Contextualized Medication Event Extraction

Ryan Friberg
The University of Chicago, Department of Computer Science

Abstract

This paper gives an overview of a submission to the Harvard Medicine National NLP Clinical Challenge. In my implementation of this task, I experimented with various versions of BERT including BERT-base, DistilBERT, and a specialized version of BERT trained on clinical notes, ClinicalBERT. Additionally, I developed a pipeline to pre-process the raw medical notes to feed to my model and employed a handful of techniques to boost performance including error analysis through confusion matrices as well as a weighted loss function to handle the extremely unbalanced distribution of the raw data. I document in detail my motivation for my approach to this challenge, the techniques I used at each step and my current results on each of the competition's sub-tasks.

1 Introduction

This competition has the task of taking raw physician clinical notes and providing the appropriate labels regarding the medications used, the past and future changes in said medications, and the context in which those changes were discussed. Each team is given the same set of training data and needs to create a machine learning model to reproduce the same specific style of annotations. As such, the overall challenge can be divided into many different smaller machine learning recognition and classification tasks. Later sections will provide a detailed overview of the each contextualized medical event extraction tasks.

2 Motivation, Related Work, & Points of Comparison

Mahajan et al. laid the groundwork for contextualized medical event extraction in (Diwakar Mahajan, 2021) in an effort to extract more comprehensive contextual information about medication changes discussed in clinical notes. Healthcare providers currently rely on structured medication orders in order to query patients' medication history, but these official orders do not capture much of the important information regarding medication change events that define the complete timeline and context of a patient's medication history. Often, richer documentation of medication events can be found in the clinical notes kept by healthcare providers. Hence, Mahajan et al. investigate the application of language models to this problem in order to extract such information and achieve a better characterization of medication events. Mahajan et al. achieve this by formulating a notion of multi-dimensional context classification in order to address the limited scope previous works have employed on similar tasks.

As paper written by Mahajan et al. was task-defining and addressed many subtasks of contextual medical event classification, I chose to model my submission to the clinical challenge based on their approach. However, since the paper does not address the application of named entity recognition (NER) to identify medication mentions in clinical notes, I simply decided to formulate NER as standard classification task using BERT and a method from (Erik F. Tjong Kim Sang, 2000) which I discuss later on in this paper in section 4.3. One other major difference between my models (in addition to my data pre-processing and the other techniques discussed in section 6), was that for the sequence-based classification tasks (event and context), I still used token classification as described in section 4.5. Mahajan et al. used sequence classification in these sub-tasks.

One of the most major benefits of (Diwakar Mahajan, 2021) was it providing me with a point of comparison in terms of model accuracy on the specific sub-tasks of the challenge. When they uitilized the ClinicalBert Model with their private configuration, they reported the following results based on the unreleased *test* data set:

CMED Event Classification

Micro Precision	0.88
Macro Precision	0.79
Micro Recall	0.88
Macro Recall	0.79
Micro F1	0.88
Macro F1	0.79

Table 1: Mahajan et. al Event classification results

CMED Action Classification

Micro Precision	0.75
Macro Precision	0.75
Micro Recall	0.75
Macro Recall	0.63
Micro F1	0.75
Macro F1	0.65

Table 2: Mahajan et. al Action classification results

CMED Temporality Classification

Micro Precision	0.83
Macro Precision	0.80
Micro Recall	0.83
Macro Recall	0.74
Micro F1	0.83
Macro F1	0.75

Table 3: Mahajan et. al Temporality classification results

CMED Certainty Classification

Micro Precision	0.90
Macro Precision	0.83
Micro Recall	0.90
Macro Recall	0.76
Micro F1	0.90
Macro F1	0.79

Table 4: Mahajan et. al Certainty classification results

CMED Actor Classification

Micro Precision	0.93
Macro Precision	0.83
Micro Recall	0.93
Macro Recall	0.72
Micro F1	0.93
Macro F1	0.76

Table 5: Mahajan et. al Actor classification results

Throughout my work, I have used these results as a grounding point of comparison to determine how well my model is doing.

3 Competition Details & Provided Data

The competition provides datasets for training and development that are composed of pairs of files that contain the raw physician note text and its accompanying annotations. The physicians' notes have no pre-processing and thus span a fairly vast range of styles and formats. Some are purely in paragraph form whereas others use bullets, sentence fragments, or longer stream-of-consciousness type writing and others are even inconsistent within the same note. The accompanying annotations, however, do follow the same general format which is derived from the methodology described in (Diwakar Mahajan, 2021). As the clinical notes themselves differ quite substantially, some notes have many more associated annotations than others.

In the annotations, there are tags for the medications present, the disposition event status and relevant tags for the five dimensions of clinical context, if applicable. The medication labels give both the name of each medication and their respective locations within their note. Each medication listed will have an associated disposition event label which states if a change in treatment for that medication was discussed, was not discussed, or is unknown. Lastly, the five context dimensions for each event span the action of the change (increase dosage, decrease dosage, etc.), negation, which logically labels if the change in medication is negated, temporality, which says when the change occurred or is intended to occur, the certainty of the change happening (although they misspelled the word certainty in every instance of the clinical annotations) and actor which describes who initiated the change.

These sets of pairs are split between the training set and the development set, 350 pairs for training and 50 for development respectively. Later on in the timeline of the competition, there will be three sets of test data that will be released. Each test set evaluates slightly different aspects of the model. The first will evaluate named entity recognition, named entity + event, and end-to-end tasks on unannotated notes. The second will only examine event + context tasks on gold standard medications. The last will evaluate only the context task on gold standard disposition medication events. Table 6 provides a high-level summary of the text data in the clinical notes.

	Train	Dev
Note Count	350	50
Token Count	299,193	47,345
Type Count	18,009	7,083
Tokens/Note	854.4	946.9
Sentence/Note	43.28	52.32
Med Count	350	50
Med Change Count	299,193	47,345
Med Tokens/Note	18,009	7,083
Med Changes/Note	854.4	946.9

Table 6: General training & dev data statistics

4 Data Pre-processing

4.1 Overview

One of the most crucial steps in the pipeline of this project is the methods used to pre-process the raw clinical note and annotation data. My methods for data pre-processing evolved over time to improve my accuracy. This section outlines my current methodology for data pre-processing which has thus far given me the highest accuracy. The primary need for pre-processing was to convert the text data from clinical notes and annotations into a dataset that can serve as input to my models. For each subtask, I converted the text data and annotations into json files that could be loaded into a HuggingFace datasets object, which can be used natively with HuggingFace's transformer models. For each subtask, I used the spaCy library to tokenize the data. Prior to being input to the model, I also applied a BERT pre-trained tokenizer to the input text.

4.2 Annotation Files

To extract the annotations, I first collected the ".ann" files and converted their contents to a json format. For each medication mentioned, I stored all of the corresponding information relating to that instance of the medication as key value pairs. This included the medication id ("T" label), medication name, the event id ("E" label), the event ("NoDisposition", "Disposition", or "Undetermined"), and any of the applicable context classifications along with their respective value (action, actor, certainty, and/or temporality). In order to stay consistent with (Diwakar Mahajan, 2021), I decided to exclude the event context of Negation as well as the Unknown values for each of the event context values (in favor of simply omitting the contexts that were not present).

Previously, I had allowed my model to classify based on the medication name which allowed for a point of failure when the same medication was named more than once within the same note but with different contexts. To fix this, I added one more element to each dictionary that would be unique to each instance of the medication. The annotation files provided the character span of a medication mention (i.e. the file gives numbers X and Y where, in the note, the instance of the medication name begins at character X and ends at character Y) so using spaCy, I converted this to a token span and stored it with each medication mention for easier classification later on as now all classifications would be made using the token span. I was then able to use this more structured form of the annotations to generate the true labels for each sub-task.

4.3 Sub-task Labeling

For NER, in order to perform token classification to identify medication mentions, I needed to assign labels based on whether or not any particular token was a medication name. While this approach was straightforward, I still ran into corner case situations. One of the most notable was in the case of multi-token medication names such as "beta blockers." In such cases, each token individually would technically not be a medication but together they would be. To ensure I handle these cases, I followed the **BIO** labelling method as defined by (Erik F. Tjong Kim Sang, 2000). In this case, I assign labels based on if a token is the **B**eginning of the medication name, **I**nside a medication name, or **O**utside the medication. Specifically, I decided to give a label of 0 to all non-medication tokens, a label of 1 for the first token in a medication, and a label of 2 for all remaining tokens in the medication name. The idea behind this is to help the model better learn the context in which the words became medications by giving it differing labels depending on the position of each token in medication names.

For the rest of the subtasks, I generated the task-specific label sets based on the number of possible values for each sub-task. For example, the event classification can have three possible values, Disposition, NoDisposition, or Undetermined, so I assign labels 0, 1, and 2 respectively. This approach thus created different label sets for each task. I then labeled only the medication names with the correct corresponding label. In my implementation, I give a label of -100 to any non-medication name token so that the model will ignore the tokens irrelevant to the task. This method was used for the event classification as well as all of the context classification tasks.

4.4 Data Chunks

I developed two main ways to go back through the data to break the notes down into chunks of tokens to be fed to the model for all of the sub-tasks.

In both methods, I defined a data point to hold all the chunks of information as a list of dictionaries where each dictionary i in the list contains both the tokens and the labels of the ith chunk. The first "chunking" procedure was simply to define the chunks of a note to be its sentences where I used the spaCy sentence tokenizer to extract the start and end of each sentence in the notes. The second was slightly more complicated and it operated on a basis of a fixed maximum number of tokens (I used 200). The reasoning behind this was due to the fact that the sentences of the notes varied greatly in length and because all the information about the same instance of a medication might not always be confined within the same sentence. Giving the model larger chunks would first provide the model more uniformity (which would particularly help with small sentences), but also provide the model with context across sentences. To accomplish this, I first tokenized the note by sentence again and composed chunks of full sequential, contiguous sentences that had a collective number of tokens under the limit. I did not split sentences across data chunks.

4.5 Token vs. Sequence Classification

In contrast to the model reported in (Diwakar Mahajan, 2021), I treated the sequence classification tasks also as token classifications. To do this, I assign the labels of the task to the first token in the medication mention, and all other tokens are given the label -100 so they are ignored. The first benefit of this implementation is that the position of the medication instance is taken into account since the label is associated with the token instead of the whole input sequence. On top of this, I no longer need a one-to-one mapping between inputs and outputs as I can make predictions for multiple medications in the same input sequence without needing to split or duplicate the sequence to avoid certain classification pitfalls that would hurt performance. This also allows me to use longer input sequences to capture more context. The results shown in section 7 are from using this form of classification.

5 Models

My models were implemented by fine tuning separate BERT-based language models for each subtask. I formulated each of the subtasks as follows:

1) Named entity recognition to identify medication mentions: I formulated this subtask as a token classification task. As input, the model takes in a list of tokens (one of my data chunks) and it performs classification on each token predicting one of three labels: whether it is not a medication (NoMedication), it is the first token of a medication (MedicationStart), or it is a token that is in a medication name but it is not the first (MedicationContinue).

2) Medication change and context classification: Each of the remaining subtasks was also formulated as a token classification problem. As input, the model for each subtask takes in a list of tokens (again, one of my data chunks) and it labels the input according to its appropriate classification task.

I experimented with BERT-based language models pre-trained on both general-domain datasets (BERT-base and DistilBERT) and in-domain datasets (ClinicalBERT), all of which yielded similar results on each subtask. For each model, I used the following configuration for training:

Training Configuration	Value
Learning Rate	1×10^{-5}
Batch size	8
Weight decay	0.01
Dropout	0.2
Gradient accumulation steps	2
Epochs	30-50
Optimizer	AdamW
Weight Decay Rate	0.01
Learning Rate Warm-Up	20%
Loss Function	Cross-Entropy

6 Techniques to Boost Performance

In addition to trial and error with various configurations of hyper-parameters, I made a handful of distinguished efforts to improve my performance.

6.1 Weighted Loss

Throughout this project I realized that a severe bottleneck to performance was the fact that the distribution of medication name tokens to non-medication tokens was vastly skewed with only around 5% of the tokens being medications. This problem was similarly represented to varying degrees in the different sub-tasks as some labels just did not occur frequently in an already-small data set.

To adjust for this, I weighted my loss function based on an inverse frequency rule. I defined my weighting scheme to be the compliment of the proportion of any particular class's frequency in the training data. Specifically, if the proportion of a certain class in training data was p, that class was weighted with $1 - p$ in the loss function. This helped improve the performance on the rare labels and mitigated some of effects of the bottleneck caused by the unbalanced distribution.

6.2 Confusion Matrices

In order to perform error analysis on my model, I employed the use of confusion matrices. For each sub-task, these graphs allowed me to see where the model's prediction most often differed from (or agreed with) the true label for each of the possible labels for that task. This helps visualize accuracy and identify the common failure cases of the model. Additionally, these matrices give the distribution of the labels on the dev data set to help demonstrate where bottlenecks may be occurring. With the common failure cases, I am able to potentially infer some information about what is going on within the model. For example, in the Temporality confusion matrix, the model seems to prefer the present class over the past class. My reasoning is that the loss function may have weighted the present class more than the past class, potentially due to overfitting on the training set. Another example is in the Action sub-task, the model never predicted a decrease label. A possible reason for this due to the fact that the decrease label appeared in the data at such a low frequency that the model never learned its significance. Alternatively, it is possible that the increasing and decreasing events had similar contexts (aside from a small number of words) and it was just hard for the model to distinguish between them.

Being able to perform error analysis at all in this competition proved to be difficult in its own due to the lack of a feasible ability to print out individual errors and directly see which sentences and words are the most confusing to the model. The analysis of confusion matrices has at least partially filled this need and helped me plan out future moves and what to focus on in order to improve the model.

All of my current confusion matrices for each task are included in the appendix of this paper.

6.3 Learning Scheduler

As part of my hyperparameter tuning, I experimented with learning rate scheduling in order to stabilize the learning over many epochs. The general trend that I observed was that learning was fairly steady overall but certain tasks had more turbulence than others. The learning rate scheduler helped the model's learning stay more consistent for longer by managing the learning rate during training and decreasing it over time. Specifically, I settled on employing a linear warm-up for the first 20% of the training time then switching to a constant value for the rest of training.

7 Results

This section gives the overview of the current performance of my model on each sub-task. Additionally, the distribution of the of training labels are also given for each sub-task. Note that the varying in epochs was dependent on how consistent the learning was and how long it took to train the model.

NER Training Label Distribution

NoMedication	266885
MedicationStart	6079
MedicationContinue	1008

NER Results

Training Loss	0.069100
Validation Loss	0.076346
Accuracy	0.991858
Micro Precision	0.991858
Macro Precision	0.936845
Micro Recall	0.991858
Macro Recall	0.972242
Micro F1	0.991858
Macro F1	0.953831

Table 7: NER classification performance after 7 epochs

Event Training Label Distribution

NoDisposition	4509
Disposition	1118
Undetermined	455

Event Classification

Training Loss	0.577700
Validation Loss	0.430546
Accuracy	0.858291
Micro Precision	0.858291
Macro Precision	0.786135
Micro Recall	0.858291
Macro Recall	0.802307
Micro F1	0.858291
Macro F1	0.786334

Table 8: Event classification performance after 30 epochs

Action Training Label Distribution

Start	436
Stop	270
Increase	97
Decrease	31
UniqueDose	258

Action Classification

Training Loss	0.747500
Validation Loss	0.891859
Accuracy	0.734375
Micro Precision	0.734375
Macro Precision	0.563952
Micro Recall	0.734375
Macro Recall	0.613500
Micro F1	0.734375
Macro F1	0.583096

Table 9: Action classification performance after 50 epochs

Actor Training Label Distribution

Physician	1017
Patient	84

Actor Classification

Training Loss	0.255300
Validation Loss	1.547757
Accuracy	0.900000
Micro Precision	0.900000
Macro Precision	0.661234
Micro Recall	0.900000
Macro Recall	0.633261
Micro F1	0.900000
Macro F1	0.645348

Table 10: Actor classification performance after 35 epochs

Certainty Training Label Distribution

Certain	932
Hypothetical	103
Conditional	81

Certainty Classification

Training Loss	0.251100
Validation Loss	1.213831
Accuracy	0.888889
Micro Precision	0.888889
Macro Precision	0.748475
Micro Recall	0.888889
Macro Recall	0.701092
Micro F1	0.888889
Macro F1	0.715140

Table 11: Certainty classification performance after 50 epochs

Temporality Training Label Distribution

Past	570
Present	417
Future	107

Temporality Classification

Training Loss	0.220100
Validation Loss	1.109499
Accuracy	0.779487
Micro Precision	0.779487
Macro Precision	0.655910
Micro Recall	0.779487
Macro Recall	0.637939
Micro F1	0.779487
Macro F1	0.623974

Table 12: Temporality classification performance after 50 epochs

7.1 Results Discussion

I can see from my results that the distribution of labels varies greatly depending on the sub-task. Many of the contexts appear with very low frequency and thus it makes sense where the model has difficulty. Again, comparing to (Diwakar Mahajan, 2021), I can see that my results are beginning to creep up on their accuracy. The following section details the methods I am going to pursue to hopefully surpass their performance.

8 Future Work

8.1 Data Augmentation

One of the biggest problems in this competition is that I simply do not have access to much data to train my model on. In computer vision, it is common and extremely beneficial to perform data augmentation on the training data to essentially artificially inject more training data without biasing the model. For vision, this usually takes the form of applying filters to images (such as grayscale), flipping images on one or more axis, cropping, or any combination of these as none of these alterations actually change what is in the image. With my remaining time, I want to explore analogous techniques in natural language processing to apply to the clinical note data. However for NLP, the task is non-trivial as naive approaches such as adding in random words or cropping sentences randomly can completely change the meaning of the sentence an thus affect the model. Future work could aim to figure out how to potentially artificially boost my quantity of training data.

8.2 Chunk Positioning

In my current method of breaking down the raw clinical notes into chunks to feed to the model, I simply go by sequential sentences until I hit my defined maximum. I also experimented with splitting my data on the medication mentions to only have one medication mentioned per sequence of tokens passed to the model for event and context classification. I believe that having a more sophisticated splitting method could perhaps improve performance. As such, I want to see if I can position my chunks to provide more context per medication instance. For example, I want to document what the effect of centering a chunk around a medication would be. My hypothesis is that the context of any instance of one medication might not always be limited to strictly before or after the medication but instead could happen in both. By centering the chunk on the medication name, the model might be able to have more relevant sentence context to hopefully improve performance.

8.3 Ensembling

I am currently using a single model to get my predictions, but I think it would be interesting to investigate how the use of an ensemble of the three BERT models might improve my accuracy. This technique would have me train all three versions of BERT and use some method of combining their collective results. Out of the options of "bagging," "stacking," and "boosting," I think that boosting would be the best for my case. This would involve adding the models' behavior in such a way that the predictions are weighted together to potentially give a overall better result than any single model could have.

8.4 Multi-label Classification

Another technique that I am actively looking into is formulating the context sub-tasks to all be trained with multi-label classification. In this, I would have the same model trained on the all of the sub-tasks and thus predict all of the sub-tasks' labels. My intuition is that this would allow the model to learn more about the data through the various contexts. This is a technique used in (Diwakar Mahajan, 2021) so it may help me bridge the remaining gap between my performance and theirs.

8.5 Further Hyperparameter Tuning

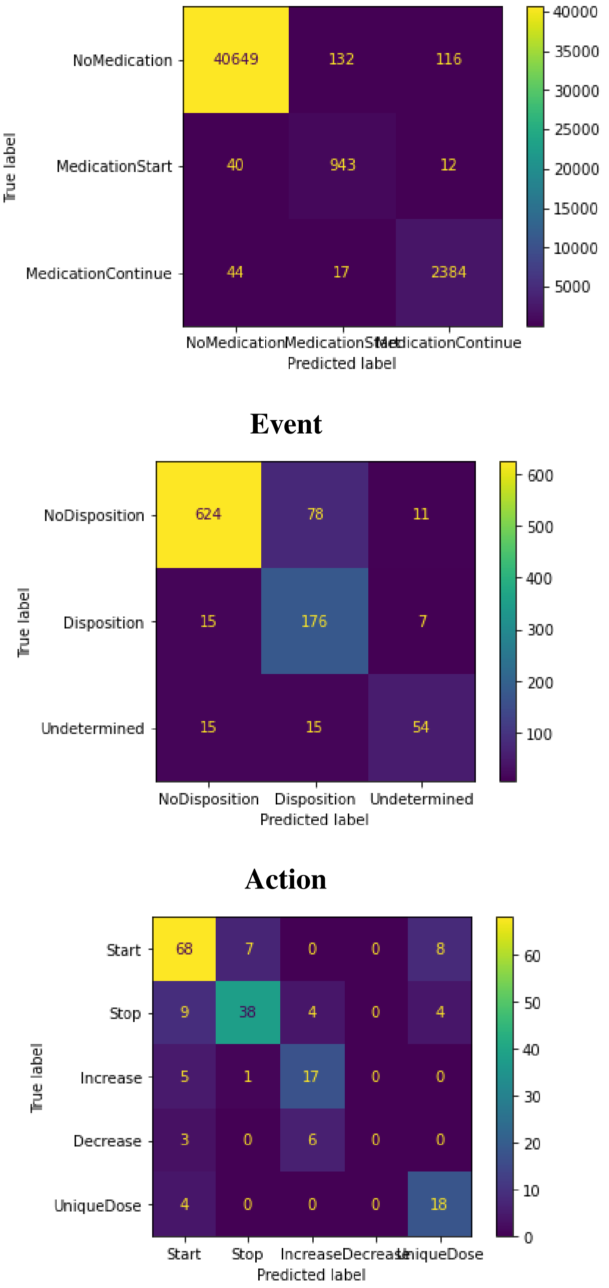
Throughout this entire project I have experimented with the hyper-parameters to run this model on. In particular, I have noticed that certain tasks' training are steady even for a high number of epochs. Because of this, one future experiment could be training one (or all) tasks on an extremely high number ($\approx$ 100) of epochs to see how it impacts my results. However, in doing so I am aware that this could increase the risk of overfitting.

Additionally, I want to experiment with how both the dropout rate and the learning rate/cosine scheduler can be optimized. In both of these, I have observed substantial shifts in the behavior and accuracy of the model based on small differences in parameter value so I want to see if I can zero in on optimal values. Lastly, I wanted to investigate the benefit of applying a hyper-parameter search algorithm.

9 Conclusion & Discussion

So far, I have been able to achieve a substantial increases in my accuracy over my initial baseline results. In some instances, I was able to see improvements of upwards of $10-30\%$. I suspect that much of this performance gain was a consequence of my overhauled data pre-processing, the learning rate scheduling, and tuning of the dropout rate. However, I am just below the accuracies reported in (Diwakar Mahajan, 2021). Almost all of the details surrounding the model they used were kept private, but they implied that it was rather simple and did not account for many corner cases. It is unclear to me how they achieved such high accuracy but some possible options include them training their model for a much higher number of epochs as well as the difference between the training data and the test data. Their reported accuracies are on the unreleased test data which has twice as many notes as the dev set, which is what my accuracies are reported on. It is pure speculation but, given that the dev set is half the size, it could be possible that certain cases that are confusing to the model are over-represented.

Confusion Matrices



Named Entity Recognition

Event

Action

References

Xudong Jia Chaojie Wen, Tao Chen and Jiang Zhu. 2021. Medical named entity recognition from un-labelled medical records based on pre-trained language models and domain dictionary. *Data Intelligence*, 3:402–417.

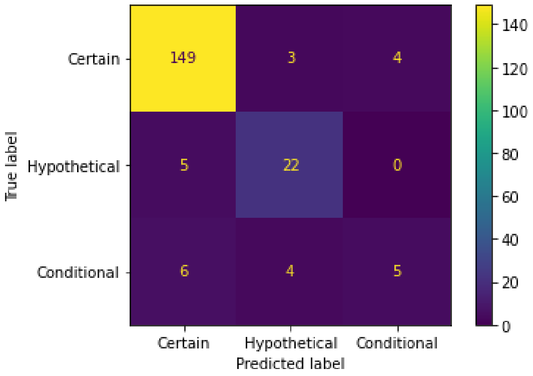
Jennifer J. Liang Diwakar Mahajan. 2021. Toward understanding clinical context of medication change events in clinical narratives. *Private Preprint*.

Sabine Buchholz Erik F. Tjong Kim Sang. 2000. *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

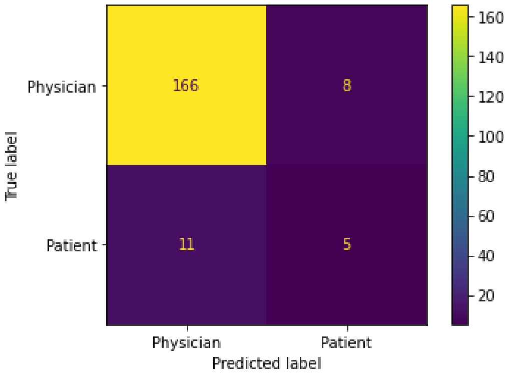
Edson Florez, Frederic Precioso, Michel Riveill, and Romaric Pighetti. 2018. Named entity recognition using neural networks for clinical notes. *PMLR*.

A Appendix

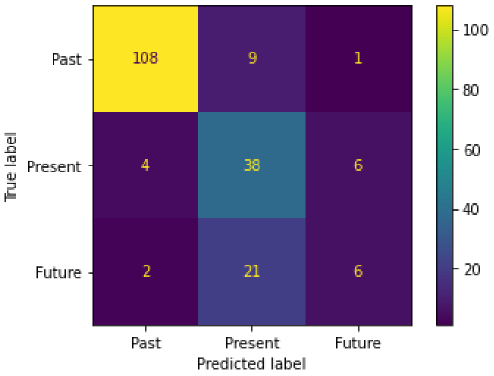
Certainty



Actor



Temporality


True label
Past
Present
Future
Past
Present
Future
Predicted label