing the tokenizer to the one intended for SciBERT, resulting in a top validation F_1-score of 0.85 when trained for 9 epochs. By combining the ChemProt corpus with a smaller sample of artificial data, equivalent to 10% of the size of ChemProt used for training, an F_1-score of 0.86 was achieved. While this does not conclusively show that adding artificial training data will improve the model, it does show a possible potential. Thus further work in improving the artificial constructor to produce more representative data could still be of interest.**

I. Introduction

The study of interactions between chemicals, proteins and DNA is key to understand the very fundamentals of life, and particularly important within areas such as medicine. These interactions may be crucial when designing a new drug in order to get the desired effect, along with avoiding undesired side-effects [1]. Consequently, there is a significant amount of research literature within this field, with new articles being published daily. It is infeasible for a human to read all these articles in order to look up some relation, especially since one interaction often results in another, potentially starting a chain reaction which causes effects several interactions away. Thus, it would be desirable to have an automated system to scan for interactions reported in literature and map them in a way that is more easily accessible.

In this project, I continue the work of previous students [2], using state of the art methods of Natural Language Processing (NLP) to construct models for classifying chemical-protein interactions using the ChemProt dataset [3], as well as investigating the potential of adding artificially constructed data to improve performance.

II. Background

A. BERT - Bidirectional Encoder Representation from Transformers

The article ”Attention Is All You Need”[4] is arguably one of the most important publications in modern machine learning, particularly for NLP. In the paper, the authors presented a new type of network architecture, called the Transformer, which is an encoder/decoder only utilizing the attention mechanism. Without going deep into the math, the self-attention mechanism can be explained as a way for the inputs to interact with themselves and try to find their most distinguishing feature, and further emphasizing it. In this project we are only interested in the encoder part of the Transformer, illustrated in fig. 1.

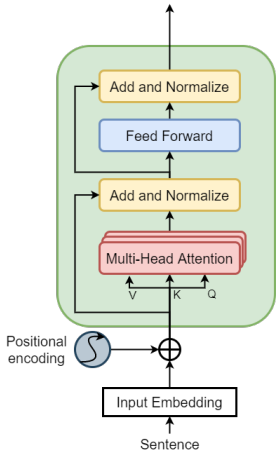


Fig. 1: The architecture of the transformer encoder.

Prior to the Transformer, recurrent neural networks (RNNs) dominated the area of NLP. While the recurrent design has many good properties it also means that each node is dependent on the one before, which has major drawbacks. Since the nodes are dependent, the calculations must be done in sequence, and with the depth scaling with sequence length, does the issue of vanishing gradient, i.e. where the gradient quickly becomes very small as it is multiplied with the weight matrix of each layer during backpropagation, making long term dependencies difficult to track. One of the more popular new language models is the Bidirectional Encoder Representation from Transformers (BERT), which

through the use of self attention rather than recurrence no longer have this issue with long term dependencies while also allowing for paralellization during training, making this process much more efficient. BERT comes in many shapes and forms, but the most common BERT$_{BASE}$ consists of 12 layers of fully connected transformers, each with 12 attention heads, and a hidden length of 768, resulting in a total of 110M trainable parameters[5]. To train such a large network is extremely costly both in training time and the amount of training data needed, so rather than training it from scratch for every task, the common practice is to utilize the method of transfer learning, where the network is pre-trained in language modeling on a large corpus, creating a model that has a good understanding of the language in general. The resulting language model can then be fine-tuned on a much smaller set of data specific to the intended task, e.g. biomedical relation classification. To put things in perspective, the pre-training of BERT$_{BASE}$ using 4 cloud TPUs takes 4 days to complete, and doing so on the machine used for this project could be expected to be well over a month, whereas the fine tuning can be done in hours or even minutes (depending on the size of training data set, number of epochs, batch size etc.).

Pre-training of large neural language models typically consists of next word predictions, but as BERT is bi-directional fully connected, it would already know what word comes next. Instead, BERT language models are pre-trained using Masked Language Modeling (MLM): hiding a few words, anywhere in the input, and running predictions on the masked words. This means that models learn context from anywhere in the sequence, regardless of direction or distance. The pre-training also consists of next sentence prediction (NSP), to also learn dependencies of entire sentences. BERT$_{BASE}$ is pre-trained on text passages from Wikipedia (2500M words) and unpublished books from the BookCorpus (800M words). However, the language used in scientific papers is very different than that generally found in novels and Wikipedia articles. SciBERT shares the exact same architecture as BERT$_{BASE}$, but has instead been pre-trained on a total of 1.14M different scientific articles.

BERT uses WordPiece embeddings to represent the words, splitting words into smaller pieces to create a token vocabulary. First outlined in "Japanese and Korean Voice Search"[6], this form of tokenization was originally introduced for Google's LSTM-based translation system as it proved useful in handling rare words[7]. It allows for a larger coverage of vocabulary with a limited number of tokens, particularly because many words share the same prefixes and suffixes, e.g. "dis", "ex" or "##ing" and "##esque" (where ## indicate that they are part of the same word as the preceding tokens). Instead of having to include multiple instances of the same words, only in different forms. Because of this the system is also often capable of representing new words never seen during training, without having to go down to single characters. BERT also uses some special tokens such as MASK for masked words, CLS for classification of the entire sequence, e.g. in NSP or sentiment analysis (SA) tasks, SEP for separation of two input parts, e.g. two sentences. Depending on the task, one can add other special tokens, making BERT an incredibly flexible language model. For relation extraction the output embedding of the two entities are concatenated and ran through a simple linear classifier.

B. ChemProt Corpus

The ChemProt corpus consists of a total of 1820 abstracts from PubMed articles with interactions between chemicals and proteins manually annotated by experts[8]. It was used in the BioCreative VI (2017) track 5 for text mining of chemical-protein interactions. The data is split into four sets, a small sample set, a training set, a development (validation) set and a test set. Each of these include four files, one with all the abstracts, one listing the entities, i.e. chemicals and proteins, and indices for their location in the text. The third is a detailed list of pairs of entities, their relation and indices in the text. ChemProt has 20 different interaction labels, grouped into 10 ChemProt relations (cpr) by semantic similarity. The fourth file is the gold standard of relations for the problem formulation of their task, where they only used a few select cpr, and as such of no importance to this project.

III. Tools and Metrics

A. Code

The scripts and models in this project were written in Python (version 3.7.13) with PyTorch (version 1.8.1, cuda 11.1) and trained on a Windows 10 machine (CPU: i7-7700K, GPU: RTX3070). Older versions were used to ensure compatibility with a larger pipeline[9] that also includes other NLP-methods, such as entity tagging, with parts written in an older version of TensorFlow (1.15) which lost support as of Python 3.8. A description of all scripts and files used in the project can be found in Appendix A, and the full code is available on GitHub[10].

B. Fixed Model Hyperparameters

All models presented in this report used a batch size of 32, Adam optimizer with $\lambda = 2\text{e}{-5}$ and $\epsilon = 2\text{e}{-8}$ as well as a linearly decaying learning schedule without warmup.

C. Metrics

The main metric used in evaluation was regular F_1-score, which is the harmonic mean of Precision (the ratio true positive predictions and total predictions of a class) and Recall (the ratio of true positive predictions and total actual positive results).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3}$$

Additionally, three types of averages were used - macro, micro and weighted. In a micro average, all samples contribute equally while a macro average first computes the score for each class and take the average of these scores. The weighted average works the same as macro, but adds weight in relation to the number of samples in each class. A micro average can be an important metric when a multi-label classifier is used, i.e. the classifier can assign more than one class for every instance. The classifier used in this project however, is limited assigning one class per instance, thus the micro average will simply be the same as accuracy, for all three metrics. Since the objective of our classifier is to work well across the board, the macro average will be more well suited as the weighted average does a poor job showing the performance on classes with few samples. Consequently, the macro average was used as the main metric, and is to be assumed henceforth if not specified otherwise.

IV. Earlier Work

As previously mentioned, this project continued the work of prior students and is a part of a larger pipeline, so in order to acknowledge the changes and additions made, we first need an understanding of the surrounding processes as well as the former state of the project.

A. Preprocessing Data

For each of the interacting pairs, the sentence was extracted and the two entities were marked within the text. The marked sentences were then put in a dictionary together with corresponding ChemProt label and cpr. Since this was done for every interacting pair, sentences containing more than two entities appeared multiple times, but with different entities marked in the text. With the ChemProt labeling being too specific for our purposes, the relations were grouped into five custom classes (as shown in table I) to reduce complexity.
The resulting distribution of the class instances can be found in table II, which shows that there is a clear imbalance, with under 5% representation for both the ”NOT” and ”PART-OF” classes, around 12.5% for ”REGULATOR-POSITIVE” while the ”INTERACTOR” and ”REGULATOR-NEGATIVE” each make up almost 40%. One attempt to minimize the effect of the imbalance was to use random oversampling on the underrepresented classes. While this helps to balance the train set, it does not add any new information.

B. Artificial Corpus

In addition to the ChemProt corpus, example sentences were constructed artificially. By assuming the description of the interaction happening somewhere between the mentioning of the first and second entity, the sentence can be deconstructed to five parts, as shown in fig. 2. Using cell names selected from the cell line ontology[11] as part of the start or end phrases (which were written by Sonja Aits), and proteins from the UniProt-database[12] as entities, one can quickly generate a large number of unique examples.

cid	**Custom label**	**cpr**	**ChemProt label**
0	NOT	10	NOT
1	PART-OF	1	PART-OF
2	INTERACTOR	2	REGULATOR
		2	DIRECT-REGULATOR
		2	INDIRECT-REGULATOR
		5	AGONIST
		7	MODULATOR
		8	CO-FACTOR
		9	SUBSTRATE
		3	UPREGULATOR
		3	ACTIVATOR
3	REGULATOR-POSITIVE	3	INDIRECT-UPREGULATOR
		5	AGONIST-ACTIVATOR
		7	MODULAR-ACTIVATOR
4	REGULATOR-NEGATIVE	4	INHIBITOR
		4	INDIRECT-DOWNREGULATOR
		6	ANTAGONIST
		7	MODULATOR-INHIBITOR
		5	AGONIST-INHIBITOR

TABLE I: ChemProt interaction labels with corresponding cpr-group and how they relate to our custom labels.

Class	**Original**				**Oversampled**	
	Train		Dev		Train	
	count	%	count	%	count	%
INTERACTOR	2583	40.13	1350	37.96	2583	26.87
NOT	241	3.74	175	4.92	1205	12.53
PART-OF	308	4.79	153	4.30	924	9.61
REGULATOR-NEGATIVE	2505	38.92	1302	36.61	2505	26.06
REGULATOR-POSITIVE	799	12.41	576	16.20	2397	24.93
Total	6436	100	3556	100	9614	100

TABLE II: Distribution of class instances in the ChemProt corpus. There is a considerable lack of support for the NOT and PART-OF classes in particular, with REGULATOR-POSITIVE also falling short. Random oversampling of these three classes with factors 5, 3 and 3 respectively yields the new distribution shown to the right.

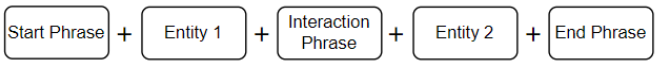


Fig. 2: Sentence syntax for the artificially constructed data.

An important thing to note here, is that the structure of all artificially constructed sentences are the same, with the interaction described somewhere in between the mentioning of the two entities. The English language does however allow for different structures, and an interaction can very well be described prior or post the mentioning of both entities, and such examples are present in the ChemProt development set.

C. Models

The previous student project presented four different models, obtained by fine tuning SciBERT on both the original and oversampled ChemProt sets as well as a completely artificial set, containing 10 000 examples of each class. All where trained for up to 5 epochs, saving the model from the epoch with highest F_1-score. The fourth model was first trained for 5 epochs on the artificial data and finally one epoch on the ChemProt train set. In table III we see that the oversampled model performed the best. The artificial model scored perfectly on both artificial sets already after the first epoch, but very poorly on the ChemProt data. It did however show slight improvement between the first and fifth epoch, which implies that it could still learn something from the artificial data. More plots and information of the models can be found in Appendix B.

Model	Best epoch	ChemProt		Artificial	
		Train	Dev	Train	Dev
Baseline	4	0.88	0.51	–	–
Oversampled	5	0.97	0.65	–	–
Artificial	5	–	0.30	1.00	1.00
Artificial + ChemProt	$5+1$	–	0.38	0.57	0.56

TABLE III: Top F_1-score of the old models

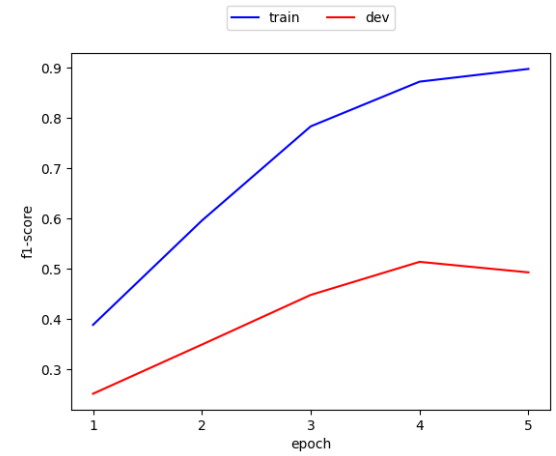
V. Method and Reuslts

Throughout this project, many models were trained, and all the presented types of models were trained, with identical starting conditions a minimum of two times, in order to ensure consistency such that when replicated, one can expect similar results.

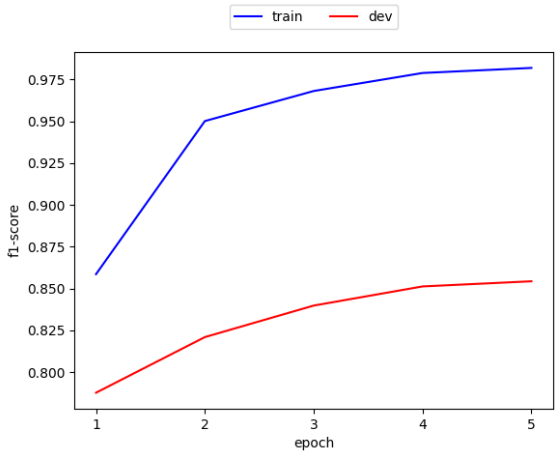
A. Tokenizer

The first discovery was that the tokenizer loaded in the script `bert_finetune` from the previous project was the one belonging to BERT$_{BASE}$, and not SciBERT. Therefore, new models were trained on the ChemProt training set with the BERT$_{BASE}$ and SciBERT tokenizer for comparison. The tokenizer had a major impact on the performance, with the SciBERT tokenizer model outclassing all old models already after the first epoch. While the F$_1$-score of the model using the BERT$_{BASE}$ tokenizer dropped after the fourth epoch (fig. 3a), the model with the correct tokenizer was still improving (fig. 3b).

In addition, another model was trained with the SciBERT tokenizer, which also included oversampling. The results displayed in table IV very similar performance of the newly trained BERT$_{BASE}$ model and the old baseline model, while clear improvements on both new models using SciBERT tokenization. Where oversampling yielded a large improvement of performance for the old models, it only resulted in a marginal increase when using the SciBERT tokenizer.



(a) BERT$_{BASE}$ tokenizer



(b) SciBERT tokenizer

Fig. 3: Baseline model macro average F_1-score over 5 epochs using different tokenizers.

Model	Tokenizer	Best epoch	ChemProt		pp-change	
			Train	Dev	Train	Dev
Old-5	BERT$_{BASE}$	4	0.872	0.514	−0.8	+0.4
Base-5	SciBERT	5	0.981	0.847	+9.9	+33.3
OS-5	SciBERT	5	0.987	0.850	+2.5	+20.0

TABLE IV: Top F_1-score for models trained for a maximum of 5 epochs. The first (Old-5) using the BERT$_{BASE}$ tokenizer (same conditions as former baseline model), the second (Base-5) using the SciBERT tokenizer, and the third (OS-5) using the SciBERT tokenizer as well as oversampling. The table also includes the percentage point change compared to the previous group's models.

tokenizer.

Given the clear advantage of using the SciBERT tokenizer, all following models were trained using this. Additionally, the Base-5 model will serve as reference for changes in performance of all following models.

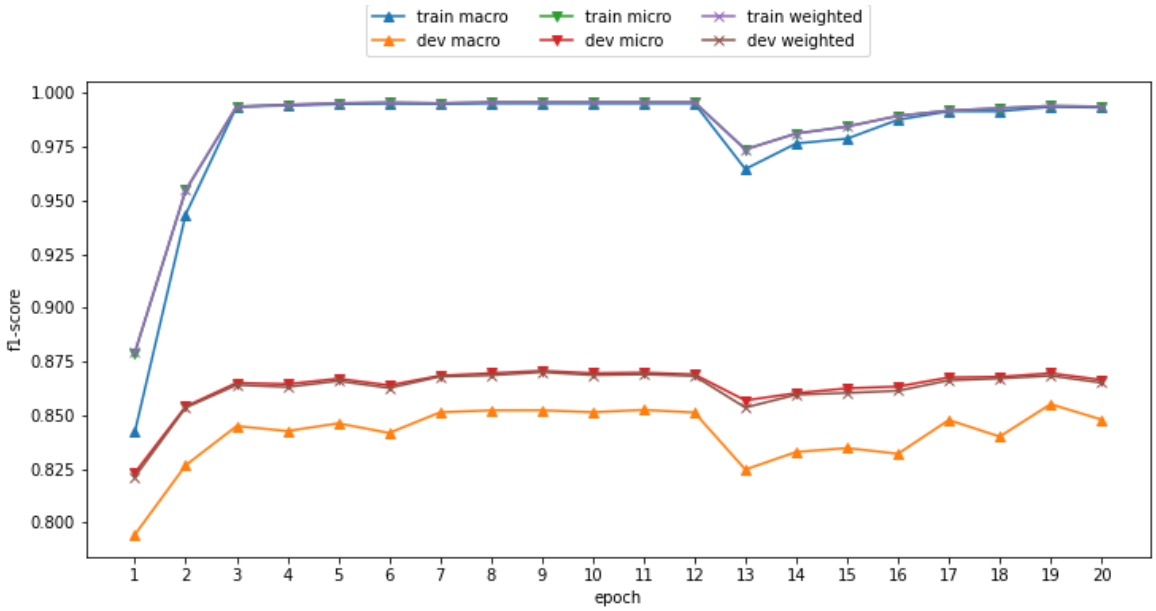


Fig. 4: F_1-score for model trained up to 20 epochs on the ChemProt training set.

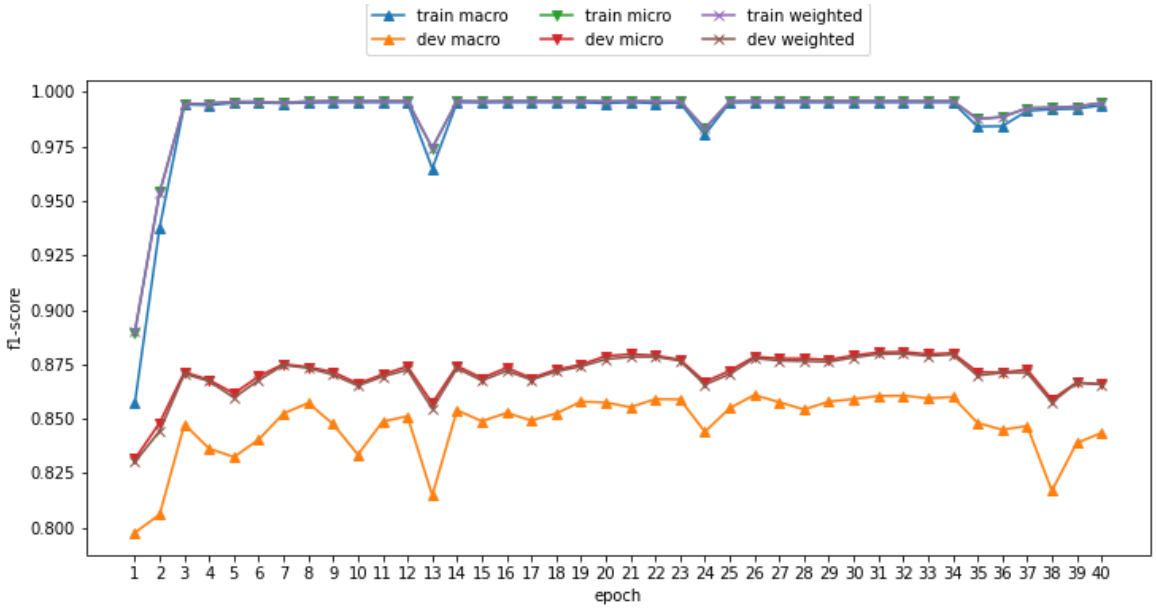


Fig. 5: F_1-score for model trained up to 40 epochs on the ChemProt training set.

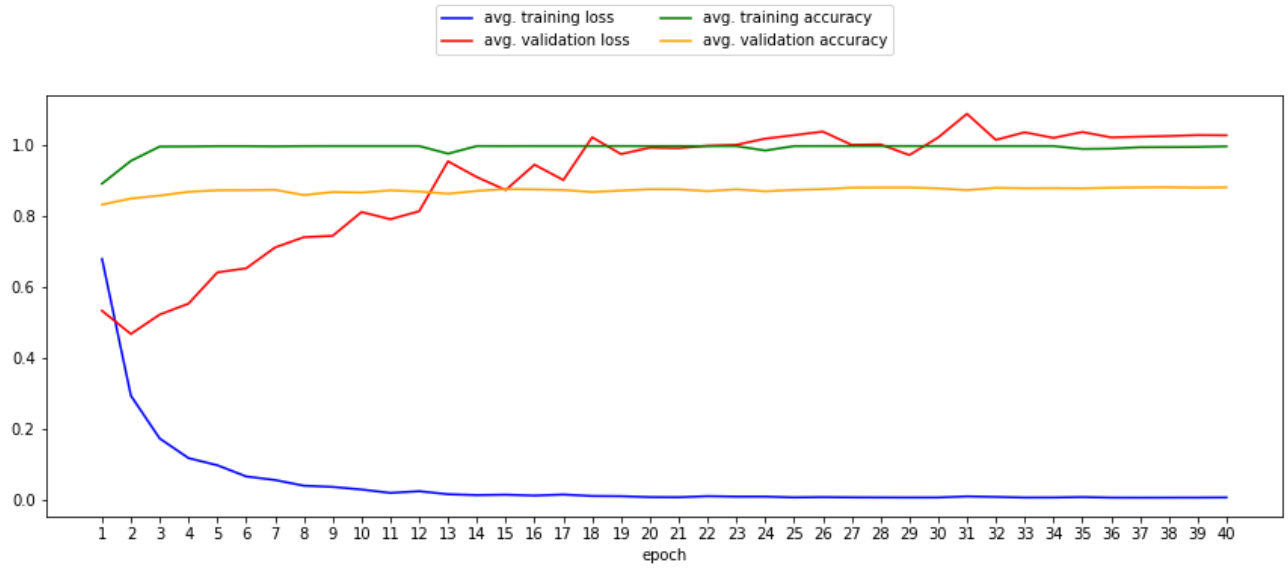


Fig. 6: Accuracy and loss for ChemProt training and development (named validation in figure) set for the model trained over 40 epochs. The training loss effectively converges at around 10 epochs while the development loss increases. Together with drops in F_1-score post 10 epochs this suggests that longer training is not beneficial.

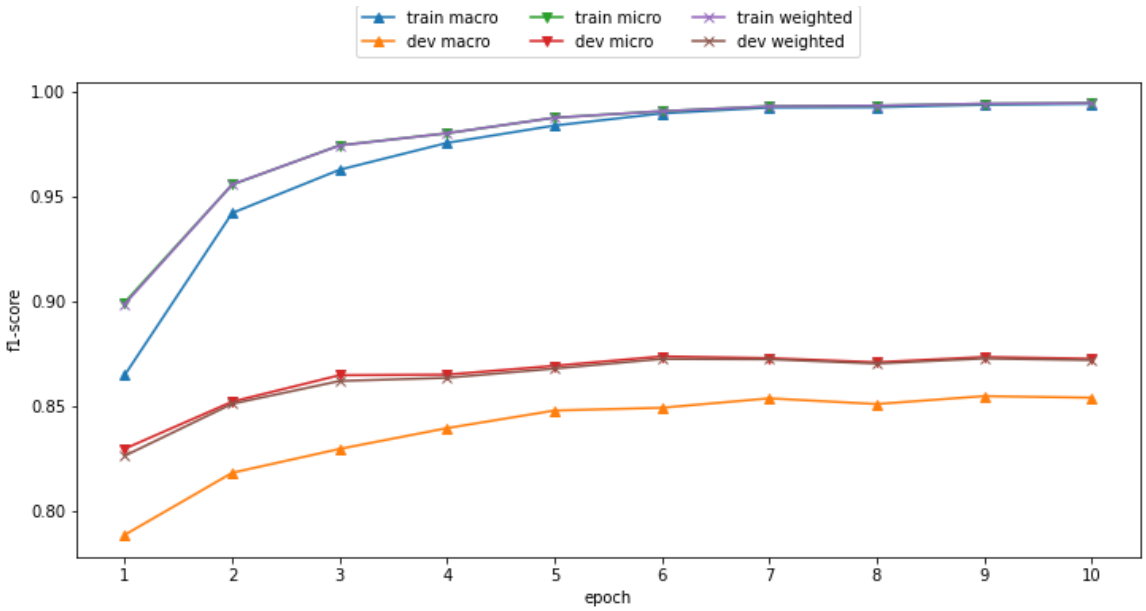


Fig. 7: F_1-score for model trained up to 10 epochs on the ChemProt training set.

B. Training Time (max Epochs)

As the performance on the development set showed no sign of peaking during the first five epochs, the maximum number of epochs was set to 20 and new model (Base-20) was trained, fig. 4. This time however, there actually was a slight drop in performance between the third and fourth epoch. Another more surprising discovery was the sudden drop in all metrics for both the development and training set after epoch 13. The performance on the training set should typically only improve during training.

To investigate this further the maximum epochs was increased to 40, and yet another model (Base-40) was trained, fig. 5. The same drop at epoch 13 was observed, together with two more at 24 and 35. Unlike after the first two drops where the model jumped back almost immediately after the drop, the model fails to recover after the drop at epoch 35, which shows more resemblance to the scores after the drop at epoch 13 for the Base-20 model. There was also a much larger variance in development performance during epochs, particularly during the earlier epochs and most noticeable between epoch 3 and 7. When examining loss during training (fig. 6) there was no significant reduction of loss for training set after epoch 10, while loss for the development set increased from epoch 3 onward.

With these discoveries in mind, a third model (Base-10) was trained for 10 epochs. This time, no sudden large performance drops were observed (fig. 7), and the overall training was much smoother. Table V shows the three models top F_1-score performances as well as their loss. While Base-20 and Base-40 had much lower loss on the training set, Base-10 resulted in lower loss for the development set, while having near identical F_1-score as Base-20 and only slightly worse performance than Base-40.

Model	Best epoch	ChemProt		pp-change		Loss	
		Train	Dev	Train	Dev	Train	Dev
Base-10	9	0.994	0.855	+1.3	+0.8	0.024	0.721
Base-20	19	0.994	0.855	+1.3	+0.8	0.007	0.957
Base-40	26	0.995	0.861	+1.4	+1.4	0.007	1.037

TABLE V: Top F_1-score with corresponding and average loss values for models trained for a maximum of 10, 20 and 40 epochs, as well as pp-change compared to the best performance of Base-5.

Furthermore, three models with identical conditions other than the addition of oversampling were trained, where the same drops at epoch 13, 24 and 35 were observed (when applicable). As shown in table VI however, unlike as for OS-5 and Base-5, the OS-10 model did not show an improvement in performance compared to Base-10, and both OS-20 and OS-40 had their peak performance much earlier (epoch 7 and 6 respectively) as compared to Base-20 and Base-40. Additionally, OS-20 showed both better performance in F_1-score and lower loss for both sets compared to OS-40. The difference in F_1-score compared to the models without oversampling still proved very small, with a mere 0.3 pp change for all three models, and only OS-20 showing improvement.

Model	Best epoch	ChemProt		pp-change		Loss	
		Train	Dev	Train	Dev	Train	Dev
OS-10	9	0.994	0.852	+1.3	+0.5	0.011	0.848
OS-20	7	0.995	0.858	+1.4	+1.1	0.028	0.782
OS-40	6	0.994	0.858	+1.3	+1.1	0.044	0.810

TABLE VI: Top F_1-score with corresponding and average loss values for oversampled models trained for a maximum of 10, 20 and 40 epochs, as well as pp-change compared to the best performance of Base-5. Not that unlike Base-20 and Base-40, both OS-20 and OS-40 showed the best performance before epoch 10, which resulted in more balanced loss.

C. Predicting Artificial Data with ChemProt Models

The results of the previous group (table III) showed that a model trained on artificial data performed poorly when evaluated on ChemProt data. This poor performance could be due to the lack of sentence structures that may be present in the ChemProt data but not in the uniform artificial corpus. This means that the model could potentially work very well on the ChemProt sentences that share the sentence structure with the artificial corpus. To test this, the experiment was done the other way around, that is evaluating the performance of a model that has been trained on ChemProt on the artificial data. The scores showed (fig. 8) that this does not appear to be the case, however. In addition to being low, there was a rather large noisy variation between epochs, that did not follow the performance changes on the ChemProt set (fig. 5. While the changes mostly behaved similarly, though of different magnitude, for the artificial training and developement set, they moved in different directions between some epochs (e.g. 3-5 and 9-12), indicating that there are some significant differences in the represented samples between the two sets, although originating from the same construction script.

D. Chemical Entities in the Artificial Corpora

The ChemProt corpora contains chemical-protein interactions while all artificial sentences used proteins for both entities. This could potentially be one of the reasons for the poor performance gained from training on artificial data. Without time to access any database of chemicals, a smaller sample was created with the script `extract_chemical_names` by taking all the chemicals from the ChemProt training corpus, and adding them to the building blocks used in `build_art_corpus`.

Two equally sized artificial training and development set pairs were created, one using only protein entities and the other with chemical-protein interactions, and two new models were trained on respective artificial train sets for up to 5 epochs. The models where then evaluated on all of the artificial sets as well as the ChemProt training and developement sets. As seen in table VII, both models performed near perfect on

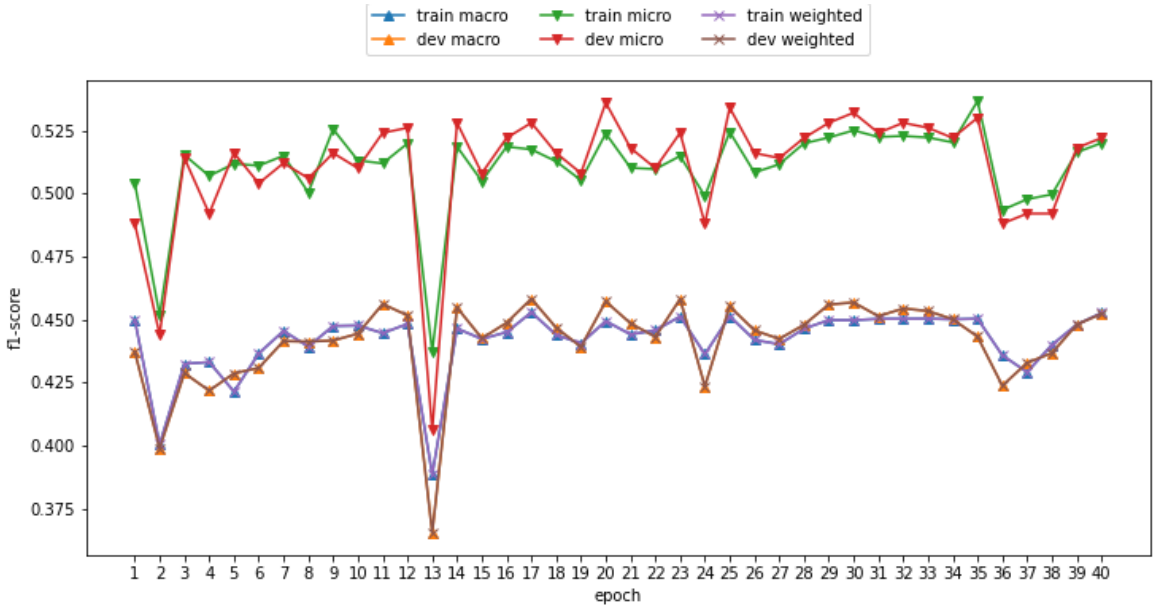


Fig. 8: F_1-score after each epoch of Base-40 when evaluated on artificial data. While the variation between epochs was continuously high, the drops on epoch 13, 24 and 35 were still clearly visible.

all artificial sets. While the performance on the ChemProt training set improved by about 1.9 pps for the model using chemical-protein interactions (Art-CP), the improvement on the development set was merely 0.3 pps, compared to the model trained on protein-protein interactions (Art-PP). Though introducing chemical names to the artificial data only displayed slight benefits, the effect of changing the tokenizer is obvious for these sets as well, with Art-PP increased by 8.9 pp and Art-CP improved by 9.2 pp on the ChemProt development set. Another interesting observation is that both Art-PP and Art-CP performed better on the ChemProt development set than the ChemProt training set.

Model	**Best epoch**	**Artificial (P-P)**		**Artificial (C-P)**		**ChemProt**	
		Train	Dev	Train	Dev	Train	Dev
Art-PP	4	1.00	0.999	0.999	0.999	0.364	0.389
Art-CP	4	0.999	0.999	1.00	1.00	0.383	0.392

TABLE VII: F_1-score for models trained on artificial data with protein-protein (Art-PP) and chemical-protein (Art-CP) interaction, using chemical names extracted from the ChemProt training set. Both models were trained for 5 epochs, with the best overall performance reached after epoch 4.

E. Mixed Corpora

So the ChemProt and artificial data have been kept separate when training our models. Next, two new training sets were created with the script `build_mixed_corpous`, which mixed the ChemProt training set with 10% and 25% artificial data (using chemical-protein interactions) respectively, and then used to train mixed models. As learning more about the potential of artificial data was of large interest during this project, several repeated models of varying training length (10, 15, 20, 30, and 40 epochs) were trained on these sets. All models trained for more than 10 epochs showed similar dips as previously observed in section V-B. While most of the models trained had top F_1-score in the same range as those trained solely on ChemProt data, two models stood out and are shown in table VIII, both trained for a maximum of 10 epochs.

Model	**Best epoch**	**ChemProt**		**pp-change**		**loss**	
		Train	Dev	Train	Dev	Train	Dev
Mixed 10%	9	0.995	0.864	+1.7	+1.7	0.021	0.664
Mixed 25%	10	0.995	0.860	+1.3	+0.5	0.011	0.728

TABLE VIII: Top F_1-score for mixed models with pp-change compared to the Base-5 model as well ass loss. Both models were trained for a maximum of 10 epochs.

The script was also designed to allow adjustment of the weight of each class for the artificial corpus, such that it could be used similar to oversampling but with artificial data instead of duplicates. Due to time constraints however, no such sets where constructed and thus no models trained on this.

VI. Discussion

A. Tokenizer

As previously mentioned in section II-A, the tokenizer essentially makes up the model's vocabulary, mapping the wordpieces to their embedding. Using another tokenizer might therefor potentially result in the model effectively reading the sentence as if made up of completely different

words. This in turn could mean that what the model have learned during pre-training would be rendered mostly useless, as the interpreted data no longer would make sense and have more resemblance to random noise. As both the BERT_{BASE} and SciBERT language models are trained on the English language, their vocabularies most likely share the majority of their tokens. Thus, it might also be possible that a large portion of the text fed to the model is interpreted as intended, even with the wrong tokenizer. That however, would further require that shared tokens also share the same token IDs and corresponding encoding, which seem very unlikely. One could potentially test this by calculating an estimated expected value of F_1-score through randomly assigned classes given the class distribution of the sets, and compare it to the result of the models using the wrong tokenizer. Due to the large amount of stochasticity involved in training neural models, it would still be difficult to draw any hard conclusions without testing using multiple sets and tokenizers. In fact, one could probably simply compare the IDs and encodings of the two tokenizers directly to test this theory.

Regardless of the specifics, the results patently showed that using the wrong tokenizer was very destructive. Since all pre-trained models that utelizes tokenization should have a defined corresponding tokenizer available, there is no real reason in using a different one, and this choice was clearly a simple error.

B. Training Time

When increasing the number of maximum epochs when training models, a much larger variance in F_1-score was observed between earlier epochs in particular. This can be explained by the linear learning rate scheduler that reduces the learning rate after each weight update by a fraction of the number of max epochs and batches. While the Base-40 model performed slightly better than Base-10 on both the training and development set, it suffered from a few other issues. To begin, having larger variance between epochs could suggest a less stable model, though through having a higher learning rate due to the scheduler this is to be expected. What is more alarming is the large difference in loss between learning and development set, which could imply worse generalization properties and much heavier bias towards the training set. Longer training time is also an inconvenience, and more wasteful in terms of resources (though, since the fine-tuning is still a rather quick process, this is not a main concern). All this, together with no dramatic increase in F_1-score, suggests that training for longer than 10 epochs is not very beneficial, at least not without adding some other changes to the training process.

Regarding the sudden drops in F_1-score, an initial suspicion was that it could be due to the randomness in batching during gradient decent, where slight differences are to be expected. Given that the drops are so significantly larger, and consistently appearing after epoch 13 and 24 during most training sessions (as seen in figs. 4, 5 and 8) however, this seem unlikely. The optimizer was another consideration, though this explanation is not overly convincing either since the loss seemingly do not follow the same pattern. While not necessarily true in all cases, one would expect the loss to increase at these large drops in F_1-score, and while true for epoch 13 (as seen in fig. 6) it did not follow the other drops at later epochs. Other spikes in loss of similar magnitude also appear after other epochs, e.g. epoch 18, where no dramatic change in F_1-score can be observed, which adds further suspicion to the behavior.

Nearing the end of the project the learning rate scheduler was once again put in consideration as a potential cause for the sudden performance drops. This type of behavior can be observed when using schedulers that utilizes periodical hard restarts, though as previously mentioned, the scheduler used follows a simple linear decay, without any restarts. There could however potentially be an error in the code that unintentionally restarts the scheduler. On the other hand, the fact the these drops get smaller for later epochs do suggest a lower learning rate at these times. Should one want to continue exploring longer training, it would likely be a good idea do more thorough examination of the schedulers behavior, e.g. monitoring the learning rate during training. Additionally one could try implementing Layer-wise Learning Decay, to have a higher learning rate for the top layers, in an effort to limit the effects of fine-tuning on the models broader understanding of language, and focus on tuning the part of the network more relevant to the classifying task. Ultimately, no definite explanation to these performance drops was found, but alas, since none of the results displayed any strong indication that fine-tuning for more than around 10 epochs had any major benefit, it might not be a top priority in further developments.

For the oversampled models however, both OS-20 and OS-40 showed best performance in earlier epochs (7 and 6), which led to the resulting models having a more balanced loss between the two sets, thus avoiding the issues that affected Base-20 and Base-40. Since the later epochs in training did not show better improvement, similar results should in theory be achieved by using a less aggressive learning rate scheduler with fewer maximum epochs, and might be worth exploring in future development.

C. Artificial Data

If the artificially constructed sentences were a good representation of at least a portion of the sentences taken from the ChemProt data, one would expect that models trained on the ChemProt training set would show some correlation of performance on ChemProt and artificial data. When comparing the results shown in figs. 5 and 8 however, this does not appear to be the chase, which implies that the artificially constructed sentences unfortunately seem to be a poor representation of sentences one would expect to find in articles, at least those included in the ChemProt set. This could also be largely due to the artificial sets being perfectly

balanced among the five class interactions, while both the ChemProt sets are heavily imbalanced, with two of the classes make up ove 70% of the sets. The ChemProt development set is slightly less imbalanced though, which could explain why the artificial models showed better performances on this set. Rerunning these tests with artificially created datasets that have similar class balance to the ChemProt sets might be a good idea for future developments.

Though the artificial model improved on the ChemProt sets when using chemical-protein interactions rather than only proteins, the improvement on the development set was small enough to be within the margin of variation due to the stochastic nature of the training process of neural networks. As for the improvement on the ChemProt training set, one must consider the potential bias due to the chemical names being extracted from that very set, thus we can not conclude that this change in the artificial data had any clear benefit from these results alone. By instead using chemical names from a separate data bank, and training multiple models on both types of artificial sets and comparing the results should give a better understanding of the potential benefits of this method.

While the mixed models provided the top results, one must still consider the user bias. More models were trained on these sets, and most showed no obvious indication of improvement in average F_1-score, so the improvement should be considered within the range of variance due to stochasticity in the learning process.

D. Additional Notes and Considerations

A potential limitation to the presented method of relation extraction is that it is built on using single sentences, but interactions could possibly be described over several. Additionally the author might refer to some previously mentioned entity by other means than it's name when describing an interaction. While this will not directly affect the classification model, it could lead to a final product that may miss important interactions described in some articles during extraction, thus never reaching the classifier.

Throughout this project, the main focus has been on the macro average F_1-score, as it is a good and simple tool to compare models due to it being represented with a single number. In order to get a deeper understanding of how different tweaks and alterations made may affect the model performances, one might need to study other metrics as well. To start, one could study the confusion matrices and the scores for each class, particularly for oversampling and mixed models, since these presumably would have a larger impact of model performance on the minority classes. While included in the scripts, not much time was spent evaluating it, in order to limit the complexity of the project due to time constraints.

As described in Appendix A, the `bert_finetune` currently saves the model after every epoch as separate models, and the evaluation is later done separately with the `evaluation` script. This method is very inefficient in several ways, especially when training lots of models and for many epochs. Every model saved is about 420 MB, which resulted in all the models trained during this project totaling over 200 GB. Additionally, since the `evaluation` script need to load a new model for every epoch trained to compute the metrics, the evaluation actually takes longer than the training. A better solution would be to compute and save the evaluation metrics after every epoch during the finetuning, and only save the best model. This would speed up the process by a lot, as well as reducing the required amount of storage.

When finishing up this report, there was a final realization of how the learning rate scheduler was set to function. It reduces the learning rate linearly, in even steps after each batch, with the total number of steps equal to `[number of batches]x[number of epochs]`. This means that the training data will not be weighted equally, since the learning rate changes within each epoch. A better practice might be to instead change the steps to the number of epochs, and only reducing the learning rate after each epoch. This would result in all of the data being accounted for equally, while still making the model less aggressive towards the end of training. It seem more likely that this would lead to better generalization.

No final evaluation was done on test set yet, as the project is to be continued. Since the models here were picked while monitoring the scores on the development set, there is a heavy risk of user bias towards it. This is one of the risks of relying on methods of early stopping or cherry picking models with regards to whatever is used as validation, and something that in my experience is often overlooked. There is no guarantee that the top performing models presented in this report will also have the best performance on the test set. However, since the goal is to continue finding ways of improving the models, it is usually a good idea to limit evaluating the models on the test set, to avoid ending up with models heavily biased toward it, which ultimately may lead to poor generalization. Additionally, since the goal of the project was not necessarily to find the best possible model, but rather exploring ways to improve them, a better representation for the improvement would have been to use the average top performance of repeated models, with accompanying standard deviation. This would have better displayed the trending effects of the changes made, and possibly simplified further work.

VII. Conclusion

When working with BERT models, or any model that utilizes tokens, it is highly important to use the correct tokenizer. The use of artificially constructed sentences could possibly help to improve the performance of relation extraction models. At this stage it is difficult to evaluate the potential due to the current method of construction being limited to a single type of sentence structure. If one would like to continue

exploring using artificially constructed sentences, the main focus should be on including a wider variety of sentences structures, and possibly use chemical names from some data bank, in order to better represent the data found in real articles.

The final summary of the best models achieved are displayed in table IX, with pp improvement compared to the corresponding models of the previous group when evaluated on the ChemProt sets. The model trained on the ChemProt training set and 10% artificial data ended up scoring the highest. As previously mentioned however, due to training more models on the mixed sets, this naturally also increase the chances of finding a better model. While the three models of choice do perform very similarly, and could potentially be more related to the variance between repeat models, the results do indicate a potential benefit in both oversampling and adding a small portion of artificial data.

Model	**Best epoch**	**ChemProt**		**pp-change**	
		Train	Dev	Train	Dev
Base-10	9	0.994	0.855	+11.4	+34.3
OS-20	6	0.995	0.858	+2.5	+20.8
Mixed 10%	9	0.995	0.864	−	+48.4

TABLE IX: Top F_1-score for models with pp-change compared to the top models of the former group. Base-10 is compared to their Baseline model, OS-20 to their Oversampled model and Mixed 10% to their Artificial + ChemProt model, as they have the closest resemblance (though one could argue that the Mixed 10% would be more comparable to the oversampled model, as the Artificial + ChemProt model was mainly trained on artificial data).

VIII. Future Developments

Several aspects that should be considered exploring in future work: (1) Widening the variety of artificial data through different types of sentence structures and authors (2) Use a larger database of chemical names for the artificial data (3) Test models trained on ChemProt data on artificially constructed sets that share the same class distribution as ChemProt (4) Creating more balanced mixed sets adding a larger portion of artificial data of minority classes, and train models on these sets (5) Investigate optimizer, learning scheduler and hyperparameters in more detail (6) More detailed analysis of measuring metrics, particularly the confusion matrices and the performance changes in each class due to the alteration made to the models

Acknowledgment

A special appreciation goes to the project supervisor Sonja Aits and Rafsan Ahmed and their wonderful support throughout this project.

References

[1] M. L. MacDonald, J. Lamerdin *et al.*, "Identifying off-target effects and hidden phenotypes of drugs in human cells," *Nature chemical biology*, vol. 2, no. 6, pp. 329–337, 2006.

[2] L. Axlin and K. Broman. Aitslab/bionlp/lykke_klara. (Retrieved 12-05-22). [Online]. Available: https://github.com/Aitslab/BioNLP/tree/master/lykke_klara

[3] BioCreative. (2017, 11) ChemProt corpus: BioCreative VI (Resources). (Retrieved 15-04-22). [Online]. Available: https://biocreative.bioinformatics.udel.edu/news/corpora/chemprot-corpus-biocreative-vi/

[4] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] J. Devlin, M.-W. Chang *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152.

[7] Y. Wu, M. Schuster *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[8] O. Taboureau *et al.*, "ChemProt: a disease chemical biology database," *Nucleic Acids Research*, vol. 39, no. suppl_1, pp. D367–D372, Oct 2010. [Online]. Available: https://doi.org/10.1093/nar/gkq906

[9] Alexander and Petter. Aitslab/nlp_2021_alexander_petter. (Retrieved 12-05-22). [Online]. Available: https://github.com/Aitslab/nlp_2021_alexander_petter

[10] N. Broman. Aitslab/bionlp/nils. (Retrieved 12-05-22). [Online]. Available: https://github.com/Aitslab/BioNLP/tree/master/nils

[11] Cell Line Ontology. (Retrieved 15-04-22). [Online]. Available: https://obofoundry.org/ontology/clo.html

[12] UniProt. (Retrieved 15-04-22). [Online]. Available: https://www.uniprot.org/

APPENDIX A
SCRIPTS AND DATA

A. Old and Updated Scripts

1) `main`*:* The main script used to run the pipeline. Calls the methods ___
Updated to also incorporate `build_mixed_corpus`

2) `add_custom_labels`*:* Adds the custom labels to the corpus
Updated to also add custom labels to the sample and test set

3) `build_art_corpus`*:* Constructs the artificial corpora using randomized building blocks. Creates a separate file for each label, then combines them into one file.
Updated to use the artificial chem-prot interactions

4) `bert_finetune`*:* Fine tunes the BERT model on the ChemProt corpus and saves as a new model after each epoch.
Updated to use the correct tokenizer for SciBERT

5) `evaluation`*:* Loads and evaluate the models saved after every epoch from `bert_finetune`, and saves the statistics of precision, recall, f-score as well as confusion matrices to a file.
Updated metrics path to fit my machine. Added micro and weighted averages as well as confusion matrices to the list of metrics.

6) `plot`*:* Plots f-score, precision, recall, accuracy-loss curves over epochs. *Outdated due to the use of a notebook (see below)*

7) `extract_relations`*:* Processes the ChemProt data to a line-by-line format that can be fed to the BERT models.

B. New Scripts

1) `plotting_notebook`*:* Notebook used for plots. Due to making many small changes in the plots while analyzing the data, a notebook allowed a smoother workflow.

2) `extract_chemical_names`*:* Extract the names of the chemicals in the ChemProt train set.

3) `build_mixed_corpus`*:* Combines the baseline corpora and a __percentage__ of artificial data (with the option of adding weighted support for the labels).

C. Corpora (.txt)

1) chemprot_train: ChemProt training set

2) chemprot_dev: ChemProt development set

3) chemprot_sample: ChemProt sample set

4) chemprot_test: ChemProt test set

5) artificial_pp_train: Artificial training set with 5000 protein-protein interactions, 1000 of each class

6) artificial_pp_dev: Artificial development set with 2500 protein-protein interactions, 500 of each class

7) artificial_cp_train: Artificial training set with 5000 chemical-protein interactions, 1000 of each class

8) artificial_cp_dev: Artificial development set with 2500 chemical-protein interactions, 500 of each class

9) mixed_train_10: Mixed training set consisting of ChemProt training set with an additional 10% artificial data, using chemical-protein interactions

10) mixed_train_25: Mixed training set consisting of ChemProt training set with an additional 25% artificial data, using chemical-protein interactions

D. Other

1) `config.json`*:* Configuration file, specifying paths for models, corpora, metrics and plots as well as details for training and construction of corpora.

APPENDIX B
EARLIER MODELS

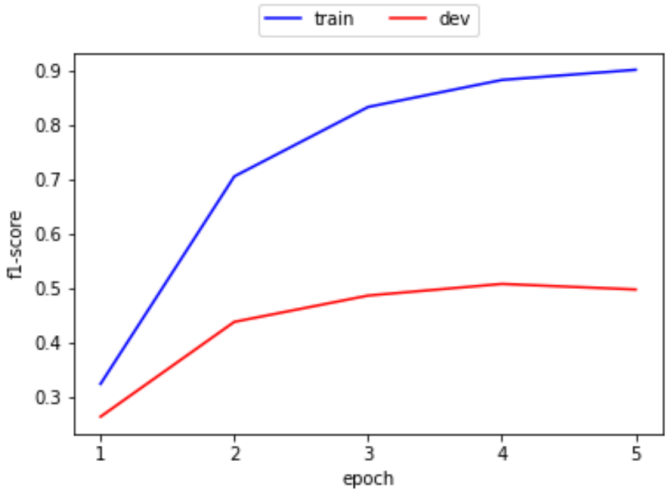


Fig. 9: Macro average F1-score for the baseline model

Model	Best epoch	Chemprot		Artificial	
		Train	Dev	Train	Dev
Baseline	4	0.88	0.51	–	–
Oversampled	5	0.97	0.65	–	–
Artificial	5	–	0.30	1.00	1.00
Artificial + ChemProt	$5+1$	–	0.38	0.57	0.56

TABLE X: F_1-score of the previous group's best models

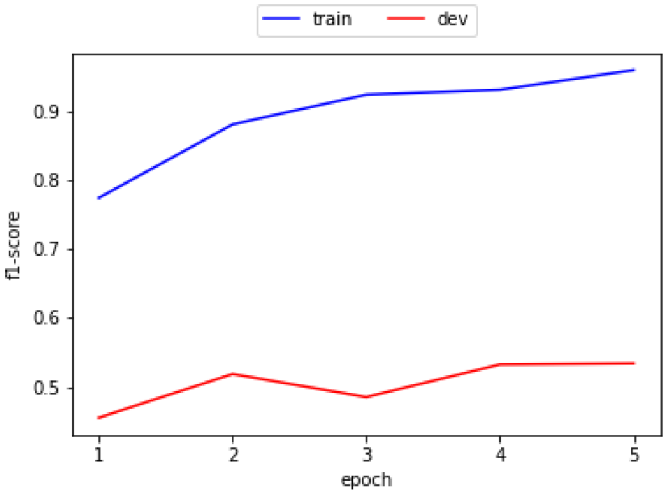


Fig. 10: Macro average F1-score for the oversampled model

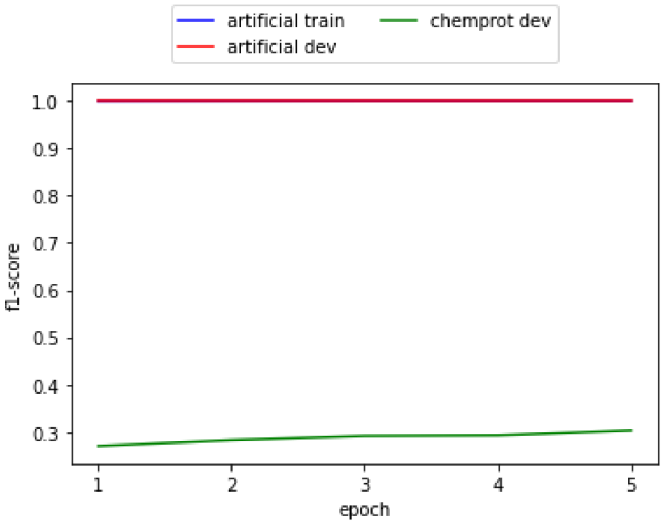


Fig. 11: Macro average F1-score for the artificial model