UPPSALA
UNIVERSITET

UPTEC F 20031
Examensarbete 30 hp
Juni 2020

Improving the speed and quality of an Adverse Event cluster analysis with Stepwise Expectation Maximization and Community Detection

Nils Erlanson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Improving the speed and quality of an Adverse Event cluster analysis with Stepwise Expectation Maximization and Community Detection

Nils Erlanson

Adverse drug reactions are unwanted effects alongside the intended benefit of a drug and might be responsible for 3-7% of hospitalizations. Finding such reactions is partly done by analysing individual case safety reports (ICSR) of adverse events. The reports consist of categorical terms that describe the event. Data-driven identification of suspected adverse drug reactions using this data typically considers single adverse event terms, one at a time. This single term approach narrows the identification of reports and information in the reports is ignored during the search. If one instead assumes that each report is connected to a topic, then by creating a cluster of the reports that are connected to the topic more reports would be identified. More context would also be provided by virtue of the topics. This thesis takes place at Uppsala Monitoring Centre which has implemented a probabilistic model of how an ICSR, and its topic, is assumed to be generated. The parameters of the model are estimated with expectation maximization (EM), which also assigns the reports to clusters. The clusters are improved with Consensus Clustering that identify groups of reports that tend to be grouped together by several runs of EM. Additionally, in order to not cluster outlying reports all clusters below a certain size are excluded. The objective of the thesis is to improve the algorithm in terms of computational efficiency and quality, as measured by stability and clinical coherence. The convergence of EM is improved using stepwise EM, which resulted in a speed up of at least 1.4, and a decrease of the computational complexity. With all the speed improvements the speed up factor of the entire algorithm can reach 2 but is constrained by the size of the data. In order to improve the clusters' quality, the community detection algorithm Leiden is used. It is able to improve the stability with the added benefit of increasing the number of clustered reports. The clinical coherence score performs worse with Leiden. There are good reasons to further investigate the benefits of Leiden as there were suggestions that community detection identified clusters with greater resolution that still appeared clinically coherent in a posthoc analysis.

Handledare: Niklas Norén, Eva-Lisa Meldau
Ämnesgranskare: Mats Gustafsson
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F 20031

Populärvetenskaplig sammanfattning

Alla mediciner har oönskade reaktioner utöver den positiva effekten. Att hitta dessa reaktioner görs bland annat genom att övervaka användningen av medicinanvändning. De vanligaste metoderna för att hitta potentiella reaktioner brukar utgå efter efter enskilda biverkningstermer i rapporter som ska beskriva reaktionen. Ett sådant tillvägagångssätt kan begränsa hur många rapporter man hittar eftersom det kan finnas flera ord som kan beskriva liknande reaktioner. Om man istället antar att alla rapporter har ett ämne som de handlar om, då genom att hitta ämnena istället skulle man både kunna hitta fler rapporter och ge mer information om reaktionen i form av dess ämne. Detta är idén bakom algoritmen i detta examensarbete, en algoritm som skapar grupper av rapporter som beskriver samma ämne. För att den ska kunna användas på stora datamängder måste algoritmen kunna köras snabbt, ett av målen har därför varit att snabba på algoritmen vilket gjordes genom att använda en förbättrad metod som använder delmängder av datan för att göra uppskattningar av grupptillhörigheten för rapporterna, samt genom att utföra algoritmen parallellt över datorns processer. Totalt gav de förbättringarna en fördubbling av hastigheten, under rätt förutsättningar. Dessutom har det varit ett mål att förbättra kvaliteten på grupperna i form av hur stabila de är och hur klinisk sammanhängande de är. Det försöktes uppnås genom att använda en algoritm som hittar gruppstrukturer i ett nätverk av rapporter. Resultaten från den förbättringen visade på stabilare grupper men mindre sammanhängande. Däremot så fanns det exempel på när den förbättrade algoritmen hittade grupper som gav mer information än den tidigare implementationens grupper. Det kan alltså finnas anledningar för att fortsätta att utforska den nya algoritmen.

Acronyms

AE Adverse Event

PV Pharmacovigilance

UMC Uppsala Monitoring Centre

ICSR Individual Case Safety Report

WHO World Health Organization

MedDRA Medical Dictionary for Regulatory Activities

CC Consensus Clustering

EM Expectation Maximization

sEM Stepwise Expectation Maximization

MAP Maximum a posteriori Estimation

MLE Maximum Likelihood Estimation

ARI Adjusted Rand Index

%Reports Percentage of Clustered Reports

CPM Constant Potts Model

SUF Speed Up Factor

Contents

1 Introduction

All drugs tend to carry the risk of unwanted effects alongside the intended benefit, the unwanted effect is known as an adverse drug reaction. In fact, some studies have shown that they may cause about 3-7 % of hospitalisations [23]. Clinical trials partly strive to catch adverse drug reactions before drugs reach the market. However, the trials typically cannot identify all possible adverse reactions to a drug. They tend to be limited in time, to exclude certain types of patients, interacting drugs and other confounding factors. Beyond that, they typically include too few patients to be able to identify very rare adverse reactions [33].

The perhaps most infamous case of an adverse drug reaction identified post-marketing occurred in 1961 when about 10,000 women gave birth to children with birth defects. The adverse event was later linked to the use of thalidomide which was marketed and sold as a sedative to pregnant mothers [19]. After this public incident the world began to take notice of the importance of post-marketing monitoring of drugs, therefore ten countries with organized pharmacovigilance centres came together and initiated the World Health Organization's (WHO) Programme for International Drug Monitoring in 1968. The intention of the programme was to help with international collaboration of finding adverse drug reactions [14]. This led to big improvements in international collaboration of Pharmacovigilance (PV). The field has grown since then and the programme now have 139 members [9]. Today the WHO defines PV as

> "...the science and activities relating to the detection, assessment, understanding and prevention of adverse effects or any other drug-related problem." [28]

While clinical trials still play a huge role in finding out as much as possible about the safety concerns, finding adverse drug reactions and reactions of broader patient groups is pivoted towards post-market monitoring. Wherein PV centres rely on spontaneous reporting of adverse events (AE) in order to find linked reactions to the drug [8]. The process of finding the reactions can in general be described as, AEs are reported through individual case safety reports (ICSR) by patients, health care professionals or pharmaceutical companies to a national PV centre, such as the Medical Product Agency[1] (MPA) or the Food and Drug Administration (FDA). Qualitative and quantitative analysis are applied to the reports in order to find not yet known adverse drug reactions, wherein statistical analysis is mainly used to find where to direct the resource of manual clinical review. Causal and risk assessments are done, and the findings are reported back to points of interests. Finally, the information is communicated back to the consumers of the drug to help patients and health care professionals make an informed choice about treatments [7, p. 9]. Much of the techniques and science is done and developed in the analysis step but the most important

[1] Läkemedelsverket in Swedish.

step is the gathering and reporting of the AEs, where a big problem is under-reporting and many AEs are never described on individual case reports [28].

Uppsala Monitoring Centre (UMC) is an independent, non-profit foundation and a centre for international service and scientific research. UMC performs PV at a global scale and is the WHO's Collaboration Centre for International Drug Monitoring [36]. Part of that responsibility is to manage and maintain WHO's global database of ICSRs, VigiBase [22]. VigiBase is a relational database management system that gathers the reports from members of the programmes' national PV centres. As of June 2020 VigiBase contains over 20 million ICSRs from over 130 countries [7, p. 109], with reports dating back to the start of the programme in 1968. The reports contain several structural fields such as gender, nationality, age etc. Some reports also contain explanatory free text narratives which gives the context of the patient and helps to tell the story of an AE. For all reports the AE is coded through the coding scheme Medical Dictionary for Regulatory Activities (MedDRA)[2]. The AE coding scheme enables the reports to have categorical labels that describe the AE, the labels are referred to as terms. The labels describe the presence of symptoms (e.g. nausea, dizziness, diarrhoea) different diagnosis and so-called signs.[22].

Quantitative techniques to find potential adverse drug reactions commonly look at single terms of the reports of a drug or an active ingredient. The most commonly used statistical approach, so-called Disproportionality analysis [1], tries to assess whether single terms are disproportionately high for the drug when compared to the frequency of that AE term for other drugs, consequently it is also based on the single term approach. The problem of only looking at single terms is that much of the context that these reports contain is left out since other reported terms and narratives are neglected during such analyses. Additionally, how AEs are reported varies in different ways. They can vary by how a condition presents itself, how the AE is translated to MedDRA and how a reporter chooses to describe the AE [26].

A consequence of the single term approach and the reporting variance is that many reports that describe perhaps the same AE are missed during surveillance for a potential adverse drug reaction because the reports do not contain the same term that is used in the search. In turn, the reports do not contain the same terms because of the reporting variance. If one instead assumes that each reported AE is connected to a certain topic, e.g. a group of terms describing a disorder, or the same AE reported with different terms. Then, by creating the topic and finding the corresponding connected reports yields more reports that describes a potential adverse drug reaction. The reports in the topic also produces more context by displaying the terms in the topic than the single term from a

[2]MedDRA® the Medical Dictionary for Regulatory Activities terminology is the international medical terminology developed under the auspices of the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH).

set of reports. Such an algorithm could improve the work done by UMC and other PV organizations.

The idea can also be described as a tool that finds clusters of reports that might differ in their AE terms, but the clusters describe a similar concept. This idea is similar to that of Topic Modelling in Natural Language Processing, where documents of words are assigned to a topic depending the on frequency of the words in the document. Analogous for this case is that the words are MedDRA terms and the documents are ICSRs.

The work of this thesis took place at UMC and is based on the idea that the reports are connected to a certain topic described above. The thesis is an extension of an implemented cluster analysis method created by UMC, referred to as AE cluster analysis. An application of the method, at an earlier stage, to HPV-vaccine has been published [10] and a manuscript describing the latest implementation has been submitted for publication [26]. The latest implementation is the foundation to this thesis work. In the implementation they have assume a probabilistic model that describes how a report is assumed to be generated, a so-called mixture model with statistical shrinkage [2]. The parameters are estimated with the algorithm Expectation Maximization (EM), which besides estimating the parameters of the model also assigns a probability to the reports of belonging to a topic, thereby clustering the reports. The clusters' quality in terms of stability and clinical coherence is then improved with consensus clustering (CC) [16] by running the EM algorithm several times and identifying groups of reports that tend to be grouped together by the EM [26] and then creating new clusters based on these co-occurrence patterns. Additionally, all clusters that are below a certain size are considered unclustered in order to not take into account outlying reports.

EM is known to be slow at converging [21]. The slow convergence makes the algorithm less feasible with regards to large data sets. By increasing the computational speed, the end-user experience will improve and enable further usage of the algorithm. Even though CC has been shown to increase the stability and coherence in this setting [26] there might still be room for improving the clusters' quality further. To these ends, the research questions for this thesis can be concretely described as.

1. To find, implement and evaluate techniques aimed at improving the computational efficiency of the implemented algorithm.

2. To find, implement and evaluate different approaches seeking to improve performance of the clusters in terms of clinical coherence and stability.

2 Theory

This theory section will cover the underlying theory concerning the algorithms implemented in the thesis. First, in sec. 2.1 theory concerning Bayesian statistics is introduced in order

to supply the reader with concepts that are used later on. Sec. 2.2 describes the theory of the old implementation, as well as the theory concerning suggested improvements studied in this thesis. In sec. 2.3 the performance measures for the produced clusters are presented, then in the last sec. 2.4, the old implementation is presented in detail to further understand how the probabilistic model works and how the suggested improvements can be applied.

2.1 Bayesian statistics

This section gives an overview of some Bayesian statistics in order to understand how the parameters from the mixture model are estimated and the meaning of the so-called statistical shrinkage, which is also used in the model. In sec. 2.1.1 there is a description of how to estimate parameters with maximum a posteriori estimation (MAP), which is used in the EM algorithm. Then sec. 2.1.2 presents the concept of conjugate priors, and two relevant examples of it. That section is intended to help understand the probabilistic model and explain statistical shrinkage.

2.1.1 Parameter estimation

Consider the Bayesian expression,

$$\underbrace{P(\theta|D)}_{posterior} = \frac{\overbrace{P(D|\theta)}^{likelihood}\overbrace{P(\theta)}^{prior}}{\underbrace{P(D)}_{marginal}} \tag{1}$$

where θ are the parameters in a model and $D = (e_1, ...e_n)$ are the observations. Maximum likelihood estimation (MLE) is a frequentist method to estimate the parameters of an assumed model. Conceptually, it proceeds by finding the parameters that maximize the probability of the observations, i.e. the likelihood, which can be written as an optimization problem,

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(D|\theta), \tag{2}$$

where Θ is the space of the parameters, and $\hat{\theta}$ the most probable parameters. Adapting this method to the Bayesian framework the expression can be rewritten to incorporate the model's prior as well, this yields the MAP instead which can be expressed as,

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(\theta|D) = \underset{\theta \in \Theta}{\operatorname{argmax}} \frac{P(D|\theta)P(\theta)}{P(D)} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(D|\theta)P(\theta), \tag{3}$$

with the same notation as in eq. 2, and where the last equality holds since the marginal does not depend on θ and is always positive. If the prior is a uniform distribution, i.e.

a constant, the MAP coincides with the MLE [2, p. 169]. The addition of the prior can conceptually be seen as a regularization of the parameters from the MLE, why will become clear in the examples of sec. 2.1.2 .

Solving for $\hat{\theta}$ can in some cases be given with a closed-form expression, e.g. if the distribution is known then the expected value of the distribution solves the problem. This is however not always the case, especially for MAP since the product between the prior and the likelihood, the posterior, does in many cases not result in a known distribution. In order to control the regularization as well as gaining a closed-form expression conjugate prior can be used.

2.1.2 Conjugate priors

When the posterior and the prior are in the same distribution family the prior is called a conjugate prior to the likelihood, the product of the likelihood and prior then results in a closed-form expression of the posterior. The likelihood is often determined as a consequence of the data gathering process while the prior can in some cases be left as a design choice when creating a model. Consequently, choosing a conjugate prior, when possible, is attractive since solving for $\hat{\theta}$ with MAP is straight forward when the posterior is a known distribution [2].

Two relevant examples of probability distributions and their corresponding conjugate priors are presented here, as well as the MAP and MLE solutions of the parameters in order to see the how the regularization from the priors affects the parameters, the regularization is also referred to as the statistical shrinkage.

Multinomial: Given a multinomial likelihood with the probability vector θ of length m, where each element, θ_j, of the vector corresponds to the probability of the j:th class, the conjugate prior is a Dirichlet distribution with the concentration parameter vector α, also of length m. The posterior then has a Dirichlet distribution and is expressed as,

$$P(\theta|D) = Dir(\theta|\alpha + \sum_{i=1}^{n} e_i).$$

where the interpretation of the observations are made sense in the context of the multinomial distribution, s.t. e_i corresponds to a vector of observations for all the classes and e_i^j is the j:th element in the vector and is a binary value of whether observation i is category j. Finally, n represents the total number of draws. The MLE gives the following expression of $\hat{\theta}$,

$$\hat{\theta_j} = \frac{\sum_{i=1}^{n} e_i^j}{n}, \tag{4}$$

and the MAP expression of $\hat{\theta}$ is

$$\hat{\theta_j} = \frac{\alpha_j + \sum_{i=1}^{n} e_i^j}{\sum_{j=k}^{m}(\alpha_j) + n}. \tag{5}$$

Consequently, the concentration vector α tunes the shrinkage in eq. 5. When α_j increases, and no other element in the vector, then that class in the multinomial becomes more likely as the ratio $\alpha_j / \sum_{j=k}^{m}(\alpha_j)$ then comes closer to one.

Bernoulli: Given a Bernoulli likelihood, with the parameter p, the conjugate prior is a beta distribution with the shape parameters α and β. The posterior then is a beta distribution and is expressed as,

$$P(p|D) = Beta(\alpha + \sum_{i=1}^{n} e_i, \beta + n - \sum_{i=1}^{n} e_i),$$

where the interpretation of the observations are made sense in the context of the Bernoulli distribution, s.t. $\sum_{i=1}^{n} e_i$ is the number of positive outcomes of Bernoulli draws and n is the total number of outcomes.

The MLE of the parameters gives the following expression of $\hat{p}$,

$$\hat{p} = \frac{\sum_{i=1}^{n} e_i}{n} \tag{6}$$

and the MAP expression of $\hat{p}$ is

$$\hat{p} = \frac{\alpha + \sum_{i=1}^{n} e_i}{\alpha + n + \beta}, \tag{7}$$

Consequently, the shape parameters α and β tunes the shrinkage in eq. 7. If α is increased the ratio $\alpha/(\alpha + \beta)$ is closer to one and the estimation of the parameter is thus weighted towards one.[2]

2.2 Cluster analysis

Cluster analysis is a collection of algorithms that attempts to find common structures in a dataset and cluster the data points in the set into meaningful groups. The big challenge of Cluster analysis is that there is no ground truth attached to the data, in that regard the classes are created by the algorithm. Without the ground truth there is both a question of the correct number of classes as well as how to find which class a data point belongs to, leading to an optimization problem that becomes intractable with large datasets [18]. Instead, several types of algorithms exist that approximates the optimal solution by defining an objective function, often based on distance, and optimizing for it. In broad terms the optimization is done through trying different permutations of the data's class assignment, which then optimizes the defined objective.

Cluster analysis can in general be divided into soft and hard clusterings. Soft cluster assignments give a probability to a data point of belonging to a class, and hard clusters give a binary answer of whether an instance is present in a class. Cluster analysis has various application and is used when big sets of data needs to be categorized without any information about the categories [18].

Following is a presentation of various important subfields of cluster analysis. Sec. 2.2.1 describes Topic models, which in some ways the AE cluster analysis mimics and it serves as a helpful analogy to aid further understanding. Then, theory concerning the mixture model is presented in sec. 2.2.2, and how to infer the parameters of the mixture model with EM and with the suggested improvement stepwise EM is presented in sec. 2.2.3. The theory of how to improve the cluster quality with CC is presented in sec. 2.2.4. Finally, sec. 2.2.5 describes Community detection which finds structures in graphs and can be used to perform cluster analysis on graphs [15].

2.2.1 Topic models

Topic models are probabilistic models that can be used to find the topics that a collection of documents is about. In that sense it can be used as a type of cluster analysis of documents and as a tool to categorize them by finding which topic a document belongs to. As the observations have no direct information about which topic a document belongs to, topic models analyses the statistics of the word frequency in each document in order to find the topic assignment. Driven by the intuition that a document about *Animals* contains *dog, cat, horse* more frequently than a document about *Politicians*. Such a document would instead have more occurrences of words such as *Obama, McCain*. Consequently, each datapoint is a document and the variables of it are the words, the goal is to find similarities in the frequency of words in the documents. The hope is that these similarities are able to describe the underlying theme of the documents, which is not always the case the as the algorithm might find similarities in frequency that does not correspond to a semantic similarity. [3]

2.2.2 Mixture models

A mixture model is a general framework of probabilistic models that consists of unobserved variables, referred to as latent classes[3], and observed variables. In detail the mixture models are made up of two parts; a multinomial distribution, with m categories, and m of the same distribution that describes the observations. The multinomial gives probability weights to the different latent classes. Given those classes the observations are drawn from the other distribution, e.g. a Gaussian. To that end the m additional distributions all have their own set of parameters, mean and variance in the Gaussian example. Consequently, the number

[3]The latent classes are in fact the topics described in the topic models.

of parameters, or set of them, is the same as the number of latent classes, m. Importantly, the observations contain no direct information about which latent class an instance belong to. Mixture models can in that regard be applied to the setting of cluster analysis, where the goal is to assign the instances to the latent classes, or clusters. Assigning the instances and finding the model's parameters is often done with the EM algorithm. [2, p. 423]

2.2.3 Expectation maximization

If the latent class membership is known in a mixture model it is possible to retrieve the parameter that maximizes the probability of observing the data using MLE or MAP as described in sec. 2.1.1. Similarly, if all the parameters of the mixture model's distributions are known, then it is possible to infer the class membership from the observations by calculating the expected class values. This chicken and egg problem conceptually describe the EM-algorithm. EM is an algorithm which is often applied to mixture models and is especially attractive for models of the exponential family as the expressions for inferring the parameters then are tractable [12]. After a random initialization, the algorithm consists of two steps, an expectation step (E-step) which calculates the expected latent class assignment of the entire data set given the current estimation of the parameters. The expected class values are used to calculate the so-called sufficient statistics, which are then in the maximization step (M-step) used to estimate the parameters through either MLE or MAP [2]. The algorithm then iterates between these steps until a local optimum of likelihood is found. EM is guaranteed to converge to a local optimum [39] but is also known to be quite slow at converging [21]

The convergence rate of EM is a known issue of the algorithm [12]. Additionally, calculating the expected class membership of the entire dataset becomes computationally heavier as the dataset increases in size. This fact only changes the E-step's computation time since the parameters in the M-step are calculated from the sufficient statistics. An algorithm that tries to target these problems is the stepwise Expectation-Maximization (sEM) algorithm [21, 6, 12]. The idea is to stochastically approximate the sufficient statistics with a randomly selected batch of size p from the entire dataset, size n. The expected class membership is then only calculated from the batch. It updates the sufficient statistics by stepping in the direction of the approximated sufficient statistics[4], how big the update is depends on the so-called step length [6, 21]. Another batch is chosen until the entire dataset has been used, thereby updating the sufficient statistics $\sim \frac{n}{p}$ times per epoch[5], whereas the EM's E-step updates once per epoch. Consequently, sEM results in more frequent updates to the parameters and an E-step with fewer data points to consider, while the M-step is the same as in EM.

[4] This can visually be seen as a gradient descent in the landscape of sufficient statistics.

[5] A epoch is taken from the Machine Learning literature and corresponds to when all the data has been used, not necessarily the same as an iteration.

The step length is defined as η_t and is exponentially decreasing over the number of updates, where results from stochastic approximation literature claim that $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$ are sufficient requirements in order to guarantee convergence to a local optimum [21]. Thus, choosing

$$\eta_t = (t + 2)^{-\alpha} \tag{8}$$

with any $\alpha \in (0.5, 1]$ fulfills the requirement. Thereby introducing a so-called hyperparameter in form of α, where a hyperparameter corresponds to a parameter that is set manually. The choice of p, the batch size, directly influences how much time each epoch will take as a smaller p will produce more updates.

Papers have empirically shown sEM to produce an increase in convergence speed [21, 6] as well as computational time [12], without loosing accuracy (in fact increasing accuracy for some cases [21, 12]) compared to EM. However, sEM has also been proved [12] to have a slower asymptotic convergence rate when approaching a local optimum compared to EM. The paper "Stochastic Expectation Maximization with Variance Reduction" [12] introduces an improvement of the sEM algorithm that has a higher convergence rate close the local optimum and no α parameter as instead of using η_t the step length depends on the variance of the data.

2.2.4 Consensus clustering

Many cluster algorithms have stochastic elements in them, such as random initialization, sampling of data points etc. A consequence of this is that the produced clusters will be different between different runs, e.g. different runs of EM result in different local optima. Consequently, given several clustering runs some instances will be clustered together quite often and others less frequently. Depending on how susceptible the algorithm is of stochastic elements, running a clustering model only once will produce different degrees of randomness in the clusters. CC is a collection of methods that tries to find an 'agreed upon' clustering from all the suggested clusters of several runs of the base clustering [32], base clustering is also referred to as the base model, thereby creating a final clustering. Such techniques yield more reliable clusters [16]. There are numerous kinds of CC, with variations on how to measure the similarity between clusters (and instances) as well as how to use that information to create a new clustering [32]. The theory presented here solely focuses on a single similarity measure, referred to as the similarity graph, and is presented below.

Given R different clusterings of the dataset, $D = (e_1, e_2, ..., e_n)$, the similarity measure is based on how frequently the instances are clustered together during the different clusterings. In that regard the measure can in detail be described by letting all the instances in the dataset be the nodes in a so-called similarity graph, where the value of the edges, referred to as the weights, depend on how frequently instances are clustered together. The weights then represent how similar two instances are. This results in a graph with highly

weighted edges between instances that were frequently clustered together. The weights are normalized by the total number of different clustering runs, s.t. the weights are expressed as,

$$w_{ij} = \frac{\sum_{r=1}^{R} \delta(c_i, c_j)_r}{R}, \tag{9}$$

where R is the total number of produced clusterings and c_i and c_j describe the cluster assignment of node i and j, and $\delta(c_i, c_j) = 1$ iff $c_j = c_i$ and the index r indicates the run [32]. An example of a similarity graph where $n = 6$ is visualized in fig. 1. The example shows both the created graph with all the weights written out and the so-called adjacency matrix. The matrix keeps track of the connections in the graph. Consequently, the values in e_i's row/column are all the edges and weights from node e_i.

e	e_1	e_2	e_3	e_4	e_5	e_6
e_1	0	.80	0	0	.83	0
e_2	.80	0	0	0.40	0	0
e_3	0	0	0	.92	0	1.00
e_4	0	0.40	.92	0	0.10	.85
e_5	.83	0	0	0.10	0	0
e_6	0	0	1.00	0.85	0	0

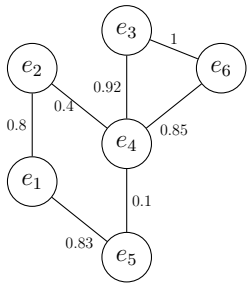

Figure 1: An example of a similarity graph with 6 nodes and its corresponding adjacency matrix. The values in the matrix are the weights in the graph.

The interpretation of this described similarity measure is an opposite distance measure since a weight with one indicates that the two instances were always clustered together and therefore very similar, or 'close' to each other. Zero indicates that two instances are never clustered together and therefore 'far' from each other. Using this similarity measure it is possible to create a final cluster over all the instances. With the similarity graph the task is to find clusters of nodes in the graph, which are then considered as the final clusters.

A simple and efficient approach of finding the clusters in the similarity graph is the so-called Single Link strategy. The strategy unfolds by manually choosing a weight threshold which creates an unweighted graph where edges whose weights are below the threshold are removed and those above it are kept in the graph. The resulting graph is then made up of several components that are disconnected if there is no edge linking them. Consequently, finding the clusters is done by finding the connected components in the graph. Continuing the example in fig. 1, choosing a threshold of 0.7 would create the two clusters, $\mathcal{C}_1 = (e_2, e_1, e_5), \mathcal{C}_2 = (e_4, e_3, e_6)$. The threshold parameter has a quite intuitive interpretation, where every edge between two reports below the threshold are not similar enough to be clustered together [16].

A less common approach of finding clusters in the similarity graph is by using community detection. The literature indicates that the more common approach is to use other types of CC on the produced clusters from several clustering runs by community detection [20, 5, 24]. However, since community detection can be used to find structures in graph and is such a promising field [15] with rapid advances happening, community detection has also been implemented as a way of finding clusters in a similarity graph. [31].

2.2.5 Community detection

Community detection is the science related to finding structures in networks, in that sense can it can be used as a clustering technique for nodes in graphs. It has risen in popularity over the last decade as the interest and availability of large-scale networks, e.g. biological, social, economical, has increased. The aim is to find clusters of highly connected groups of nodes in the graph and form communities around such groups. This is a notoriously hard problem both in terms of how to define a community as well as designing algorithms that find them [15]. The common denominator of community detection algorithms is the idea that the nodes in a community will have many edges between them, referred to as edge density which corresponds to the edges (or weights) per nodes in a community. The edge density is then the separator of communities, based on the intuition that a community will have a high edge density within the community and lower edge density between communities, illustrated in fig. 2.

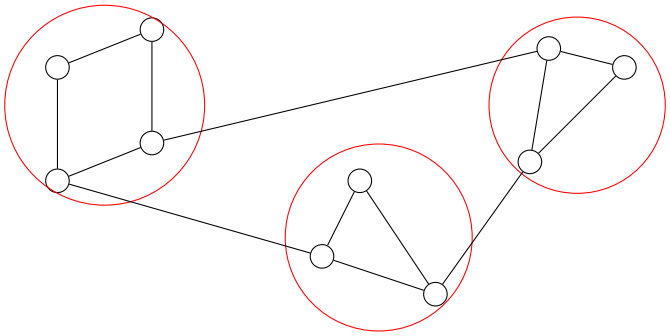

Figure 2: Graph with a single connected component but with three visually distinct communities, highlighted by the red color.

The vast landscape of community detection is much too large for the scope of this thesis. For interested readers the paper "Community detection in graphs" by Fortunato [15] gives a thorough description of the field. With that in mind, only a brief introduction and the necessary details of two related community detection algorithms are provided, the Leiden algorithm [35] and the Louvain algorithm [4]. They are conceptually similar as they find the communities by optimizing either for modularity, defined as,

$$Q = \frac{1}{2m} \sum_{i,j} \Big[w_{ij} - \frac{k_i k_j}{2m} \Big] \delta(c_i, c_j), \tag{10}$$

where m is the number of edges, w_{ij} is value of the edge, the weight, between node i and j, k_i is the number of edges of the corresponding node, also known as the degree, c_i tracks which community a node is assigned to and $\delta(c_i, c_j) = 1$ iff $c_i = c_j$. The other measure to optimize for is introduced by the same authors of the Leiden algorithm, it has the advantage that it tries to keep the communities relatively small while keeping a high edge density [34]. It is called the Constant Potts Model (CPM) and can be expressed as either a summation of nodes,

$$Q = \sum_{i,j} (w_{ij} - \gamma) \delta(c_i, c_j), \tag{11}$$

or by summing per community,

$$Q = \sum_{c} e_c - \gamma n_c^2, \tag{12}$$

where e_c is the total weights of community c, n_c is the number of nodes within community c and γ is a hyperparameter. The hyperparameter is known as the resolution parameter [34] as it can be used to design the size and the number of communities. Consequently, there is freedom to tune the output of the algorithm, γ has a straightforward interpretation where it is the threshold of inter/intra cluster edge density. In fact, given the two communities r, s where e_{r-s} denotes the total weight of the edges between r and s, it can be shown [34] that whenever $\frac{e_{r-s}}{2n_r n_s} < \gamma$, CPM will increase by separating r and s. Consequently, $\gamma = \min_{ij} w_{ij}$ will create a single community consisting of every node and when $\gamma = \max_{ij} w_{ij}$ all instances get clustered in their own community. Thus, $\gamma \in [\min_{ij} w_{ij}, \max_{ij} w_{ij}]$.

It is easier to understand Leiden and its advantage over Louvain by first reviewing how the Louvain algorithm functions. The algorithm increases the modularity by first initializing all nodes to separate communities. One iteration then consists of two steps, (1) merging and (2) aggregating. During the merging, it lists all nodes randomly[6] and calculates for every node i the change in Q by potentially merging i with a neighbour j. The merge that increases Q the most is chosen, and the merge is done only if ΔQ is positive. Those two nodes are then considered a community for the next node in the list. Note that during this process many nodes can be considered repeatedly, and that this process continues until a local optimum is found, i.e. no single movement of a node increases Q. Consequently, the first step creates a partition of the graph. Secondly, in

[6]Thus, the ordering matters but was showed to not change the outcome significantly [4].

the aggregation step the communities are created of the partition. The communities are treated as nodes in the first step again, the two steps continue until there is no increase in the modularity [4].

The Leiden algorithm, referred to as Leiden, was proposed in "From Louvain to Leiden"[35], and is an improvement of the Louvain algorithm, where one issue with the Louvain algorithm is that it can create communities that are internally disconnected [35]. Leiden solves this by also having a refinement step after the merge step. During the refinement the clusters are ensured to be connected. Leiden also has some technical changes in the implementation that increases the speed.

In the Louvain algorithm the aggregation is created from the partitions of the merge step, the Leiden algorithm instead uses a refined partition to create the aggregated network. The refined partition is found through a similar process as the merge step but with two additional rules for merging nodes. The first is that the nodes needs to be merged in the merge step, in that regard is the refinement step occurring separately on each partition. The second rule is that the nodes must be sufficiently well-connected to be merged together. After the refinement communities from the first step are often divided into smaller communities. The result of using Leiden is both better connected communities as well as better computational performance [35].

2.3 Performance measures

A well-known problem in cluster analysis is how to measure the performance of the produced clusters, a consequence of there being no available ground truth. Various methods exist that tries to capture objectively valuable features of the clusters, such as instances within a cluster being similar in terms of a defined distance [18]. In this thesis three measurements are used which try to capture different features of the clusters, all are adopted from the paper of the existing model [26]. First, in sec. 2.3.1 the number of clustered reports is presented then in sec. 2.3.2, the adjusted Rand index (ARI) is presented which is used to measure the cluster stability. Even though stable clusters are found it need not be based on a similarity that is interpretable for the viewer of the clusters, e.g. a semantic meaning in topic modelling. The instances in a cluster needs to be coherent when inspected which is what the measure term intruder detection tries to capture, presented in sec. 2.3.3.

2.3.1 Number of clustered reports

Usually in cluster analysis all the instances are clustered, however, in the paper [26] there are two criteria for an instance to be considered clustered. First, from the EM algorithm's soft cluster assignments, the probability of belonging to a class must be above a certain probability for an instance to be clustered. The second one is the that all the clusters that are below are certain size are excluded. Recall that a reason for the AE cluster analysis is that we want to find more reports connected to a AE than was possible before using other

methods based on single terms. Therefore, it is important to see how many reports gets clustered.

To that end the first measure is simply how many reports of the dataset are clustered, which is presented as a percentage out of all the reports and referred to as %Reports.

2.3.2 Adjusted Rand index

The idea of measuring stability is that if the instances are clustered together over different clustering runs the model has found a consistent similarity within the data set. The Rand index (RI) is a simple way to measure this, it is the proportions of the agreeing pairs over the total number of pairs, where agreeing pairs corresponds to both the set of pairs that are clustered in the same cluster, referred to as a, and the set of pairs that are in clustered into different clusters, referred to as b. The RI can be expressed as,

$$RI = \frac{a+b}{\binom{n}{2}} \in [0,1], \tag{13}$$

where n is total number of data points, and a and b are the agreeing pairs.

An example is provided to further understand the somewhat complicated a and b. Let the five data points $(e_1, e_2, e_3, e_4, e_5)$ be clustered two times, producing the two clusterings, $\mathcal{C}_1 = (1,1,1,2,2)$ and $\mathcal{C}_2 = (1,2,2,3,3)$, where the number indicates which cluster the corresponding datapoint belongs to. The first counter, a, is the number of pairs that are clustered together across the runs, which in this case are the pairs, $\{e_2, e_3\}, \{e_4, e_5\}$, therefore $a = 2$. The other counter is b and it represents the number of pair of elements that are not co-clustered, which in this example are the pairs $\{e_1, e_5\}, \{e_1, e_4\}, \{e_2, e_4\}, \{e_3, e_4\}, \{e_2, e_5\}, \{e_3, e_5\}$. Consequently, $b = 6$ and since there are 5 data points there is a total of 10 unordered pairs which gives $RI = \frac{2+6}{10} = 0.8$

The ARI is more complicated and it is easier to understand with a the help of a contingency table, which is expressed in tab. 1. The table represents how a dataset of size n is clustered in two different sets of clustering results $\mathbf{X}$ and $\mathbf{Y}$.

Table 1: A Contingency table of two clustering results $\mathbf{X}$ and $\mathbf{Y}$. X_i indicate the i:th subcluster and n_{ij} the number of instances that are clustered in i and j. Note that the number of clusters are not necessarily the same, highlighted by the different last indices in $\mathbf{X}$ and $\mathbf{Y}$.

$\mathbf{X}$ \ $\mathbf{Y}$	Y_1	Y_2	$\dots$	Y_m	*sums*
X_1	n_{11}	n_{12}	$\dots$	n_{1m}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2m}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rm}	a_r
sums	b_1	b_2	$\dots$	b_m	n

With the contingency table's notations $a + b$ can be simplified to linear transformation of $\sum_{ij} \binom{n_{ij}}{2}$, which we call RI' [17]. This expression is then used as the basis for calculating the ARI which is the RI adjusted for chance. Conceptually, it can be written as,

$$ARI = \frac{RI' - \mathbf{E}[RI']}{RI'_{max} - \mathbf{E}[RI']}, \tag{14}$$

where the $RI'_{max} = \frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]$ and it can be showed that $\mathbf{E}\Big[\sum_{ij} \binom{n_{ij}}{2}\Big] = \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}$ [17]. Thus, ARI is expressed as,

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \in [-1, 1], \tag{15}$$

with notations from the contingency table. A value of zero would indicate that the clustering is as stable as if the instances had been randomly allocated to clusters of the same size as those at hand. A value close to one would indicate a perfect match up to some permutation.

2.3.3 Term intruder detection

To retrieve a measurement of how coherent and 'meaningful' the clusters are the same measurement that was implemented in the original paper [26] is used in this thesis, where they in turn drew inspiration from Topic Modelling literature. Specifically, the paper "Reading Tea Leaves: How Humans Interpret Topic Models" [11] which introduces a novel approach to get a quantitative value of how coherent a cluster is by setting up tasks for a human to do, referred to as the examiner. The task is quite simple, for each produced cluster a subset of three common words in the cluster is chosen. A word with low probability in the current topic and high probability relative to the entire corpus is intruded to the

subset. These probabilities are chosen to ensure that the word is not common in the topic and that it is not an entirely uncommon word in the entire corpus. This results in four words in total with three common words from the cluster and one intruded word, such a task is created for each of the clusters in the model. One coherent and one incoherent example of what the examiner is presented with is displayed in fig. 3. The red color is not there during the task as it is there to indicate which one is the intruder.

Figure 3: Two examples of the task that is set up for the human examiner, the task is to find which word does not belong. The intruder is highlighted in red.

The examiner is tasked to find the intruded word among the four[7], consequently the idea is that if a cluster is coherent it will be easy to find the intruder, i.e. *cat* is easy to see that it does not belong. On the other hand, if it is hard to spot the intruder, such as *cancer* in the incoherent example, the connection between the words in the topic is harder for the examiner to make. Thus, a high accuracy of correctly found intruders indicates that the clusters are internally coherent and that it is possible for a human to interpret them.

The change needed to be made from topic modelling to the case of ISCR with MedDRA terms is that the examiner needs to have medical knowledge, as the produced clusters will appear random when the reader does not understand the terms and the potential link between terms. The need of domain expertise makes the measurement expensive in time and resources [26].

2.4 Existing implementation

This section first introduces how the actual data is structured and the fundamental mixture model which describes how a report is assumed to be generated. Then follows how the underlying clusters are inferred using EM and improved with CC and the Single Link strategy.

2.4.1 Mixture model

The probabilistic model of a how report is generated is a mixture model with statistical shrinkage, where the statistical shrinkage comes from using MAP in the EM algorithm..

[7]For those interested in the connection between topic modelling and kids tv shows in Sweden during the 70s, this method is very similar to *Brasses Djurlåda* in the popular show "Fem myror är fler än fyra elefanter" [25]

The statistical shrinkage is applied to ensure that a report can be reclassified to a class in which none of the current members had one of its medical events [26].

The observed variables are the MedDRA terms of an ISCR. The reports consist of observations regarding the presence of the possible MedDRA terms. The data is thus represented as,

$$\mathcal{D} = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,l} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n,1} & e_{l,2} & \cdots & e_{n,l} \end{pmatrix}, \tag{16}$$

where l is the number of different terms in the dataset, n denotes the size of the dataset and $e_{k,i}$ is a binary representation of whether a report k have reported term i. D is sparse matrix as most reports only have a few terms. The unobserved variables are the m different the classes, where the class of a report is referred to as C_k^j. The indices that are introduced here are kept throughout the section, where k indicates the report, i which MedDRA term and j the class. However, for the sake of notional abbreviation e_i and C^j corresponds to the terms and class of a single report, if not otherwise specified.

A report is assumed to be generated by two steps. In the first step a class is randomly chosen from a multinomial distribution with the probability vector θ of length m, where each element θ_j indicates the probability of that class. Then, given the class each term i is randomly drawn from a Bernoulli distribution with the parameter p^{ij}, and every term is conditionally independent given the class.

The θ is distributed as $Dirichlet(\alpha)$, where $\alpha_j = 1$ for $j = [1, 2, ...m]$. The p^{ij} is distributed as $Beta(s\phi_i, s(1-\phi_i)$, where ϕ_i is chosen with an empirical Bayes approach[8] s.t. ϕ_i is the relative frequency of term i in the dataset, and s is a hyperparameter. The generation process of a single report is graphically illustrated in 4.

[8]Empirical Bayes approach means that the parameters are chosen from the observations.

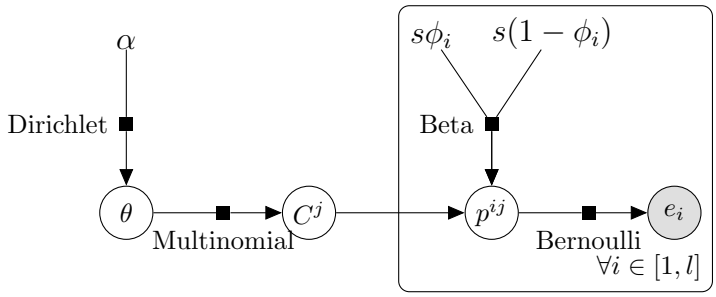

Figure 4: Factor graph visualizing the how the constructed model produces a single ICSR with reported MedDRA terms. The plate indicates that the part is repeated for each of the l terms in the dataset. The circles are the random variables in the model and the grey color indicates that e_i is observed. The observations, e_i, corresponds to the presence of the i :th MedDRA term. The black squares are the factors which indicates the relationship between the variables. The variables without a circle are the priors' parameters, which are chosen manually.

From the observed terms in a report the goal is to find the expected class assignment of the report, which is done with the EM algorithm. With the EM algorithm we also find the model's parameters $\hat{\theta}$ and $\hat{p}$ that belong to a local optimum of the likelihood [26].

2.4.2 Expectation maximization

The joint likelihood expression of a single term i in the class j of a report with index k is given by

$$\boldsymbol{\ell}(e_i, C^j|\theta^j, p^j)_k = \theta^j \big(p^{ij}\big)^{e_{k,i}} \big(\underbrace{1-p^{ij}}_{q^{ij}}\big)^{1-e_{k,i}}, \tag{17}$$

since the terms are conditionally independent given the class assignment the complete likelihood of a report k is,

$$\boldsymbol{\ell}(\underbrace{e_1, ..., e_l}_{\mathbf{e}}, C^j|\theta^j, p^j)_k = \theta^j \prod_{i=1}^{l} \big(p^{ij}\big)^{e_{k,i}} \big(q^{ij}\big)^{1-e_{k,i}}. \tag{18}$$

The total likelihood of the dataset given the parameters is given by summing the likelihood of all the reports. The total likelihood is used to see when convergence is met.

After each report is randomly assigned to a class the EM algorithm's two steps follows which are described as follows [26].

E-Step: The sufficient statistics are retrieved by summing over all reports' expected class membership which is calculated using the current parameter estimation. The expected class membership of a report is given by using Bayes' formula and the likelihood expression from eq. 18, resulting in the following expression,

$$\mathbf{E}(C^j|\mathbf{e},\theta,p)_k = P(C^j|\mathbf{e},\theta,p) = \frac{P(\mathbf{e}|C^j,\theta,p)P(C^j)}{\sum_{j=1}^{m} P(\mathbf{e}|C^j,\theta,p)P(C^j)} = \\ = \frac{\theta^j \prod_{i=1}^{l} \left(p^{ij}\right)^{e_{k,i}} \left(q^{ij}\right)^{1-e_{k,i}}}{\sum_{j=1}^{m} \theta^j \prod_{i=1}^{l} \left(p^{ij}\right)^{e_{k,i}} \left(q^{ij}\right)^{1-e_{k,i}}}. \tag{19}$$

Eq. 19 can then be used to give the sufficient statistics, which are expressed as,

$$\mu_{t+1} = \sum_{k=1}^{n} \Big[E(C^1|\mathbf{e},\theta_t,p_t)_k, ..., E(C^m|\mathbf{e},\theta_t,p_t)_k \Big], \tag{20}$$

$$\rho_{t+1} = \sum_{k=1}^{n} \begin{pmatrix} E(C^1|\mathbf{e},\theta_t,p_t)_k e_{k,1} & \cdots & E(C^1|\mathbf{e},\theta_t,p_t)_k e_{k,l} \\ \vdots & \ddots & \vdots \\ E(C^m|\mathbf{e},\theta_t,p_t)_k e_{k,1} & \cdots & E(C^m|\mathbf{e},\theta_t,p_t)_k e_{k,l} \end{pmatrix}, \tag{21}$$

where the index t indicates the iteration. The interpretation of the sufficient statistics is that μ is the expected value of the total number of reports in each class j, and ρ the expected number of reports in class j that list term i.

M-step: The θ parameter is estimated with MAP in order to achieve the statistical shrinkage. Using eq. 5 and eq. 20 we can get the expression,

$$\theta^j_{t+1} = \frac{\sum_{k=1}^{n} E(C^j|\mathbf{e},\theta_t,p_t)_k + \alpha_j}{\sum_{j=1}^{m}(\sum_{k=1}^{n} E(C^j|\mathbf{e},\theta_t,p_t)_k + \alpha_j)} = \frac{\mu^j_{t+1} + 1}{n+m}, \tag{22}$$

Consequently, the parameters shrink towards $1/m$.

Similarly the p^{ij}'s are also estimated with MAP. By using eq. 7, eq. 20, eq. 21 and realizing that the total number of occurrences in a class is given by $\sum_{k=1}^{n} E(C^j|\mathbf{e},\theta_t,p_t)_k$. The parameters are then expressed as,

$$p^{ij}_{t+1} = \frac{\sum_{k=1}^{n} E(C^j|\mathbf{e},\theta_t,p_t)_k e_{ik} + s\phi_i}{s\phi_i + \sum_{k=1}^{n} E(C^j|\mathbf{e},\theta_t,p_t)_k + s(1-\phi_i)} = \frac{\rho^{ij}_{t+1} + s\phi_i}{\mu^j_{t+1} + s}. \tag{23}$$

Consequently, the parameters shrink towards ϕ_i[26].

The value of the model parameter s and the number of classes m are set to a standard setting. The standard setting was found through a so-called cross-validation where a subset of the data is left out when running the algorithm over different combinations of s and m and then calculating the total likelihood of the subset with the final parameters.

That likelihood is then used as a measure of how well the model performed with those hyperparameters. Several runs on different drugs have shown that choosing $s = 50$ and $m = 100$ are sufficient for gaining a consistently high likelihood and therefore used as the standard setting. The algorithm runs for a fixed number of epochs, 50, a value manually chosen s.t. the total likelihood does not increase significantly after additional epochs, which is interpreted as convergence. At the very end of the algorithm, if there are any classes without any reports the classes are removed, and one additional step of EM is executed.

Many of the reports only contain a single term and using single terms for calculating the co-occurrences of reports does not add any value to the CC. In the sense that a report with just the term *Headache* might get clustered with many other completely different reports. Reports that besides *Headache* also contain terms that describe some other disorder where headache is a symptom of the disorder. Thus, all single term reports are removed from the dataset when calculating the parameters.

2.4.3 Consensus clustering

The EM algorithm is executed 100 times, and is used as the base models to calculate how often reports are clustered together, where those values are used as the weights in the similarity graph. Then the Single Link strategy with threshold 0.8 is used to produce the final clustering. The value of the threshold is chosen from empirical investigation as well as the rationale that in order for reports to be in the same cluster they need to clustered together a majority of the times which translates to a high threshold [26].

3 Method

This section gives an overview of how the work of the thesis has proceeded by giving a description of problems I found with the existing implementation, and why they are a problem. Then follows my suggested improvements and how they were implemented. It is structured into two sections, EM, in sec. 3.1, and CC, in sec. 3.2. Time improvements are suggested for both but only a cluster quality improvement is suggested in CC. In sec. 3.3 there is a short section about hardware and implementation details and at then end, in sec. 3.4 there is a description of how the evaluation of the suggested improvements is done by describing the study design. The introduced variables and notations in sec. 2.4 are kept in this chapter to not confuse the reader with additional notation and clarifying the connection between the old implementation and the new.

3.1 Improvements to expectation maximization

To solve the problem of the known slow convergence rate of the EM-algorithm emphasis is put on finding an improved version of EM, since finding another way of inferring the class

membership and retrieving the parameters might lose some of the nice features connected to the EM algorithm, such as the closed-form expressions. In that regard sEM is the first suggested improvement to decrease the computation time.

Adapting the mixture model to sEM instead of EM results in the following expression of the E-step instead,

$$\mu_{t+1} = \mu_t(1-\eta_t) + \eta_t \overbrace{\sum_{k\in B}\left[E(C^1|\mathbf{e},\theta_t,p_t)_k, ..., E(C^m|\mathbf{e},\theta_t,p_t)_k\right]}^{\mu'} \tag{24}$$

$$\rho_{t+1} = \rho_t(1-\eta_t) + \eta_t \underbrace{\sum_{k\in B}\begin{pmatrix} E(C^1|\mathbf{e},\theta_t,p_t)_k e_{1k} & \cdots & E(C^1|\mathbf{e},\theta_t,p_t)_k e_{lk} \\ \vdots & \ddots & \vdots \\ E(C^m|\mathbf{e},\theta_t,p_t)_k e_{1k} & \cdots & E(C^m|\mathbf{e},\theta_t,p_t)_k e_{lk} \end{pmatrix}}_{\rho'}, \tag{25}$$

where a new batch, B of size p is selected until the entire data set is used. Recall that $\eta_t = (t+2)^{-\alpha}$ and that p is the batch size, thus there are two additional hyperparameters α (not the same α as in fig. 4) and p. These are chosen from empirical investigation and results from other implementations of sEM [21]. Based on those sources, a p was empirically found that gives a higher convergence rate while still sufficiently speeding up the execution time of an epoch. Additionally, choosing a low α gave a higher likelihood during the empirical investigation and was thus chosen to be as low as possible, which translates to a bigger η_t in eq. 24 and eq. 25. Both parameters where, however, quite arbitrarily chosen to values that yielded a faster algorithm with no significant loss in performance than EM during development. In that regard where they chosen to $p = 0.17n$ and $\alpha = 0.51$ as standard settings for all runs.

In order to add the old sufficient statistics with the approximated, they need to be expressed as a ratio of expected class membership per batch size instead of the counts, i.e. normalize them. To see why let us take the update of μ as an example, if the μ_t had a batch size that is bigger than that of μ' then there will be more counts in certain classes because of the size and not because of the estimated parameters and observations. Consequently, the batch size would impact the update, again it is helpful to see μ' and ρ' as directions only consisting of the direction not the magnitude. Then, in the M-step the sufficient statistics are expressed as counts again as the expressions in the M-step are derived from using the entire data set.

Analysing EM and sEM it is clear that there are pros and cons with them both. EM has faster computations per epoch whereas sEM converges with fewer epochs. Additionally, with the theoretical results [12] of the convergence rate of sEM I implemented a third algorithm, referred to as *hybrid*, that first goes through a 5 epochs of sEM and then

when approaching the local optimum it changes to the faster epochs of EM and runs for 15 epochs. With an algorithm that converges faster the idea of improving computational speed is to iterate the algorithm less times while still reaching a sufficiently high likelihood.

3.2 Improvements to consensus clustering

This section is divided into two parts, one section concerning how to speed up the CC, sec. 3.2.1, and one section that tries to improve the cluster quality, sec. 3.2.2.

3.2.1 Computational efficiency

When investigating the implementation details of the CC part two potential time improvements were found. The first concerns how to acquire the weights of the graph. The old implementation calculated this serially, which is not necessary as all the different EM models are independent from each other. The suggested improvement is thus to do it in parallel instead, which requires multiproccessing hardware. Conceptually, the weights, from eq. 9, are calculated in different processes, where each process calculates the weights from a subset of all the runs, such that

$$w_{ij}^{p} = \sum_{r \in R^{(p)}} \delta(c_i, c_j)_r, \tag{26}$$

where w^p are the weights from the p:th process, and $R^{(p)}$ indicates the clusterings in the p:th process. The output from the processes are then aggregated together to create the final weighted graph, s.t.

$$w_{ij} = \frac{\sum_{p=1}^{P} w_{ij}^{p}}{R} \tag{27}$$

where P is the number of processes and R is the number of different clusterings.

The implementation creates memory constraints as each process creates a sparse adjacency matrix that is $\propto \mathcal{O}(n^2)$. By realizing that the weights of the graph are always positive is is possible to use unsigned integers instead which allows the weights to be stored with a smaller datatype than in the old implementation. Additionally, in order to not overload the data writing capability of the computer[9], only the necessary information is sent to the different processes. In detail, the class assignment is sent instead of the class probabilities which changes the datatype from a float16 to a Boolean. In order to be assigned to a class the instance's class probability needs to be above 0.5.

[9]The multiprocessing library of Python sends all information from different processes through serialization with Pickle. Thus, it is important not to send too big files between processes as it will slow down computations as well as clot the writing capabilities.

Secondly, finding the final clusters in the similarity graph was done with a brute force Python implementation, which is a problem due the nature of Python's high-level implementation. In most cases it is beneficial to use available libraries as they are implemented in C/C++, often with clever ways of speeding up the functionality in terms of computational efficiency and the algorithm.

I found the connection between the old implementation and the problem of finding the connected components in a graph. Consequently, my suggested improvement is to use the available SciPy package that implements the algorithm from "An improved algorithm for finding the strongly connected components of a directed graph" [29]. The implementation is in this thesis referred to as Pearce from the name of author of the paper. Both the suggested time improvements to the CC do not change the produced clusters as these suggestions have the same functionality in terms of outcome.

3.2.2 Cluster quality

With regards to improving cluster quality the CC part of the implementation was the focus, as the simplicity of the Single Link strategy produces some issues. For instance, consider a generic report that could be clustered into several different topics, that report could then become assigned to otherwise distinct clusters during the different runs, thereby creating an edge and merging the two distinct clusters. A scenario that would decrease how coherent the cluster are, and how much information can be extracted from them.

Additionally, when a high threshold is chosen the number of instances which are either alone or only connected to a few others becomes a higher proportion and thereby creating more clusters below the minimum cluster size and are therefore unclustered. Finally, if a lower threshold is chosen more edges are kept in the graph and therefore connect clusters unnecessarily since a single edge is enough to merge clusters. The two last points are problems concerned with refining the clusters and finding clusters with more niched information.

These are conceptual problems derived from the rationale behind Single Link, there are still benefits from the strategy that makes it valid choice for finding the clusters, especially since it does indeed increase the clinical coherence [26]. The previously described problems are, however, the driving intuition of why the clusters from Single Link might suffer. Fig 2 displays a graph that the Single Link strategy would consider one single cluster, my suggested improvement is the algorithm that is able to find three distinct clusters in that graph, community detection.

The community detection algorithm I chose is the Leiden algorithm, see sec 2.2.5, as it is proven to be fast for big datasets and to produce highly connected refined communities [35]. Additionally, it has an open source implementation [37], making it easier to implement. The problem of losing reports to the minimum cluster size will not be the same problem since no edges are removed. In fact, it might even be possible to cluster more reports than

a single EM clustering, since left out reports from a single EM run might be clustered in a different run and thereby still make connections. Another attractive feature of the Leiden algorithm is the resolution hyperparameter, γ, which provides the ability to fine tune how big the clusters are. Therefore, the implementation uses CPM and not modularity as its measure to optimize. By trying different values of γ and comparing it to the result from the Single Link strategy γ is chosen. The empirical results can be seen in fig 5, where all the experiments are done with drugs used under the development stage. The weight threshold for Single Link is 0.8 and the similarity graph is created from 100 runs of the hybrid model.

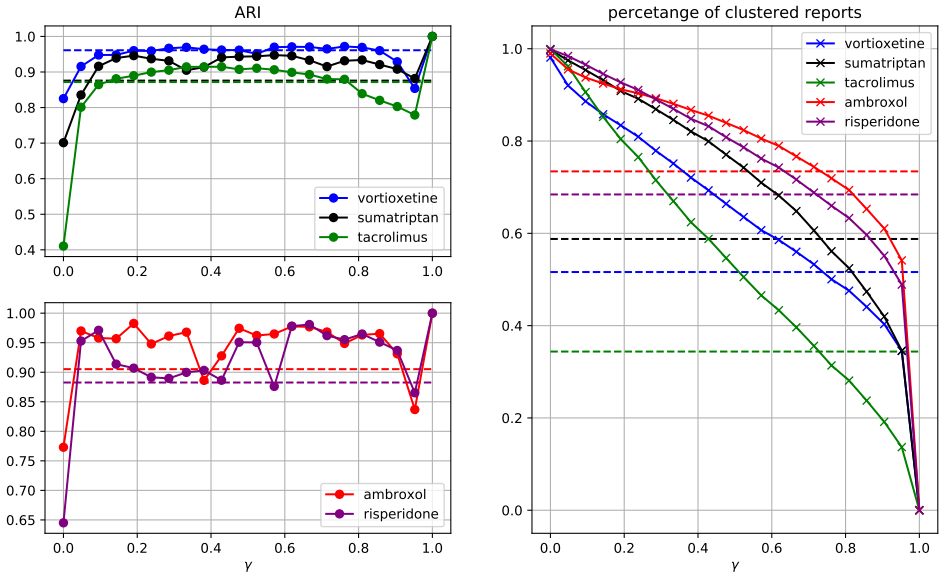

Figure 5: Plots showing different values of γ, which is the basis for setting the hyperparameter. The dashed line indicates the value from the Single Link strategy with a threshold of 0.8. The drugs are chosen from previous implementations of the algorithm. All runs are made with hybrid as the base model.

The figure tells us that it is possible to get a higher ARI score while still clustering more reports than Single Link does. In order to achieve both attractive features a $\gamma = 0.35$ is chosen. A value that clusters significantly more reports than Single Link.

A problem of the algorithm is that it, compared to Single Link, is computationally expensive. Running the algorithm will be more time consuming as well as having higher memory requirements. However, with the changes made to how the weights are acquired

and by using hybrid as the base models there might still be room for allowing an algorithm that takes more time without actually becoming slower than the existing implementation.

3.3 Implementation

The implementation of this thesis is mainly done with Python and the Python packages *igraph* [13] for the Leiden algorithm, *sklearn* [30] for the ARI, *SciPy* [38] for the connected components and together with *NumPy* [27] for the calculations. The data is extracted from VigiBase using SQL queries. Finally, the hardware which produces all results is an Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s) with Total Physical Memory 31,7 GB.

3.4 Study design

In order to see how the suggested improvements is affected by the size of the dataset, n is the main choosing criteria. First, all drugs that had already been used during previous research or during the initial trial and error process were excluded from the list of potential drugs. The excluded drugs are

- Vortioxetine
- Ipilimumab
- Nivolumab
- Pembrolizumab
- Escitalopram
- HPV vaccines
- Haloperidol
- Nintendanib
- Methylphenidate
- Desogestrel
- Clonazepam
- Bevacizumab
- Levothyroxine
- Risperidone
- Ibuprofen
- Irinotecan
- Celecoxib
- Ritonavir
- Nifedipine
- Digoxin
- Glimepiride
- Valsartan
- Levonorgestrel
- Sumatriptan
- Ambroxol
- Tacrolimus

The aim for the selection is to find six drug sets in increasing size which had not been used before. The smallest four drug sets are chosen randomly from manually chosen intervals; the intervals increase by a factor of $\sim$2 s.t. the different intervals are

$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$
10,000-12,000	20,000-25,000	50,000-60,000	100,000-120,000

As the size of the drug sets increases the number of available drugs in VigiBase decreases, thus also the possibility to choose an interval and randomly selecting a drug set from it. The two largest drug sets are instead chosen by manually looking through VigiBase and creating two groups that contain drug sets whose size almost follows the increase of ~ 2. The groups and their sizes are

Group 1 $\sim 200,000$	Group 2 $\sim 500,000$
Influenza vaccine	Etanercept
Pneumococcal vaccine	Adalimumab

Then, from the drugs in the intervals and the two manually produced groups a random draw is made and all the single term reports are removed, which gives the following list of drugs with their corresponding sizes,

Drug	Cefixime	Dupilumab	Furosemide	Ciprofloxacin	Pneumococcal vaccine	Adalimumab
Size,	3436	10998	21254	45688	150304	277603

Because of the high memory requirement of the Leiden algorithm, Pneumococcal Vaccine and Adalimumab can not be executed on the available computer in the current state of the implementation. Hence, the results of the attempted improvement to the clusters' quality is only presented with the smallest four drug sets. Additionally, only three are chosen to evaluate the clinical coherence by the term intruder detection, due to the term intruder detection's cost. The randomly selected subset is

- Cefixime
- Furosemide
- Ciprofloxacin

In all the experiments the hyperparameters are all set to the standard values introduced in the suggested improvements and the existing implementation. To get an easy overview, a summary is presented here.

Mixture Model			**EM**	**hybrid**			**Single Link**	**Leiden**
m	s	runs	Epochs	α	p	Epochs	Threshold	γ
100	50	100	50	0.51	$0.17n$	20	0.8	0.35

4 Results

The results are presented in a structure that mimics the method section. First in sec. 4.1 the results of the sEM are presented, in terms of the change in the convergence rate, the impact of the hyperparameters and then to see the impact of the base model on the CC with the Single Link strategy the ARI and %Reports are presented.

Then in sec. 4.2 the results from the changes to the CC part is presented, first the speed up then in terms of cluster quality.

When comparing the execution time the speed up factor is also presented, which simply is $\frac{T_{old}}{T_{new}}$.

4.1 Expectation maximization

In fig. 6 the change in convergence rate is shown by how the normalized log likelihood tapers off while the number of epochs increases, remember that hybrid is run for 20 epochs while EM for 50. The axes are all normalized with respect to the number of reports in order to make a fairer comparison between the drugs. The percentage of clustered reports over the epochs is also presented. The plots represent a single run of each algorithm using a fixed seed.

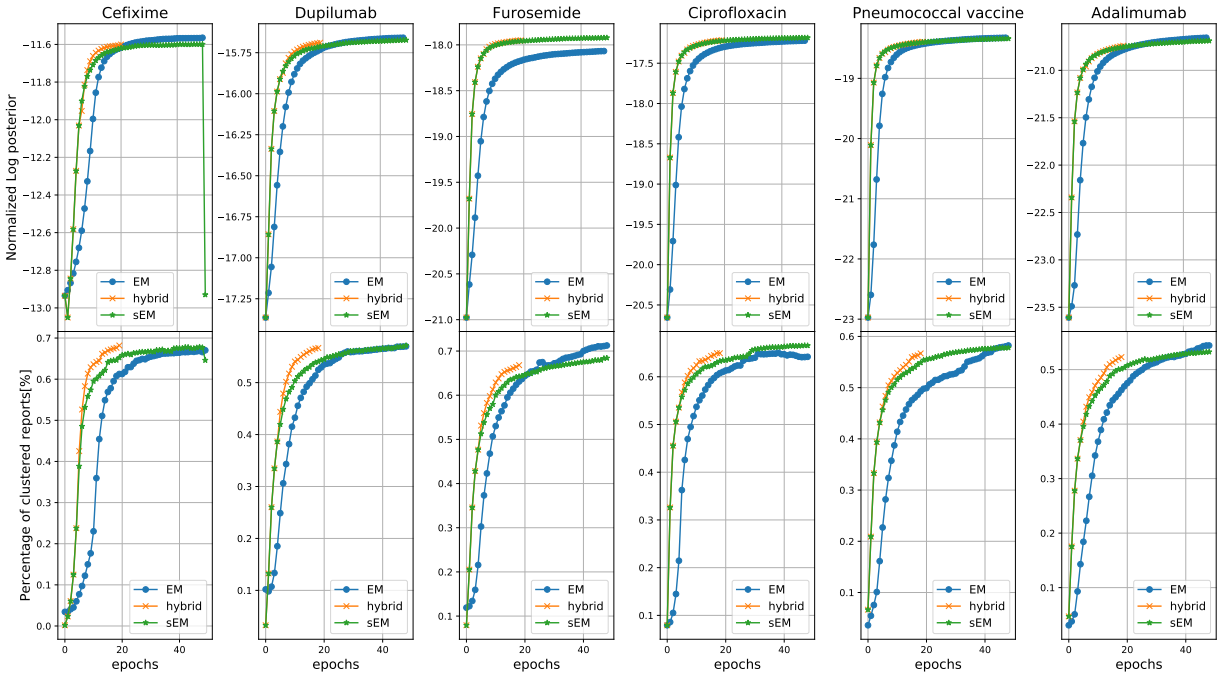

Figure 6: Plots showing the change to the convergence rate of the likelihood, in the first row. In the second row the percentage of clustered reports is presented. One tick of the x-axis corresponds to an entire epoch.

The plots in fig. 6 show that increase in likelihood and % Reports increases faster with hybrid and sEM than EM for each of the drugs. The final values after 20 epochs of both measures with the hybrid is similar to the final values of EM after 50 iterations.

Fig. 7 provides a further investigation regarding the impact of the hyperparameters, α and p. Each drug is represented with the normalized loglikelihood after 20 epochs and the time taken as function of p, p is presented as a percentage of the entire dataset. Each square in the heat map indicates a single run, with a fixed seed, and the time is the mean over the different α's given a p, i.e. the mean time along the columns of the heat map. A higher likelihood after the same number of iterations is interpreted as a higher convergence rate since a higher likelihood after fewer epochs indicates that the algorithm arrived at the optimum sooner. The red line in the color bar corresponds to the value from the EM model. The red line is not present in Cefixime's color bar as it was too high to be captured, a value of -11.56.

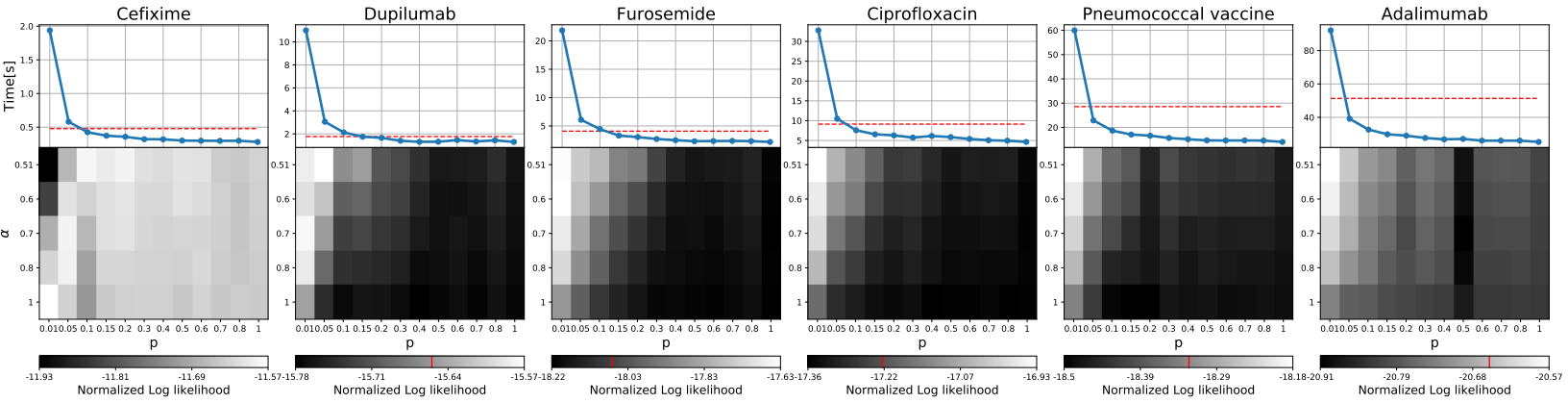

Figure 7: Plots showing how the hyperparameters α (controls the step length) and the p (the batch size) affects the likelihood and the computational time after 20 epochs. The red line indicates the value from the EM algorithm after 50 epochs. It is not present in Cefixime as the hybrid algorithm never reached that value, -11.56.

The heat maps and plots of fig. 7 shows that using a smaller batch size generally produces a higher likelihood after 20 epochs and that a low α is in general better for the final likelihood. It is also visible how the execution time is heavily dependent on p and that choosing a too small batch size would make the execution time longer using the hybrid.

Using the standard setting of the hyperparameters, see sec. 3.4, the impact on the clusters' quality in terms of ARI and %Reports is presented in tab. 2. The values are taken from the mean over ten comparisons all containing 100 runs. Creating a total mean over 1000 different runs all with different seeds. It is structured in this way since the CC

uses 100 different runs to create the similarity graph. Each drug then has ten comparisons to make of the CC part as well.

Table 2: The ARI and %Reports from the EM and hybrid produced clusters.

Drug		**ARI**		**%Reports**	
Name	n	EM	hybrid	EM	hybrid
Cefixime	3436	0.68	0.64	86	87
Dupilumab	10998	0.52	0.47	82	79
Furosemide	21254	0.58	0.50	91	89
Ciprofloxacin	45688	0.54	0.46	91	88
P. vaccine	150304	0.40	0.37	86	84
Adalimumab	277603	0.34	0.33	82	79

Tab. 2 shows that the stability measure is lower with all of the drugs when running with hybrid instead of EM. The %Reports also decreases, but only slightly.

However, since the final clusters are built from the CC the more important metric is the impact that the different base models have on the final clusters from CC. These results are presented in tab. 3, and are also run with 10 comparisons with 100 base models. However, since the base models are used to build the final graph the mean is only taken over 10 comparisons, with the exception of Adalimumab which only has 4 comparisons as one CC takes around 13 hours (see tab. 6).

Table 3: The impact on the CC's ARI and %Reports when the base model is hybrid and EM, the underscore indicates the higher value.

Drug		**ARI**		**%Reports**	
Name	n	EM	hybrid	EM	hybrid
Cefixime	3436	<u>0.95</u>	0.88	70	69
Dupilumab	10998	<u>0.94</u>	0.84	52	47
Furosemide	21254	0.71	<u>0.85</u>	60	55
Ciprofloxacin	45688	0.82	<u>0.89</u>	65	58
P. vaccine	150304	0.57	<u>0.70</u>	64	58
Adalimumab	277603	<u>0.59</u>	0.56	52	45

Tab. 3 shows that the ARI changes from drug to drug and that the consistent stability decrease from hybrid in tab. 2 is not present with the CC. The %Reports is still consistently lower.

Finally, to view the actual difference in time tab. 4 presents the speed up from running hybrid and EM each with 100 runs. The results are presented in terms of computational time as well as the speed up factor, referred to SUF. The time is calculated from the mean over ten comparisons.

Table 4: The change in computational time for creating 100 base models with the EM and the hybrid . SUF is the speed up factor and the computational time is given in seconds or minutes.

Drug		**Time[s]**			**Intuitive units**	
Name	n	EM	hybrid	SUF	EM	hybrid
Cefixime	3436	30	21	1.40	30 s	21 s
Dupilumab	10998	90	63	1.41	1.5 min	1 min
Furosemide	21254	180	124	1.44	3 min	2 min
Ciprofloxacin	45688	360	245	1.47	6 min	4 min
P. vaccine	150304	1166	732	1.59	19 min	12 min
Adalimumab	277603	2082	1138	1.83	35 min	19 min

Tab. 4 shows that the execution time for all drugs is sped up with the use of hybrid and it shows that the speed up is increasing as the dataset increases in size.

To further aid with how the computational time depends on size of the data, fig. 8 tells a visual story about the complexity.

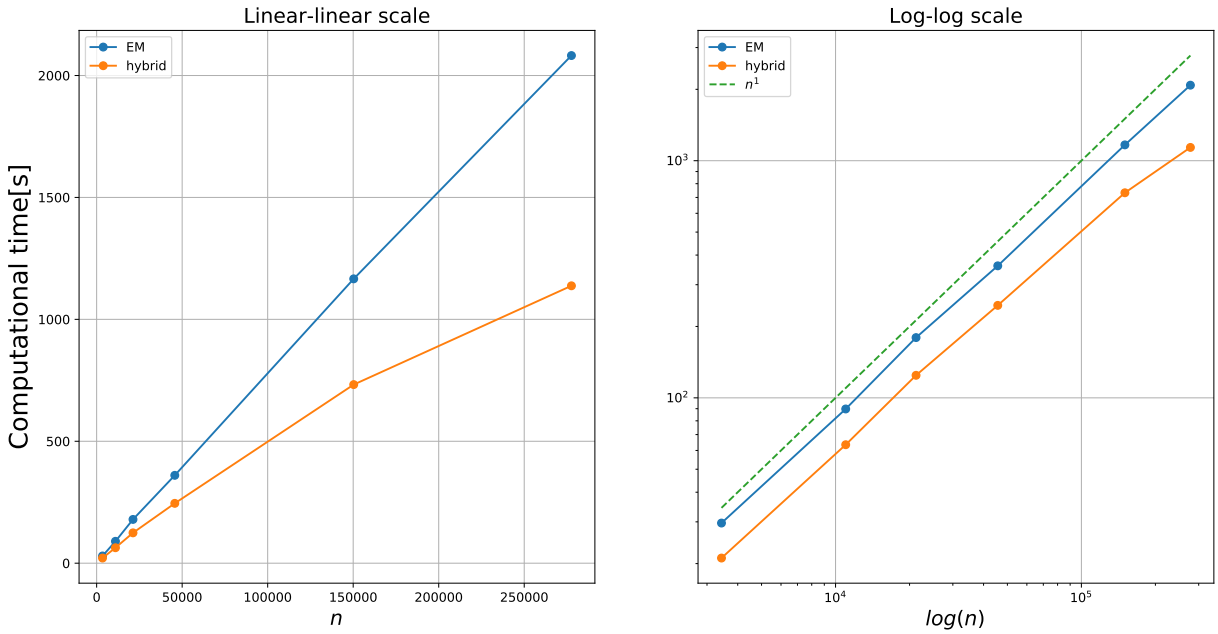

Figure 8: Two plots showing how the execution time depends on the size of the data. In the right plot a 'guide' line is presented that shows a linear increase.

The plots in fig. 8 shows that the hybrid's execution time increases at a different rate than the EM's. Looking at the logarithmic scale one sees that the slope of the curve is close to, indicating an almost linear computational complexity.

4.2 Consensus clustering

This part is divided into two sections, first regarding the computational efficiency and then the cluster quality improvement. The base model for all results here is hybrid.

4.2.1 Computational efficiency

Tab. 5 presents the two suggested speed up improvements of the CC, the parallelization and using the SciPy implementation. These are referred to as Acquiring weights and Connected Components. Acquiring weights corresponds to creating the similarity graph from 100 base models and Connected Components is finding the connected components once the edges below the threshold are removed. The comparison is done only over those two functionalities. Any additional code that is needed to change the structure of the data is not taken into account. In the parallel implementation all the runs are made with the maximum number of available cores in the computer, eight. With the exception of Pneumococcal vaccine as running on eight cores took too much memory and is instead executed with three cores. Adalimumab was not possible to execute for any number of

cores. In fact, it was not possible to execute with the old implementation of the algorithm either. The implemented changes of datatype management were needed in order to execute Adalimumab. Thus, the comparison is not completely the same as of the other drugs. The time is calculated as the mean over 10 ten comparisons except for Adalimumab which uses only 4, due to time concerns. This basis for calculating the time remains the same for all the following results in this section.

Table 5: The time improvement with the suggested changes. Serial is the time for creating the similarity graph serially, the old implementation, and parallel the suggested improvement. Brute Force corresponds to the old implementation of finding the connected components, and Pearce is the SciPy implementation of finding the components. The * indicates that it was run using 3 cores instead of 8, the ** indicates that changes were made to the original implementation in order to get a result.

Drug		**Acquiring Weights[s]**			**Connected Components[s]**		
Name	n	Serial	Parallel	SUF	Brute Force	Pearce	SUF
Cefixime	3436	6	11	0.55	0.52	0.01	44.25
Dupilumab	10998	23	15	1.47	5	0.10	55.04
Furosemide	21254	74	30	2.43	16	0.3	53.68
Ciprofloxacin	45688	364	118	3.10	74	1	61.33
P. vaccine	150304	2384	1601*	1.49	922	10	95.80
Adalimumab	277603	48134**	–	-	3420	39	87.52

By first reviewing the Pearce implementation in tab. 5 one sees that there is a speed up with all of the drugs and that the speed up is not strictly dependent on the size of the data. In the other part, acquiring the weights, there is a speed up, although this speed is dependent on the size of the dataset.

Combining these two results from tab. 5 gives almost the entire time of the CC part. It is not the complete time as there is some additional code between the two functionalities that changes the data type and other minor adjustments. In tab. 6 the total speed up of CC is presented, the times between tab. 5 and tab. 6 does not match up as a consequence of the additional code. Consequently, the columns represent the entire time of CC, i.e. producing a final cluster from the different base models. The changes to Pneumococcal vaccine and Adalimumab in tab. 5 are the same in this table, as well as the number of comparisons.

Table 6: The time improvement with the suggested improvements. Original is the old algorithm in its original form, Pearce indicates using the improved way of finding the connected components, Serial represents finding the weights serially and Parallel represents acquiring the weights in parallel. The * indicates that it was run using 3 cores instead of 8, the ** indicates that changes were made to the original implementation to get a result.

Drug		**Original**[s]	**Serial + Pearce**[s]		**Parallel + Pearce**[s]	
Name	n	Time	Time	SUF	Time	SUF
Cefixime	3436	6	6	0.99	11	0.55
Dupilumab	10998	27	21	1.28	15	1.76
Furosemide	21254	95	80	1.19	44	3.03
Ciprofloxacin	45688	416	342	1.22	112	3.71
P. vaccine	150304	3141	2521	1.25	1582*	1.98
Adalimumab	277603	51577**	48196	1.09	-	-

Tab. 6 describes that the increase of finding the connected components with the Pearce algorithm gives a speed up ~ 1.2, in all cases except for the smallest and largest drug set. Acquiring the weight in parallel as well as running the Pearce algorithm gives a bigger speed up but is even more dependent on the size.

As an additional visualization of the increase in time as the number of reports increases fig. 9 is presented. It has both a linear and logarithmic scale of the increase in n, the number of reports. In the logarithmic plot there is a guiding line with n^2 to see the complexity of the algorithm. The right most plot shows the increase without Adalimumab. There is no value for Adalimumab for Updated as it was not executed in parallel and would therefore not give a fair continuation of the trend.

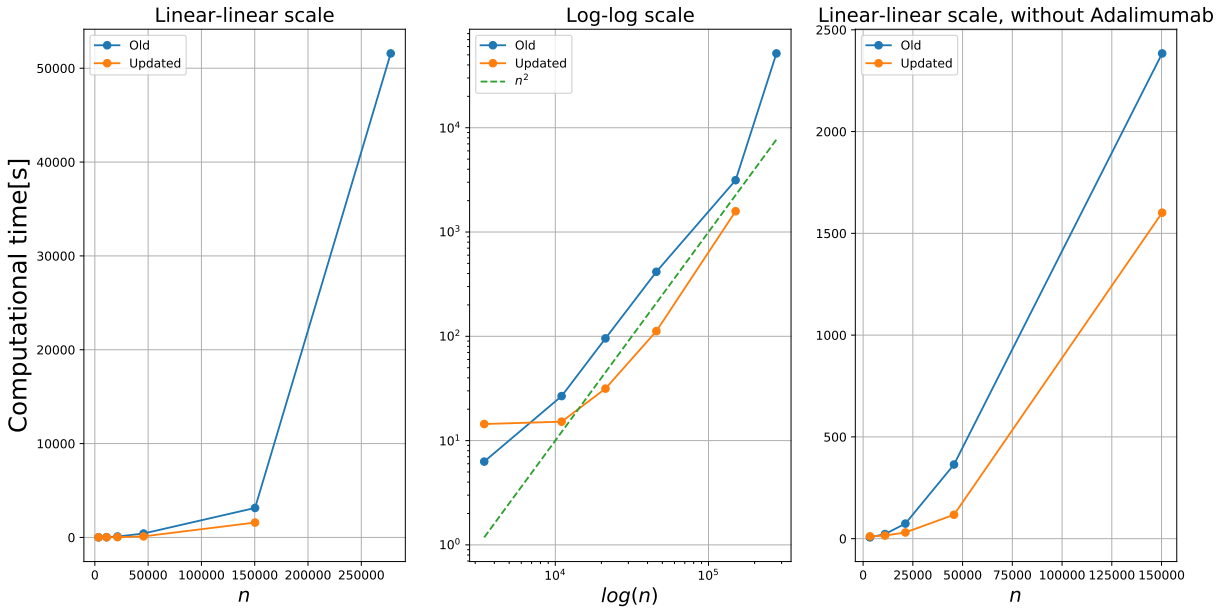

Figure 9: The execution time when executing the entire CC implementation, acquiring weights and finding components, with both the updated version and the old version. The middle figure displays a 'guide line' as well which traces a n^2 curve in a logarithmic setting.

The plots in fig. 9 shows that the complexity for both the old and the new is about $\mathcal{O}(n^2)$ and that Adalimumab does not follow that complexity pattern.

With all these speed ups it is possible to present the complete speed up of all the suggested improvements. The changes to Adalimumab and Pneumococcal vaccine in the CC are the same as before, there is no changes to EM or hybrid. The time is given in more intuitive units than the previous tables in this section for easier interpretation of how the improvements actually change the execution time.

Table 7: The total time improvement with all the changes. Original is the existing implementation which uses EM, serially acquired weights and the brute force way of finding the connected components. With the exception of Adalimumab which uses the new memory management. Updated is then all the suggested improvements with hybrid, weights acquired in parallel and the Pearce algorithm to find the connected components. Pneumococcal vaccine is run with 3(*) cores. Adalimumab acquires the weights serially in the CC(***) and uses the updated memory management in the original implementation (**). The units are presented in intuitive units to easier interpret the how long the runs take.

Drug		**Total speed up**		
Name	n	original	updated	SUF
Cefixime	3436	36 s	27 s	1.1
Dupilumab	10998	2 min	1 min 30 s	1.48
Furosemide	21254	6 min	3 min	2.13
Ciprofloxacin	45688	13 min	6 min	2.17
P. vaccine	150304	1 h 12 min	39 min*	1.86
Adalimumab	277603	14 h 54 min**	13 h 42 min***	1.09

From tab. 7 it is possible to see how the speed up is dependent on the size and that the significance of hybrid's increase in speed up fades away as the dataset increases. The outcome for Adalimumab gives a smaller speed up, however, the gain in absolute time is still the greatest.

Finally, to see the impact the Leiden algorithm has on the execution time tab. 8 presents those results. The table presents the time for finding the clusters when the similarity graph is created, how much time the entire CC takes and finally the time to run the entire old implementation and the new with Leiden instead of Pearce to find the final clusters.

Table 8: The impact on time of the four smallest drug sets that the Leiden algorithm is applied on. Finding the clusters refers to finding the clusters when the similarity graph is created, CC corresponds to the time for acquiring the weights and finding the clusters and finally With base model corresponds to the entire execution time from every part. In terms of abbreviations B.F is brute force way of finding the connected components; S is finding the weights serially and P is in parallel.

Drug		**Finding the clusters[s]**		**CC[s]**		**With base model[s]**	
Name	n	B.F	Leiden	S+B.F	P+Leiden	EM+S+B.F	hybrid+P+Leiden
Cefixime	3436	0.52	2	6	13	36	34
Dupilumab	10998	5	3	27	32	116	95
Furosemide	21254	16	78	95	107	275	231
Ciprofloxacin	45688	74	406	416	519	776	764

Tab. 8 shows that Leiden takes more time to execute than the old implementation of finding the clusters. Even with the faster way of acquiring the weights, the Leiden still takes longer time, only when the speed up from the hybrid is considered Leiden is able to execute faster.

4.2.2 Cluster quality

The objective measures of cluster quality are presented in tab. 9, which shows how the different strategies of Leiden and Single Link impacts the performance measures. The number of clusters are also shown here in order to highlight the refinement. The values are taken as the mean from ten different comparisons.

Table 9: Comparison of the performance measure between Leiden and Single Link (S.L) strategies.

Drug		**Clusters**		**%Reports**		**ARI**		**Term intruder**	
Name	n	S.L	Leiden	S.L	Leiden	S.L	Leiden	S.L	Leiden
Cefixime	3436	25	39	69	86	0.88	0.97	0.87	0.74
Dupilumab	10998	37	97	47	72	0.84	0.94	-	-
Furosemide	21254	79	196	55	77	0.85	0.93	0.86	0.71
Ciprofloxacin	45688	139	288	58	84	0.89	0.91	0.87	0.77

Tab. 9 displays how there are more clusters, the % Reports increases and that the

stability increases when using Leiden instead of Single Link. However, the measure for clinical coherence is consistently lower with Leiden.

In order to see whether the minimum cluster size has an impact on the term intruder score fig. 10 shows the recalculated score with more clusters removed.

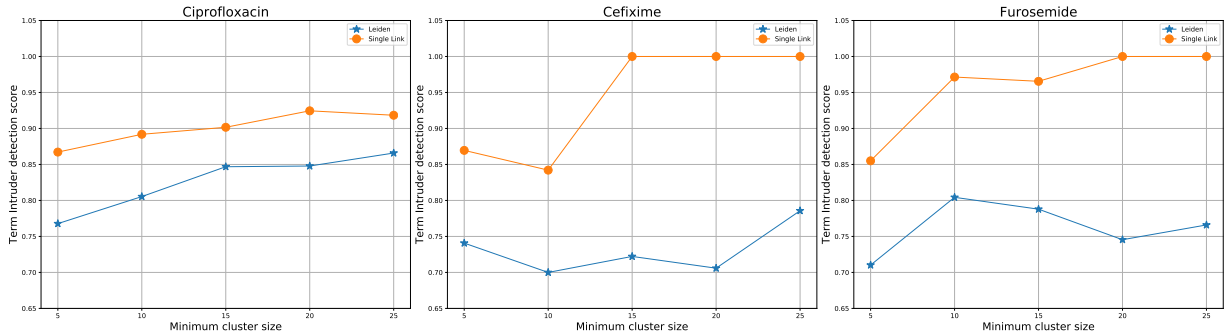

Figure 10: Graphs that show how the term intruder detection score changes as the minimum cluster size increases

The results from the graphs in fig 10 shows that the minimum cluster size does give both strategies a better result but it does not reduce the difference between Leiden and Single Link strategy.

In order to see what the clusters actually looks like as well as displaying how Leiden in some cases was able to refine and find additional reports four examples are provided. They are chosen manually by reviewing the output of the two strategies and are therefore not necessarily representative of the algorithm. The clusters are presented by providing statistics about the frequencies of the terms. The percentage refer to how common that term is in the cluster, only terms with a frequency above 5 % are displayed.

Cefixime: Fig. 11 shows an example found in the final clusters of Cefixime. One cluster by the Single Link strategy and two clusters from the Leiden strategy. The intersection of the reports from Single Link and Leiden is 936, thus Leiden clusters $\frac{936}{939}$ reports of the Single Link cluster and also clusters 87 additional reports.

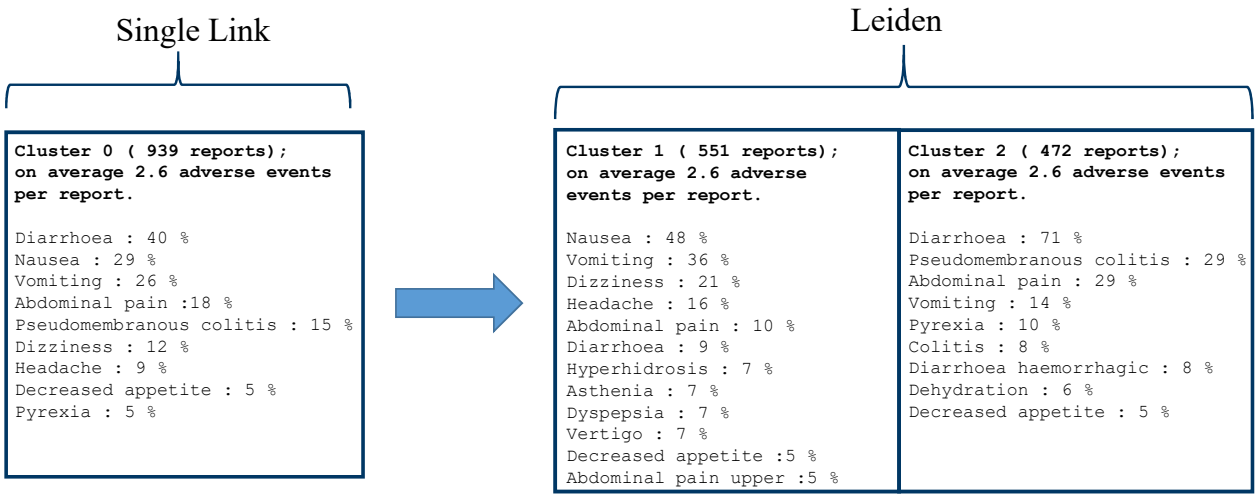

Figure 11: The produced clusters from Single Link strategy and the Leiden strategy. The percentages refer to how common the term is in the cluster.

Fig. 11 shows how Cluster 0 of Single Link is separated into Cluster 1 and Cluster 2 by the Leiden strategy, while keeping most reports from the Single Link strategy.

Furosemide Fig. 12 shows an example found in the final clusters of Furosemide. One cluster by the Single Link strategy and three clusters from the Leiden strategy. The intersection of the reports from Single Link and Leiden is 2021, thus Leiden clusters $\frac{2021}{2162}$ reports of the Single Link cluster and also clusters 534 additional reports.

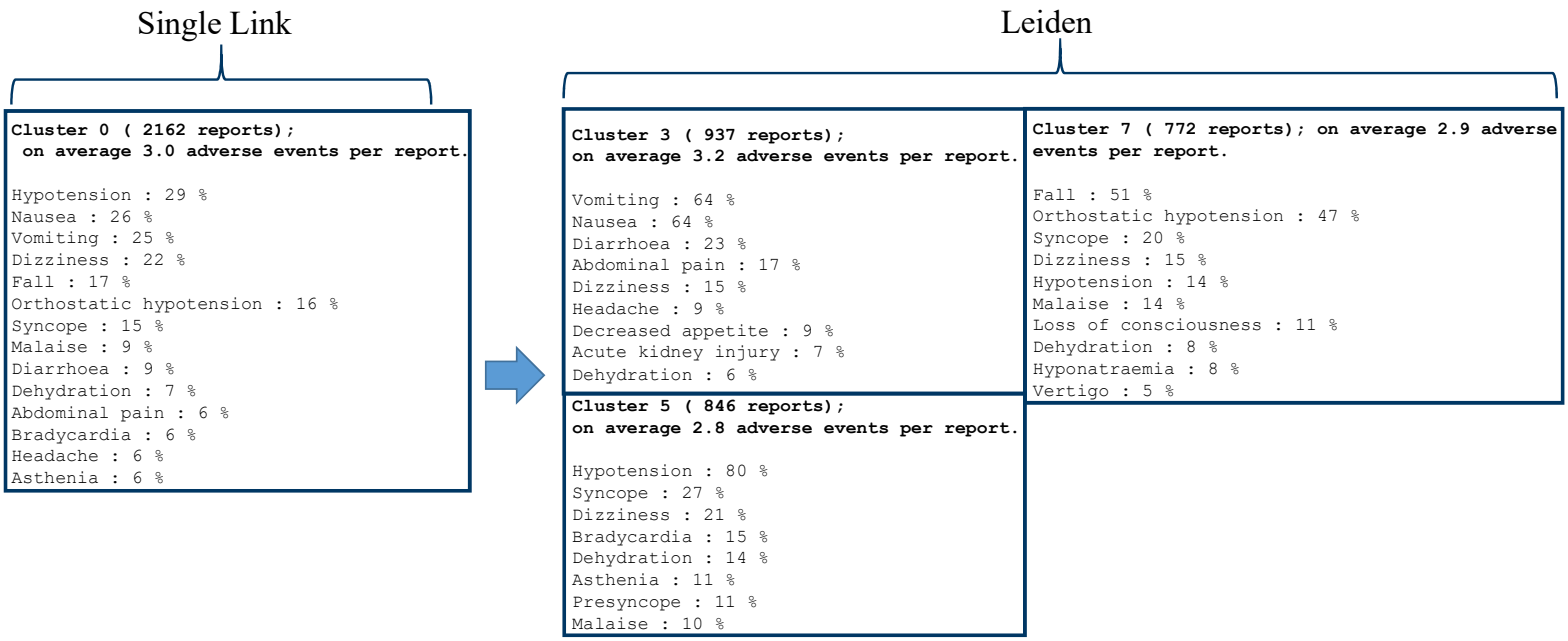

Figure 12: The produced clusters from Single Link strategy and the Leiden strategy. The percentages refer to how common the term is in the cluster.

Fig. 12 shows how Cluster 0 of Single Link is separated into Cluster 3, Cluster 5 and Cluster 7 by the Leiden strategy, while keeping most reports from the Single Link strategy.

Ciprofloxacin Fig. 12 shows an example found in the final clusters of Ciprofloxacin. One cluster by the Single Link strategy and three clusters from the Leiden strategy. The intersection of the reports from Single Link and Leiden is 2875, thus Leiden clusters $\frac{2875}{2929}$ reports of the Single Link cluster and also clusters 1063 additional reports.

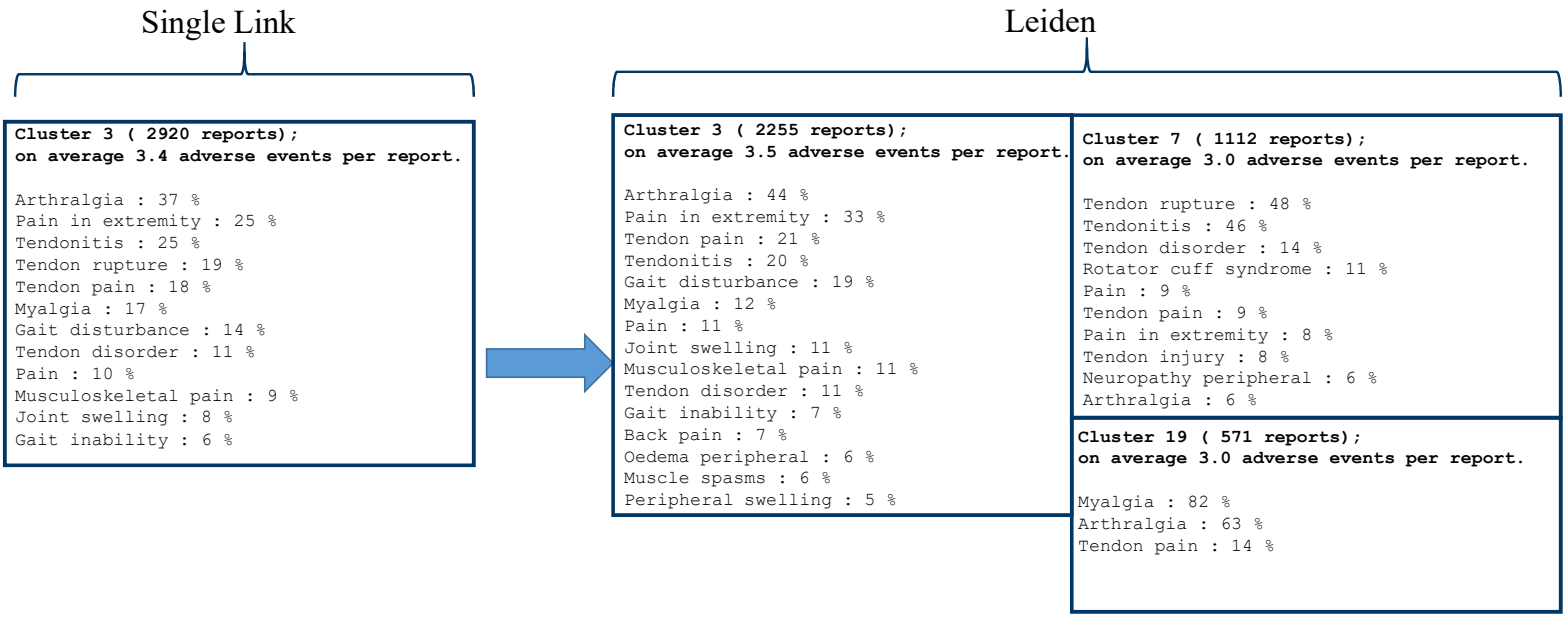

Figure 13: The produced clusters from Single Link strategy and the Leiden strategy. The percentages refer to how common the term is in the cluster.

Fig. 13 shows how Cluster 3 of Single Link is separated into Cluster 3, Cluster 5 and Cluster 19 by the Leiden strategy, while keeping most reports from the Single Link strategy.

The other aspect that Leiden aimed to achieve is to find more reports connected to a topic than Single Link. In fig. 14 two corresponding clusters were found in Single Link and Leiden with Dupilumab, Leiden found 171 additional reports whereof five randomly selected reports are displayed.

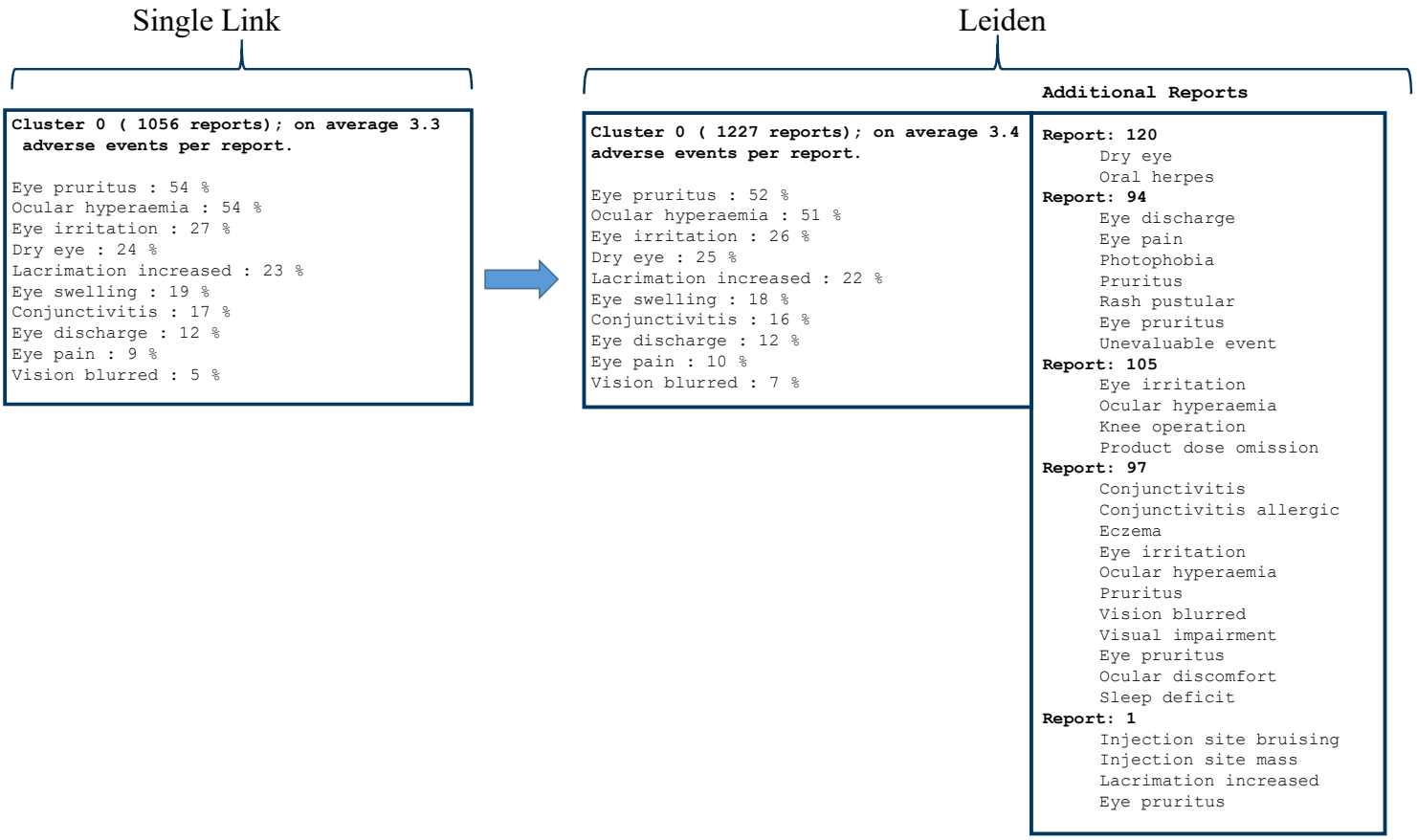

Figure 14: Two clusters connected to the same topic where Leiden found 171 more reports, where 5 are randomly chosen. The percentages refer to how common the term is in the cluster. The model was executed with Dupilumab.

The additional reports in fig. 14 all contain at least one similar term as the common ones in the Cluster 0 by the Single Link and the Leiden.

5 Discussion

Before diving into the details of the results a short summary of how to interpret the big picture outcome is presented. Then follows a discussion of the specific results which mimics the structure of the results section. Finally, a discussion of how the work can be applied to work done at UMC and other PV centres is provided.

The results show that the suggested improvements are able to speed up the execution by a factor of ~ 2, if the dataset is not too big nor small, without a significant loss in cluster quality. These results also give room for using the suggested cluster quality improvement

with the Leiden strategy instead. Improving the quality with Leiden is both successful and unsuccessful, the objective measures of stability and how many reports that are clustered perform better. Unfortunately, this trend does not continue with the clinical coherence score, there are, however, reasons to further investigate why this score is lower since there are also examples of Leiden finding clusters that provides more precise information while apparently retaining clinical coherence.

The appearance of the likelihood curve in fig. 6 mimics the results presented in the papers [21, 12], indicating a successful implementation of the sEM algorithm. The change in convergence speed is greatest during the first epochs, where sEM have great changes of the likelihood and the EM slowly increases and then they behave similarly approaching the optimum.

One could also argue for EM reaching a similar likelihood after 20 or 30 epochs, thus questioning the need for hybrid with 5 epochs sEM as running EM for 20 epochs would give an even greater speed up. However, by investigating the %Reports it is apparent that EM needs many more iterations to reach the same number of clustered reports as achieved by the sEM and especially hybrid. With that noted running fewer epochs of EM could have been an improvement that also would have decreased the computational time without losing too much performance as after around 30 iterations not much gain in likelihood is made.

Regarding the hybrid model it achieves the wanted functionality. It first gains the faster convergence rate of the sEM algorithm and finds the local optimum with the final EM steps. The theoretical results of sEM having a slower convergence close to local optimum [12] can also be seen in the convergence rate especially with Cefixime and Dupilumab. The faster convergence rate may also explain that the %Reports of hybrid increases significantly faster than sEM's %Reports. Thus, even with a sEM implementation that had a similar time per epoch to that of EM, these results would still indicate that it is more valuable to use the hybrid model instead.

Going forward there are of course areas of improvement. Firstly, choosing the hyperparameters α and p creates more variance and choices for the user to make. Implementing the variance reduction sEM [12] would remove the α and maybe even increase the convergence rate as it has a higher asymptotic convergence rate than both EM and sEM. Secondly, the normalization is currently implemented in a way that works but can be made more efficiently by realizing how to normalize the $E(C^j|\mathbf{e}, \theta, p)$ over the batch size. By doing that there is no need to normalize the old and the approximated sufficient statistics. However, at the M-step, the sufficient statistics still needs to be scaled back to the counts in order to retrieve the correct parameters.

Convergence is reached for all drugs, less so with Adalimumab where EM still has a noticeable slope after 50 iterations. This can be explained through the choice of number of

classes, since the Adalimumab is almost 100 times bigger than Cefixime it is reasonable to assume that there might be more topics in the dataset and therefore choosing 100 classes is not sufficient. This is in fact also the case when running the cross-validation, which indicates the number of classes ought to be 200.

In the case of Furosemide, the reached altitude of the likelihood is clearly higher than that of EM, this is most likely explained by sEM and hybrid finding a different local optimum. Given another random seed EM might find it as well, this is important to note as there had been speculation in the papers, that introduced sEM, of increasing accuracy with sEM. A similar argument can be made with Ciprofloxacin. There is one big deviance in the convergence, by the sEM of Cefixime. After finding an optimum, at the very last step the drug's likelihood drops to a level of likelihood similar to the random initialization. It happens after the extra step that the algorithm takes after it has removed classes with no members, which is also implemented in the sEM. This could be explained by sEM not working with regards to that extra step. Another explanation might be a simple implementation error, meaning it could be solved through some debugging and investigation. This explanation is also supported by the fact that there is a drop in the %Reports, which should not happen as only classes with zero members are removed. Due to time limitaitons was this potential bug backlogged. However, as the code is implemented and set to be used in future cluster projects this is not a problem since hybrid is the method that will be used and not sEM. Hybrid will not suffer from the error as it will always finish with an EM step and consequently the final implementation still works.

The visualization to the impact of the memory parameter α and the batch size p shows a similar result as the empirical results from other papers [21], where a smaller batch size reaches a higher altitude with the same number of epochs, i.e. a higher convergence rate. However, actual computational time is of greater importance to the aim of the thesis and thus the main choosing criteria of the value of p is the time. Which indicates that after the "L-turn" in the plots, the time taken does not alter significantly, but anything before the turn greatly changes the computation time. The plots indicate that the chosen value of 17 % is more than sufficient and that a smaller value might increase the final likelihood even above the EM model without increasing the computational time. However, as other results have shown, such as tab. 3, a smaller likelihood does not change the final clusters from the CC to a large extent. Even though choosing a smaller p, which still is below the red line, would give a higher likelihood the small decrease in time by choosing a small p is more valuable than the gain that can be made by increasing p.

In tab. 2 there is a decrease of the hybrid's ARI in varying degrees with all of the drugs. One reason for this is that since the batches are randomly selected there is more randomness in the algorithm, making it harder to produce the same clusters. EM also takes more steps close to the optimum which enables it to make small adjustments to the class assignments and thereby creating a more stable clustering. The %Reports decreases by a

few percentages, nonetheless the trend is the same as in fig. 6 and there is no significant change to the percentage. This could indicate that the speed up was done too greedily and that it would be more valuable for the clusters' quality if hybrid was run for a few more epochs. However, the drug seems to be a bigger factor in the resulting ARI. This indicates that the hybrid's ARI is not different in relation to the drugs, i.e. when EM's ARI is lower because of the drug the hybrid's ARI is also lower. If, on the contrary, hybrid would produce a significantly lower ARI for a drug with a high ARI from EM, there would be a voice of concern as the hybrid has captured something else between different clustering. Indicating that the produced clusters are similar and still suitable for CC. The more important measurement is how the different base models impacts the final clusters' ARI, which are the clusters from the CC. In the tab. 3 it is first and foremost noticeable how valuable the CC is. The ARI is higher with all drugs at the expense of losing reports, seen by the %Reports decreasing.

With regards to the impact the different base models have there is no significant difference of the ARI. Thus, the small change noticed between the models before does not translate. It is however also important to note that there still might be a loss in cluster quality as some drugs do actually lose stability. The %Reports is consistently lower, which might be explained through the fact that even though hybrid almost clusters the same number of reports it is still a few percentages lower. These percentages aggregates and produces less links above the weight threshold, which then results in fewer reports of the CC.

Therefore, I would argue for the use of hybrid instead of EM going forward as tab. 4 shows that there is a significant time improvement. A speed up of at least 1.4 with all drugs and it increases as the size increases with the caveat that increasing the number of epochs might produce a better result, or potentially decreasing p as it perhaps converges even faster.

Why the speed up increases over the size might be explained by the E-step's need to sum over the entire dataset to retrieve the sufficient statistics in the EM algorithm. Fig. 8 shows how the complexity of the algorithm has decreased from a linear dependence. Highlighting the potential to scale sEM to even larger datasets. This also fits nicely with the target to increase user friendliness, the absolute time that is gained for the smaller drugs is still quite small as they are completed relatively fast. The absolute time for the bigger drugs is a lot bigger and the time a user must wait is much lower, i.e 19 minutes instead of 35. Consequently, the potential for using the hybrid on bigger data sets is promising, both due to the complexity and that there are no memory concerns for either of the drugs.

The results from the other part of the speed up, the CC part, also show promising leads. Starting with the more straight forward improvement, the Pearce algorithm to find the connected components, the speed up factor is dramatically improved. A non surprising result as the implementation both uses an efficient algorithm to find the components and

the fact that it is implemented in C/C++ instead of Python. Simply one of those two would most likely increase the speed, and the two of them together give the dramatic change. Importantly, this part of the CC algorithm takes less time than acquiring the weights to the graph, therefore even though the speed up factor is as big as it is, looking at tab. 6 we see that it only gives a speed up at around 1.2 when running the complete CC algorithm.

The other implemented speed up is the parallelization, also visible in tab. 5. Where the speed up factor is heavily dependent on the size of the data set, where it first increases and then decreases as the size increases. Why it is lower for the small dataset is because initiating the parallelization takes more time than simply running it serially. To that end, there is an overhead rule implemented, after the output of the results, which ensures that data sets below a certain size will not be run in parallel. As it becomes more and more valuable to run it in parallel on all eight cores the speed up increases and reaches a peak at Ciprofloxacin, then as it is not possible to run it on all cores with Pneumococcal vaccine the speed up decreases again. This highlights the problem with the memory when speeding up the algorithm. An improvement that appeared as a consequence of enabling the parallelization to run on as big datasets as possible is that Adalimumab was possible to be executed with the new memory management, which also is a step in direction for it being feasible to run even larger datasets.

The fig. 9 shows a behaviour that indicates another problem as the data set increases, the first 5 drugs have a complexity of $\mathcal{O}(n^2)$, which can be seen as the curves follows the dashed green line in the logarithmic plot. With Adalimumab the behaviour drastically changes, and it does not follow the same complexity. A potential explanation is that in order to acquire the weights the sparse adjacency matrices are added together, when there is not enough memory to store them Python serializes the data and writes it to the hard memory instead. The serialization takes more time, especially when the data is large, and consequently Adalimumab takes disproportional long time to run. The figure also shows that even though a speed up is achieved the complexity of the new algorithm is still $\mathcal{O}(n^2)$.

Tab. 6 shows how the speed changes of the entire CC part, in it one can infer that given the 'suitable' size of the data it is possible to achieve a high speed up. Together with the results from the base model improvement the complete picture is painted in tab. 7. The table describes how the improved algorithm gives a speed up factor of two in some cases. The two different complexities, from the hybrid and CC, also make its presence known as the time that the hybrid takes becomes a smaller proportion of the entire execution time. From tab. 8 one can clearly see the impact on the time taken when using Leiden. However, together with all the other improvements Leiden is still able to execute faster than the original implementation. Highlighting the value of the achieved speed up from suggested improvements. As the dataset increases in size this speed becomes harder to achieve as Leiden's complexity is dependent both n and the number of edges in the graph.

Before diving into the objective measures of the Leiden algorithm it can be helpful to discuss the presented examples of the difference between the Leiden and the Single Link strategy as shown in figures 11, 12 and 13. In all examples Leiden is able to refine the cluster without losing reports, in fact Leiden increases the total number of reports, and most of the reports from the Single Link cluster are kept. The important question, however, is if this separation makes clinical sense.

By looking at fig. 11 we can see that cluster 1 and cluster 0 are similar by viewing the terms' percentages and the topic is some kind of gastrointestinal problems. In cluster 0 there is also the term *Pseudomembranous Colitis*, which Leiden, in cluster 2, is able to separate together with other *Colitis* terms into its own cluster. Consequently, Leiden found the Colitis reports in the gastrointestinal cluster, terms that were more diluted in the Single Link's cluster.

Looking at fig. 12 the Single Link cluster is split into three clusters where cluster 3 catches a different trend than cluster 7 and 5. The terms that cluster 3 produces is not apparent in the Single Link cluster and the refinement appears valuable. Cluster 7 and 5 are quite similar and separating those might not be a valuable distinction. It could in fact make it harder for the term intruder detection as a term such as *fall* then could be picked as the intruder in cluster 5. That intruder would then make clinical sense and the measurement punishes the unnecessary split, a problem of over refinement.

In fig. 13 the Leiden strategy creates three clusters of the Single Link cluster's reports. The clusters all have similar terms in them and are perhaps about the same topic, indicating a non valuable refinement. Clinical expertise highlighted that Cluster 7 could however be considered more severe as the reports with the term *Tendon rupture* have been distinctly separated to that one. Cluster 19 might not give additional clinical information but it makes sense algorithmically as the reports with *Myalgia*, muscle pain, have all been clustered there.

The other wanted improvement of the Leiden strategy was to cluster more reports than with Single Link, which objectively can be seen to have worked looking at the %Reports. However, there is a question if the extra reports have increased or decreased the clinical coherence. In in fig. 14 the clusters from the strategy display a topic concerned with eye complications. Leiden found many more reports than the Single Link strategy did, and the randomly selected reports all have terms connected to that problem, and none of them are clearly reports that should have been left out of the cluster. It could, however, be argued that there are terms within in the report that seem out of place. Take report 105, it has the term *Knee operation* which does not have a clear clinical connection to *Eye irritation*. Consequently, the topic that report 105 is connected to might not be the same as the cluster's topic and Leiden might have unnecessarily clustered that report.

The attempt of improving the cluster's quality with the Leiden algorithm appears both successful and unsuccessful. In terms of the objective measures Leiden was able to improve

both the %Reports, which was expected, and the ARI. One explanation for the increase in stability might be that Single Link is more vulnerable to single unexpected links which may introduce discrepancies, whereas the community detection may be more robust to such spurious associations between two reports. The fact that the ARI does increase is a testament to the strength of Leiden, it is able to find a high level of consistency in the different adjacency matrices, while still creating clusters with significantly more reports than the Single Link strategy. This is also the result that the empirical investigation, seen in fig. 5, indicated. In these regards Leiden performs objectively better than the Single Link strategy. However, the problem regarding the lack of ground truth makes itself known here, even though these measures are improved it may not actually provide a higher cluster quality. A clustering with a perfect ARI and all the reports clustered would be a singleton cluster, which scores higher at those two measures but is obviously not an improved cluster analysis. To evaluate the clusters quality for clinical usefulness the term intruder detection is the more important metric.

Term intruder detection scores worse with Leiden on all drugs, indicating an unsuccessful cluster quality improvement algorithm, even though examples such as discussed above were found. A speculative and non-observed explanation for the lower score is the problem of over refinement, as displayed in fig. 12. In the example cluster 5 and 7 could still be considered coherent by themselves even though there might be a wrongly spotted intruder term if the term is chosen from 5 to 7 or vice versa. To that end is refinement perhaps punished disproportionately with the term intruder detection.

This fact could help to explain why Leiden gets a lower term intruder detection score. However, the fact that Leiden scores worse is a reason to further investigate how to fine-tune the strategy in order to maximize the potential of the Leiden algorithm, and to note that the algorithm performs worse. There is also the problem of memory with this algorithm, which is a greater issue as Leiden cannot be executed for the biggest drugs in VigiBase. Consequently, running with bigger datasets is also out of the question. Solving this problem requires either implementing the algorithm manually in order to find areas of the algorithm that might not be needed and thereby reduce memory requirements. The other potential solution is improving the hardware.

Improving the hardware would not only help with the Leiden algorithm's memory concern but also with acquiring the weights. The latter is true both in terms of being able to run bigger datasets on 8 cores as well as the serialization concern that completely changes the complexity of Adalimumab. As the CC part is of $\mathcal{O}(n^2)$, both in memory and time, complexity running this algorithm for millions of reports would require either a vastly bigger memory capacity of the hardware or a revised algorithm with a lower complexity. An improved hardware with more cores might also increase the speed up even further as then more work is evenly dispersed throughout the CPU.

As mentioned in the section about EM, sec. 2.4.2, if the class membership is known it is possible to retrieve the mixture model's parameters. This gives the opportunity to create a classification algorithm from the final clusters from CC, with the help of Leiden the classifying model would then be able to classify reports to a more specific topic. This is of great value as problem in PV is that when an adverse drug reaction first appears the number of reports connected to it from a drug are often few, many of these would also be coded differently. The drug's few reports could then be applied to the classifying algorithm the reports could then get the context of the topic, and even though an adverse drug reaction may be coded in different ways the algorithm might find more reports. With more refined clusters the context would then be more specific. This is of course also possible with the Single Link strategy.

A future research idea is to take the idea of creating a classification model to the extreme by executing the cluster analysis on the entire VigiBase. Then estimating the parameters from the class assignments would create a great variety of topics. It would then be possible to perform a sort of disproportionality analysis with topics instead of single terms, by letting a new drug go through the model and checking whether topics are disproportionately high instead of single terms. This example both highlight the value of running the algorithm on big datasets as well as the potential gain of then finding more refined clusters in the huge dataset. However, the outlook for this is quite far away as the CC part of the algorithm with Leiden in its current state is not executable for datasets above 100,000. Additionally, even though Adalimumab was possible to run it was at the very edge of the memory capacity of the computer. Consequently, for the idea to be feasible improved hardware is needed and perhaps a novel approach of doing the CC.

6 Conclusions

The goal of investigating ways of improving the computational efficiency has been achieved. The resulting implementation is dependent on the size of the data set. However, given a suitable size of the data set the execution time can be reduced in half. The implementation of a hybrid approach combining EM with sEM has been successful as the speed up is above 1.4, with the added benefit that the computational complexity has been reduced, without any significant change to the final clusters' quality. Additionally, speeding up the CC part of the algorithm is also achieved but with the caveat of being severely memory dependent, except for finding the connected components which is a clear-cut significant improvement. In order to achieve a high speed up for bigger data sets either improvements to the hardware are needed or a change in the CC part of the algorithm as that part does not scale to bigger sets.

Improving the clusters' quality with community detection shows potential as the measures of stability and how many reports are clustered are both improved. The clinical coherence, as measured by term intruder detection performs worse. However, the results

are somewhat conflicting as there are also examples of clusters with improved resolution that still maintains clinical coherence after posthoc review. Consequently, there are reasons to further investigate why Leiden objectively performs worse in clinical coherence and further develop the strategy for future cluster analysis endeavours.

7 Acknowledgements

My greatest gratitude to my supervisors, Eva-Lisa Meldau who helped me greatly with understanding the old implementation, and Niklas Norén who gave valid input regarding results and directions to take the thesis. I would also like to thank Mats Gustafsson who helped me make sure that the report kept an academic quality. Mónica Tarapués' work as examiner of the clusters and finding the intruders deserves an acknowledgement as it gave the thesis an important clinical connection. Finally, thanks to the people in the data science department of UMC for valuable coffee breaks, and to the rest of UMC for providing such a pleasant experience.

References

[1] A. Bate and S. J. W. Evans. Quantitative signal detection using spontaneous adr reporting. *Pharmacoepidemiology and Drug Safety*, 18(6):427–436, 2009.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55:77–84, 2012.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[5] Matthew Burgess, Eytan Adar, and Michael Cafarella. Link-prediction enhanced consensus clustering for complex networks. *PloS one*, 11(5), 2016.

[6] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[7] Uppsala Monitoring Centre. *Making medicines safer*. Uppsala Monitoring Centre, Uppsala, Sweden, 2018.

[8] Uppsala Monitoring Centre. A beginners guide to pharmacovigilance, 2020. `https://www.who-umc.org/global-pharmacovigilance/global-pharmacovigilance/a-beginners-guide-to-pharmacovigilance/`, Last accessed on 2020-06-03.

[9] Uppsala Monitoring Centre. Members of the who programme for international drug monitoring, 2020. https://www.who-umc.org/global-pharmacovigilance/who-programme-for-international-drug-monitoring/who-programme-members/, Last accessed on 2020-06-15.

[10] Rebecca E Chandler, Kristina Juhlin, Jonas Fransson, Ola Caster, I Ralph Edwards, and G Niklas Norén. Current safety concerns with human papillomavirus vaccine: a cluster analysis of reports in vigibase®. *Drug safety*, 40(1):81–90, 2017.

[11] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.

[12] Jianfei Chen, Jun Zhu, Yee Whye Teh, and Tong Zhang. Stochastic expectation maximization with variance reduction. In *Advances in Neural Information Processing Systems*, pages 7967–7977, 2018.

[13] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[14] I. Ralph Edwards and Sten Olsson. The who international drug monitoring programme. volume 25 of *Side Effects of Drugs Annual*, pages 589 – 598. Elsevier, 2002.

[15] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[16] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):4–es, 2007.

[17] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[18] Abonyi Janos and Feil Balazs. *Cluster Analysis for Data Mining and System Identification*. Birkhauser, 2007.

[19] James H Kim and Anthony R Scialli. Thalidomide: the tragedy of birth defects and the effective treatment of disease. *Toxicological sciences*, 122(1):1–6, 2011.

[20] Andrea Lancichinetti and Santo Fortunato. Consensus clustering in complex networks. *Scientific reports*, 2:336, 2012.

[21] Percy Liang and Dan Klein. Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 611–619, 2009.

[22] Marie Lindquist. Vigibase, the who global icsr database system: Basic facts. *Drug Information Journal*, 42(5):409–419, 2008.

[23] Jonas Lundkvist and Bengt Jönsson. Pharmacoeconomics of adverse drug reactions. *Fundamental & clinical pharmacology*, 18(3):275–280, 2004.

[24] Stefano B.B.R.P. Mathias, Valério Rosset, and Mariá C.V. Nascimento. Community detection by consensus genetic-based algorithm for directed networks. *Procedia Computer Science*, 96:90 – 99, 2016. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 20th International Conference KES-2016.

[25] Fem myror är fler än fyra elefanter. Brasses djurlåda. `https://www.youtube.com/watch?v=iMx36mlZEpE`, 2020.

[26] Niklas Noren, Eva-Lisa Meldau, and Rebecca Chandler. Consensus clustering for case series identification and adverse event profiles in pharmacovigilance. submitted.

[27] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[28] World Health Organization. The importance of pharmacovigilance, 2002.

[29] David J Pearce. An improved algorithm for finding the strongly connected components of a directed graph. *Victoria University, Wellington, NZ, Tech. Rep*, 2005.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] Javier Rasero, Mario Pellicoro, Leonardo Angelini, Jesus M Cortes, Daniele Marinazzo, and Sebastiano Stramaglia. Consensus clustering approach to group brain connectivity matrices. *Network Neuroscience*, 1(3):242–253, 2017.

[32] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[33] Janet Sultana, Paola Cutroneo, and Gianluca Trifirò. Clinical and economic burden of adverse drug reactions. *Journal of pharmacology & pharmacotherapeutics*, 4(Suppl1):S73, 2013.

[34] Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

[35] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[36] Uppsala Monitoring Centre. Who we are, 2020. http://www.who.int/mediacentre/factsheets/fs282/fr/, Last accessed on 2020-06-03.

[37] Vincent A Traag. Igraph:leiden algorithm, 2020. https://igraph.org/python/doc/igraph.Graph-class.html\#community_leiden, Last accessed on 2020-06-03.

[38] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[39] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.