Chemical & Disease Named Entity Recognition

Erik Ziegler
Hasso-Plattner-Institut
Potsdam
Erik.Ziegler@student.hpi.de

Marcel Schmidberger
Hasso-Plattner-Institut
Potsdam
Marcel.Schmidberger@student.hpi.de

Jan Philipp Sachs
Hasso-Plattner-Institut
Potsdam
Jan-Philipp.Sachs@hpi.de

Leo Heineken
Hasso-Plattner-Institut
Potsdam
Leo.Heineken@student.hpi.de

Tillmann Int-Veen
Hasso-Plattner-Institut
Potsdam
Tillmann.Int-Veen@student.hpi.de

Abstract

Project Summary - We propose a bidirectional LSTM (BiLSTM) network architecture for named entity recognition (NER) of chemicals and diseases in the BioCreative V challenge dataset. We evaluate the effects of pretrained word embeddings, additional char inputs and the batching method on the overall model performance. The model with the best performance used pre-trained embeddings, and char input, and yielded F1 scores of 0.85 and 0.79 for chemicals and diseases.

1 Introduction

Natural language processing (NLP) refers to a set of methods to extract, analyze, or generate text data. One of these approaches is named entity recognition, a task that aims at automatically identifying words belonging to pre-defined categories of interest. However, due to characteristics like descriptive entity names and multiple spelling variants for one entity, the task of automatic extraction is challenging [38]. Since manual entity extraction, albeit accurate, is labor- and time-intensive, various methodological approaches to NER promise a considerable acceleration of this task. With an increasing level of automation and categorization quality, NER has also become interesting for the biomedical domain. In addition to the advances that NER brings for of the categorization of words, it lays the foundation for more complex tasks such as relation extraction. For the latter one, the most prominent areas are gene-disease or chemical-disease relation extraction from scientific literature [39, 2]. Similarly, NER may even serve as a medical knowledge generation or disease surveillance tool, as has recently been shown in the context of the SARS-CoV-2 pandemic, e.g., to gain geographically locatable information about symptoms and healthcare system treatment resource availabilities by using data from Twitter [24, 22].

In in our work we focus on the automated extraction of diseases and chemicals from scientific articles in the biomedical domain. In section 2 of this paper, we highlight the most relevant existing approaches to the same challenge. Section 3 is dedicated to a more in-depth description of the dataset resource; we tapped into the preprocessing we applied on it. section 4 covers the major architectural decisions we took and section 5 evaluates these. In the last section 6, we embed our results into those from literature on the same task and give an outlook on those methods that we deem to be promising to improve our results.

2 Related Work

The emergence of automatic NER systems has a long history with five categories of NER methods evolving over time: rule-based, dictionary-based, machine learning-based, deep learning-based and hybrid/ensemble [13][17].

Rule-based systems detect entities by scanning the text for tokens that match predefined rules like a regular expression. This works well for information that follows certain structural rules like emails or zip codes, but has its shortcomings if there is no clear structure to match, e.g. surnames. In this case a dictionary-based systems would be more applicable. By using a dictionary of all the possible values of entities, e.g. a register of all known names, tagging the entities by matching occurrences in the text becomes easy, but may at the same time lead to false positives as ”Fox”can both be an animal as well as a common surname. Machine learning-based approaches use different multi-class classification techniques. However, just as the previous systems, most machine learning-based approaches do not consider the context of the sentences or previous word labels for the classification. This imposes a drawback for learning more complex structures. Conditional Random Field (CRF) models use the classification information from previous tokens and infer the probabilities of certain entities following in this context to improve the recognition. Handling of forward labels is difficult however. Most deep learning models overcome these limiations and by taking into account the context of adjacent (i.e., previous and next) labels as well as far more previous information, which greatly improves their ability to learn label patterns in texts. Hybrid approaches combine two or more systems for better results, e.g. use a rule-based system for email tagging and deep learning for other labels.

Current state-of-the-art deep learning approaches utilize bidirectional long short-term memory (BiLSTM) networks - with or without conditional random field models (BiLSTM-CRF) [11, 14, 21]. Modifications such as multi-task learning [32] or transfer learning [9] are common and have shown to improve results on the individual tasks [33].

The main drawback of BiLSTM-CRF networks is a rather weak generalizability. The models can perform worse when applied to domain-specific datasets which they have not been trained on [7]. Compared to feature-based approaches, deep learning approaches can discover hidden characteristics of unlabelled data automatically [25], but typically require a large dataset to be trained on. Recently, Bidirectional Encoder Representations from Transformer (BERT) models were applied to biomedical NER (BioNER) tasks, and the experimental results indicated slight performance improvements over existing BiLSTM approaches [16]. As of July 2020, peak F1-score performance across several different BioNER datasets is achieved by the DTranNER framework [12]. This archtecture achieves F1 scores of 0.94 on the chemical NER and 0.87 on the disease NER on our dataset (see section 3).

State-of-the-art models need a useful and sensible representations of the text. Pre-trained words embeddings have been proven to be a valuable asset to that end [11]. Albeit the fact that they are not the only representation at hand, they can - if trained on a domain-specific corpus - help to solve embedding tasks within the same domain [34, 31, 27]. Another rather recent but canonical addition to representation of text is the addition of character embeddings in terms of char input at an a single character or n-gram level [10]. Approaches leveraging this technique benefit from improved recognition and suffer less of word variants including prefixes, suffixes, orthographic features (casing), and out-of-vocabulary issues.

A comprehensive overview of the history of NER and of the methods that are applied in this field can be found in [17]. The potential impact of some of the methods that were not applied in this iteration of our NER architecture, is explained in the discussion section.

3 Dataset

The *BioCreative V* challenge dataset, *BC5CDR*, serves as the primary dataset of this project[1]. The BioCreAtIvE (Critical Assessment of Information Extraction Systems in Biology) organization and its challenge aimed to create a gold standard corpus for text processing, as previous datasets were

[1]https://biocreative.bioinformatics.udel.edu/tasks/biocreative-v/track-3-cdr/

sparse and siloed, leading to an inability to evaluate text extraction system performance and scalability.

The dataset itself consists of 1,500 PubMed articles with 4,409 annotated chemicals, 5,818 diseases and 3,116 chemical-disease interactions [18], out of which we use the first two label types. Therefore, non-null entities make up only about 11% of our dataset. The entity annotations of the chosen corpus include normalized concept identifiers (MeSH term IDs [2]) and mention text spans. Using the Jaccard similarity coefficient, it achieved an inter-annotator agreement of 96.05% and 87.49% for chemicals and diseases. For training, development (validation), and testing, the dataset is split into three distinct, equally sized subsets consisting of 500 annotated articles each.

3.1 Preprocessing

The entire dataset is provided in XML format. Each example contains the title and body of a PubMed article and annotations for the chemicals and diseases of these two components. Annotations consist of the char offsets of the entity mention in the text and the entity of that mention.

The target format is a list of sentences where each sentence has a list of tokens and a list of ground truth labels. This format is achieved by the following preprocessing steps:

1. A custom XML parser is used to retrieve the raw text and annotations. The text is then split into sentences. Each sentence is tokenized. Then, for each token, the label is received from the raw annotations by comparing the char offsets with the original text. If no annotation is given for a token, it is assigned the null class label. This first processing step results in a list of sentences containing a list of tokens and a list of labels.

2. In this step, the raw text tokens and labels are transformed into numbers. The three labels are just encoded as 0, 1 and 2. For the text tokens, we assign each individual unique word to a unique number, resulting in a word-to-index (and index-to-word) mapping that also can be stored separately to encode any arbitrary sentence. For the case that a word should be encoded that is not in the original vocabulary, we introduce an *unknown token* with own index. For dummy tokens that are added to sentences for padding, we introduce a *padding token* with own index. In total, the word-to-index mapping has a size of $V + 2$ where V is the original vocabulary size and the additional two are mappings for unknown and padding token. All tokens in the data resulting from step one are encoded with the mapping. For creating the mapping the whole vocabulary of train, development and test set should be used to have a vocabulary that optimally covers the data domain.

With pretrained embeddings If we make use of pretrained embeddings the word-to-index mapping should be built from the vocabulary of the embeddings instead of the dataset, because out-of-vocabulary words would have no embedding anyway. Additionally, the embeddings' vocabulary may also cover words that are not in any of the sets, making it possibly perform better for unexpected input. Using pretrained embeddings also requires building a word-to-index-to-embedding mapping during preprocessing.

4 Architecture and Training

The core of our model consists of a bidirectional LSTM, meaning that it uses two LSTMs: one in forward direction, one backwards - starting with the end of a sentence. Thereby the model is able to incorporate the context of previous and next labels into the classification of the current token. As discussed in section 2 this follows state-of-the-art approaches.

The detailed model architecture is shown in Figure 1. It has two inputs: the token input that accepts a batch of B sentences with N tokens where each token is encoded with the word-to-index mapping; and the char input where for each token in each sentence the chars padded to length C are used (also in char-to-index encoding). The token input is embedded with an embedding layer to a vector of size E_W. The embeddings are either pretrained or learned. Both options are evaluated in subsection 5.5. The char input is also embedded with an embedding layer to vectors of size E_C. On the char embedding, 2D convolution is applied followed by a 2D maxpooling layer. The use of

[2]www.nlm.nih.gov/mesh

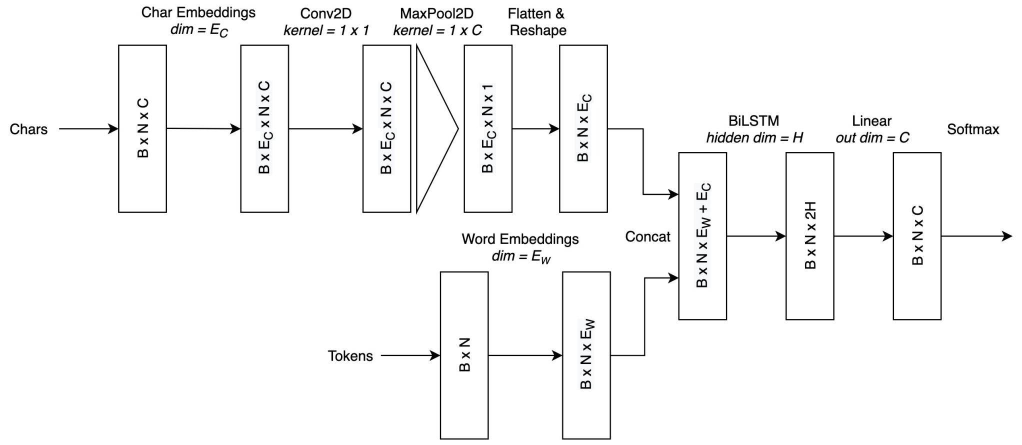


Figure 1: Model architecture. B denotes batch size, N denotes sentence length, C denotes the padded word length, E_W denotes word embedding dimensions, E_C denotes the char embedding dimension, H denotes hidden dimension, C denotes number of classes.

char embeddings is evaluated in subsection 5.8. The results is flattened and concatenated with the token embedding. The concatenated tensor is fed into an bidirectional LSTM with a hidden size of H resulting in a $B \times N \times 2H$ tensor. Lastly, a linear layer is applied bringing down the size to the number of classes $2H \rightarrow C$ with a softmax activation layer to create a probability distribution along the classes. This approach proved to be a suitable choice for many classification problems [1].

The model is trained with a weighted cross entropy loss because of the data sparsity. An stochastic gradient descent (SGD) optimizer with weight decay is used to regulate model complexity. Different options for batching are discussed in subsection 4.1.

The model was implemented using PyTorch (version 1.5.1) and the frameworks loss and optimizer functions were used for training.

4.1 Batching

Many deep learning methods have a straightforward approach to mini-batching by splitting the dataset into mini-batches of a fixed size, feeding them into the network and applying stochastic gradient descent. In NLP tasks we commonly deal with sentences as the input for the network, which have no uniform length. Feeding a mini-batch of sentences that have not the same length is not possible, because the data would not be rectangular. While e.g. in image detection the problem of images of different size is tackled by warping all images to a uniform size, there are several options to create mini-batches from data consisting of sentences. We evaluated three approaches: stochastic learning, padded sentence batching and unique sentence length batching.

Stochastic learning Feeding each sentence in the network individually (resulting in mini-batches of size 1). This way the input data is obviously always rectangular. This method negatively affects the training duration since the model can not parallelize computation over more than one sample. This method is less memory demanding because only one sample has to be stored in memory. The estimate for of the true gradient computed by a single example can be very noisy and may not move the weights precisely down the gradient at each iteration. However, this ”noise” is considered advantageous and actually might improve results [15].

Padded sentence batching All sentences in the training set are padded to a uniform length using a special padding token with the null vector as it’s embedding. As the uniform length, the longest sentence length in the dataset can be used, or, if the longest sentence is disproportionately long, a reasonable length can be used and all longer sentences are removed from the training set. This method allows for proper mini-batching on the padded data because of the uniform size, which improves training duration due to the ability to parallelize over the whole mini-batch. But because we introduce a large number of tokens that are labeled with the null class, the data sparsity problem gets even worse, resulting in the increased need for countermeasures like weighting the cross entropy loss in favour of non-null class labels.

Unique sentence length batching All sentences that are from the same size are considered as a mini-batch. Very small mini-batches of just a few examples (occurring especially for very large sentence lengths) may be omitted, because they negatively affect the training duration. Very large mini-batching may be split into to multiple batches with a fixed maximum size for mini-batch, because large batches heavily impact memory usage.

Method	**Epoch duration**
Stochastic learning	$\approx 30s$
Padded sentences (batch size 100)	$\approx 3s$
By sentence length	$\approx 20s$

Table 1: Average epoch duration for different batching methods. Values are only approximation and actual value is dependant on model architecture and implementation.

Table 1 shows the impact on training duration for the described approaches to mini-batching. As discussed before, padded sentence batching yields the lowest average epoch duration.

4.2 Padding

As discussed using a fixed sentence length has several advantages like allowing proper mini-batching and therefore faster training. If the model is flexible enough to allow variable sentence lengths, the fixed length padded sentences can be used during training to benefit from the mentioned advantages and during inference or for evaluating on other sets, single batching can be used to allow prediction on sentences of any length.

But if the model is not flexible and requires to manual set an exact sentence length that every input sentence has to have, this result in implications for evaluation of the model and running inference. This is for example the case if the model uses convolutions which are used in our model on the character embedding. If the sentence length of the input is fixed, sentences that are longer have to be split in chunks of the fixed length and fed into the network separately. This negatively affects prediction accuracy because word relation information within the sentence is lost. But setting the padding to high increases data sparsity and is unnecessary for majority of sentences in the dataset.

The similar discussion can be led about padding the chars of a word, which is mandatory to have rectangular data regardless of whether the sentence is padded. Padding chars with a high number makes the data more sparse and padding chars with a low number increases the risks of information loss when running inference on a long unexpected word because overhanging chars must be omitted (the concept of feeding multiple times into the network does not make sense in the context of character input).

4.3 Embeddings

Word embeddings are used to convert words into vectors. Here the objective is to use similar vectors for words with similar meanings. Thereby word embeddings provide a semantic clustering, which helps the model to better understand the meaning of words as similarity and unlikeness in represented in vector distances. These vectors are usually of high dimensionality.

For our model we can use either pretrained or selftrained embeddings.

As pretrained embeddings we use BioWordVec [36], which are biomedical word and sentence embeddings trained using PubMed and the clinical notes from MIMIC-III Clinical Database. They make up 13 GB of data and cover over 99% of word occurrences in our data set.

	Chemical			**Disease**		
Model type	train	dev	test	train	dev	test
BiLSTM (Baseline)	0.96	0.63	0.61	0.94	0.67	0.67
BiLSTM + Dropout (0.5)	0.92	0.62	0.61	0.89	0.67	0.67
BiLSTM + Char CNN	0.98	0.69	0.67	0.97	0.67	0.69
BiLSTM + Pretrained Embeddings	0.91	0.85	0.85	0.85	0.80	0.79
BiLSTM + Pre Embeds + Char CNN	0.93	0.86	**0.85**	0.86	0.80	**0.80**
sota DTranNER framework [12]	-	-	*0.94*	-	-	*0.87*

Table 2: **F1 scores for all evaluated models.** F1 scores are shown for chemical and disease on the train, validation, and test sets for all evaluated methods. Bold font result indicates best performance per entity category. Italic shows the comparison to the state-of-the-art performance.

5 Evaluation

5.1 Experiments and Metrics

All experiments where carried out on a dedicated NVIDIA GeForce RTX 2080 GPU with 12 GB of RAM, running NVIDIA-SMI driver version 450.36.06 and CUDA version 11.0. TensorBoard has been used to track training and test losses online.

To evaluate the entity recognition accuracy, F1 scores have been calculated according to

$$F1\,score = 2\cdot\left(\frac{precision \cdot recall}{precision + recall}\right) = \frac{2 \cdot true\,positives}{2 \cdot true\,positives + false\,positives + false\,negatives}$$

for the chemical and disease classification bins, respectively. This evaluation approach follows the one chosen in the related literature [3][12]. Throughout the entire documentation, the F1 score for the null class is not shown here, but was reported $>$ 0.99 for all experiments.

5.2 Baseline Model

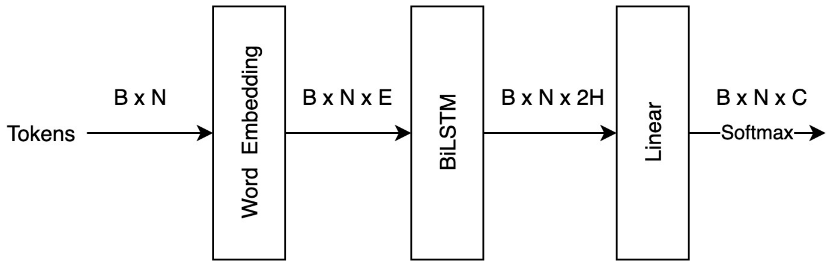


Figure 2: Baseline model architecture. B denotes the batch size, N the sentence length, E the embedding dimension, H the hidden dimension of the LSTM and C the number of classes (3).

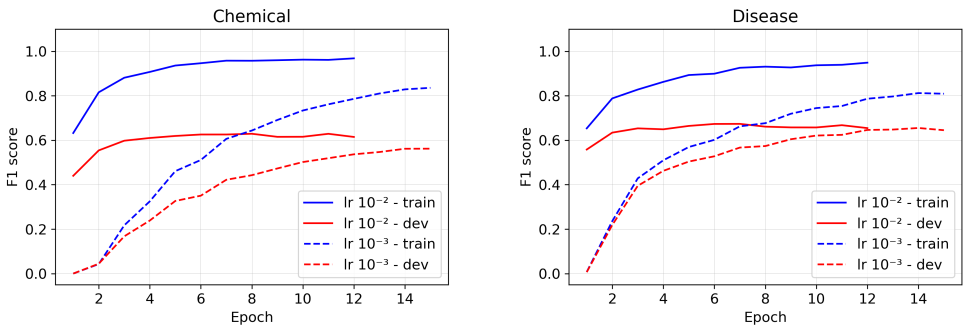
The baseline for our experiments is a simple version of the model architecture: The input only consists of tokens, followed by an embedding layer (embedding dimension of 200), a bidirectional LSTM (hidden dimension of 100) and a final linear layer with softmax activation. The architecture of that model is visualized in Figure 2. It is trained with an SGD optimizer with a learning rate of 0.01, momentum of 0.9 and weight decay of 10^{-4} and a normal (non-weighted) cross-entropy loss. Within one epoch, each sentence is fed individually into the network (stochastic learning / single batching).

The details on the performance parameters of this model are shown in Table 2, evaluated with regards to F1 scores of the two candidate classes. The baseline model performs reasonably well on the data very quickly and converges after only a few epochs. At the same time, the model overfits the training data, resulting in a significant difference between train (around 0.95) and dev / test score (around 0.65).

Figure 3 shows the F1 score for the two classes after each epoch, showing that methods like early-stopping would not improve the result because the train and dev graphs are not diverging at any time, but rather following a similar trajectory and the dev graph just converges at a lower score.

5.3 Hyperparameters

Learning rate For the baseline model, we evaluated the reduction of the learning rate to 0.001 to verify that the gradient descent did not just find a non-optimal local minimum because of a potentially outsized learning rate. As shown in Figure 3, the development graph still converges on roughly the same F1 score, but just slower.



(a) F1 score *Chemical* (b) F1 score *Disease*

Figure 3: **Effect of different learning rates on F1 scores.** F1 scores during training for different learning rates: Baseline model - learning rate of 10^{-2}, dashed line - learning rate of 10^{-3}).

Momentum Using no momentum instead of the 0.9 of the baseline SGD shows roughly the same results as reducing the learning rate to 0.001 and is therefore not displayed here. The model converges slower and does not improve in performance in comparison to the baseline results.

After evaluating different values for learning rate and momentum, we decided to keep the original values of a learning rate of 0.01 and momentum of 0.9 because the model converges faster without negative impact on performance.

5.4 Dropout

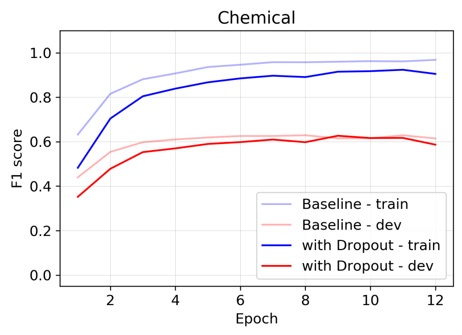
Due to the overfitting of the baseline model on the training data, we evaluated the effect of adding a dropout layer to the network after the bidirectional LSTM that randomly sets values in the tensor that is fed into the final linear layer to zero with probability p . In our experiment, we used a dropout probability of 0.5.

The resulting performance is listed in Table 2 and the progression of the F1 scores during training is visualized in Figure 4. The model performance is slightly worse on the training set and the performance on development and test sets does not change significantly. The big gap between the sets (overfitting) remains.

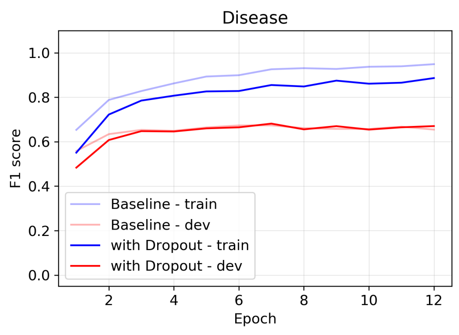
5.5 Pretrained Embeddings

In the baseline model, the word-to-vector embeddings are approximated on the fly during training. Since this process only creates useful embeddings for thosen tokens that are part of the training set, the usage of pretrained embeddings that represent the whole domain could improve performance on the development and test set.

We used the BioWordVec embeddings that are pretrained on the PubMed text corpus, because our training data is from the same source and from the same domain.



(a) F1 score *Chemical*

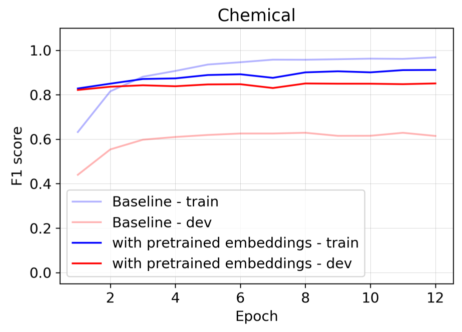


(b) F1 score *Disease*

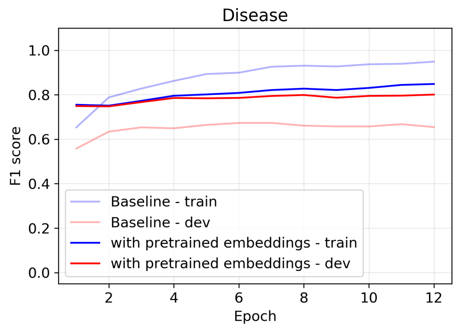
Figure 4: **Effect of dropout on F1 scores**. F1 scores during training for baseline model and model with dropout at probability p = 0.5 .

As shown in Table 2 the pretrained embeddings improve the performance of the model significantly. Although the F1 score for training set is lower than before with 0.91 for the Chemical class and 0.85 for the Disease class, the F1 scores on the development and test sets improve to around 0.85 for the Chemical class and 0.80 for the Disease class.

As shown in Figure 5, the model also achieves high F1 scores faster and is above 0.8 F1 score for the Chemical class and above 0.75 for the Disease class after the first epoch. Performance still increases slightly for a larger number of epochs.



(a) F1 score *Chemical*



(b) F1 score *Disease*

Figure 5: **Effect of pretrained embeddings on F1 scores.** F1 scores during training of baseline model compared to similar model with pretrained embeddings.

5.6 Padding

So far, all experiments used single batching / stochastic learning during training. Though, as mentioned in subsection 4.1, this only works for models that have the ability to process variable input sizes. Later experiments using char inputs require the network input to be of fixed size, meaning that padding has to be introduced. Before evaluating methods that rely on padded input we want to evaluate how the baseline model is impacted by padding the input.

In this experiment, we only pad the input sequences during training, the evaluation still feeds sentences individually into the network (because the baseline model technically still allows that). The sentences of the train set are padded to the maximum length of sentences in the train set, which is 111 for our tokenization. This introduces a total of 408440 null-class labels to the training data, increasing data sparsity significantly. Now that the data is rectangular, we can use a batch size of 100 which makes training faster.

As shown in Figure 6 by the dotted lines, the network training with sentences padded to length 111 and no weighted loss struggles to learn to label non-null class tokens correctly, because the number of null-class labels is too high. Predicting the non-null classes has such a low impact on the loss that the model just learns to predict the null-class. Only after many epochs the F1 score starts to increase.

As described in subsection 4.1, a possible countermeasure is to reduce the maximum length of the sequence for the training set. This is accomplished by cutting off sequences that are longer than a specific threshold, reducing the amount added null-class tokens. We evaluated the reduction of the padded sentence length to 20. This adds a total of 13722 null-class labels and removes 28250 tokens from sentences that are longer than 20 tokens. The performance is shown in Figure 7 by the dotted lines. Although the model learns better than with a padded sentence length of 111, it still learns only slowly and has not converged after 100 epochs.

Generally, training with mini-batches of padded sentences increases the number of epochs that are necessary to achieve a similar performance than the model with stochastic learning. This is simply because the model makes fewer optimizer steps (only after a mini-batch of 100 samples instead of every sample). But on the other hand, epochs with mini-batching are faster as explained in subsection 4.1.

5.7 Weighted cross-entropy loss

Since the training data is sparse with approximately 11% of the training data tokens labeled as a non-null class and using padded sentences only increases this problem, weighting the loss in favour of the non-null classes might help the model learn to pay more attention to labeling non-null class labels correctly than labeling the null class.

The weights are applied to the cross-entropy loss as follows:

$$loss(x, class) = weight[class] \left(-\log \left(\frac{\exp(x[class])}{\sum_j \exp(x[j])} \right) \right)$$

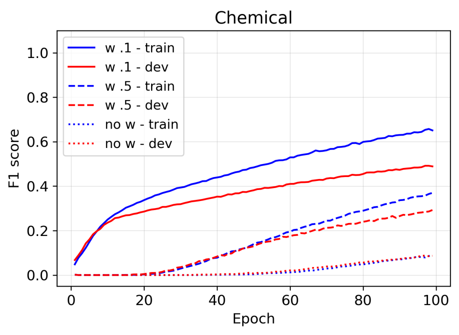
We evaluated two weighting schemes in combination with the two maximum sequence lengths from subsection 5.6: *(1)* non-null class weight 1.0, null-class weight 0.5 and *(2)* non-null class weight 1.0, null-class weight 0.1. The performance of different weightings is shown in Figure 7 and Figure 6 for the sentence lengths 20 and 111. It shows that weighting the loss helps the model to converge faster but does not influence the overall performance. In Figure 7 it becomes obvious that non-null class weights of 0.5 and 0.1 converge to the same F1 score and that the 0.1 weight performs better throughout the training.

Based on this experience, further experiments with padded sentences are performed with a maximum sentence length of 20 and a weighted loss with weighting scheme *(2)*.

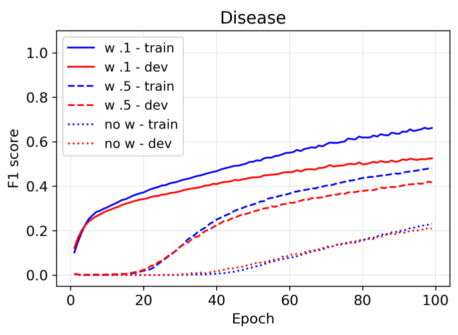
5.8 Char embeddings and Convolutions

The baseline model uses tokens as its input. Inspired by other works in the field of NER, we evaluated adding character information as a second input. The model architecture with character inputs is shown in Figure 1. The characters of each token are fed into the network and each character is vectorized by an embedding layer (embedding dimension of 30). Then, 2D convolution is applied on the embedding followed by a 2D maxpooling layer. The result is flattened and concatenated with the token embedding. The rest is the same as the baseline model. The goal is to give the model additional information to learn from in order to improve results.

The performance is listed in Table 2 and shows an F1 score improvement for the chemical class. The model is successfully able to utilize the additional char inputs to learn certain chemicals labels. This is most likely the for chemical annotations that dependent on specific chars which are not well represented in the token embeddings like *[3H]-naloxone*. The training is shown in Figure 8, the model achieves similar results as the baseline model for the Disease class after around 100 epochs.

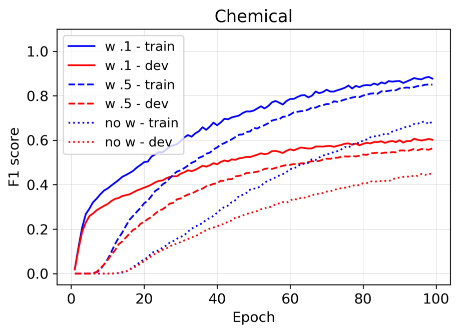


(a) F1 score *Chemical*

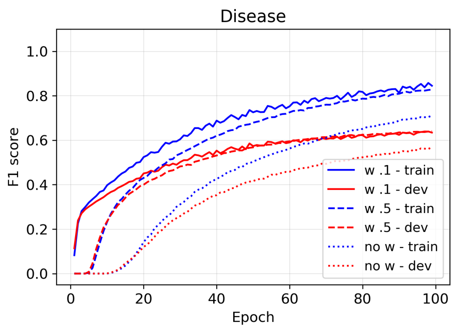


(b) F1 score *Disease*

Figure 6: **Effect of loss weights on F1 scores, sentences padded to maximum length.** Baseline model trained with sentences padded to length **111** (maximum sentence length in the training set) and batch size 100. Comparison of different loss weights for the null-class.

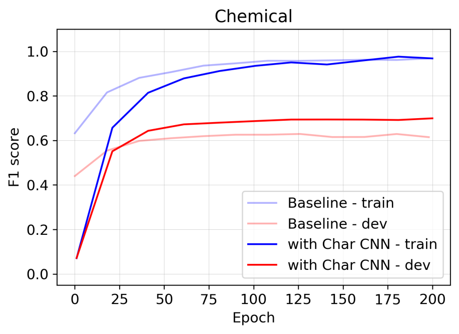


(a) F1 score *Chemical*

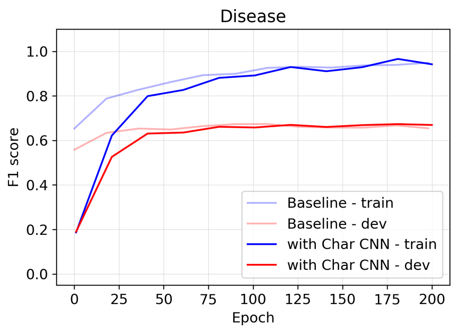


(b) F1 score *Disease*

Figure 7: **Effect of loss weights on F1 scores, sentences padded to length 20.** Baseline model trained with sentences padded to length **20** and batch size 100. Comparison of different loss weights for the null-class.



(a) F1 score *Chemical*



(b) F1 score *Disease*

Figure 8: **Effect of character input convolution on F1 scores.** Comparison of baseline model with additional Char CNN input. (Note that the baseline graph is scaled up along the epoch axis to better visualize the differences. It does not display the actual epochs.)

5.9 Method combination

Based on the previous result, combining the char inputs with pretrained word embeddings should yield the best results. As shown in Table 2, this model configuration performs the best on all of the sets but only slightly better than just using the pretrained embeddings an no character inputs.

6 Discussion

In the following, we shortly assess our results again and then give an extensive outlook on methods and approaches that have already shown to improve the performance on biomedical NER tasks.

To start with, our best results have been achieved by a combination of a BiLSTM architecture, using pre-trained, domain-specific word embeddings and adding char embeddings to the latter representation of content. Our (average) F1 score of 0.82 on both target classes is in close vicinity to the best literature-reported (averaged) F1 score on the same dataset achieved by BiLSTM models without further architectural modifications (0.839) [3]. Adding a CRF layer or completely replacing the architecture by BERT adds 0.02 to 0.06 to the F1 scores [12]. The recently published DTranNer architecture even reaches a peak performance increase of 0.09. The performance increase may at a first glance not be groundbreaking, however, such improvements can be noticeable in a real-life applications. As a general observation, it seems that further performance gains above a combined F1 score of approximately 0.85 are becoming increasingly difficult to achieve.

In order to understand more details about the model performance, it would also be necessary to investigate the model performance on a word-by-word level. Thus, words with poor performance in the test set may unveil patterns of shortcomings. Another limitation of the current approach is that it has not been evaluated on another test set. Despite the fact that with 500 articles (although selected from a proprietary database which is a subset of PubMed), the chances of introducing a bias to the dataset is rather low, the model might improve or deteriorate on a different existing annotated PubMed subset [18]. Training could therefore incorporate more datasets. Considering similar applications but on different data modalities such as clinical notes, the current model would have to be re-evaluated and might benefit from more sub-domain-specific embeddings and training data. The same direction of thoughts applies when looking at the age of the currently used dataset and that after five years, more and especially new concepts and terms might have entered the biomedical field. To our knowledge, there has been no evaluation yet to take this into consideration.

Within the network parameters themselves, there is more exploration depth including, but not limited to: the number of hidden layers, the number of output (fully connected) layers, the use of activation functions.

Looking beyond the adjustments we made to the model and had subsequently evaluated, there is a large number of concepts that can serve as leverage points for performance improvement. Their common denominator is that they require significant changes of the underlying architecture and/or need to leverage new information that either is already in the data or comes as additional data resource.

In the context of LSTM architectures for NER, recent works have shown that the network performance becomes better in multi-task learning settings, both in the general[30] and in the biomedical domain [33]. These approaches add **part-of-speech (POS)** tagging to the named entity recognition. Similar to NER, POS tagging is a classification task with labels corresponding to the grammatical group, e.g., noun, verb, or adjective. Noteworthy is the fact, that tokens/words (separated a blank space) have a 1:1 relationship to POS tags, whereas NER tags follow the BIO (Begin, In, Out) structure and therefore do not necessarily correspond to an exact word but often to shorter or longer character sequences. POS tagging as one further input feature for NER tasks promises to yield better results, as the NER component of an architecture can focus on those tokens labelled with a noun tag and thus refine the entity tags [5]. POS-annotated, ready-to-use datasets are available [35].

Unlike some of the traditional, rule-based architectural approaches to language modeling and specifically NER, the results of those involving deep learning typically improve with increasing size of training data. However, this is a common bottleneck since it requires a labor-intensive labelling process involving people who have extensive domain knowledge. One of the approaches to overcome these hurdles is **distant supervision** [23]. With the help of a (domain-specific) dictionary, unlabelled text can thus be labelled automatically without the necessity to involve human annotators, either as an additional training resource [4], or independently from any manually annotated datasets [8]. The eventual insertion of additional noise into these training labels are addressed by recent approaches of denoising [29, 19, 26]

Similarly, existing (domain-specific) dictionaries can be used for entirely dictionary-based entity extraction. This, however, is a rather linear and early approach to biomedical NER and heavily relies on the completeness of the dictionary and does not achieve the same classification quality as newer non-dictionary or non rule-based methods [20]. Except for a combination with newer architectures or for post-processing purposes, **dictionary-based methods** have been gradually superseded by deep learning approaches until the present day [28].

Any NER approaches just relying on (Bi)LSTM network architectures share the shortcoming of a sequence-based view on the sentences, i.e., one individual label is (learnt and) predicted based on the current word and the left- (and right-)sided surroundings. One of the two major approaches to overcome this limitation is the application of **conditional random fields (CRF)**. Instead of calculating the conditional distribution of the current class given a vector of input features (for BiLSTM: text sequence before and after each word), the approach aims at modelling a joint conditional distribution of a sequence of classes/labels given a sequence of inputs. Thus, more of the context and semantic information can be taken into account. CRF-based algorithms are amongst the best-performing on the present dataset, typically combined with a (Bi)LSTM component [11, 37].

The second major approach to taking into account more of the context information, is the use of **BERT (Bidirectional Encoder Representations from Transformers)**. Introduced in 2019, BERT is a deep learning architecture specifically designed for language modelling purposes[6]. As such, it is using a neural network architecture similar to BiLSTM, but goes beyond taking a uni- or bi-directional sequence prediction perspective by using masked sentences during training. In more detail, a (random) subset of the words in a sentence is hidden from the model in the training phase to learn their context within a whole sentence. The second part of the original BERT architecture consists of a next sentence prediction (NSP) task by randomly learning the true next sentence and a random other sentence from the training set. That part of the architecture is more geared towards Question-Answering scenarios and typically omitted in purely named entity recognition settings using BERT. While typically being pre-trained on large general, i.e., not domain-specific text corpora, BERT shows some performance shortcomings when applied in the biomedical context. An alleviation to these are proposed by BioBERT, a BERT-based architecture pre-trained on the PubMed (and PubMedCentral) abstract (and article) corpora [16]. Whilst outperforming the non domain-specific BERT, the net increase in performance compared to BiLSTM-CRF architectures for NER tasks such as the one in this project is negligible [16].

Acknowledgments

This project was initiated by the organizers of the Deep Learning course at the Hasso Plattner Institute, led by Christoph Lippert, and supported by Aiham Taleb, Jana Fehr, and Benjamin Bergner. The authors want to appreciate the guidance and feedback throughout this project, provided by their supervisor Raunak Agarwal.

References

[1] Behnam Asadi and Hui Jiang. On approximation capabilities of relu activation and softmax output layer in neural networks, 2020.

[2] Balu Bhasuran and Jeyakumar Natarajan. Automatic extraction of gene-disease associations from literature using joint ensemble learning. *PLOS ONE*, 13(7):1–22, 07 2018.

[3] Hyejin Cho and Hyunju Lee. Biomedical named entity recognition using deep neural networks with contextual information. *BMC Bioinformatics*, 20, 12 2019.

[4] M Craven and J Kumlien. Constructing biological knowledge bases by extracting information from text sources. *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, 02 1999.

[5] Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. A neural network multi-task learning approach to biomedical named entity recognition. *BMC Bioinformatics*, 18, 12 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Dieter Galea, Ivan Laponogov, and Kirill Veselkov. Exploiting and assessing multi-source data for supervised biomedical named entity recognition. *Bioinformatics (Oxford, England)*, 34, 03 2018.

[8] Omid Ghiasvand and Rohit J. Kate. Learning for clinical named entity recognition without manual annotations. *Informatics in Medicine Unlocked*, 13:122 – 127, 2018.

[9] John Giorgi and Gary Bader. Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics (Oxford, England)*, 34, 06 2018.

[10] Mourad Gridach. Character-level neural network for biomedical named entity recognition. *Journal of Biomedical Informatics*, 70:85 – 91, 2017.

[11] Maryam Habibi, Leon Weber, Mariana Neves, David Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics (Oxford, England)*, 33:i37–i48, 07 2017.

[12] Lee-J. Hong, S.K. Dtranner: biomedical named entity recognition with deep learning-based label-label transition model. *BMC Bioinformatics*, 21(53), 2020.

[13] Ming-Siang Huang, Po-Ting Lai, Pei-Yen Lin, Yu-Ting You, Richard Tzong-Han Tsai, and Wen-Lian Hsu. Biomedical named entity recognition and linking datasets: survey and our recent development. *Briefings in bioinformatics*, 06 2020.

[14] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. pages 260–270, 03 2016.

[15] Yann Lecun, Leon Bottou, Genevieve Orr, and Klaus-Robert Müller. Efficient backprop. 08 2000.

[16] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics (Oxford, England)*, 36, 09 2019.

[17] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *CoRR*, abs/1812.09449, 2018.

[18] Sun-Y. Johnson R. J. Sciaky D. Wei C. Leaman R. . . . Lu Z. Li, J. Biocreative v cdr task corpus: A resource for chemical disease relation extraction. *Database(Oxford)*, 2016.

[19] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. Bond: Bert-assisted open-domain named entity recognition with distant supervision, 06 2020.

[20] Hongfang Liu, Zhang-Zhi Hu, Manabu Torii, and Cathy Wu. Quantitative assessment of dictionary-based protein named entity tagging. *Journal of the American Medical Informatics Association : JAMIA*, 13:497–507, 09 2006.

[21] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. pages 1064–1074, 03 2016.

[22] Tim Mackey, Vidya Purushothaman, Jiawei Li, Neal Shah, Matthew Nali, Cortni Bardier, Bryan Liang, Mingxiang Cai, and Raphael Cuomo. Machine learning to detect self-reporting of symptoms, testing access, and recovery associated with covid-19 on twitter: Retrospective big data infoveillance study. *JMIR Public Health Surveill*, 6(2):e19509, Jun 2020.

[23] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[24] Bharat A. Panuganti, Aria Jafari, Bridget MacDonald, and Adam S. DeConde. Predicting covid-19 incidence using anosmia and other covid-19 symptomatology: Preliminary analysis using google and twitter. *Otolaryngology–Head and Neck Surgery*, 0(0):0194599820932128, 0. PMID: 32484425.

[25] Farag Saad, Hidir Aras, and René Hackl-Sommer. *Improving Named Entity Recognition for Biomedical and Patent Data Using Bi-LSTM Deep Neural Network Models*, pages 25–36. 06 2020.

[26] Jingbo Shang, Liyuan Liu, Xiang Ren, Xiaotao Gu, Teng Ren, and Jiawei Han. Learning named entity tagger using domain-specific dictionary, 09 2018.

[27] S. Silvestri, F. Gargiulo, and M. Ciampi. Improving biomedical information extraction with word embeddings trained on closed-domain corpora. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1129–1134, 2019.

[28] Min Song, Hwanjo Yu, and Wook-Shin Han. Developing a hybrid dictionary-based bio-entity recognition technique. *BMC medical informatics and decision making*, 15:S9, 05 2015.

[29] Peng Su, Gang Li, Cathy Wu, and K. Vijay-Shanker. Using distant supervision to augment manually annotated data for relation extraction. *PLOS ONE*, 14:e0216913, 07 2019.

[30] Masaya Suzuki, Kanako Komiya, Minoru Sasaki, and Hiroyuki Shinnou. Fine-tuning for named entity recognition using part-of-speech tagging. In *PACLIC*, 2018.

[31] Buzhou Tang, Hongxin Cao, Xiaolong Wang, Qingcai Chen, and Hua Xu. Evaluating word representation features in biomedical named entity recognition tasks. *BioMed research international*, 2014, 2014.

[32] Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics (Oxford, England)*, 35, 01 2018.

[33] Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. Cross-type biomedical named entity recognition with deep multi-task learning, 2018.

[34] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, 87:12 – 20, 2018.

[35] Leon Weber, Jannes Münchmeyer, Tim Rocktäschel, Maryam Habibi, and Ulf Leser. HUNER: improving biomedical NER with pretraining. *Bioinformatics*, 36(1):295–302, 06 2019.

[36] Zhang Yijia, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific Data*, 6, 12 2019.

[37] Lou Yinxia, Yue Zhang, Tao Qian, Fei Li, Shufeng Xiong, and Donghong Ji. A transition-based joint model for disease named entity recognition and normalization. *Bioinformatics (Oxford, England)*, 33, 03 2017.

[38] Guodong Zhou, Jie Zhang, Su Jian, Dan Shen, and Chew Lim Tan. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics (Oxford, England)*, 20:1178–90, 06 2004.

[39] Huiwei Zhou, Chengkun Lang, Zhuang Liu, Shixian Ning, Yingyu Lin, and Lei Du. Knowledge-guided convolutional networks for chemical-disease relation extraction. *BMC bioinformatics*, 20(1):260, 2019.