Bachelor thesis
Computing Science

Radboud University

Evaluation and analysis of a transformer model for detailed PIO element extraction on medical literature

Author:
Luke van Leijenhorst
s1045958

First supervisor/assessor:
Prof. M.A. Martha Larson
m.larson@cs.ru.nl

Second assessor:
Prof. dr. ir. Arjen de Vries
a.devries@cs.ru.nl

January 22, 2022

Acknowledgements

I would like to thank my primary supervisor Martha Larson for the guidance and the many useful suggestions and comments made through the entirety of this thesis.

I also wish to show my appreciation to Klaus-Michael Lux and Dr. Jan Schouten of the CWZ hospital in Nijmegen whose insights and knowledge on the subject steered much of the research.

Abstract

In Evidence-Based Medicine (EBM), medical practitioners make use of the best available evidence to make decisions about the care of each individual patient. A large amount of this evidence is hidden in Randomized Controlled Trials (RCTs). RCTs are considered to provide the purest amount of evidence for a potential causal connection between a medical intervention and a measured outcome. This evidence is hidden in the papers describing these RCTs.

The PICO (Population, Intervention, Comparator and Outcome) elements give a lot of information about the trial and can be used in many subsequent automation tasks. Automatic PICO element extraction using NLP techniques can aid EBM practitioners substantially.

In this thesis, we will look at the extraction of the PIO elements subdivided into 16 more granular entities. Here the I and C class are both encapsulated in the I class. We show the potential of using transformer models for this task of detailed PIO extraction by showing that it outperforms the current best-performing model by a large margin. Afterwards, we show that the model still seems to benefit from more data by showing that data saturation has not been achieved yet. Finally, we perform a detailed error analysis and address systematic errors and common mistakes, which should guide how new abstracts should be selected and annotated.

Contents

Chapter 1

Introduction

Knowing whether a certain medical intervention has a desired outcome is crucial for determining the right intervention for a patient. Using the best evidence available to make the decision about the treatment of patients is at the core of Evidence-Based Medicine (EBM).

A SR (Systematic Review) of RCTs (Randomized Controlled Trials) is considered to provide the largest amount of evidence for medical decision making. This evidence is hidden in the lengthy papers describing the RCTs. These papers are time-consuming to analyse and also costly since they have to be analysed by medical professionals due to the technicality and complexity of the texts. On top of that, the number of RCT studies and SRs published in the top 30 contributing countries has been increasing between 1995 and 2015 [1]. By (partially) automating this process, time will be saved and it will allow for faster incorporation of new knowledge into the knowledge base for EBM.

The identification of the elements of the PICO (Population, Intervention, Comparator and Outcome) framework in abstracts of RCT studies could play an important role in identifying the important elements of a study, assisting in selection procedures of studies for a SR and it could play an important role in large scale automation systems.

In this research we will focus on the extraction of the PIO elements subdivided into a total of 16 sub-categories for more detailed identification. The Comparator (C) and the Intervention (I) elements are both encapsulated into the I class. A short sample span annotated with some of these detailed PIO elements can be found in figure 1.1.

We have compared the efficacy of aspirin I: Drug in comparison to a placebo I: Control for the prevention of headaches O: Physical in children P: Age aged P: Age 8-16 P: Age years P: Age .

Figure 1.1: Sample span of text annotated with detailed labels.

The research question that we will be answering in this thesis is:

How can spans of text containing detailed PIO elements be extracted from RCT studies to speed up systematic reviews and facilitate Evidence-Based Medicine (EBM)?

In the process of answering this question we will answer the following sub-questions:

- How well do transformer-based models perform on entity extraction of detailed PIO values on RCT studies in comparison to earlier attempts?
- To what extent does the accuracy measurement of the model reflect the usefulness of the output to medical professionals?
- How to sample additional RCT abstracts for annotation to improve the classifier?
- In what way can data be annotated more effectively to improve the classifier?

To answer these questions, we will first look at existing work on PIO extraction and other relevant literature. Afterwards, we will train a transformer model for this entity extraction task. We will report on the results of this model and compare it against earlier attempts to see the potential a transformer model has on this task.

With the resulting model, we will then perform an analysis. In this analysis we will first explore whether data saturation has been achieved. We will train the model on training sets of different sizes and inspect whether the model still seems to benefit from additional data or if a plateau has been reached on the performance of the model and thus no benefit is gained from adding additional data.

Secondly, we will perform an error analysis to look for ways in which new data should be selected and annotated. For the error analysis we will create our own evaluation method. We do this because evaluating against what is considered to be the ground truth for the task, does not fully represent the usefulness of the output to medical professionals. In this error analysis we will look for systematic errors and their possible cause, what types of mistakes the model makes and the severity of those types of errors to medical professionals.

Chapter 2

Literature review

In this literature review we will first give an overview of larger scale systems with the goal of (partially) automating systematic reviews and what role PICO extraction plays here. Then we will look at previous work on the task of PICO extraction and the EBM-NLP dataset that we will be building on in this research. Lastly, we will look at models that were previously used for this task (Linear Regression, Conditional Random Fields and Long short-term memory networks). Finally, we will look at transformer models to better understand the benefits a transformer model could have over the previously trained models on this task.

(Semi-)Automated SR

For the task of (partially) automating SR we will review two different systems: Evidence Inference and RobotReviewer.

Evidence Inference

In two papers by DeYoung et al [2] [3], a task called *Evidence Inference* was introduced. This task consists of two parts:

1. Identifying spans of text containing PICO elements in a RCT study
2. Determining the effect of the **I**ntervention to the **O**utcome for a certain **P**opulation in comparison to the **C**omparator.

The papers showed the complexity of the task with f1-scores of around 0.52 on a BERT pipeline for the full task. The f1-scores for the same model with the correct PICO spans already marked, is around 0.77. This significant difference in scores shows the importance of correct PICO element extraction out of RCT studies in the performance of larger automation systems.

RobotReviewer

The RobotReviewer system [4] takes the automation one step further and has additional components that also perform a risk of bias review to analyse the quality of the study and determine whether the given text is actually a RCT study. The system also extracts other relevant information like sample sizes and key findings of the study in addition to the standard PICO elements.

Trialstreamer

A recent addition to the RobotReviewer system was the Trialstreamer database [5] which automatically updates itself by adding new RCT studies from PubMed and the International Clinical Trials Registry Platform of the World Health Organization to the database. The RobotReviewer system is then run on this database to extract all the information. Such large scale automation systems could be very beneficial for EBM but the output of the individual components are often too inaccurate to fully automate this process. In this research we will look at the PICO extraction component and look for improvements.

PI(C)O extraction

In the domain of PICO element extraction there are two different approaches to classify the entities in a text: Sentence classification and Named Entity Recognition (NER) of the PICO entities. Often the I and C class are combined into a single I class. We will refer to this as PIO extraction. We first look at current work in the area of sentence classification of the PI(C)O elements.

PICO sentence classification

Sentence classification of PI(C)O elements has seen some promising results. A LSTM model trained on abstracts achieved f1-scores of around 0.82 for the PIO elements [6]. Another study [7] applied a transformer-based model to the same problem and achieved f1-scores of around 0.88. The limitations with sentence classification however, is that sentence level classification does not provide information that is detailed enough for complex QA systems or summarization tasks. On top of that, subsequent steps in SR automation systems could benefit greatly from more detailed annotated PI(C)O elements.

PICO-NER

We now look at PICO extraction as a NER task where each token in a text gets a PICO entity or a *None* entity assigned to it. At first, due to a lack of

labeled data, only smaller datasets existed or larger datasets were derived through distant supervision which resulted in noisy labels [8]. To overcome this problem the EBM-NLP dataset was introduced.

EBM-NLP

The EBM-NLP dataset was introduced in an accompanying paper to provide a large labeled dataset for the task [9]. The dataset consists of 5000 abstracts from PubMed which were primarily sampled on three subjects: Cancer, autism and cardiovascular diseases. These subjects were chosen to capture a high amount of common conditions. The sampled abstracts were annotated with spans of text containing PIO elements. 4800 of those annotations were annotated by crowd workers and 200 were annotated by medical professionals. After identifying the PIO elements, the annotators were also asked to sub-divide the PIO elements in more fine-grained entities. After filtering out abstracts that described non-RCT studies or were missing data, 4641 abstracts remained. The full list of the fine-grained categories and the counts for each category in the used development set (10% of the crowd workers' annotations) can be found in figure 2.1. Along with the dataset, the authors also provided two baseline models for the task: A Logistic regression (LR) model and a Conditonal Random Field (CRF). The authors also created a leaderboard (`https://ebm-nlp.herokuapp.com/`) which includes additional models for the task. At the time of writing, those models are another LR model and a Bidirectional LSTM-CRF network.

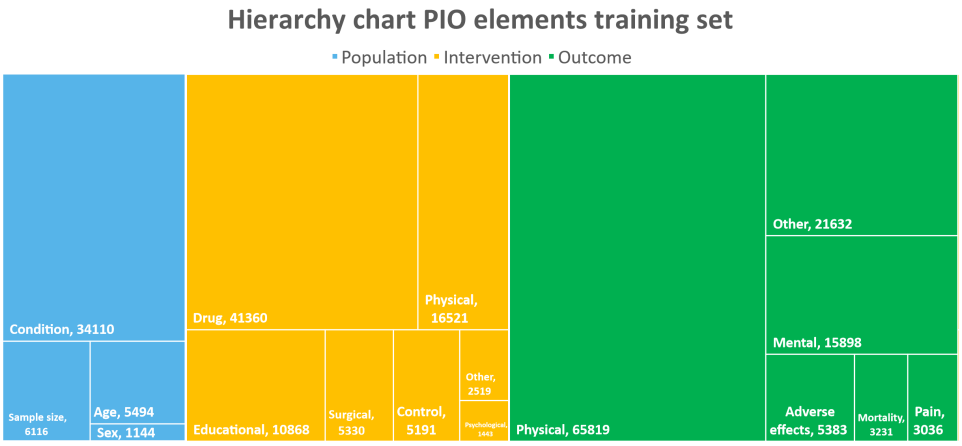


Figure 2.1: Hierarchy plot of the PIO sub-categories together with their counts in the development set.

Models

Now we will look at the existing models trained on the task and see what advantages a transformer model could give over these models.

Logistic regression

Logistic regression [10] models predict the probability of an outcome based on the feature vector that is given to the model. It does this by multiplying the feature vector by a weight vector and adding a bias term. If the probability for a class is over a certain threshold, it predicts this class. The weight vector and bias term are trained through an optimization algorithm like gradient descent. For handling text (NLP), obtaining feature vectors is often not trivial and you have to manually generate input values from the text. Features often include POS (part of speech) tags, n-grams, character information (e.g. number of special characters or capital letters) and more.

Conditional Random Fields

In a 2001 paper by Lafferty et al. [11] Conditional Random Fields (CRFs) were introduced. A CRF simply applies logistic regression to sequential data and makes use of the fact that the predictions of the model depend on each other. In NLP, this is useful because context carries a lot of information about the meaning of a word or sentence.

LSTM

LSTM [12] is a type of Recurrent Neural Network (RNN) architecture. Just like CRFs, RNNs make use of the dependencies between the models' output. A big difference however, between RNNs and CRFs is that RNNs can find non-linear relations and scale much better to larger contexts. As a result RNN networks often perform much better than CRFs.

One of the limitations of standard RNN architectures is that long term dependencies often get lost. LSTMs solve this by having input gates, output gates and forget gates. These gates allow the cells in the network to keep or drop information from its context. As a result, LSTM can capture both long term and short term dependencies and outperforms the standard RNN architectures.

Bidirectional LSTM-CRF

A bidirectional LSTM runs the input both from beginning to end and end to beginning. Because of this, it can capture both dependencies to the left in the sequence of input but also to the right.

A Bidirectional LSTM-CRF [13] is simply a Bidirectional LSTM with a CRF layer on top. This CRF layer makes use of additional features like POS tags and token-level information (e.g. special characters, capitalization). For POS tagging and NER tasks, this model seems to outperform the regular Bidirectional LSTM and LSTM-CRF models by a small margin. However, this model does require you to manually generate input features for the CRF layer.

Transformer models

LSTM networks and its variants still come with some limitations. Some of these limitations are solved by transformer models [14].

The first limitation is that in order to capture the dependencies, input sequences have to be processed input by input. As a result, training can not be done in parallel which increases training time vastly. Transformer models pass all arguments of the sequence at once so computations can be done in parallel.

Another limitation in the LSTM network, is that in order to get information about dependencies that occurred much earlier or much later in the sequence, many computations have to be made. During these computations, information loss can occur. In transformer models you have something called a self attention layer which has direct access to the dependencies. By solving these limitations, transformers have been shown to outperform LSTM networks in many tasks.

For the task of detailed PIO extraction, no transformer had yet been trained. We believe models for this task can greatly benefit from the improvements a transformer gives in comparison to the Bidirectional LSTM-CRF model which is the current best-performing model for this task. That is why we will now look at an implementation of a transformer model and report on the results.

Chapter 3

Model

For the task of detailed PIO extraction we use spaCy 3.1's en_core_web_lg pipeline based on RoBERTa with a built-in NER component and customize this NER component by changing the entities to match the detailed PIO entities and then fine-tuning the existing NER component on the EBM-NLP dataset.

The EBM-NLP dataset contains a total of 4641 abstracts with detailed annotations for all the sub-categories (figure 2.1). An example of such an annotated abstract can be found in Appendix A. 4457 of the abstracts were annotated by selected crowd-workers and 184 abstracts were annotated by medical professionals. Due to the complexity of the annotation task, the annotations of the medical professionals are used as a test set for more reliable results. This is also in line with the way the models to which we compare the transformer-based model were evaluated. As a validation set, the first 10% of the training set was set aside.

Results

The accumulated results of the model for the PIO classes can be found in table 3.1. We compare our results to the baseline models provided in the original paper and the best-performing models on the leaderboard (`https://ebm-nlp.herokuapp.com/`) for this task. We also provide the results for a naive baseline model where we simply always guess the most frequently assigned entity from the training set (*O: Physical*). The individual scores for each different entity on the transformer model can be found in table 3.2. The code, data, step by step instructions and information about the used hyperparameters can be found on the following GitHub page: (`https://github.com/Lukevanl/Thesis_PIO_Extraction/tree/master`).

Model	Precision	Recall	F1-score
Most Common Entity	0.06	0.27	0.09
LogReg - Baseline	0.46	0.27	0.34
LogReg - Leaderboard	0.3	0.47	0.36
CRF - Baseline	0.54	0.33	0.41
LSTM-CRF - Leaderboard	**0.64**	0.38	0.46
Transformer	0.56	**0.50**	**0.53**

Table 3.1: Weighted precision, recall and f1-score of transformer model in comparison to the leaderboard and a baseline model all evaluated on the same test set.

Entity	Precision	Recall	F1-score
P: Age	64.15	66.58	65.34
P: Condition	80.66	24.44	37.51
P: Sample size	77.39	67.79	72.27
P: Sex	25.53	75.00	38.10
I: Control	65.75	43.44	52.32
I: Drug	70.67	65.20	67.83
I: Educational	47.01	38.46	42.31
I: Other	0.00	0.00	0.00
I: Physical	21.35	51.63	30.21
I: Psychological	0.00	0.00	0.00
I: Surgical	21.58	41.04	28.29
O: Adverse effects	53.47	29.17	37.75
O: Mental	67.20	25.97	37.46
O: Mortality	67.53	68.42	67.97
O: Other	54.57	36.35	43.63
O: Pain	71.03	59.38	64.68
O: Physical	56.01	63.30	59.43

Table 3.2: Precision, recall and f1-score of transformer model on each separate entity evaluated on the test set

Looking at the results, we see that the model outperforms the existing Logistic Regression and Bidirectional LSTM-CRF models on the leaderboard with a difference of 0.17 and 0.07 on the f1-score respectively. We also see that there seems to be a correlation between the amount of instances for an entity and the f1-score (figure 3.1). There are some exceptions to this

rule when the entity is easier to assign. For example, the entities *P: Sample Size* and *P: Age* have a high f1-score despite a low entity count. This can be explained by the simplicity of assigning these two entities since they are both often numbers. The correlation of elements with higher entity counts having higher f1-scores can be caused by class imbalance and/or by data saturation not having been achieved.

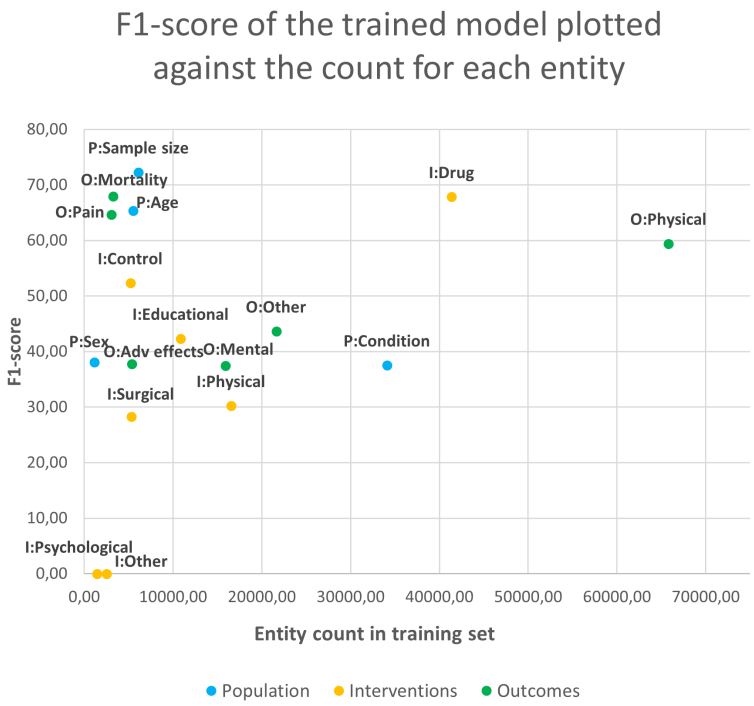


Figure 3.1: Scatter plot of the F1-score on the test set plotted against the count for each entity in the training set.

Chapter 4

Data saturation

In the previous section we showed that there seemed to be a correlation between the f1-score and the count of an entity in the trained model (figure 3.1). To see whether this correlation was caused by data saturation not having been achieved, we have run the model on 20%, 40%, 60%, 80% and 100% of the training data with the same early stopping conditions as the original model. The only modification that was made was to the `eval_frequency` hyperparameter which determines after how many iterations the model gets evaluated on the development set. This value was changed with respect to the percentage of the training set to compensate for the decreasing size. So `eval_frequency` is 200 for 100% of the training set, 160 for 80% etc.

The weighted precision, recall and f1-score of the model on the test set for the different training sizes can be seen in figure 4.1. The individual f1-scores of each entity for the different training set sizes can be found in figure 4.2. The entities in these figures are sorted on the entity counts in the training set. We validated the models on the test set since the labels of the test set are of higher quality and there is no feedback loop going back into the model so we do not overfit to any of the characteristics of the data in the the test set.

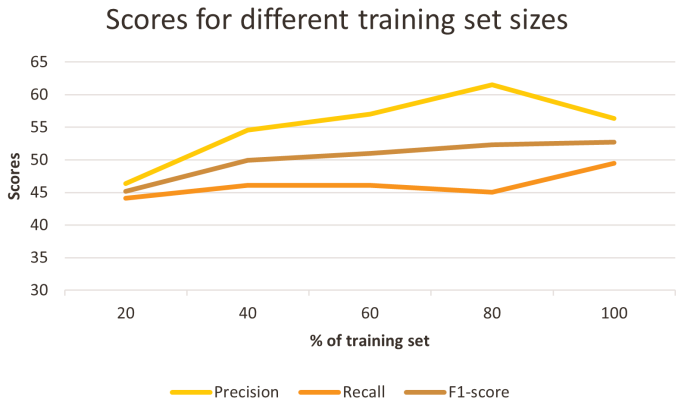


Figure 4.1: Precision, Recall and F1-score on the test set for models trained on different sizes for the training set.

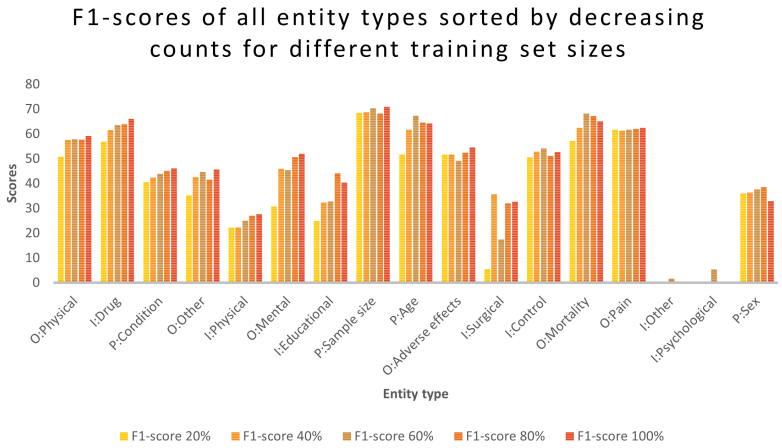


Figure 4.2: F1-scores for each entity evaluated on the test set trained on different training set sizes sorted in decreasing order on the count of each entity.

These figures show us that the model still benefits from more data. In figure 4.1 we see an upwards trend for the combined metrics when the training size increases and in figure 4.2 we can see that in most elements with low and high entity counts, more data increases the f1-scores. This holds for the entities that are easier to assign like *P: Sample Size* and *P: Age* but also for the more complex entities like many of the interventions. In conclusion, the model could benefit substantially from more labeled data.

To determine how new RCT studies should be sampled and annotated, we first take a closer look at the type of errors the model makes by performing an error analysis. In the error analysis we will look for systematic

errors our model makes on samples of the development set. We do this by first comparing the output of the model to the used ground truth using the models' confusion matrix. Afterwards, we point out problems with the ground truth and create a ranking of different types of errors which we use to perform a detailed evaluation of the model by hand to get a better idea of the type and severity of the errors that are being made.

Chapter 5

Error analysis

To know what type of errors the mode makes, we first create the confusion matrix (figure 5.1) on the development set. Because of the high class imbalance which is primarily caused by the *None* class, we take the logarithm of each number in the confusion matrix to generate the heatmap for the confusion matrix (figure 5.2).

	None	I: Drug	O: Physical	P: Condition	O: Mental	I: Physical	O: Other	P: Age	P: S. size	I: Control	O: Mortality	I: Surgical	O: Adv. eff.	I: Educat.	O: Pain	P: Sex	I: Psychol.	I: Other
None	94833	1325	2034	1168	464	550	312	391	207	115	127	192	219	143	46	120	0	0
I: Drug	1435	3642	48	0	0	99	0	0	0	20	0	3	0	0	0	0	0	0
O: Physical	2204	32	4781	96	136	16	150	4	0	0	25	1	102	0	11	1	0	0
P: Condition	1642	13	38	1573	11	4	2	50	52	0	0	36	10	4	6	28	0	0
O: Mental	527	1	51	5	895	2	56	2	0	0	0	0	6	8	0	1	0	0
I: Physical	815	42	23	4	0	698	1	0	0	3	0	92	2	22	0	0	0	0
O: Other	1283	26	444	2	60	24	695	0	0	0	11	6	31	11	4	0	0	0
P: Age	78	0	1	8	0	0	0	321	2	0	0	0	0	0	0	0	0	0
P: S. size	229	3	0	5	0	0	0	4	408	0	0	3	0	0	0	13	0	0
I: Control	262	92	0	0	0	22	0	0	0	362	0	1	0	7	0	0	0	0
O: Mortality	102	0	10	0	0	0	12	0	0	0	388	0	4	0	0	0	0	0
I: Surgical	337	15	5	29	0	73	0	0	0	0	0	330	0	0	0	0	0	0
O: Adv. eff.	141	0	107	0	0	0	1	0	0	0	7	0	418	0	2	0	0	0
I: Educat.	618	2	7	3	1	28	0	3	0	2	3	2	0	321	0	0	0	0
O: Pain	129	1	22	0	19	0	12	0	0	0	10	0	8	0	235	0	0	0
P: Sex	7	0	0	0	0	0	0	1	3	0	0	0	0	0	0	93	0	0
I: Psychol.	89	13	0	0	0	54	0	0	0	3	0	0	0	36	0	0	1	0
I: Other	161	11	1	1	0	64	0	0	0	0	0	0	0	12	0	0	0	0

Figure 5.1: Confusion matrix of the detailed PIO entities for the development set. The entities are sorted by the entity counts in decreasing order in the development set.

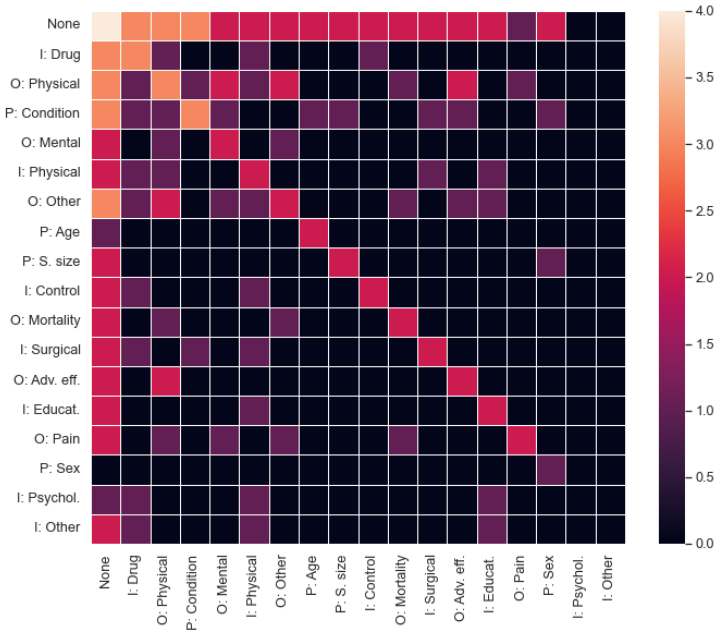
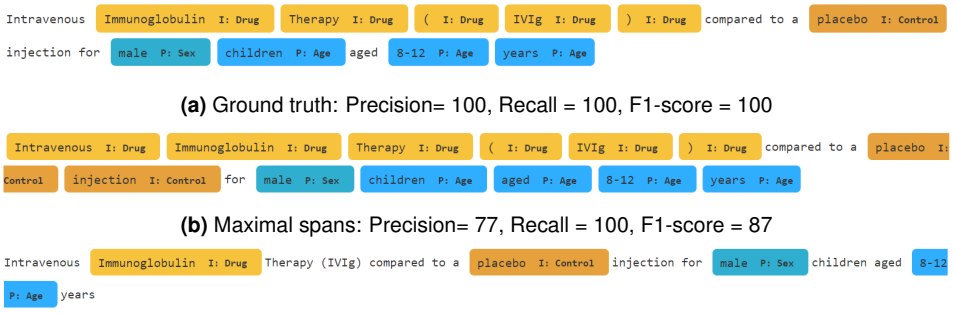


Figure 5.2: Heatmap of the detailed PIO entities where the log is taken for each count in the development set. The entities are sorted by the entity counts in decreasing order in the development set.

In the confusion matrices we see that very few mistakes are made where a token was assigned a P, I or O element but the actual entity was an element of a different PIO class entirely. Mistakes between the different classes in the PIO elements are slightly more common but still relatively rare. Instead, the most common mistakes that are made, are false positives (FPs) and false negatives (FNs) on the *None* class. Of the 19684 total errors 17472 ($\approx$ 89%) are made on the *None* class. Of the errors on the *None* class, 7413 were cases where the model assigned some PIO class while the *None* class should have been assigned and 10059 of the errors were cases where the model assigned the *None* class while one of the PIO classes should have been assigned.

After manually inspecting some of the output of the model and the annotations we considered to be the ground truth, we noticed that evaluating on the ground truth does not always give a good indication about the actual usefulness of the output of the model for its purpose. To show this, consider the (made up) span annotated with what we consider to be the ground truth in figure 5.3a. The same span could be annotated using many different strategies while losing little to no information. For example, in figure 5.3b and 5.3c we annotate with maximal and minimal spans respectively. For some applications, these maximal or minimal annotations might

be considered better. However, we can see that the f1-score is considerably lower for annotations b and c because we consider annotations a to be the 'ground truth' which our model evaluates against.



(a) Ground truth: Precision= 100, Recall = 100, F1-score = 100

(b) Maximal spans: Precision= 77, Recall = 100, F1-score = 87

(c) Minimal spans: Precision= 100, Recall = 40, F1-score = 57

Figure 5.3: Span of text annotated with the ground truth (a), maximum spans (b) and minimum spans (c) together with the precision recall and f1 scores of the annotations evaluated against the ground truth (a).

To get a better overview of the type of mistakes the model makes, we first subdivide the types of errors into 10 classes which are ranked on what we, after a discussion with a medical professional, have determined to be from most to least critical:

1. Annotation completely missing causing the loss of critical information
2. Annotation was too short causing the loss of critical information
3. Entity assigned where none should have been
4. Entity assigned where another entity should have been of a different PIO element
5. Annotation was too long causing the inclusion of irrelevant tokens
6. Annotation completely missing causing the loss of less critical information
7. Entity assigned where another entity should have been of the same PIO element
8. Annotation was too short causing the loss of less critical information
9. Annotation was too long causing the inclusion of less important tokens
10. Repetition of entity missed

In general we consider false negatives (FN) to be more critical than false positives (FP) because from a clinical perspective, it is often better to include non-important information than to miss important information.

We also make a distinction between information that we consider to be critical like the measured outcomes (e.g. level of stress) and less critical information like the way the outcome was measured (e.g. questionnaire). We give repetitions a low rank (type 10) if it was assigned at some stage but missed later on. We do this, because very little information is gained if it is annotated a second or third time. If all the occurrences of a certain P, I or O entity were missed they are all assigned a type 1 error. If an error can be assigned to multiple types, we always assign the lowest type of error.

Now we evaluate the output of the model by hand using the ranked types of errors described above. The evaluation process is separated into two rounds:

1. Sample the 10 abstracts from the development set that have a minimum of 100 words, a maximum of 500 words and the highest ratio of FNs for the PIO elements. Now we evaluate the resulting 10 abstracts by hand.

2. Randomly sample two abstracts for each of the intervention classes (except for *Control*) which have that intervention as main intervention. We again filter out abstracts with fewer than a 100 words or more than 500 words. Wee use the gained insights from round 1 and a discussion with a medical professional to annotate the resulting 12 abstracts.

Round 1

For round 1, we have decided to sample based on the ratio of FNs of all the PIO classes (including the *None* class) in the abstracts. We sample on the ratio of FNs because FNs are a type of error that is common and we also consider this type of error to be more critical than FPs. At first this only returned very short abstracts. Therefore, we removed abstracts with fewer than a 100 words (the average length of an abstract is about 271 words). We also removed abstracts with more than 500 words to make the sample more representative. During the evaluation process we keep track of the following information:

- The number of errors of each type in each abstract for every PIO element.

- The subject of each abstract. Most sampled RCT studies in the EBM-NLP dataset were either about cardiovascular diseases, cancer or

autism. We keep track of the subject to see whether the sample was representative.

- The main intervention class of each abstract.
- Common mistakes and other abnormalities found during the evaluation process.

Results and discussion round 1

The results of the types of errors in round 1 can be found in figure 5.4 and the subjects and main interventions can be found in figure 5.5 and 5.6. All the individual evaluations, the subjects, main interventions and the full texts of all sampled abstracts of round 1 can be found on GitHub `https://github.com/Lukevanl/Thesis_PIO_Extraction/tree/master/round1`.

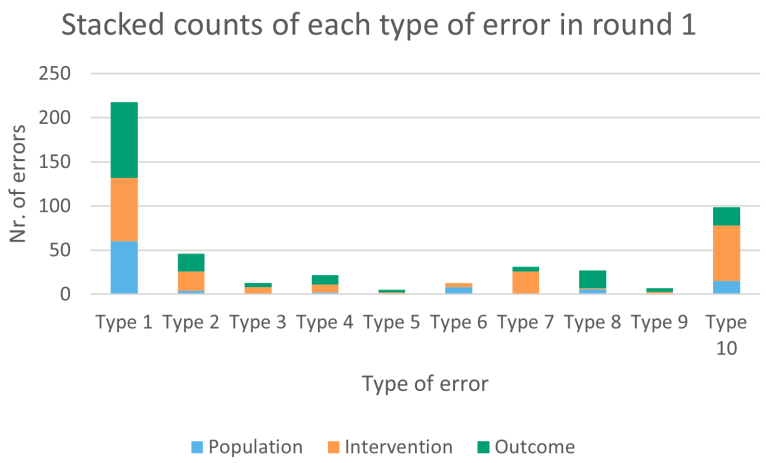
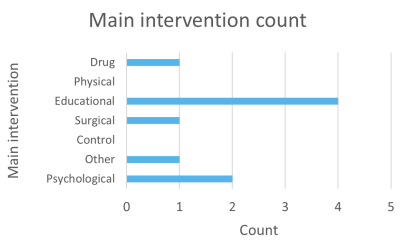


Figure 5.4: Stacked bar chart of counts for each type of error for all PIO elements for the abstracts of round 1.



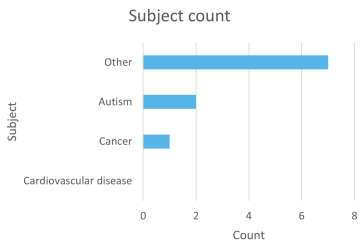


Figure 5.5: Counts main interventions round 1. **Figure 5.6:** Counts subjects round 1.

Figure 5.4 tells us that severe type 1 errors are still very common in the sampled abstracts. Type 10 errors (missed repetitions) are also common

especially in interventions where abbreviations and subsequent occurrences of interventions often get missed. In figure 5.5 we see that the sampled abstracts mostly contain types of interventions which are less common in the dataset (see figure 2.1) like *Psychological* and *Educational*. Figure 5.6 shows us that the sampled abstracts for round 1 have subjects that are often not one of the three common subjects (autism, cancer and cardiovascular diseases). The full dataset was sampled with an emphasis on those three subjects with the goal of sampling a high amount of common conditions. Since the 10 sampled abstracts were selected on the highest ratio of false negatives (FN), the poor performance on these abstracts could be caused by the model not properly generalizing to uncommon subjects and/or types of interventions.

Common mistakes and abnormalities

During the evaluation process we also kept track of common mistakes and abnormalities in the output of the model and looked for possible causes. To classify whether valuable information was lost with those mistakes, we used the insights from a medical professional through a discussion. It is also important to note that whether something is important to annotate depends on the task the PIO elements are used for and thus it is hard to classify something as a 'valuable' mistake. So we try to look for mistakes that would be considered valuable for a wide range of possible applications for which the PIO elements could be used.

Population

In the population class, most errors were made on the *P: Condition* class because it is by far the most common class and because of the difficulty of this entity compared to the other population entities. This extra difficulty arises due to the fact that *P: Sample size*, *P: Age* and *P: Sex* all have very limited domains (e.g. only numbers or only one of the sexes) while the *P: Condition* entity can comprise a much wider variety of values.

There also was a surprising amount of errors in the *P: Sample size* class. Sample sizes of the RCT studies regularly occur in the format ($n = x$) or ($N = x$) where x is the sample size. N describes the sample size of the entire population while n describes the sample size of the subgroups. Sample sizes in these formats were often missed by the model and only sometimes annotated despite having a very standard structure which should be easy for the model to detect. If we look at the crowd annotations of the development set, we see that these sample sizes were annotated very irregularly. Sometimes all the samples sizes in these formats were annotated, sometimes only the samples sizes of the entire population and sometimes neither were

annotated. The model seems to copy this irregular behaviour.

The sample sizes of the subgroups do not carry much information for physicians. They do however, provide information that can be used for assessing the risk of bias in the RCT study. The sample size of the entire population on the other hand, provides important information for physicians because a larger sample size allows for a more precise estimate of the results for the intervention applied in the RCT study. And thus, if a sample size is larger, a RCT study carries more value because it has a smaller margin of error.

We suggest, when annotating additional data, to provide clear instructions for the annotators on which of these sample sizes to annotate in order to increase the consistency of the model's output for these instances. For many tasks, solely identifying the sample size of the entire population is sufficient.

Interventions

The most errors were made on the intervention class in the samples abstracts. Type 1, type 2 and type 10 errors were all common. Common mistakes include missing abbreviations describing interventions, missing subsequent occurrences of the same intervention and also missing a single intervention at every occurrence in the text causing a lot of errors for the same one mistake. The wrong intervention class also gets assigned quite often (type 7 error). On top of that, information about how interventions were applied (e.g. intravenously) often gets included in the annotations. In the instructions, the annotators are instructed to not annotate this information. However, in the annotations it was quite regularly included causing the model to replicate this behaviour. In the sampled abstracts, the intervention annotations were often the most difficult class to evaluate because of the high complexity of the class and the medical expertise that is sometimes required to properly assign the correct type of intervention. This complexity can also be the reason why the model frequently makes mistakes on these classes in the sampled abstracts.

Besides that, types of interventions that were less common in the dataset (e.g. Psychological and Educational) were overrepresented in the sample while the two most common classes (Drug and Psychical), despite accounting for about two-thirds of the intervention class, only appeared once as main intervention in the ten sampled abstracts. The subjects of the abstracts also mostly included subjects which were not in the three common subjects (autism, cancer and cardiovascular diseases) on which the dataset was primarily sampled. So the large number of errors in the sample could

be caused by a lack of labeled data for the uncommon interventions, poor generalization to subjects outside of the sampled subjects or the model is not able to fully capture the high complexity of the task.

Outcomes

Considering the Outcome classes are the most common classes by quite a large margin, the model seems to perform quite well on this PIO element. The model does however have the highest amount of type 1 errors on this element. These errors mainly come from the same outcomes being missed in all occurrences of the text or long spans of text describing an outcome that were missed in its entirety which amount to a large number of errors for a single missed span.

Type 8 errors are also quite common. For these errors the main outcome is captured but the span was too short. As a result the model misses information that would be considered less critical for most tasks. For example, in the span *adherence to prescribed medicine* the word adherence describes one of the outcomes which would be considered critical information and the remainder of the sentence describes for what event the outcome was measured. This information might be relevant for some tasks but would in general not be considered a critical error.

In the sampled abstracts, there was no clear pattern in the errors of the *Outcome* class nor do we see any systematic errors that could easily be solved.

Round 2

For round 2 we look at the differences in errors between abstracts with main interventions that have a high entity count (e.g Drug) and interventions that have a low entity count (e.g Psychological). We do this, by randomly sampling for each intervention class, two abstracts which has that intervention as the main intervention (except for *Control* since it is never the main intervention). Then after evaluating the abstracts by hand just like in round 1, we compare the types of mistakes that are being made between the different abstracts. Again, we filter out abstracts with fewer than a 100 words or more than 500 words to only consider abstracts that are representative of the dataset.

We also excluded the abstracts that were already sampled in round 1. This does introduce a bias especially towards the uncommon interventions since most abstracts in round 1 had those intervention as main intervention and thus, some of the worst performing abstracts of those interventions are fil-

tered out.

As a heuristic for the initial sampling, we considered the most common type of intervention to be the main intervention. After the initial sampling, four abstracts were resampled because they either described non-RCT studies or the most frequent assigned intervention was not the actual main intervention.

In the evaluation process of round 2 we use the insights from round 1 about what should be classified as a mistake to evaluate the newly sampled abstracts more accurately. We keep track of the same information as in round 1 but now, we also keep track of the total number of correct entities for each PIO element so that we can calculate the precision, recall and f1-scores and compare the results of our evaluation against the crowd annotations.

Results and discussion round 2

All the individual evaluations, the subjects, main interventions and the full texts of all final sampled abstracts of round 2 can be found on GitHub (`https://github.com/Lukevanl/Thesis_PIO_Extraction/tree/master/round2`.)

The results of the types of errors in round 2 can be found in figure 5.7. The evaluation metrics for each PIO class can be found in figure 5.8.

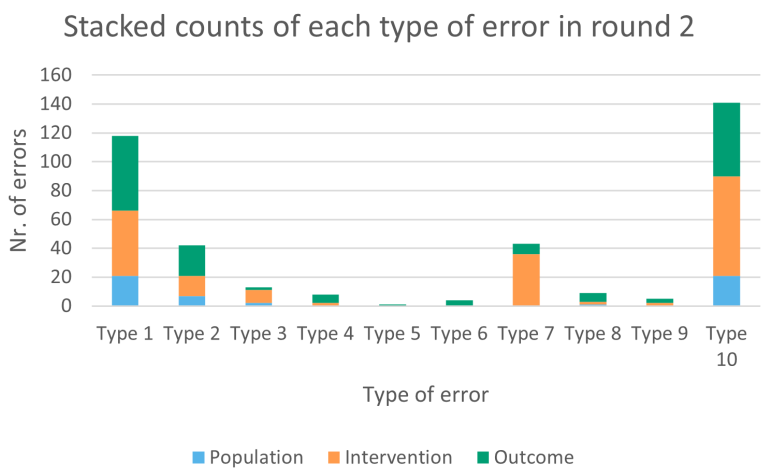


Figure 5.7: Stacked bar chart of counts for each type of error for all PIO elements for all the abstracts of round 2 combined.

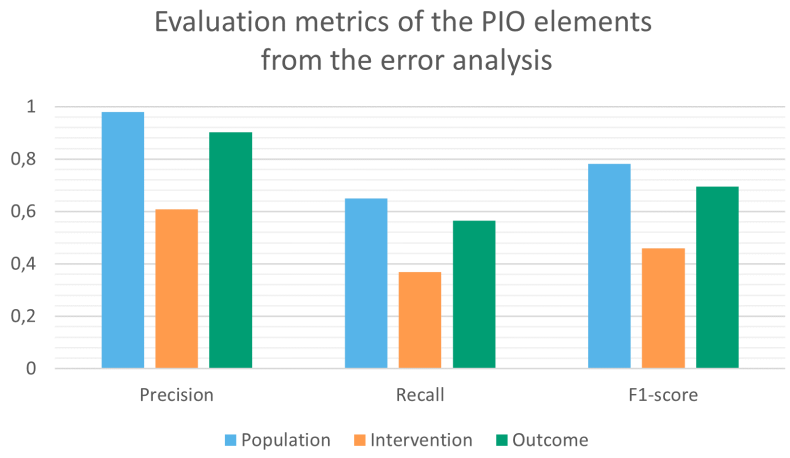


Figure 5.8: Evaluation metrics for the P, I and O elements resulting from the error analysis.

Figure 5.7 tells us that FNs (Type 1, 2, 6, 8 and 10 errors) are far more common than FPs (Type 3, 4, 5, 7 and 9 errors) in the sampled abstracts for all the PIO elements. Especially type 1 and 10 errors are still very common. As a consequence, the recall scores are considerably lower than the precision scores for all three PIO elements as illustrated in figure 5.8. The most mistakes were made on the *Intervention* class despite only being the second most frequent class after *Outcome*. Very few mistakes are made on the Population class. Especially the precision score (=0,979) for the population class is very high.

In figure 5.9 we can see the types of errors for each abstract grouped by their main intervention. Here we see that most types of errors seem to appear consistently for all the different types of interventions and the performance only decreases slightly for interventions with lower entity counts. Only type 7 errors (correct PIO element but incorrect fine-grained element) seem to mostly occur in types of interventions with low entity counts which can be explained by class imbalance because the model is more likely to guess frequent classes since it is more often correct. This results in a larger number of type 7 errors for the interventions with a low count. We also see that the *Other* intervention type performs quite poorly especially on the outcomes. Because of the small sample however, this can also quite possibly be caused by chance.

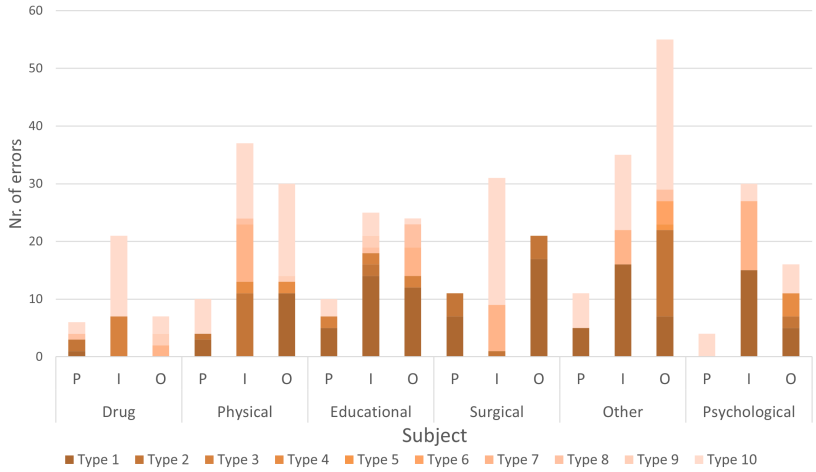


Figure 5.9: Number of each type of error for each subject separated by the PIO classes.

In figure 5.10 we can see the f1-scores for each abstract grouped by their main intervention for each separate PIO element. This figure again shows that the model seems to generalize quite well to abstracts with uncommon types of interventions. Only the f1-score of the *Intervention* class drops drastically for the *Other* and *Psychological* classes. However, this can again be explained by class imbalance because both of these classes have very low counts (see figure 2.1).

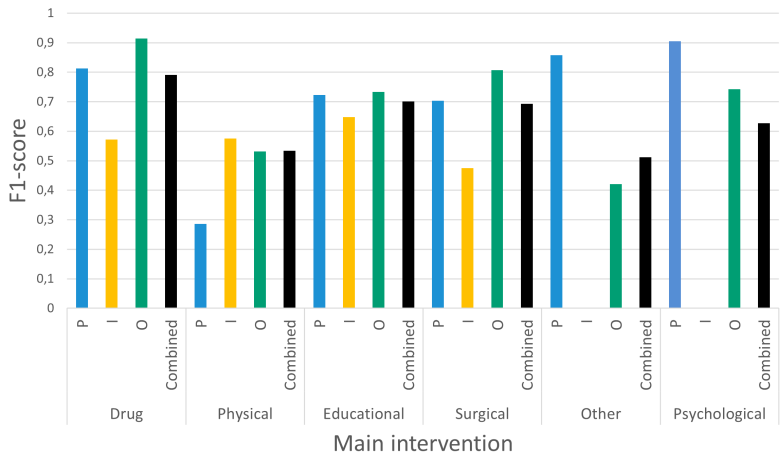


Figure 5.10: Number of each type of error for each subject separated by the PIO classes.

We also compared the evaluation metrics of our own evaluation against the evaluation on the ground truth. In figure 5.11 and 5.12 we can see the precision and recall scores for each abstract and the combined scores.

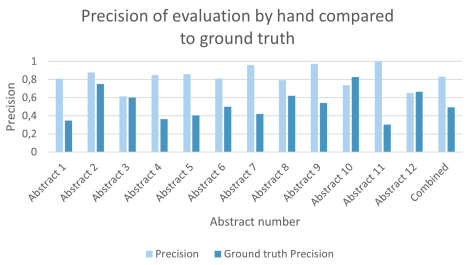


Figure 5.11: Precision of error analysis compared to dataset annotations.

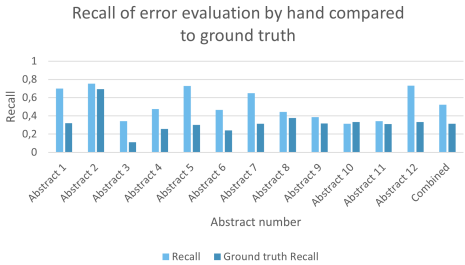


Figure 5.12: Recall of error analysis compared to dataset annotations.

Figures 5.11 and 5.12 show that both the precision and recall of the evaluation against the crowd annotations is lower than the evaluation done by hand on the error analysis. This means that the crowd annotations include a large amount of annotations that we deem to be FPs which result in lower precision scores for the model but also misses information that we consider to be valuable resulting in a higher number of FNs and thus lower recall scores for the model. As a result, the model will be likely to partially adopt this annotation strategy and replicate those errors.

Conclusions round 1 and round 2

In round 1 we saw that there seemed to be very few systematic errors. The systematic errors that did occur were most likely caused by irregular annotations. These irregularities can be reduced by providing additional instructions for the crowd workers or aggregating the results over a higher amount of workers. We also saw that the model seemed to perform worse on abstracts with uncommon subjects and main interventions.

In round 2 we looked at the differences between more common types of interventions and less common ones. By comparing the different types of errors we saw little difference between the different interventions. The performance only seemed to decrease slightly for abstracts which had interventions with lower counts. Only the performance on the intervention class decreased drastically due to class imbalance on interventions with lower entity counts. This especially affected the number of type 7 errors. To reduce this class imbalance, one may try to sample additional abstracts on subjects that have these uncommon types of interventions. We also saw that the evaluations against the ground truth resulted in much lower precision and recall scores than the evaluation of our error analysis showing some of the inaccuracies of the crowd annotations.

Chapter 6

Conclusion

In this research, we looked at the research question *"How can spans of text containing detailed PIO elements be extracted from RCT studies to speed up systematic reviews and facilitate Evidence-Based Medicine (EBM)?*". Before we could answer this question we looked at four sub-questions.

First, we looked at the sub-question *"How well do transformer-based models perform on entity extraction of detailed PIO values on RCT studies in comparison to earlier attempts?"*. To answer this question, we trained a transformer model and saw the potential of using a transformer model on the task of detailed PIO extraction with the model outperforming the existing models by a large margin. We also explored the data saturation and saw that the model still seemed to benefit from additional data.

We then explored how useful the output of the model is to medical professionals by answering the sub-question *"To what extent does the accuracy measurement of the model reflect the usefulness of the output to medical professionals?"*. Here we saw that the used metrics do not always give a good representation of the usefulness of the output.

Next, we performed an error analysis with our own evaluation method to better capture the usefulness of the output and to answer the sub-questions *"How to sample additional RCT abstracts for annotation to improve the classifier?"* and *"In what way can data be annotated more effectively to improve the classifier?"* by looking for improvements in the annotation and sampling process. We highlighted systematic errors, common mistakes and looked at the type and severity of the mistakes that were made. We also analysed the differences between interventions with high entity count and low entity count and saw that the model seemed to generalize quite well. Only the score of the intervention classes reduced drastically due to class imbalance.

With these answers to our sub-questions we answer our main research question "*How can spans of text containing detailed PIO elements be extracted from RCT studies to speed up systematic reviews and facilitate Evidence-Based Medicine (EBM)?*" by saying that transformer models present much potential for extracting detailed PIO elements out of RCT studies. We also saw that the classifier could still improve even further from additional data and from more consistent annotations to reduce systematic errors and common mistakes. However, in the error analysis we also saw that there were still a large number of critical errors. As a result, valuable information gets missed and thus, the model is not consistent enough to fully automate systematic reviews and some form of human verification is still required.

In future research one could explore whether pre-trained models on scientific texts like sciBERT provide better results than the currently used spacy pipeline based on RoBERTa. One could also explore how well the model generalizes to abstracts that have subjects that were not in the three subjects (cancer, cardiovascular diseases, autism) on which the dataset was primarily sampled.

Bibliography

[1] P. Fontelo and F. Liu, "A review of recent publication trends from top publishing countries," *Systematic Reviews 2018 7:1*, vol. 7, pp. 1–9, 9 2018.

[2] E. Lehman, J. DeYoung, R. Barzilay, and B. C. Wallace, "Inferring which medical treatments work from reports of clinical trials," in *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 3705–3717, 2019.

[3] J. DeYoung, E. Lehman, B. Nye, I. J. Marshall, and B. C. Wallace, "Evidence inference 2.0: More data, better models," 2020.

[4] I. J. Marshall, J. Kuiper, E. Banner, and B. C. Wallace, "Automating biomedical evidence synthesis: RobotReviewer," *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, vol. 2017, pp. 7–12, July 2017.

[5] I. J. Marshall, B. Nye, J. Kuiper, A. Noel-Storr, R. Marshall, R. Maclean, F. Soboczenski, A. Nenkova, J. Thomas, and B. C. Wallace, "Trialstreamer: A living, automatically updated database of clinical trial reports," *Journal of the American Medical Informatics Association*, vol. 27, pp. 1903–1912, 12 2020.

[6] D. Jin and P. Szolovits, "PICO element detection in medical text via long short-term memory neural networks," in *Proceedings of the BioNLP 2018 workshop*, (Melbourne, Australia), pp. 67–75, Association for Computational Linguistics, July 2018.

[7] L. Schmidt, J. Weeds, and J. P. T. Higgins, "Data mining in clinical trial text: Transformers for classification and question answering tasks," 2020.

[8] B. C. Wallace, J. Kuiper, A. Sharma, M. Zhu, and I. J. Marshall, "Extracting pico sentences from clinical trial reports using supervised distant supervision," *J. Mach. Learn. Res.*, vol. 17, p. 4572–4596, jan 2016.

[9] B. Nye, J. J. Li, R. Patel, Y. Yang, I. J. Marshall, A. Nenkova, and B. C. Wallace, "A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature," *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 197–207, 6 2018.

[10] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[11] J. Lafferty, A. Mccallum, F. C. N. Pereira, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," pp. 282–289, 2001.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, p. 1735–1780, nov 1997.

[13] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

Appendix A

Appendix

Local injection of bupivacaine I: Drug after rubber band ligation of hemorrhoids P: Condition : prospective, randomized study.

PURPOSE The aim of this study was to determine if local injection of bupivacaine I: Drug after hemorrhoidal banding causes a decrease O: Pain in O: Pain pain O: Pain and O: Pain in O: Pain the O: Pain incidence O: Pain of O: Pain associated O: Pain symptoms O: Pain . O: Pain

METHODS After hemorrhoidal P: Condition banding P: Condition , patients were randomly assigned to receive a local I: Drug injection I: Drug of I: Drug bupivacaine I: Drug with I: Drug 1:200,000 I: Drug epinephrine I: Drug , an injection I: Drug of I: Drug normal I: Drug saline I: Drug , or no I: Control injection I: Control , I: Control just I: Control superior to each band. Pain O: Pain was graded by the patient and by the study nurse within 30 minutes, and any associated O: Physical symptoms O: Physical were recorded. At intervals 6, 24, and 48 hours postbanding, the patient recorded pain O: Pain , O: Pain limitation O: Pain of O: Pain activities O: Pain , O: Pain and O: Pain analgesic O: Pain requirements O: Pain . O: Pain Associated O: Physical symptoms O: Physical while at home were recorded.

RESULTS Of 115 P: Sample size patients studied, 42 received bupivacaine I: Drug injection, 42 received normal I: Control saline I: Control injection, and 31 received no I: Control injection I: Control . In patients receiving bupivacaine I: Drug compared with no injection, within 30 minutes postbanding there was a significant reduction O: Pain in O: Pain pain O: Pain graded by the patient (P = 0.000002) and by the nurse (P = 0.000005) and a significant reduction in incidence O: Adverse effects of O: Adverse effects nausea O: Adverse effects (P = 0.01) and shaking (P = 0.008). However, in the bupivacaine I: Drug group compared with the other two groups, at the intervals of 6, 24, and 48 hours postbanding there was no sustained reduction in the severity O: Pain of O: Pain pain O: Pain and no reduction in analgesic O: Pain requirements O: Pain or O: Pain limitation O: Pain of O: Pain normal O: Pain activities O: Pain . O: Pain In the week after banding, there was no difference between groups in symptoms of nausea O: Physical , O: Physical shaking O: Physical , O: Physical lightheadedness O: Physical , O: Physical urinary O: Physical retention O: Physical , O: Physical or O: Physical bleeding O: Physical . O: Physical

CONCLUSIONS Bupivacaine I: Drug injection may be useful for reducing pain and associated symptoms long enough to tolerate a trip home from the outpatient department but does not show a sustained effect.

Figure A.1: RCT abstract annotated with detailed annotations from the EBM-NLP dataset.

Appendix B

Appendix

README.md of the GitHub page:
`https://github.com/Lukevanl/Thesis_PIO_Extraction/tree/master`

B.1 Thesis PIO element extraction on RCT studies

B.1.1 Repository content

In this repository you can find the project described in the thesis (*Thesis_v1.pdf*). We provide the following information:

- Replication instructions for the model described in the thesis (see replication instructions below)
- Code with which the scores for the naive baseline was calculated (*naivebaseline.py*)
- Brief EDA on the dataset and the counts of each element in the training set (*eda.py* and *picocounts.xlsx*)
- Results of round 1 and round 2 of the error analysis as described in the thesis (read README.md in *error_analysis* folder for more information)
- Results of exploring the data saturation (*saturation.xlsx*)

To replicate the data saturation numbers simply take the first 20%, 40%, 60%, 80% and 100% of the dataset and run the model with this data on the same config file with just one modification: change the eval_frequency hyperparameter inside the config.cfg file with respect to the percentage of the training set to compensate for the decreasing size. So, eval_frequency is 200 for 100% of the training set, 160 for 80% etc.

B.1.2 Dependencies:

- python 3.8.

- spaCy version 3.1.3 (pip install spacy==3.1.3) + en_core_web_lg pipeline (python -m spacy download en*core*web_lg).
- numpy version 1.18.5 (pip install numpy==1.18.5).
- pandas version 1.0.5 (pip install pandas==1.0.5).
- matplotlib version 3.2.2 (pip install matplotlib==3.2.2).
- seaborn version 0.10.1 (pip install seaborn==0.10.1).
- scikitlearn version 0.23.1 (pip install scikit-learn==0.23.1).

B.1.3 Table 1: All entities with their label inside the transformed dataset.

```
P = Population, I = Intervention, O = Outcome
*----------------------------*
| Label | Entity             |
|-------|--------------------|
| 0     | None               |
| 1     | P: Age             |
| 2     | P: Sex             |
| 3     | P: Sample Size     |
| 4     | P: Condition       |
| 5     | I: Surgical        |
| 6     | I: Physical        |
| 7     | I: Drug            |
| 8     | I: Educational     |
| 9     | I: Psychological   |
| 10    | I: Other           |
| 11    | I: Control         |
| 12    | O: Physical        |
| 13    | O: Pain            |
| 14    | O: Mortality       |
| 15    | O: Adverse Effects |
| 16    | O: Mental          |
| 17    | O: Other           |
*----------------------------*
```

B.2 Replication instructions main model:

B.2.1 1. Clone the repository

B.2.2 2. Unzip the 'ebm_nlp_2_00.tar.gz' file so you get the ebm_nlp_2_00 folder in the root directory.

B.2.3 3. Execute the prepare.sh script which prepares the data by running the following files (could take up to a couple minutes depending on your hardware):

loaddata.py:

This file first loads all the .tokens files which hold the tokenized texts and all the .txt files which hold the full texts. The tokens and documents are stored in seperate arrays together with the document ID for each document. Now all the individual annotations are read into arrays. The dataset provides individual annotations for every PIO element but since we want to have a dataset where all entities are combined, we merge the annotations. We do this by first making all the assigned labels distinct (see table 1) so that we can uniquely identify each of the 17 elements (18 if you count the 'None' class).

reformdata.py:

Here we use the loaded data from the last file to apply some transformations. Firstly, the model requires the annotations to provide the character indices of the beginning and the end of all the annotated tokens instead of the text of the tokens. Therefore, from the token-level annotations we generate character-level annotations instead. Next, we match the annotations with the corresponding texts and tokens using the document ID's we stored. Lastly, we map the numerical labels to the entity names enumerated in table 1 so that the labels are more intuitive.

ner_pico.py:

After the previous steps the data is in the following format: [('None', 'Effect'), ('None', 'of'), ('I: Drug', 'aspirin'), ('None', 'for'), ('O: Outcome', 'headaches')] Where the left element of each tuple is the entity and the right element the token. We want to transform it into the following format: ['Effect of aspirin for headaches', {entities: [(8, 14, 'I: Drug'), (18, 26, 'O: Outcome')]}] Here we have the full text on the left and all the entities (excluding the 'None' class) with the character indices in a dictionary. This was the original format for spacy version 2.x and can easily be converted to the .spacy format used in spaCy 3.x. To get this result we first must change the tokens by the character indices we generated in reformdata.py

and filter out the 'None' instances. Now we make a tuple of the full text and the dictionary containing the entities. Out of this format we can make instances of spaCy's 'Doc' objects. All these 'Doc' objects are stored in a DocBin object. This DocBin gets stored in the .spacy format and now we are ready to train our model with this .spacy file.

B.2.4 4. Execute the *model.sh* script inside the *spacy_acc* folder which trains and evaluates the model and saves the model in the *output* folder.

For training the model, the hyperparameters described in the config.cfg file are used. The output can be seen in the file scores.txt after running the script. Running this script could take up to a couple hours. If you have a GPU available, you can add *–gpu-id = 0* at the end of the spacy train command to speed it up considerably.