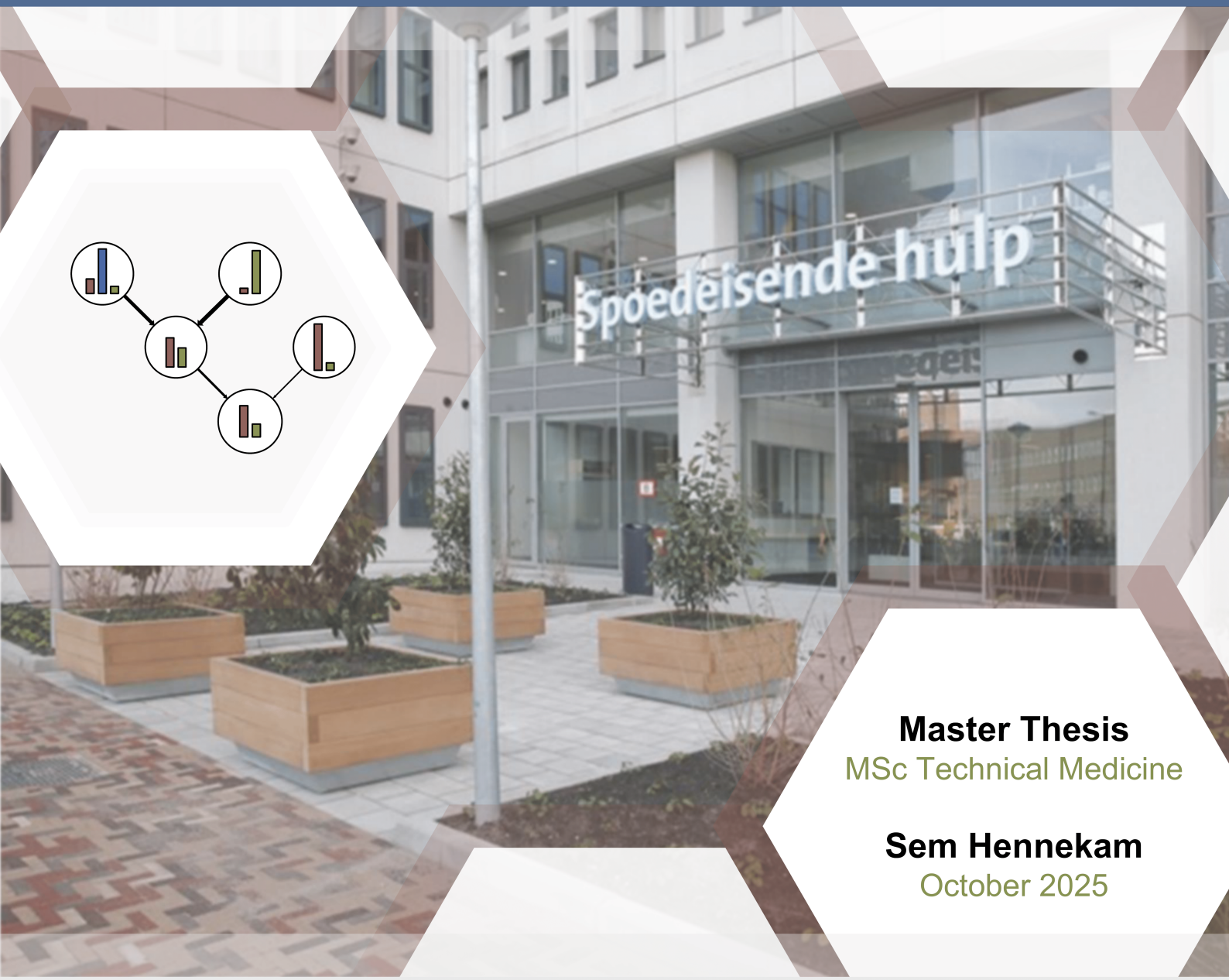
Promedas: Bayesian Network-Based Clinical Decision Support in Emergency Care



Master Thesis
MSc Technical Medicine

Sem Hennekam
October 2025

TU Delft

Erasmus MC
Universitair Medisch Centrum Rotterdam

LUMC Leids Universitair Medisch Centrum

This page was intentionally left blank.

PROMEDAS: BAYESIAN NETWORK-BASED CLINICAL DECISION SUPPORT IN EMERGENCY CARE

Sem Hennekam

Student number : 4802578

30 oct 2025

Thesis in partial fulfilment of the requirements for the joint degree of Master of Science in

Technical Medicine

Leiden University ; Delft University of Technology ; Erasmus University Rotterdam

Master thesis project (TM30004 ; 35 ECTS)

Dept. of Acute Internal Medicine, Internal Medicine, Erasmus MC

6 jan 2025 – 14 nov 2025

Supervisors:

Dr. Jelmer Alsma

Dr. Wim Wiegerinck

Thesis committee members:

Dr. Jifke Veenland, TU Delft (chair)

Dr. Jelmer Alsma, Erasmus MC

Dr. Wim Wiegerinck, Radboud Universiteit

...

Universiteit Leiden

TU Delft Delft University of Technology

Erasmus Universiteit Rotterdam

This page was intentionally left blank.

Preface

Hello there and welcome to my Master Thesis Report. This report concludes my incredibly educative 7-year journey at the TU Delft, Erasmus MC and LUMC; a period of my life that truly has flown by, looking back on it, but I guess time just flies when you're having fun. As much as I find it hard to say goodbye to it just yet, I also look forward to what is to come after this.

I find it appropriate to say a word of gratitude to those who have helped me out throughout this project. The team of Promedas for one, which not only has seen the attentive involvement and insights from Jelmer, Wim and Alisa, but also the input of countless Technical Medicine interns besides me throughout these months. Make no mistake, this thesis has from the very start been part of a group effort and I could not imagine the progress remotely near where it currently stands without their contributions. I also want to thank the staff of the internal medicine department in the Erasmus MC for giving me an environment to practice my clinical skills. But most of all I would like to thank my family at home for their tremendous love, care and strength during what has been an absolute rollercoaster of a year for all of us, in particular Karlijn, Remy and our cat Wiebel.

Having said that, the matter is bound to become complex here and there: the Bayesian Network is not a very well-known data structure in general and, despite my education, I still regularly find myself in awe with the complexity of the human body as well. I did take the liberty to provide some context for those less familiar with Bayesian Networks and/or diagnostic errors on the emergency department, which I hope avoids most of the confusion on these topics. I also couldn't help but occasionally refer to the work of other students, which is unpublished but available within the Promedas team. Lastly, I will move over to passive/third-person from this point, except for a handful of footnotes. I hope you will find joy in reading this report as much as I found it in my thesis itself.

- Sem

Contents

1. Introduction

Diagnostic Errors (DEs) are an important challenge in modern medicine. Estimated to occur in roughly 5.7% of Emergency Department (ED) admissions throughout the USA and Europe, DEs affect the everyday standard of care [1]. Additionally, approximately 2% of ED admissions suffer from Adverse Events (AEs) due to a DE, 0.3% of ED admissions suffer Serious Adverse Events (SAEs), and, among the SAEs, 0.2% of ED admissions have a fatal outcome due to a DE. In the USA, these numbers would imply 2.6 million SAEs and 250,000 deaths per year. If the same numbers would apply to the Netherlands, which sees 1.97 million ED admissions per year, this would translate into 112,000 DEs, 39,400 AEs, 5,910 SAEs and 3,940 deaths per year.

DEs occur due to many factors. The direct cause being the clinician's misjudgement: wrong diagnostic assessment, test ordering and interpretation, which on their turn can be attributed to a lack of (adequate) clinical knowledge, skill or reasoning [1]. The faulty estimation of (conditional) probabilities is known to be an important cause for DEs in medicine [2, 3]. Additionally, human judgement can only account for up to 20 variables, or 40 with proper decision support tools, and will decrease when more variables are added beyond those [4]. Clinical decisions, on their turn, require clinicians to take many variables into account: the anamnesis of a patient alone could already exceed this amount of 20 to 40 variables.

This is exacerbated by the fact that clinicians in the ED regularly work under pressure. Patients may be admitted in a critical state and the ED has a limited capacity in terms of space and personnel. And the admitted patient cases can be insidious. In fact, the strongest and most consistent predictors for diagnostic errors are atypical presentations, cases with non-specific symptoms and diseases that are "out of place" [1].

In reducing the rate of DEs and its risks, a multitude of approaches can be postulated. The most evident approach is to increase ED capacity and staffing, while increasing the involvement of more experienced staff members, but this is costly. Alternatively, one could attempt to increase the knowledge and skill of clinicians, while decreasing the potential harms from DEs. A good clinical decision support system (CDSS) can kill two birds with one stone, by generating a second opinion for the clinician and transparently providing the reasoning behind its diagnosis.

Promedas (PRObabilistic MEdical Diagnostic Advisory System) is being

developed for this specific purpose [5]. Built as a Bayesian Network (BN) model, Promedas is designed with the aim to provide an accurate differential diagnosis for clinicians in the ED [6].

A BN is a probabilistic graphical model, which represents a set of variables, their prior chances, and their interdependent probabilistic dependencies. A BN can make use of these conditional probabilities and a set of observed variables and states, to calculate the probability of unobserved nodes to have a certain state in the model. In the case of Promedas, a set of symptoms and findings from physical and additional examinations; and predictive factors like smoking and a history of cardiovascular disease, can be used to calculate the probability of each diagnosis in the system. The conditional variables of interest, the parameters of the model, are sensitivity values. The probabilities of these diagnoses can then be sorted to create a differential diagnosis, which approximates the real chances of each disease being true. BNs are explained in more detail in Section 2.2.

The development of Promedas is still on its way. The last iteration in the configuration of Promedas, before the start of this thesis, was implemented by colleagues Scholten, E., Memelink, A. and Zwart, G. After their work, 22 out of 37 patient cases in Promedas had their correct diagnosis in the top 3 of the differential diagnosis. Promedas still had some big challenges with regard to the adequacy in which it represents clinical practice and, conversely, the set of patient data also contained illogical data, which could not be traced back.

This thesis continues building on that version of Promedas. In the weeks this thesis spanned, other efforts were made by colleague students from the TU Delft Technical Medicine master program and the Radboud University Artificial Intelligence bachelor program.

The overarching goal of this thesis can be summarized as to determine whether Promedas, as a BN model, can exceed the clinicians' overall diagnostic accuracy of 94.7%. This summary is odd, though, as the sheer number of steps involved towards that goal still puts it out of reach within the given time span. Instead, an overview is provided of the steps that have been made towards this goal, each with their own subgoals.

The subgoals towards this goal are as follows: to obtain a larger dataset of patient cases, that both provides insight into the source material and increases the amount of reference material for Promedas; to facilitate the translation from unstructured clinical documents to the standardised format of Promedas; to improve the efficiency of the development process of Promedas; and to propose and test changes to the configuration and optimisation of Promedas.

The next chapter continues to provide more context on both the technical and the clinical side of this thesis: Bayesian Networks, Promedas itself and misdiagnoses. Chapter 3 continues to explain how the rest of this report is structured with respect to these subgoals.

2. Context

The goal of this chapter is to illustrate the problem definition underlying Promedas in more detail. There is ample room for misunderstandings between the technical side and the medical side of this story. So it stands to reason to 'bridge the gap' between these two worlds by introducing each side to the base principles of the other's.

Put briefly, it describes the 'what', the 'how' and the 'why' of it: so what is this technology? How is it planned to be implemented? And why do we do this? The first two sections will elaborate the clinical necessity of Promedas: how are diagnostic errors established, and how are they caused? The last two sections cover what Bayesian Networks are and how Promedas is applied as one.

2.1 Misdiagnoses in the Emergency Department

One theme that keeps recurring is this 5.7% misdiagnosis rate in Emergency Departments throughout the USA and Europe. I have cited the work of Newman-Toker et al. for this [1]. But still, I want to elaborate how Newman-Toker et al. obtained this number. This was important for me to understand in the beginning, in order to be able to execute performance analysis on Promedas. But seeing as I went through this anyway, I will dedicate this section to summarize the most important elements from this (rather extensive) review.

2.1.1 The basics

The underlying study was a systematic review of general clinical research studies. Their main objective was to determine the most frequent diseases and clinical presentations associated with DEs in the ED, to measure the error and harm frequency, and to assess causal factors. These studies had to be conducted on the Emergency Department of any hospital in North America or Europe. The authors included 279 studies after screening. From these studies, they were able to estimate that in every ED visit, there is a 5.7% (95% CI: 4.4-7.1%) chance of misdiagnosis. Aside from that, the authors spent attention to disease prevalences in the ED at large and how

misdiagnoses vary between different groups of diseases. They found that myocardial infarctions are misdiagnosed in only 1.5% of cases, where spinal abscesses could be misdiagnosed in up to 56% of presentations. The authors additionally investigated common reasons behind these diagnostic errors, and found that approximately 89% of diagnostic errors could be traced back to failure of decision-making or judgement.

2.1.2 When is a diagnosis wrong?

So far so good. This 5.7%, or really the DE rate that each study observed, is just found by dividing the number of cases with wrong diagnoses by the overall number of ED admissions. The latter is simple enough, but the first creates an important question: how do we know that a diagnosis is wrong? After all, the clinician just found this diagnosis to be the most adequate according to their experience and knowledge and the patient's presentation.

Newman-Toker et al. don't define some approach to avoid this, although they do mention some attempts by other researchers. They mention the National Academy of Medicine definition of misdiagnosis, which is "the failure to (a) establish an accurate and timely explanation of the patient's health problem(s) or (b) communicate that explanation to the patient", but criticize it for its practical vagueness [7]. And they mention the work of Hautz et al. which aims to establish a more concrete definition [8]. In their own review, however, they simply use the definition as is employed by each individual included study, which was necessary too, for its large scope.

These studies had a lot of different measurements, each with their own caveats. These include but are not limited to: autopsy reports, which are definitive, but underestimate the real number; misdiagnoses recognized by coworkers, which depends on these coworkers' attention and accuracy; malpractice claims, which were biased towards diagnostic errors with worse implications; and incident reports, which were also biased towards more outstanding incidents.

The reported 5.7% DE rate is a weighted average of two studies, which took systematic approaches to DEs. One was conducted in an academic hospital in Spain [9]. Here, they used an adverse event trigger, which was defined as a return for the same chief complaint within 72 hours. This was triggered in 250 out of the 32523 eligible patients. The researchers then created a new independent control population of 250 patients, with similar characteristics. Upon review, they estimated a 4% diagnostic error rate within this control group. Newman-Toker et al. argue that this is likely a minimum, as it was solely based on charts and not on patient follow-up. The other study was conducted in a tertiary care facility in Switzerland [8]. This study considered

'clinically meaningful discrepancies', which was defined to be a difference in diagnosis given during admission from the ED and the diagnosis upon discharge, where this difference would lead to a clinically meaningful difference in hospital length of stay and/or mortality. They found this rate to be 12%. It is difficult to decide whether this is an under- or over-estimate, though, as this relies on the assumption that the patient was discharged with the correct diagnosis and that all misdiagnoses on the ED would correctly be caught.

2.1.3 More statistics

Newman-Toker et al. went on with more findings. They approximated DE-related adverse events at 2% (95% CI: 1.0-3.6); DE-related mortality at 0.2% (95% CI: 0.1-0.4); and DE-related permanent disability at 0.3% (95% CI: 0.1-0.7).

The report additionally examined DE rates by respective diseases, going by malpractice claims and incidence reports, the five most common diseases in these files and their occurrence are fractures (26.2%), stroke (9.9%), myocardial infarction (6.5%), appendicitis (2.9%) and venous thromboembolism (2.8%). When factoring in the harm from misdiagnosis, however, fractures are far less severe. Especially risky diagnoses are stroke, myocardial infarctions, aortic aneurysm/dissection, spinal cord compression/injury, venous thromboembolism, meningitis/encephalitis, sepsis and lung cancer, among others. The authors noted that a majority of these claims could be attributed to the "Big Three" disease categories. Going by high-severity harm cases, 41.5% are attributed to vascular events, 22.5% to infections, and 7.9% to cancers. Trauma, at 11.6%, is a notable other category, though. By organ system, the neurologic tract occurs most often in high-severity harm cases, at 34.1%. After that is the cardiovascular tract (22.8%) and the pulmonary tract (7.6%). But, whereas the top 5 organ systems make up 78.6% of high-severity harm cases, they constitute 'only' 45.1% of low- and medium-severity harm cases.

The study also mentions, for highly common diagnoses, the false positive and false negative rates, which are in their table 9. These could be quite significant too: false negative rates could reach up to 56% for spinal abscess, or 36% for aortic aneurysm and dissection. False positive rates could be as high as 34% for pneumonia, or 49% for transient ischaemic attacks. The authors did not supply an overall number for the sensitivity and specificity in all patient cases, but remarked its high inter-diagnosis variability and there being a rough inverse proportionate relationship between a disease's false negative rate and its occurrence in the ED. I.e. rarer diseases are missed more often. The authors lastly remarked that there was large variability

between different EDs.

2.1.4 Reasons behind diagnostic errors

Newman-Toker et al. finally raised the question of which factors underlie DEs. They found that 90% of ED errors could be attributed to cognitive factors: diagnostic assessment, test ordering and test interpretation; which on their turn were attributed to inadequate clinical knowledge, skills or reasoning. The authors also noted that atypical presentations, non-specific symptoms and "out of place" diseases (e.g. a very young stroke patient) were the strongest and most consistent predictors for diagnostic errors. In other words, clinicians do not miss obvious diagnoses, they miss them when they are subtle. They concluded that clinical risk management should rather be focused on these subtle patient cases instead. Additionally, the authors found a roughly 20 to 30 percent increased DE risk in women and minorities throughout the included studies, which were attributed to the lack of clinical knowledge about differences in patient presentations by gender and ethnicity.

On top of these findings, other sources have investigated the processes behind DEs as well. It is known that faulty estimation of probabilities is an important cause of DEs [2, 3]. Additionally, a 1999 report by the Central Intelligence Agency concluded that humans can reasonably make decisions when they consider up to 20 information entities [4]. This can be increased to 40 information entities, by use of good decision support tools. Beyond 40 entities, human decision quality declines.

2.1.5 Implications for Promedas

The underlying reasons for DEs highlight where a good CDSS is most needed. Namely, where humans fall short. In practice, where clinicians are exposed to tens or even hundreds of variables, mistakes are unavoidable - a 94.3% overall accuracy rate is quite impressive in this respect.

Two important considerations to begin with are related to the observed 5.7% error rate. For one, this number is not representative of the performance of Dutch hospitals, as it is based on a Spanish and a Swiss medical centre. The diagnostic accuracy and error rate of Dutch medical centres remains as yet unknown.

Secondly, one must remain mindful of the data that is used to configure Promedas, and its implications. A straight-forward registration of findings and diagnoses from patient cases in the past inherently contains this same 5.7% DE rate. If Promedas is 100% accurate in this dataset, it is just in full agreement with clinicians and their flaws. On the other hand, if

Promedas proves to be hypothetically 100% accurate, accounting for DEs, then it would only be 94.3% accurate on the dataset of clinician-made diagnoses. If Promedas approaches this 95% accuracy rate in a later stage, it is critical to test the performance of Promedas in cases that were wrongly diagnosed, to validate the consistency of its performance in such cases.

Promedas could prove its worth by providing a graphic insight for the clinician with all findings and its differential diagnosis and show why it finds some diagnoses plausible. And it could provide sources for the conditional probabilities in the network from literature, which could further educate the clinician. This would both serve as a decision support tool and as the backbone of a learning environment. On top of that, it provides certainty about test interpretation.

Conversely, with additional mathematics, it seems plausible that Promedas could also recommend some tests or findings to distinguish diagnoses that seem highly plausible, at any given moment. E.g. if somebody is admitted with chest pain, Promedas could suggest to make an ECG to provide more insight. Note that many such tests are already commonplace in practice, especially upon admission, but they may be useful at a later stage. For instance, uncertainty about the pathogen underlying some infectious presentation, could lead Promedas to suggest inquiring new information, when the clinician is still in doubt. This would be especially useful in detecting rarer diagnoses, of which clinicians may overlook their plausibility, when all evidence points in favour of this rarer diagnosis as well.

Whether a Bayesian Network model could also be beneficial in atypical patient cases and "out of place" diseases, is yet to be discovered. One could speculate in two ways. On one hand, a BN remains truthful and objective to the probabilities within it. If a handful of symptoms point in the direction of some disease, the BN should find that disease more likely. A BN is less susceptible to deceptions like that. As for racial and gender inequality in DEs, a BN could be an interesting candidate to create more insight as well. On the other hand, an experienced clinician may have a powerful gut feeling, which could be invaluable in getting to the bottom of such cases. And atypical case presentations may also, on a statistical level, remain very unlikely.

Lastly, Promedas raises some questions about responsibility. I like to see Promedas as a tool: one that maps the clinician's own reasoning process, while highlighting the pitfalls it expects. But then again, if the only tool you have is a hammer, every problem becomes a nail [10]. If Promedas were to become commonplace, it would inevitably shape the way in which clinicians reason about cases, both in individual cases and in the general population. So when a DE occurs in a clinician that uses Promedas, it would be unfair to hold the clinician fully accountable for the final diagnosis. On the other

hand, one couldn't expect Promedas to be perfectly accurate either. All this considered, the 94.3% to 100% accuracy range would be challenging for Promedas from a regulatory perspective.

2.2 Bayesian Networks explained

Promedas is built as a certain computer model, namely a Bayesian Network (BN). Although handy, Bayesian Networks are not as commonly known as some other data structures, which tends to create some confusion about their design and behaviour. This section aims to elaborate what BNs are, how they are designed, and what behaviour one could expect from it.

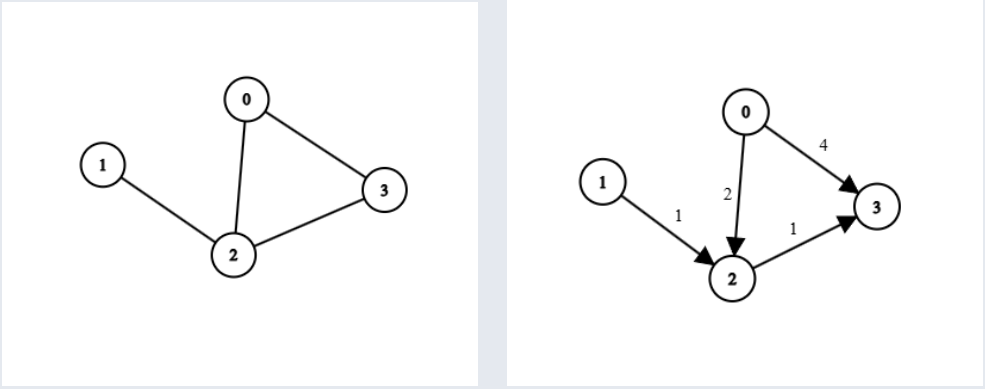
2.2.1 What is a Bayesian Network?

A Bayesian Network is generally defined as a probabilistic graphical model [6, 11]. This already reveals three properties: it is a graph; it is used for probabilistic functions; and it is used to model some occurrence from the real world (i.e. provide insight or simulate). Each of these are elaborated in the next sections. BNs are interesting for scenarios where many probabilistic dependencies interact. These are all over in modern medicine, for instance in diagnosing a patient on the basis of their symptoms and examinations, but there are many more contexts in which these can be used.

2.2.2 Graphs

Starting with the 'graphic' property, a BN is built on top of a data structure called a graph (Figure 2.1) [12]. A graph is a data structure that consists of vertices (also called nodes) and edges. Edges exist between any two vertices in this graph. There are a variety of ways in which graphs can be built in code, but that is outside the scope of this report. It is possible to add extra information to a graph. An edge may have a weight, for example, which could resemble a distance, cost or resistance. Graphs underlay a lot of applications today: GPS navigation makes use of graph representations of roads to calculate the shortest route towards the user's destination; feed-forward, recurrent and convolutional neural networks in machine learning are built as graphs, to emulate the function of neural tissue in animals; and finite state machines, which are commonly employed in embedded systems, for instance in robotics or simple lifts, are graphs as well.

By introducing extra restrictions to a graph, one can turn it into a Directed Acyclic Graph (DAG). Such a graph has two properties. First, its



(a) A simple graph (b) A DAG with weights

Figure 2.2: Two example graphs

edges need to be directed, i.e. an edge that goes from node x to node y does not automatically go from y to x. Second, a DAG may contain no cycles, i.e. it is impossible from any of the nodes in this graph, if one moves along the edges, to end up at the same node.

2.2.3 A Bayesian Network as a graph

A Bayesian Network is a DAG as well, see an example in Figure 2.3. In the case of a Bayesian Network, a node represents an observable outcome with some state s. This state could be Boolean, i.e. True or False, consist of multiple categories or even numerical values. For example, a node could represent the statement 'It is raining right now', with states 'yes' (True) and 'no' (False). The probability of the node would then represent the chance that it is raining right now.

The edges in a Bayesian Network are directed by definition. The direction of the edge represents the existence of a conditional probability between the child node and its parent node. In the example, the edge from node A to node B resembles the existence of some probability $P(s_B|s_A)$. Also note that the contrary is implied: when an edge does not exist between two nodes, they are assumed to be conditionally independent.

The edges do not receive a weight. Instead, they are represented through a Conditional Probability Table (CPT) in the child node. Going by the example above, assuming the nodes A and B are Boolean variables, the CPT receives two values: $P(s_B|\neg\ s_A)$ and $P(s_B|s_A)$.

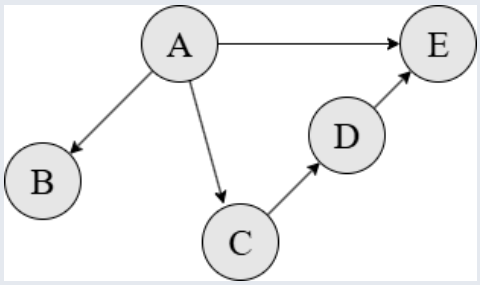


Figure 2.3: An example of a Bayesian Network graph

The CPT is stored in the child node, because these nodes may have more than one parent. The rule above can be generalised to account for this scenario: The CPT of a child node C with parent nodes $B_0, \cdots, B_n$ represents $P(s_C|s_{B_0} \cap \cdots \cap s_{B_n})$. Nodes without a parent, on the other hand, do not receive a condition at all. The CPT of node A just defines $P(s_A)$, by default its prior probability.

2.2.4 Inference

Nodes in a Bayesian Network can be 'clamped' to a certain state. This represents an observation of the variable it represents, so as to say 'yes, it is raining right now'.

After clamping these observations, a Bayesian Network can be inferred to calculate the probabilities of all unobserved variables. To establish this, we express the probability of all nodes in the network. A BN with nodes $V_0, \cdots, V_n$ can be expressed according to Equation 2.1.

$$P(s_{V_0} \cap \cdots \cap s_{V_n}) \tag{2.1}$$

For example, the network from Figure 2.3 can be expressed according to Equation 2.2

$$P(s_A \cap s_B \cap s_C \cap s_D \cap s_E) \tag{2.2}$$

This expression can be taken apart using the chain rule of probability (Equation 2.3) [13]. Using this formula, the equation of the network can be rewritten as the product of the probability of the state of each node in the network, given the state of their parents.

$$P(A_1 \cap \cdots \cap A_n) = \prod_{i=1}^{n} P(A_i | A_1 \cap \cdots \cap A_{i-1}) \tag{2.3}$$

Going by the same example, the probability of node E can be expressed according to Equation 2.4.

$$P(s_E | s_A \cap s_B \cap s_C \cap s_D) \tag{2.4}$$

Note that no conditional relation is defined between B and E and between C and E. Therefore, the expression can be reduced to Equation 2.5.

$$P(s_E | s_A \cap s_D) \tag{2.5}$$

This can be expanded to the entire network to form Equation 2.6.

$$P(s_A, s_B, s_C, s_D, s_E) = P(s_E | s_A \cap s_D) P(s_D | s_C) P(s_C | s_A) P(s_B | s_A) P(s_A) \tag{2.6}$$

This expression can be used to formulate a strategy to solve the state of the network. The analytical approach involves reducing the conditional probabilities back to a marginal probability, e.g. to reduce $P(s_E | s_A \cap s_D)$ back to $P(s_E)$. This relies on marginalization (Equation 2.7). This way, through iterations, nodes can be eliminated [14].

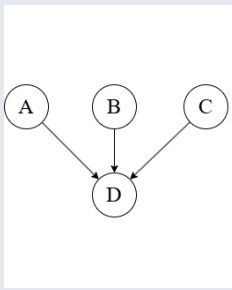
$$P(X) = \sum_{Y} P(X \cap Y) \tag{2.7}$$

node A can for instance be eliminated through the expression in Equation 2.8.

$$P(s_C, s_B) = \sum_{s_A} P(s_C | s_A) P(s_B | s_A) P(s_A) \tag{2.8}$$

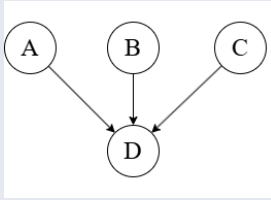
The probability of any node in the network can be calculated through multiple iterations of this process, eventually returning a probability value for the unobserved variable of interest.

However, this analytical algorithm is NP-hard [15]. This means, in practical terms, that the algorithm cannot solve the network within a reasonable amount of time, as the network grows larger. A multitude of alternative algorithms have been developed since, mainly relying on approximation of these probabilities.



$P(s_A \mid s_B, s_C, s_D)$				
s_B	s_C	s_D	$s_A = 0$	$s_A = 1$
0	0	0	0.8	0.2
0	0	1	0.7	0.3
0	1	0	0.9	0.1
0	1	1	0.4	0.6
1	0	0	0.2	0.8
1	0	1	0.1	0.9
1	1	0	0.2	0.8
1	1	1	0.4	0.6

(a) A default node in a BN, A, and its truth table



$P(s_A \mid s_B, s_C, s_D)$			
		$s_A = 0$	$s_A = 1$
Leak		0.6	0.4
s_B	1	0.8	0.2
s_C	1	0.9	0.1
s_D	1	0.2	0.8

(b) A noisy-OR node, A, and its truth table

Figure 2.5: A default and a noisy-OR node. Note that $P(s_A = 1) = 1 - P(s_A = 0)$

2.2.5 Simplifications

Promedas ultimately relies on a few more assumptions to reduce its runtime and storage requirements.

2.2.5.1 Noisy OR-gates

One problem arises in the CPTs, because the probability value of the state of a node must be defined for all combinations of states of its parent nodes. In the case of Promedas, medical literature will fall short in describing the interplay of multiple variables. Even worse is the fact that the number of CPT entries grows exponentially with the number of parents: a binary node with n binary parent nodes has to define 2^n conditional probabilities, which takes up a lot of storage (Figure 2.5a).

The solution to this is to assume that all conditional probabilities are mutually independent. A node that is based on this assumption is also known as a noisy OR-gate [16]. A binary noisy OR-gate only needs to define $n + 1$ values in its CPT (Figure 2.5b). One for each conditional chance and one more to define a 'leak' value, the prior chance of a node being True when all its parents are False.

2.2.5.2 Quickscore algorithm

Quickscore algorithm is another inference algorithm for Bayesian Networks with more acceptable runtime and space requirements [17]. Quickscore enforces noisy OR-gates along with a strict two-layer network structure towards

this goal (Figure 2.6). In this case, diseases are stored in the 'parent' layer and findings in the 'child' layer. The links therefore are defined as the probability of a finding being true, given that someone has a certain disease.

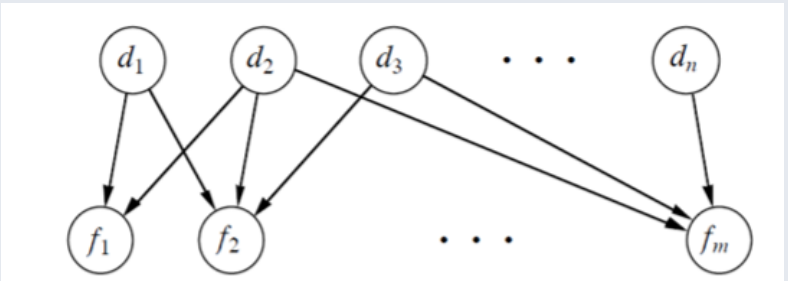


Figure 2.6: The two-layer Quickscore network layout, adapted from [17].

Heckerman argued that Quickscore is suitable for medical diagnostics, because the runtime complexity only grew exponentially with the number of positive findings, which tends to remain relatively small in real patient cases. This comes at a tradeoff with the following assumptions: diseases are marginally independent; findings are conditionally independent; and diseases act independently to cause any given finding to be present. The author highlights the drawbacks of these assumptions in clinical practice. Most importantly, diseases and findings are not always independent in reality, and some diseases and findings may mutually be conditionally dependent.

2.2.6 Bayesian Network behaviour

These properties have several implications for Bayesian Networks. On the upside, BNs are by design robust against missing data, as their very design principle is to provide reasoning about unobserved variables. Additionally, it is relatively intuitive to hard-code BNs, as the same conditional probabilities can be measured in the real world. However, this is a blessing and a curse. The BN also forces its users to translate the observations in the real world into static, discrete concepts. A lot of dynamic information and nuance tends to get lost in such a translation. If somebody wishes to employ a BN, they must always be mindful of the way in which they translate the language of that BN to reality and vice versa.

The architecture of BNs also create implications for their application. For one, the fact that it is modelled as a DAG means that it cannot represent

feedback loops by default. Another issue is that a BN by default acts as a momentary observation: a BN calculates probabilities of unobserved variables assuming that all other variables are observed 'now'. It only handles the past at best through data preprocessing tricks [18, 19]. And lastly, observed variables can only be read at best in a standardized format. Both of these prerequisites are in stark contrast to medical practice, which consists of textual clinicians' notes, highly dimensional patient symptoms and diagnoses and continuously developing patient states.

All these things considered, one can reason about the behaviour one could expect from a BN when optimising its configuration. First, there is a complex interplay of variables in a BN. Probabilistic relationships can cascade throughout the network to lead to different results. By the example of Figure 2.3, the state of B may influence the state of E, even though they are not directly connected. These chain reactions can be difficult to reason about.

Lastly, it is important to consider that a Bayesian Network is a model. Which means that it should be considered viable if it is closely represents reality, rather than form a perfect match. Its advantage of evidence-based, transparent reasoning about probabilities is an obvious argument in favour of its adaptation and the architecture of the BN should be malleable enough to fit in a clinical context.

More implications of these design decisions within the context of Promedas are discussed in Section 7.4.1, after the results of this thesis are known.

2.3 Explaining Promedas

2.3.1 Structural implementation of Promedas

Promedas has been proposed as a Bayesian Network model, with the goal to generate a differential diagnosis for patient cases [5]. In brief, Promedas takes clinical observations in a patient as entries, then is inferred to calculate the probability of each diagnosis. These probabilities can then be used to generate a differential diagnosis of the patient population.

At the very core of Promedas is a Bayesian Network, which relies on the Quickscore inference algorithm and its noisy OR-gate-based, two-layer structure (Figure 2.7) [17]. Promedas introduces a new layer above the diagnosis layer for conditions, which model risk factors for different diseases, like smoking.

The links between the diagnoses and the findings layers still represent the probability of a finding being present, given that a diagnosis is present. In other words, this is the sensitivity of that finding for a diagnosis. Links be-

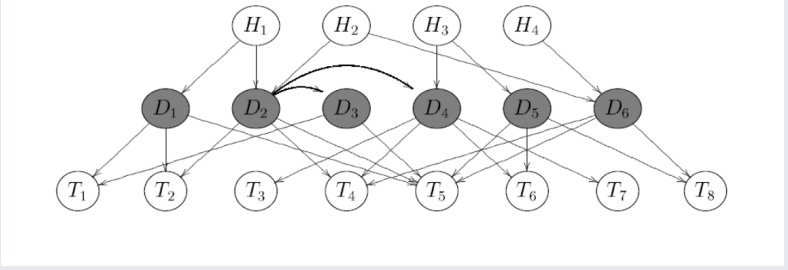


Figure 2.7: The architecture of Promedas as shown in [5]

tween conditions and diagnoses are defined by a risk ratio instead [20]. These sensitivity values and risk ratios can be found in medical scientific literature. Specificity values from the literature are not supported by Promedas, but estimated by dividing over the number of other connected diagnoses to the finding. Specificity values are much lower in Promedas than in medical literature in general, because Promedas handles a high amount of different diagnoses, as opposed to a typical cohort study's population with a single diagnosis.

In theory, by identifying links between diagnoses and findings and by supporting them through literature, Promedas should be able to generate a differential diagnosis that relies purely on scientific findings and mathematical operations.

Promedas additionally divides findings into binary, continuous and categorical modelling types. Binary findings represent the default True/False functionality, e.g. whether some symptom may be present. Continuous findings represent observations like blood pressure and are handled using cut-off points. Categorical findings are currently mostly implemented to represent different ECG wave morphologies. Continuous and categorical findings require additional logic to work with the rest of the network.

Promedas critically relies on this discrete, logical representation of dynamic variables. The discrete structure of the entities within the system, diagnoses, conditions and findings, are henceforth referred to as 'Promedas language'.

2.3.2 Organisation and cooperation

Having defined how the Bayesian Network exists, the rest of this section explains how the development of Promedas is organised. After the initial

design by Kappen et al., the codebase and database are now predominantly managed by Gumerova, A., who by all means has been the *code guru* of this project since.

2.3.2.1 Database

A database is used to configure the network. All parameters of the BN are stored in a Microsoft Access database. First, a list of all diagnoses, findings and conditions are defined, each in separate tables. For all categories goes that these entities have a name, a source, and are associated with a SNOMED ID. For diagnoses, a prior probability is also supplied; and they are placed in categories, thanks to the work of Bakker, G and Meijer, J [21]. As an example, stable angina exists in the 'angina pectoris' category, which on its turn exists within the 'disorder of the cardiovascular system' category. This way, it is possible to zoom out to different categories of diseases while configuring and inferring the network. Additionally, findings with a continuous or categorical modelling type are registered as well. Continuous findings have cutoff values for (extremely) below and above the normal range; categorical findings have additional categories associated to them. Findings are categorised to distinguish their path of establishment, for instance 'symptom' or 'ECG'.

Links are defined as well. Between diagnoses and findings, these are defined by the finding and diagnosis IDs and their sensitivity, in case of a binary finding for being True; a continuous finding for being in a given range; or a categorical finding for being some category. These sensitivity values are also supported with a source. The links between conditions and diagnoses are defined by the risk ratio [20]. This risk ratio also receives a source.

The same MS Access database also contains a list of patient cases, each of which is associated with a list of findings and conditions with their results, and known and new diagnoses. Associations between patients and their findings, conditions and diagnoses are henceforth referred to as 'observations'. Their source and creation date are added as well as a list of case-specific notes from the editors.

For all entries in this database goes that they are manually configured by experts, which in this case means that they are managed by Technical Medicine interns. The experts will henceforth be referred to as administrators or admins. Internship projects mainly target a smaller subset of this database.

2.3.2.2 Inference

Inference of the network is executed using a separate Matlab codebase, which is outside the scope of this thesis. The party that is responsible for inferring the model is henceforth referred to as the programmer. What is known is that it interprets the MS Access database before inferring the network using the Quickscore algorithm [17]. The results of this code are then stored in an Excel workbook, which contains a sheet for each patient case. Moreover, the inference algorithm also has some additional options, such as to infer assuming 0.5 sensitivity on all links (also known as a 'zero-leg model') or close to 1 sensitivity on all links ('one-leg model').

2.3.2.3 Inference results

The Excel sheets of these patient cases list once again all findings and conditions as well as the patient's known diagnoses and the list of correct diagnoses. The Excel sheet also displays a differential diagnosis by calculated probability of this diagnosis being present and another differential diagnosis by number of explained positive tests. For each of these differential diagnosis entries and the correct diagnosis, Promedas also lists their calculated probability, their prevalence, and for each positive and negative finding the sensitivities of the links with that diagnosis.

This way it is relatively intuitive to reason about the network's behaviour, as many patterns emerge here. One can see how many positive and negative tests are associated with the correct diagnosis and either discover that a case has an atypical presentation or that a link is missing in the network. Conversely, the absence of findings could be reason to review the sensitivity. Other patterns could be the existence of highly-specific tests where they shouldn't be, and thus add more links to that test. Iterations of these small updates have slowly but steadily made Promedas more accurate over the years.

2.3.2.4 Summary of roles

The roles surrounding Promedas have already been mentioned throughout the context, but are once more summarized in Table 2.1. These descriptions are not exhaustive. Note that many of the working points come from cooperation between different roles; the weekly meetings were essential to discuss objectives that transcended individual roles. The structure of the model and assumptions in its implementation were often-recurring subjects.

Role	Description
Supervisor	Oversee structural and methodological considerations within the model with respect to its performance
Programmer	Infer the network using the registered data
Administrator (admin)	Change the database that directs the prior probabilities and conditional probabilities that dictate the behaviour of Promedas in the population. E.g. Technical Medicine interns
User	The people that eventually use Promedas in practice, e.g. clinicians

Table 2.1: Roles of parties in Promedas

2.3.3 Summary

To summarize, Promedas can functionally be described as a white box model with inputs and outputs. The inputs of Promedas are observations in a patient: findings like symptoms and test results; conditions like risk factors from prior history or smoking; and diagnoses that are already confirmed. The output of Promedas is a differential diagnosis, with a probability for each diagnosis in the BN. The BN can be inferred for a patient with a set of these observations to generate a differential diagnosis for that patient. On a population level, on the other hand, we can observe the probabilities that make up the underlying BN and use those to improve its accuracy. Medical literature is by default the prime source of information for these population-level observations.

3. Organisation of this report

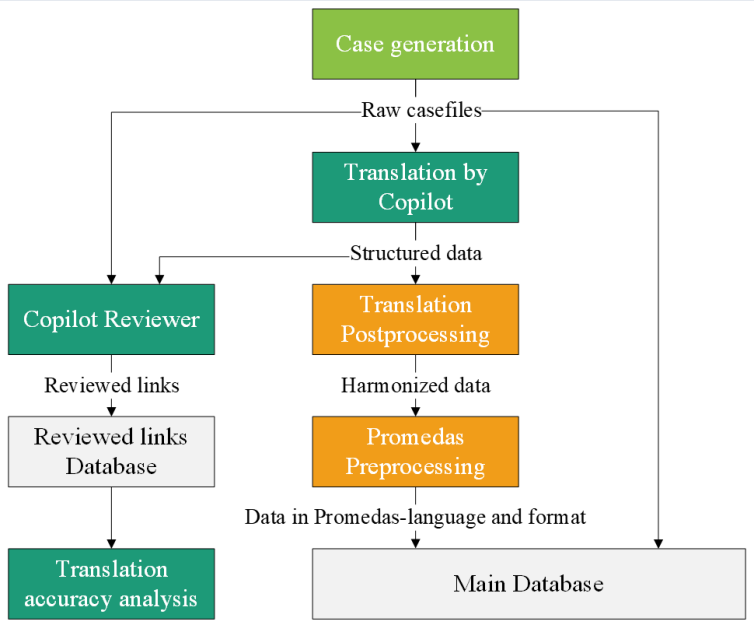
To reiterate, the goal of this thesis is to determine whether Promedas, as a Bayesian Network, can exceed the diagnostic accuracy of clinicians. The steps towards that goal were identified to be to expand the dataset, to translate unstructured patient data to structured 'Promedas language', to improve the efficiency of the development process of Promedas and to test changes to the optimisation of Promedas.

The development of Promedas is described in the next three chapters. Much has happened over the months between the start and the end of this thesis, which has also seen contributions from other Technical Medicine students. To funnel this into giant methodology and results sections would make the material hard to digest, hence each stage of the development is described with a brief introduction as well as methods, results and quick remarks sections each.

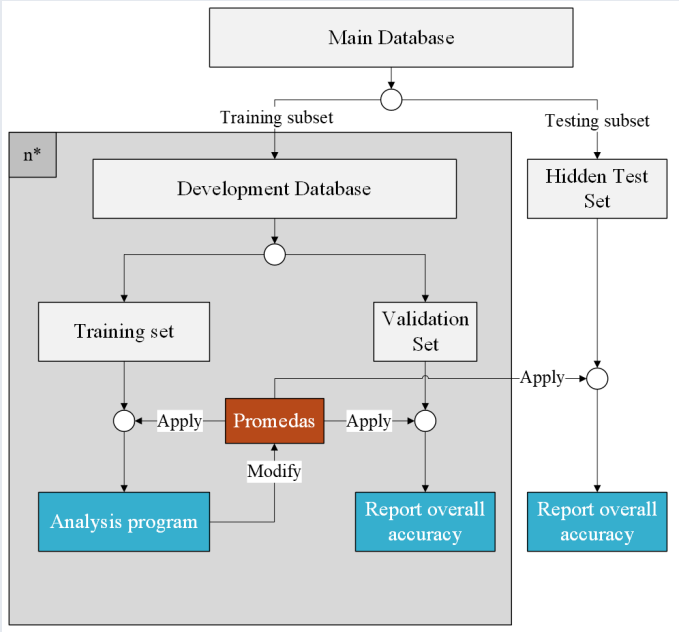
From a bird's-eye view, the development process involved 5 concrete milestones, split into two parts. First is the creation of unstructured and structured patient data (Figure 3.2a). This step includes generation of fictive patient case files (Section 4.1). These text-based files must then be translated into the standardized format of Promedas (Section 4.2). This section also elaborates how the structured data is post-processed to harmonize the data, as well as how it is further processed to match the input format of Promedas. Additionally, a program was written to conveniently review the accuracy of the translation step and used to analyse this translation accuracy (Section 4.3).

Second is the development of an optimisation or training cycle for Promedas (Figure 3.2b). This is structured very similar to the classic train-test-validate setup in Machine Learning, but distinguishes itself by handing control over the parameters of the network over to the user. The optimisation cycle at large is described in Chapter 5. For analysis of the results, another program was developed, as described in Section 5.2. The analysis program is used to inspire manual modifications for Promedas. Strategies towards this goal are described in Section 5.3.

The report wraps up with two thought experiments in Chapter 6, which were executed within the span of this thesis. The final discussion in Chapter 7 evaluates how the individual outcomes of the steps in the development process and the thought experiments stand compared to the main goal of increasing the accuracy of Promedas overall.



(a) An overview of the steps involved in creating the dataset.



(b) An overview of the proposed design-development cycle

Figure 3.2: A bird's-eye view of the development cycle

4. Data acquisition

This chapter describes how the dataset is created (Figure 3.2a). The first section explains how fictive patient cases were generated to serve as an unstructured dataset. The translation step from unstructured data into Promedas language along with its post-processing steps are elaborated next, followed by an analysis of the quality of this translation step.

4.1 Case generation

Promedas employed an old dataset of 37 patient cases before the start of this thesis. Each case being defined by their positive and negative findings, their conditions, their correct diagnoses and known diagnoses from the patient history.

This dataset had issues, however. The source of these cases could not be traced back to its unstructured raw case file. For this reason, it was impossible to backtrack why certain findings were made. This was problematic: the dataset performed well in about two thirds of the dataset, but many of the other cases presented trouble. Findings could be unexpected with some diagnosis, such as high pulse rate with sinus bradycardia; some cases could have highly complex interactions, such as multiple simultaneous cardiac arrhythmias; and in the dataset at large, some diagnoses were overrepresented where other diagnoses were completely absent. Moreover, over the years multiple students were expected to have altered patient cases as findings were added and removed from the database.

This was reason enough to create a new dataset to cover all diagnoses and the expected findings.

Unfortunately, it was not possible within the realm of this thesis to test Promedas on a real set of patient data. In the absence of access to real patient data, the validation set was instead generated using Microsoft Copilot, which was chosen for it being the Erasmus MC's supported and safe large language processing model[1]. This dataset could be used to generate a structurally correct dataset, which serves as a backbone for the rest of this methodology.

[1]This obviously has implications for the methodological soundness of the rest of the development cycle that I'd love to address right away, but unfortunately have to postpone until the discussion (Section 7.3).

4.1.1 Methods

To generate patient files, 5 templates were used to inspire MS Copilot's case generation. These templates follow the structure and style of cases available in HiX Zandbak, by employing similar headers, abbreviations, jargon and overall use of language. The cases do not resemble patient records as they were; their ultimate aim was to illustrate the style of reporting.

These templates were then added to MS Copilot (Figure 4.1). Copilot first reads a template and prompt 1 to initialize the chat (Appendix A.1). Through prompt 1, Copilot was essentially placed in a role for generating discharge letters for a fictive patient. Copilot was requested, upon the submission of a given diagnosis, to generate a patient with that diagnosis, fabricating any other information like age and sex. The prompt provided additional requirements on the header structure and requested additionally that Copilot would generate a list of citations from its own file that would contribute to its final diagnosis.

After initializing the chat, prompt 2 only submitted a diagnosis. Prompt 2 was sent a total of 5 times per chat, each time to create a new case in the same template with a different, random diagnosis (Appendix A.2). Diagnoses for the query were selected according to a uniform distribution. The returned cases were stored as fictive patient records in a .txt file. The number of 5 cases per chat was arbitrarily chosen, but intended to minimize the chance of Copilot forgetting its task while wasting as little time as possible on reinitialising the chat.

The documents that MS Copilot generates are ultimately aimed to be structurally very similar to patient records in the Erasmus MC, while their manifestations are entirely made up.

MS Copilot does not receive more requirements, such as the list of findings and conditions in Promedas, in order not to bias the patient files to accommodate Promedas language. This way if Copilot generates a symptom that does not yet exist in Promedas, it could be reason to investigate that symptom's significance for said diagnosis. The downside is that misunderstandings may arise from the differences between Copilot and Promedas language.

Since the data extraction procedure from MS Copilot was predictable and repetitive in its nature, it could be automated with a Python script [22]. Since Microsoft Copilot does not provide an Application Programming Interface (API) yet, the script relied on the libraries PyAutoGUI and Pyperclip for interaction with its UI and clipboard [23, 24]. The code is stored within the Promedas Teams files and can be reused in the future.

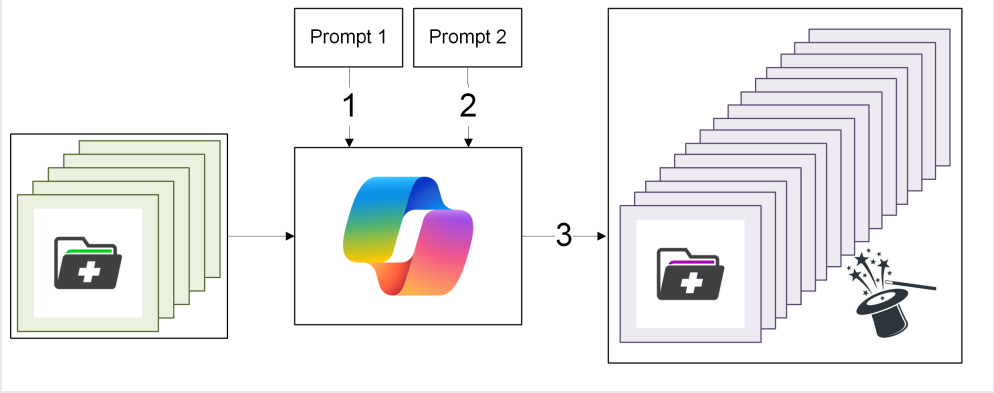


Figure 4.1: Fictive case generation using templates. (1) One of the template files and prompt 1 are loaded into Copilot to initialize the chat for generating cases. (2) Prompt 2 is sent to Copilot to generate a case with a random diagnosis. (3) The new case Copilot generates is saved as a fictive patient case.

4.1.2 Results

By using the code, 150 fictive patient files could be generated by Copilot per hour. The code was finished after 600 distinct cases were generated. Case files that weren't properly copied into a text file were discarded, which would leave a total of 512 case files at the basis of the dataset. The cases have all received any of the 58 cardiovascular diagnoses in Promedas.

The contents of the file were structurally consistent with case files in the Erasmus MC. A few exceptions lie in the fact that tract anamnesis was not included in both the templates and the prompt, causing Copilot not to generate such sections. Additionally, the generated files appeared considerably more verbose than everyday doctor's notes, missing abbreviations and jargon.

4.1.3 Discussion

The fact that the format of the case files is largely consistent with the format of real doctor's notes is good news. This has to do with the next step of translating the unstructured data into structured data (Section 4.2). If that step turns out to be executed successfully, then the general structure of the case files should not be reason for worse performance in real cases. On the other hand, the files seem to be lacking in jargon and abbreviations, thus the accuracy of the translation step may end up lower in the case of real doctor's

notes.

The implications are worse for the optimisation cycle later on (Section 5). Firstly because the tract anamnesis is missing in all cases and, therefore, it is likely that some symptoms are underrepresented in the dataset. Findings that may predominantly be missing are negated findings, as tract anamnesis aims to rule out some symptoms. These symptoms are also less likely to be main complaints, such as swelling of the ankles or nocturia. But more important is that it is impossible to verify to what extent the relations between diagnoses, findings and conditions are clinically plausible. Thus, this step may retrieve data that is structurally consistent, but may not be realistic.

4.2 Translation step

With the 512 new cases at disposal, the rest of the backbone of this project could be created. The goal of this specific step is to translate the unstructured clinical data into Promedas language.

This is mandatory regardless of the input data: these cases had to be interpreted in one way or another; but in the case of a final product, the doctor's notes that must be analysed have to be translated in a way too. The alternative would be to force clinicians to use a form, but that would be rather restrictive.

4.2.1 Exploring alternatives

The translation step proved to be surprisingly challenging, but was eventually solved by two colleague students, Q. Langens and S. Kuiper. In their works, they respectively explored the routes through which this translation step could be done and how a prompt could be developed to reliably execute this task [25, 26].

The report of Q. Langens mentions multiple alternatives: Natural Language Processing (NLP) and Large Language Model (LLM) systems [25]. Medspacy was briefly tested as an NLP but quickly showed to miss many terms and failed to interpret contextual information. Microsoft Copilot was considered as the LLM alternative, again for the reason that it is the recommended system in the Erasmus MC. LLMs turned out to be more promising in terms of quality and speed, but could hardly be automated without an API.

In the report of S. Kuiper, the LLM path is further explored [26]. This report contains a brief literature review on prompt engineering strategies

(Figure 4.2). From the discussion section of this review, the following excerpt elucidates the different strategies:

> Prompt Design focuses on the structure and content of the prompt itself. Strategies in this category include Chain-of-Thought Prompting, which encourages the model to reason step-by-step before providing an answer; Custom (Rule-Based) Prompting, where prompts are manually designed based on expert knowledge or predefined rules; and Task Specific Examples, in which demonstrations are included that closely match the intended task. Also, part of this category are Multi-Shot Prompting, where multiple examples are included to guide the model's output; Role Prompting, which instructs the model to adopt a specific identity such as "medical expert"; Premise/Context Prompting, which incorporates relevant background information into the prompt; and Single-Task Prompting, which limits the prompt to one clearly defined question or task.
>
> Data Preparation involves modifying or structuring the input data before it is presented to the model. This includes Chunking, which splits large texts into smaller segments to make them more manageable for processing; Chunk Selection, which involves identifying and selecting only the most relevant portions of the text; Preprocessing to Structured Data, where inputs or outputs are formatted to match predefined data structures; and Translate to English, which involves translating clinical texts to improve performance with English-trained models.
>
> Execution Flow refers to strategies that structure or guide the sequence of steps in the prompting process. This includes the Pipeline Approach, which breaks down complex tasks into smaller subtasks, and introduces a structured prompt format with context, instructions, input data, and output specifications. Multiple Iterations, in which outputs are refined through repeated prompting cycles; Iterative Prompt Refinement, where the prompt itself is adjusted over time based on model performance or feedback; and Retrieval-Augmented Generation (RAG), which enhances output quality by combining knowledge retrieval with text generation.

Based on the list of these strategies, multiple prompts were designed and tested on 5 handwritten fictive cases. The seventh and last prompt proved to be highly accurate in the use case of translating these handwritten fictive cases into Promedas language, being based on a combination of 7 unique strategies in prompt design, 2 strategies in data preparation and 4 strategies in execution flow. The prompt achieved a sensitivity of 0.913, a specificity of

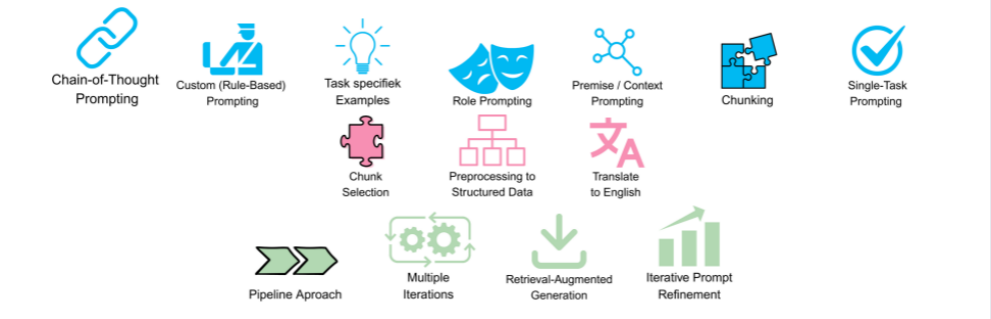


Figure 4.2: An overview of prompt engineering strategies, copied from [26].

0.99 and an overall accuracy of 0.975, even surpassing the accuracies of one general practitioner and another student[2].

4.2.2 Automation

This translation step could be automated too, using a similar approach to the dataset creation step. The prompts had to be adjusted slightly to harmonise the output format of Microsoft Copilot (Appendix B). All steps had to be executed over the UI, without an API for Copilot at disposal. A new chat was opened for each patient case. The first prompt was then sent to assign the role to Copilot. After receiving an answer, the second prompt was sent with the case file attached, querying Copilot to break up the text into digestible bits. The third prompt was sent with the structure of Promedas attached, querying Copilot to indicate for each discretized entity in Promedas what the outcome is for the patient, ignoring any entries that do not exist in the case file. Prompt four kept querying Copilot until all entities were seen. The code further relied on error catching logic to avoid crashes when these steps couldn't be executed.

Copilot was requested to stick to a given format when returning the results, which made the response string predictable. The response was copied to clipboard and deconstructed using a regex pattern, which was designed to match the response format. For the case under consideration, the identifiers of each finding, condition or diagnosis were added to a database if present, along with the outcome and a citation from the case file (Appendix C.1). No additional processing was done during the translation stage, but the database

[2]It's a shame that this report is not publicly available. I have found myself looking back at this report quite a lot over the last months.

was manually checked on cases with suspiciously few or many observations. Suspicious cases were rerun.

The naive translation dataset was forwarded to a different environment that is further elaborated in Section 4.3. This program was developed to review the accuracy of this translation step in this new independent dataset.

4.2.3 Translation post-processing

After Copilot translated the case, the data was post-processed to harmonise it. Copilot turned out to be inconsistent in reporting outcomes. For example, in continuous findings, Copilot would report units sometimes while reporting the number alone at other times. Moreover, Copilot was not prompted to distinguish diagnoses from prior history from new diagnoses. Diagnoses were therefore manually set to 'KNOWN' if they existed in prior history, or 'NEW' if they were found during this visit. Diagnoses needed no more post-processing after this.

The conditions in Promedas are all binary. The returned outcome values from Copilot would require simple string matching logic to be converted into TRUE/FALSE variables.

The findings were split by their type: binary, categorical and continuous. Binary findings required similar simple string logic to the conditions. Categorical findings were naively mapped to one of 18 keys by looking up any of the 50 associated substrings in the outcome field. Numerical outcomes were split up into multiple fields when they held a number: a relational operator; the coefficient; A base and exponent in case of scientific notation; and the numerator and denominator of the measuring units.

The post-processing step would only raise warnings when entries behaved unexpected, for instance when it encountered textual results on numerical findings, but did not remove any information from the database. The warnings could be used in part to finetune the logic of the program to cover more edge cases that did not exactly match the aforementioned patterns.

4.2.4 Promedas Preprocessing

The preprocessing script would take the harmonised dataset and adjust it for the input format of Promedas. For this, a few measures were made. In the case of conditions, any conditions without a clear Boolean value were removed. Diagnoses were dropped if it was unknown whether they were known from prior history or new.

Similar to conditions, binary findings were removed when they did not match a clear Boolean value. Categorical findings were set to NORMAL if

they contained a negation and removed when their harmonised key did not exist for the finding ID they were supposedly associated with or when no key matched at all. Continuous findings with numerical outcomes would have their coefficient adjusted so that it would match the exponent and unit prefixes in Promedas. Next, a threshold was applied to the coefficient to indicate their degree of deviation: VERY LOW, LOW, NORMAL, HIGH or VERY HIGH. The continuous findings with textual results would be matched to another dictionary of keys and substrings, assuming the clinician properly thresholded this variable by themselves. All entries that couldn't be interpreted to match any of these degrees of deviation were removed from the dataset.

Both steps would produce log files, respectively displaying warnings for entries that showed issues in the harmonisation step or printing which entries were dropped on what ground. The final file would come in a raw and a truncated format. The raw entries still stored information such as citations from the original document and intermediate steps, whereas the truncated file would filter out all data that wouldn't be relevant for Promedas (Appendix C.2).

4.2.5 Results

Copilot would typically take between 3 and 5 minutes to process one patient case, which was much shorter than manual translation, which came down to approximately 20 minutes per case on average [26]. Translating the entire dataset would take approximately 6 weeks, though. The script was not completely robust against inconsistencies in Copilot's user interface and Copilot would regularly time out or stop responding[3].

The final dataset managed to have a translation of all 512 fictive patient cases. Within this dataset, 3401 condition observations, 1483 diagnosis observations and 19182 finding observations were registered. 7 cases had suspiciously few or many observations, but these turned out to be reproducible after being rerun. A copy of this database of raw output was forwarded to the Copilot reviewer environment, to review the accuracy of the translation step at large. The output was then post-processed for harmonisation as described above.

During pre-processing for Promedas, the findings were split up into 14344 (74.8%) binary findings, 4503 (23.5%) continuous findings and 334 (1.7%) categorical findings (Figure 4.3). Out of the binary findings, 41 (0.3%) en-

[3]I assume the latter was an intentional measure from Microsoft's end to relieve stress on the server, in which case I hereby apologize to Microsoft.

tries did not match any of the given substrings, all of whom were dropped. Out of the continuous findings, 611 (13.6%) did not hold a numerical value. 36 (0.8%) were dropped because they could not be mapped to a degree of deviation. Another 70 (1.6%) elements were dropped because of unsolvable unit mismatches. Out of the categorical findings, 33 (9.9%) did not match any of the substrings in the dictionary. 6 (1.8%) of these were removed after having included negations as NORMAL values. Another 28 (8.4%) categorical findings were removed because their outcome category did not match the finding ID. A total of 19001 (99.1%) finding observations made it into the final dataset.

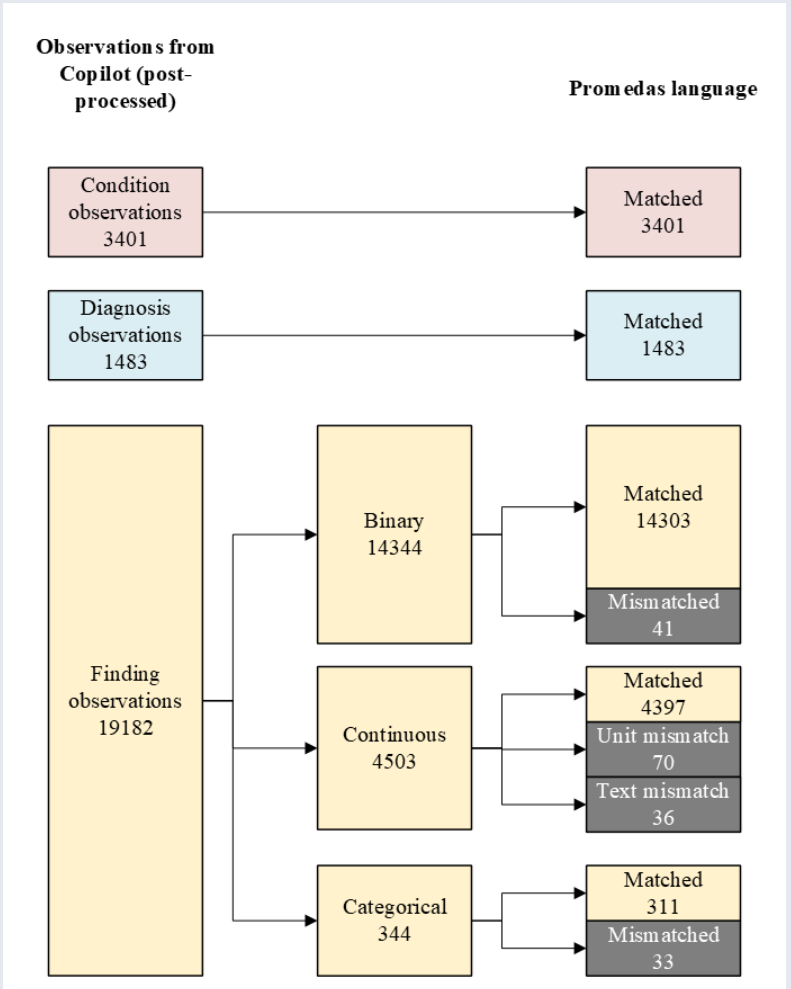


Figure 4.3: An overview of the observations that could be mapped to Promedas language

No string mismatches occurred for the condition observations, thus no conditions needed to be removed because they could not be matched to the input format of Promedas. Similarly, no diagnoses needed to be removed because they could not be matched to the input of Promedas.

4.2.6 Discussion

The final dataset turned out to be sizeable, compared to the original dataset, being roughly 14 times larger in terms of patient cases and roughly 35 times larger in terms of the number of observations. This was good news: the size of this dataset could create insight in how the links in Promedas would relate to the observations in this population. This would be especially valuable with real patient data.

An important observation is that most patient cases would be associated with multiple diagnoses after translation, rather than the single diagnosis this case was originally queried with. Especially diagnoses like essential hypertension and sinus tachycardia, which come down to respectively an unidentified elevated blood pressure and an elevated pulse rate alone, would be much more frequent than more specific diseases like cardiac arrhythmias. This will become relevant in later stages, as the assumption that all diagnoses are present equally often is no longer valid.

The major issue of this translation step lies in the fact that it depends on Microsoft Copilot. It is an odd choice to use an LLM in an application that is principally set up to avoid the risks of hallucinations, which LLMs are known to be prone to. One cannot be sure that an LLM like Copilot ever achieves a rate of 100% translation accuracy, which might affect the performance of Promedas overall as it reaches higher accuracies.

Ideally, one would adapt an analytical solution for this translation step instead. How such an application would look is difficult to envision from a Technical Medicine standpoint, but might be an interesting topic for linguists[4].

In defence of Microsoft Copilot, it does make for a reasonable practical approach. It managed to exceed the translation accuracy of one domain expert

[4]One of the things that jumps to mind is the PSOLMO-method that is commonplace in high school classical language courses in The Netherlands, which essentially is a formula for translating phrases from ancient Latin or Greek. How one would go from these dissected phrases to such discretised entities is more challenging already, though. I would have loved to take a deeper dive into this, but find myself missing too much knowledge of the basic principles of linguistics to make a solid attempt. Therefore, I deemed it more appropriate to acknowledge this boundary and formulate that this would be better trusted in the hands of the right experts.

within a comparatively short processing time. Moreover, one can question whether an analytical tool would ever reach or exceed the accuracy rate at which Copilot manages to make this translation. Analytical tools are likely to remain more vulnerable to discrepancies in the end user's use of language, regional differences, dialects and typological errors. And ultimately, the development of such an analytical tool that covers these edge cases may be more complex than the naive development of an LLM.

Additionally, Bayesian Network models are known to be fairly robust against small discrepancies: missing data entries do not disrupt the execution of the model, but merely are assumed not to be observed; and a single hallucination are only likely to outweigh the vast amount of correctly interpreted variables when they are highly specific for a small group of diagnoses. One must also note that this translation step is currently run on the entire patient file, whereas a large part of the medical data infrastructure already exists in a standardized format by default. If the translation step were to be skipped for all fields that already exist in a structured form, the accuracy of this step would only rise.

Nevertheless, the fact that the translation step currently doesn't and may never reach a 100% accuracy should be ample reason to ensure its transparency for the end user. Promedas should ultimately not only provide their predicted diagnoses, but also justify on which grounds it came to that conclusion. These grounds must point to the findings that the clinician has entered, and the clinician should have the means to overwrite this discretized data when they detect an error. Assuming the 97.5% accuracy rate is for the files overall, this inconvenience would occur once in every 40 findings, which should not be too much overhead for the clinician.

4.3 Translation Reviewer

The next goal was to measure the accuracy of Copilot in translating unstructured data into Promedas language. Therefore, the observations between patients and their findings, conditions and diagnoses had to be reviewed. These observations will be referred to in this section as 'associated entities'. When the translation prompts were first tested on handwritten cases, it turned out that humans had a low accuracy [26]. This is presumably because of the large quantity of unique findings, conditions and diagnoses in Promedas, all of which might be present or absent in the case. Since no other programs could be found to provide a convenient interface, one was created manually to compare the output of the translation step to the raw text file. These next subsections describe the operating principles of the program and how

its results could be used, then reports the translation accuracy of Copilot in these new fictive cases.

4.3.1 Program Overview

To enable reviewing, the program requires both the folder of text files that were analysed and the database that was generated by the translation step. The program displays the original text file and highlights the citations attached to that file (Figure 4.4). The highlighting colour depends on the category of the associated entity (finding, condition or diagnosis) and whether it is implied by the citation or directly mentioned.

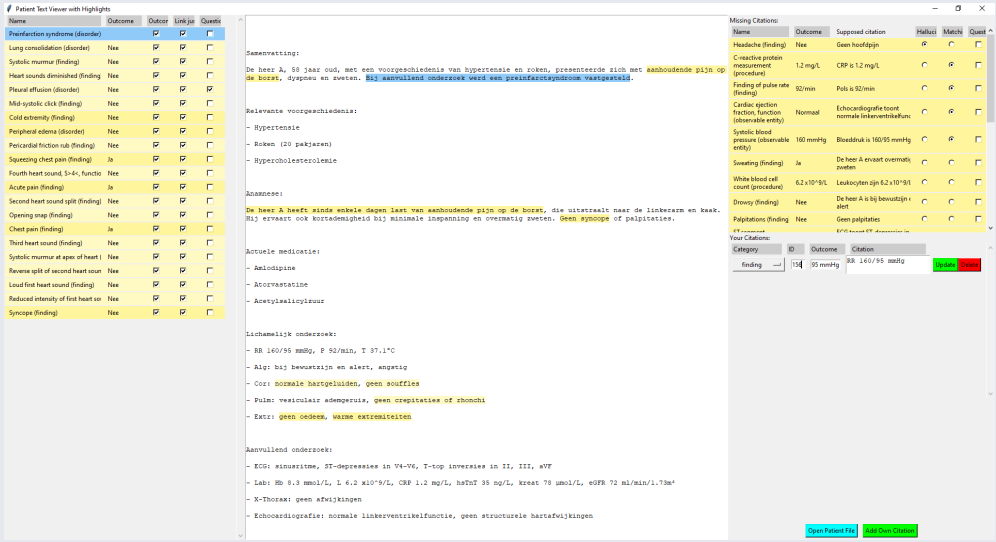


Figure 4.4: The user interface of the Translation Reviewer program with a random fictive patient case included as an example.

The program essentially serves as a database interface (Appendix C.3). The database on its turn distinguishes three types of observations: DetectedLinks, NotFoundLinks and OwnLinks[5]. Each type of observation has different criteria by which they are reviewed.

DetectedLink entries are observations whose citation could be matched to at least one section of the text. Three fields are attached for review. The first asks whether the outcome of the observation is appropriate in context;

[5]I sincerely apologize for the inconsistent naming here. There was no initial distinction between science-based links and individual observations and to correct in order for it to be consistent with the code would require a large overhaul of the codebase.

for instance whether it distinguishes new diagnoses from diagnoses in prior history (which are conditions), or whether it correctly handles negations. The second reviews whether the observation is justified at all, so whether the citation logically corresponds to the reported entity and outcome. Wrongly reported lab values or simply false associations between associated names and the citation would fail here. Lastly, one field questions whether the observation is questionable, i.e. when the clinician too struggles to interpret a citation as the name of an associated entity. Reasons for this might be region- or dialect-dependent use of language or vague complaints.

NotFoundLink entries are observations whose citations could not be matched to at least one section of the text. The error type field in this case would question the cause of this mismatch: either the observation was hallucinated by the translation step; or the given citation is correct but could not be aligned to a part of the text. This type of observation was also associated with a questionable field, with the same definition as under DetectedLink entries.

OwnLink entries are observations that could be added by the user, when they see an informative text snippet that wasn't observed by the translation step. The OwnLink entries hold fields for the category and identifier of the associated entity, along with a citation field. The user can optionally leave out the ID field when they identify a text string that does not exist in the Promedas database at all.

4.3.2 Data Analysis

The reviewed observations database would later be analysed to provide insight into the accuracy of the translation step. The observations classification could roughly be turned into a binary confusion matrix. Within this confusion matrix, the four fields are composed of the following:

- True Positives were DetectedLink entities that were neither inappropriate nor unjustified as well as NotFoundLink entities that were mismatched against the text.
- False Positives were DetectedLink entities that were either inappropriate or unjustified as well as hallucinated NotFoundLink entities.
- False Negatives were OwnLink entities with an associated category and ID in Promedas.
- True Negatives were the total number of findings, conditions and diagnoses in Promedas minus the sum of all other fields in this confusion matrix. I.e. the total number of potential associated entities in

Promedas that both the translator and the reviewer agree are not reported in the text.

The relative amount of questionable items are reported as a separate statistic. The OwnLink entities without an associated ID are not considered in this analysis but instead moved to a different environment, where they could inspire additions to the Promedas database.

This confusion matrix and the questionable items statistic could be created for each considered case. The binary confusion matrix is further used to calculate sensitivity, specificity, PPV and NPV of Copilot for detecting observations in the free text. The results are summarised in a binary confusion matrix with 95% confidence intervals for the entire population.

These results are also analysed per entity type, so findings, diagnoses and conditions each. Findings are further subdivided into binary, continuous and categorical modelling types. Their confusion matrices are not reported, but outstanding observations are described.

The observations were reviewed both by the author and by another expert.

4.3.3 Results

17 cases with a total of 860 observations were reviewed by the author and another 10 cases with 513 observations by the expert.

The overall accuracy of Copilot was high, at 0.968 (95% CI: 0.955-0.981) for the author and 0.996 (95% CI: 0.993-0.998) for the expert (Figure 4.5). Copilot only has a comparatively low positive predictive value overall at 0.851 (95% CI: 0.797-0.905) and 0.978 (95% CI: 0.967-0.989), meaning that there is a relatively large portion of false positives among positive findings. The dominant group of false positives comes from Promedas entities that do not match their text snippet.

The results were fairly consistent across each entity type, with all sensitivity, specificity, PPV and NPV values roughly staying above 0.9. Comparatively vulnerable were conditions and diagnoses. Conditions were found by the author to only have a PPV of 0.738 (95% CI: 0.648-0.828). The PPV for diagnoses was 0.802 (95% CI: 0.689-0.915). The categorical findings category had few reports overall within the dataset and, despite having good overall metrics above 0.9, still lacked some statistical power due to their relatively small number of observations.

Out of the 860 observations, 64 were found to be questionable by the author in total. The expert reported a total of 2 questionable observations in their set of 513 observations.

	Copilot +	Copilot -	Total
Reviewed present	0.146 [0.127-0.164]	0.004 [0.001-0.007]	0.851 [0.797-0.905]
Reviewed absent	0.028 [0.016-0.041]	0.822 [0.798-0.846]	0.996 [0.992-0.999]
Total	0.969 [0.942-0.996]	0.966 [0.951-0.982]	0.968 [0.955-0.981]

(a) Confusion matrix of translation review by author

	Copilot +	Copilot -	Total
Reviewed present	0.173 [0.144-0.203]	0.001 [0.000-0.002]	0.978 [0.967-0.989]
Reviewed absent	0.003 [0.002-0.005]	0.820 [0.791-0.848]	0.999 [0.997-1.000]
Total	0.994 [0.986-1.003]	0.996 [0.994-0.998]	0.996 [0.993-0.998]

(b) Confusion matrix of translation review by expert

Figure 4.6: Confusion matrices of overall link reviews.

4.3.4 Discussion

The results of manual reviewing were roughly on par with the results from S. Kuiper [26]. Knowing this high accuracy rate, translation inaccuracies can be assumed to be of minimal influence on the dataset in later stages.

That being said, Copilot can be relied upon as a fairly accurate means to translate cases from raw text into Promedas language. Some additional improvements may be made to avoid false positives, like hallucinations and wrong citations. This might become relevant for conditions and diagnoses. Categorical findings require more evaluation.

One issue that showed up was that MS Copilot would still reason about diagnoses, even though it was explicitly queried not to do so. One frequently recurring example is that Copilot would highlight sections of elevated pulse rates to diagnose a patient with sinus tachycardia. This explains in part why the PPV of diagnoses remains comparatively low, because these reasoning steps would be interpreted as a false positive. Since the translation outcomes were directly used to create the main dataset, it must also be noted that it in part reflects the reasoning of the LLM.

5. Optimisation cycle

5.1 Optimisation cycle

With a much larger fictive dataset at disposal, focus turned towards the optimisation of Promedas. This has been the main task of the administrators (experts) in Promedas.

5.1.1 Former method

Promedas had in the past been configured through iterations. The admin would receive the results of Promedas on a dataset, containing for each patient case the findings and conditions; the known diagnoses; the top of the calculated differential diagnosis; and the correct diagnoses with their assigned probability scores. The admin would receive additional information about the related links for each of these diagnoses.

The admin could then reason where the model fails, for instance due to wrong prevalences or conditional probabilities; a shortcoming of findings; or a lack of links. Such an improvement could then be added to Promedas, with the goal to increase its accuracy.

This method proved to be pragmatic in steadily increasing the accuracy in the old patient set, but also had a few caveats. Firstly, it was rather tedious and time-consuming to both identify issues in the population with this top-down method, and then to find appropriate literature and a sensitivity value.

Additionally, this strategy is at risk of confirmation bias and a form of overfitting. The admin is inclined to find literature and adjust the network to match the population, and conversely the network will also accommodate this same population. An important demonstration of this issue came from the TreLynCa model for laryngeal cancer, which had a perfect accuracy on the original dataset, but dropped to 0.769 when tested in a new patient population [27, 28, 29].

5.1.2 Proposed optimisation cycle

A new optimisation cycle was proposed with respect to these caveats (Figure 5.1). This cycle takes inspiration from the train-test-validate strategy that is commonplace in Machine Learning [30]. The reason for this was to counteract the risk of confirmation bias and overfitting without sacrificing

the pragmatism of the former method, that used patient cases to inspire changes to the network structure.

From the 512 initial patient cases, 108 (20%) were randomly split off to create the testing set. The only criterion for this split was that the training set contained at least 1 instance of each diagnosis. Stratified sampling was difficult to conceptualise beyond this point, since all 58 diagnoses in Promedas make up the output space. It wasn't really necessary either for this purpose. The diagnoses that were overrepresented in the remainder of the dataset were predominantly 'intermediate' diagnoses, which might as well act as a finding for many of the other diagnoses[1]. As an example, the two most common diagnoses in the input dataset were sinus tachycardia and left ventricular hypertrophy. Other than those, the BN assumes mutual independence of all links, so if a diagnosis is to be overrepresented, it should not affect the probability of the underlying diagnosis much.

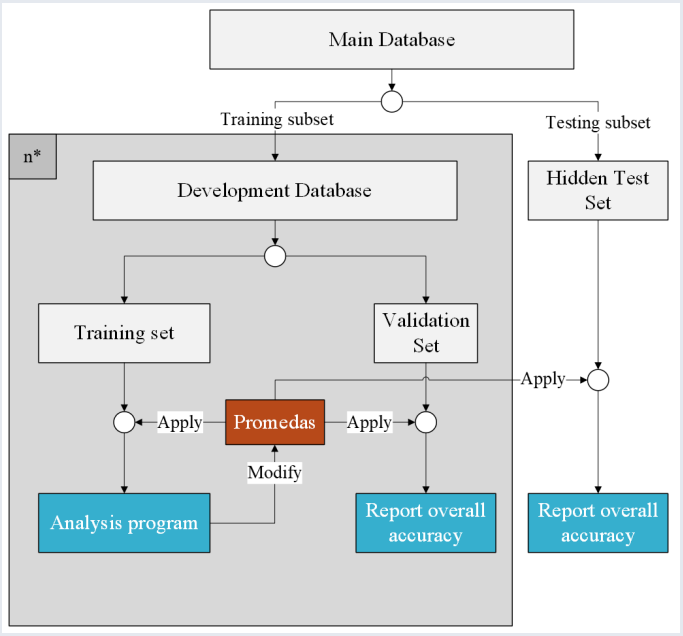
The testing set was to be stored only on the PC of the programmer, so that the administrators could not observe it. The administrators instead used the remaining 404 samples to improve the network. Admins on their turn take a random subset of the training population on each iteration, then use it to guide changes to the network. A script was implemented to create a transparent comparison between the population and Promedas to guide such changes (Section 5.2). The remaining samples on that iteration can be used to stand in as an 'independent' test set, which shows the results with less transparency (Section 5.2.4).

While this approach does not completely get rid of the risk of confirmation bias, it at least makes it measurable in the future. The test set in the programmer's environment can be used to analyse the accuracy of Promedas on unseen data. The programmer can use the same nontransparent analysis code as described in Section 5.2.4. The difference between the two datasets then indicates how much accuracy is lost by focusing on the training set alone.

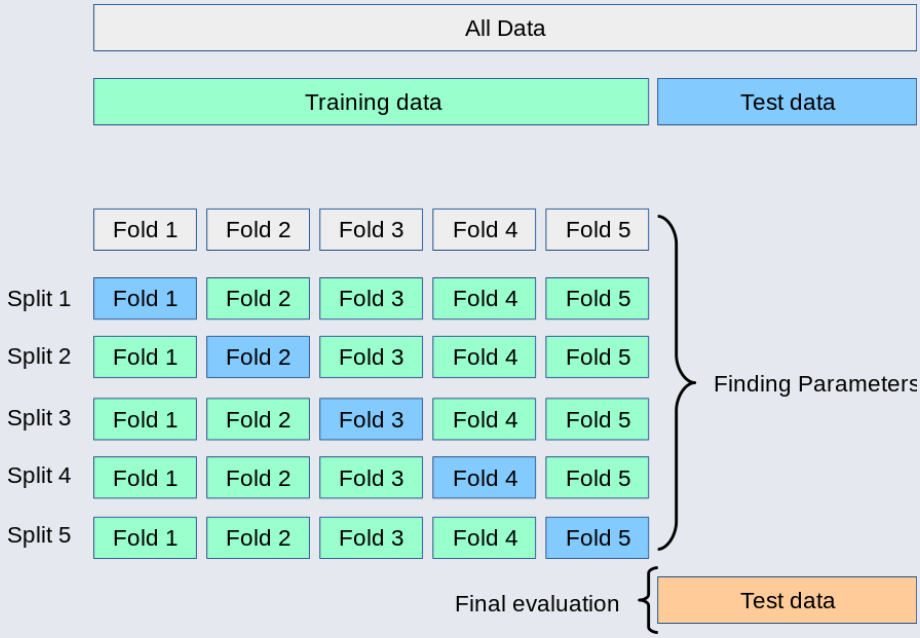
5.2 Analysis program

A program was written to mitigate the other issue, which was the tedious experience of identifying and changing issues. The main goal of this program was to provide insight into the BN structure, to guide the administrator in detecting shortcomings and proposing changes for issues in the structure of Promedas.

[1] These intermediate diagnoses have been a bit of a recurring theme, I am going to pay some attention to it in Section 7.4.1



(a) Proposal



(b) Inspiration

Figure 5.2: A diagram of the proposed optimisation cycle, compared to a diagram of a default train-test-validate strategy, adapted from [30]

This section elaborates by which intentions this program was written and how this program was implemented. It is not supposed to be a final product, if anything it would be a proof of concept (POC) application. This section and the program's development did not utilise standardised software design principles like MoSCoW prioritisation, as they would have been excessive for the small scale of the program.

5.2.1 Required features

As indicated above, the main goal of this program was to create more insight into Promedas. The main factors of influence on the model's behaviour could be reasoned about from the very inception, namely:

- How do the probabilities of Promedas' differential diagnosis compare to the correct diagnosis in each individual? Have the correct diagnoses been assigned a high probability and have the other diagnoses received a lower probability? And if not, is it reasonable to assign a relatively high probability to that diagnosis given the findings in that individual?

- How does the point above work out for the population at large? What probabilities have been assigned to each correct diagnosis, on average? Are they being mistaken for other diagnoses and, if so, can these be explained through known comorbidities or simple similarity between the two?

- How do the links in Promedas compare to the same observations in the population at large? If a link between some finding and some diagnosis exists in Promedas, does that then also mean that patients with that diagnosis also have that finding? And how does the sensitivity value in Promedas compare to that in the population?

This list is not exhaustive, but would be a solid start for the program under development. Section 5.2.5 goes on to list other functions that would be interesting for future developments.

Other than this function, maintainability and expandability would be important quality attributes of this program, as this program is going to be expanded on by future students in order to provide more insight into Promedas. Hence, the underlying codebase had to be written in such a way that future, unfamiliar technical medicine students would understand.

Other quality attributes received much lower priority: usability was considered in the form of defensive programming and by changing the layout,

whenever necessary; and performance would only be addressed when the program could quickly be sped up. Other attributes like security, availability, reliability and so on were hardly relevant in this stage.

5.2.2 Proof of concept description

The program evidently had to take the form of multiple graphical interfaces. Matlab version 2021b was the programming language of choice in the face of this, as it is optimised for vectorised data processing[2] [31]. Matlab app designer could be used to design the user interface (UI) of the program [32]. The rest of the program was written in an object-oriented form.

The app consists primarily of multiple tabs. Each tab presents the input- and output-data of Promedas in a way that illustrates the factors of influence as mentioned before. The third tab for links between findings and diagnoses had to be split into three subtabs, depending on the modelling type of the findings. I.e. binary, continuous and categorical findings each needed to be displayed differently.

Classes were introduced to distance the logic of the program from the code within the Matlab app file (Appendix D). Note that classes only serve as a template for the arrangement of data. One can create an 'instance' of a class when they want to have an entity according to that template. One can think of a class as a blueprint and of an instance as the actual house that is built from that blueprint. The main file's task is to generate these class instances together with the UI, so that they can form a functional whole

The classes were created so that their instances could be arranged in a tree structure, similar to the tabs in the user interface. These so-called nodes would store pointers to objects in the UI to facilitate the initialisation of and interactions with the UI. Conversely, UI objects would store pointers to these classes in their 'UserData' fields. Code was kept on the side of the nodes as much as possible, because they allowed more control than Matlab app designer.

Each of the bottom nodes receives data upon initialisation for their specific purpose, processes it and displays its results. For each of the tabs this looked as follows:

- NHTab or the 'Naive Heatmap' tab displayed for each patient their differential diagnosis. Patients are on the vertical axes and all diagnoses in the system are on the horizontal axis. Each intersection receives a

[2] I only later realised that this is not sustainable for future TM students, since Matlab has been phased out of the study program to make place for Python. So the codebase will have to be translated to Python at some point in the future.

colour, representing the probability Promedas assigned to the diagnosis for that patient. Another blue box would be drawn around each box if that diagnosis was actually found in that patient. In its desired state, all boxes with a blue outline have a yellow (close to 1) probability, while all other boxes have a dark green (close to 0) probability.

- CMTab or the 'Confusion Matrix' tab displays a summary of the population. Correct diagnoses are on the vertical axis this time. For each correct diagnosis, all patients with that diagnosis were selected. Their differential diagnosis probabilities were averaged over the entire selection. These averaged probabilities were then laid out against the horizontal axis. This time the diagonal would preferably be yellow and entries outside the diagonal would be dark green, unless such 'comorbidities' or 'co-occurences' are expected.

- The last group of tabs, FDBinTab, FDConTab and FDCatTab, compares links in Promedas to observations in the population. The vertical axis contains once again correct diagnoses, while the horizontal axis contains the findings. At each intersection, the fill colour represents the sensitivity based on the population alone. The border colour represents the sensitivity that is stored in Promedas, being a dashed white line if it exists without a known sensitivity value. Close calls between Promedas and the source data additionally receive a purple ellipse, indicating that they should be in order. A callback was put into place to highlight some outstanding links for review, which were stored in an output file. Each of these subtabs do these same tasks, but they are plotted separately to accommodate their different outcome options.

5.2.3 Purpose of the program

Due to this program, the optimisation cycle of Promedas became more instinctive. The reasoning from before is in a large part reduced to comparing colours. Students that were introduced to the program tended to be able to make sense of it within a few days.

With this implemented, it would only take one more step to close the loop of this optimisation cycle, which is described in Section 5.3. The output of the program can be fed into this final step to inspire potential links between findings and diagnoses in the literature.

5.2.4 Nontransparent analysis

A smaller script was added as well to analyse results without providing transparency into relations within the dataset. Central to it is a confusion matrix, similar to the one in the main analysis code. The nontransparent analysis file would further calculate the average probability of each correct diagnosis in the differential diagnoses and their average and median positions. These last two metrics would be calculated naively as well as adjusted to ignore other correct diagnoses in a better position.

5.2.5 Maintenance and development

Although this POC makes for a solid backbone, a number of features could already be identified to further increase the understanding of the model. The most significant extra features are listed below:

- Links between conditions and diagnoses must also be implemented similarly to the current findings and diagnoses. Note that this must be the Positive and Negative Predictive Values (PPV, NPV), but the rest of the logic works out the same. Similarly, known diagnoses and new diagnoses should also be displayed in this form. Promedas does not take this into consideration yet, but some diseases can also increase the risk of other diseases.

- Co-occurence of diagnoses within the population should be displayed as well, similarly as the aforementioned confusion matrix. These co-occurences can point to intermediate diseases. These may become relevant in the future, if the model is decided to incorporate these intermediate steps. Additionally, such a matrix could identify systematic recurrences in the dataset.

- Co-occurence of findings and conditions would be informative for a similar reason. But the additional benefit of these is that they may provide insight into issues in the underlying dataset. The templates from the very beginnings may for instance have biased Copilot to generate cases with similar findings to the template, whereas their actual diagnosis would require a different set of findings to discover.

- Additionally, if needed, with a sufficiently large population, one may be able to observe interactions between multiple diagnoses and a finding, i.e. in the case of multiple parents. This might help to validate the appropriateness of the noisy-OR assumption in the model.

- For the program at large, a history function should be considered that automatically tracks the improvements throughout multiple iterations through the optimisation cycle.
- Extra filters could be introduced on the population, like sex and age range.

The major challenge of this program, it seems, exist in the remaining lack of insight into the underlying population, rather than Promedas. These additional features above may help to further increase the understanding of the patient set.

Interestingly, this method of display might be a candidate for mapping out health data at large, outside the scope of Promedas: it carries a lot of information about a patient population and allows the administrator, in this case, to detect unexpected associations within this population. This could be used both to increase understanding of diseases at large, but also to track systematic errors in medical practice. And after all, each medical centre is expected to see different relations, depending on their specialisations, environment and overall complexity of care that they offer.

One large point of caution here is that the data, as displayed, only displays correlations. For one, correlation does not imply causation. Any viewer must remain critical of why a correlation shows up. And at that point, one would be hypothesising after the results are known (HARKing) [33]. The displayed results may be influenced by out-of-scope confounders, that could be wrongly identified or completely omitted.

This same risk of HARKing exists within the scope of Promedas, but has been made measurable by splitting the dataset into a training and a test set. Still another risk exists of correlation being interpreted as causation, because Promedas assumes causal probabilistic relations between findings and diagnoses either way.

For example, this program found that nearly all patients with the diagnosis of sinus tachycardia, when tested, had a positive blood culture. Knowing this alone, one could for instance hypothesise that bacteria increase the heart rate. Of course this negates two important facts: for one, this measurement was only made for about 7% of patients with sinus tachycardia. Thus, it was not expected to be meaningful in the vast majority of patient cases. Secondly, it is a long-established fact that sinus tachycardia exists in cases of severe inflammation, including severe infections [34]. The (fictive) clinicians must have ordered the test because they reasonably suspected an underlying infectious disease.

This highlights an interesting point of conflict between pragmatic medical organisation and scientific soundness. This being that any finding with a high

PPV would still be a helpful clinical question, albeit a creative one. Let's assume that the PPV of blood culture for sinus tachycardia would also be close to 1. Then, if a positive blood culture was found in a patient without a heart monitor, the clinician can be quite confident that their pulse rate is elevated as well[3].

But finding these relations does not necessarily exemplify a good understanding of the underlying physiology. This can only be achieved through reasoning, which is an essential element of the scientific method. The downside of this, compared to this more pragmatic approach, is that these iterations of reasoning, experimenting and evaluating are incredibly slow.

The same risk is present in Promedas, so long as this method is applied. The only reliable, scientifically grounded way to develop Promedas would be to actually reason about potential associations within the network, discover the work of colleagues in this respect, and to experiment for each individual relation inside the network.

The codebase for the analysis programs with dummy data were uploaded to a public Github repository[4].

5.3 Closing the loop

The final step to complete the optimisation cycle is to use discrepancies from the analysis program in a way to improve the accuracy of Promedas. Technical Medicine interns have joined forces during the last four months to tackle this issue. This turned out to be challenging to the point where no consistent, highly accurate strategy was found within the term limit. This section describes these strategies and evaluates the results of each step, as far as they were known.

5.3.1 Problem definition

As explained in Section 5.2.2, the analysis program compares conditional probabilities between Promedas and the population. The values in Promedas should align as much as possible with the values from the population in

[3]This may not be a very good example to illustrate this point. For one because, when you find a positive blood culture, you have a larger problem on your hands than a high pulse rate, namely sepsis [35]. But this was one of the simplest and clearest examples I could find for this purpose.

[4]Which can be found here: https://github.com/Semmenekem13/TM3-Bayesian-Network-Analysis

order to be an accurate model. The mismatches can therefore inspire an administrator to alter links in the model.

The challenge arises from the fact that these links must be supported by literature. Thus, administrators are required to supply a source for each link. This is fairly straight-forward as it is, but the work adds up when hundreds of links are considered.

The goal therefore is to find a method that reads these mismatches after identification and reliably finds an appropriate source that either supports or rejects this link, with additional sensitivity data.

5.3.2 Strategies

Multiple strategies were tried throughout these months. The most straightforward would be to search for one link in PubMed in the format of (”¡finding¿”[MESH] OR ”¡finding¿”[tiab]) AND (”¡diagnosis¿”[MESH] OR ”¡diagnosis¿”[tiab]). But the results for this search query turned out not to be highly reliable per se: the query would include many irrelevant studies and was not guaranteed to include studies that would describe these links as secondary findings. For this reason as well as its labour-intensive nature, this attempt was abandoned.

The most prominent strategy would be to adapt an LLM again. Its success during the translation step was reason for optimism. The first attempt was to natively prompt an LLM to find a source for a list of findings and diagnoses. The prompt would not receive a hint of the presence or absence of this link in the population. This turned out to perform poorly [36]. In a sample of 50 cases, only two explicit probabilities were found in a credible source. Most of the other sources would not exist at all, going by title and author. This behaviour was consistent across Grok xAI, ChatGPT-4o and EvidenceHunt. This attempt was therefore abandoned.

An alternative was to use an LLM to extract links from a medical text book instead. This was performed using Microsoft Copilot on 1118 links that existed in Promedas at the time, with the intention to remove links that were not supported yet by a source [36]. The LLM gave up 191 out of these links for deletion. A rerun of Promedas without these 191 links lead to a minor overall improvement in performance: an additional 7 out of the total 808 diagnoses (0.87%) appeared at the top of their differential diagnosis. The number of correct diagnoses in the top 10 of their differential diagnoses did not change. The accuracy of the ‘keep’ group was later analysed as well. In a sample of 272 links, only 60.3% of links turned out to be supported by the book at all. Most of these links would occur in other words and on a different page than was returned by the LLM.

In the next stage, the book query would be simplified by breaking up the book into chapters, as smaller containers of information. An additional attempt was made to further improve on direct prompting in OpenEvidence. The outcomes of these two strategies have not been fully disclosed yet. The intermediate results have not turned out to have exceptionally high accuracy, but were sufficient to identify future steps.

5.3.3 Discussion

None of the strategies turned out to find links in the literature with a consistent high accuracy. For this reason, manual literature search would remain the method of preference. The research trajectory of Promedas would benefit to retry this occasionally. One can reasonably expect that LLMs soon acquire the ability to interpret scientific literature, especially those that are explicitly trained on scientific literature.

Another important observation is that sensitivity and specificity values are underreported in the literature, even when a link between a diagnosis and some finding is known to exist. For instance, chest pain as a symptom of myocardial infarction is an established fact, but it is unknown what portion of the population of myocardial infarction patients actually experiences chest pain [37].

This underreporting is not alarming in clinical practice. For one because these actual values may differ for each medical centre. But also because positive predictive value is more interesting for diagnostics, in general: given that a patient presents with symptom Y, what are the odds that they have diagnosis X? Going by the example of chest pain: there are many alternative diagnoses that could cause it, but the symptom of chest pain prompts clinicians to apply ECG testing and blood troponin measurement, which do have higher sensitivities and specificities [37].

But this is a serious issue for Promedas, which relies explicitly on reported sensitivity values. A good compromise for Promedas would be to only use literature to indicate simply whether a correlation exists within the literature. The exact probabilistic values should then be reconstructed in a way.

6. Thought experiments

6.1 Best guess strategy

As is briefly mentioned in section 5.3.3, sensitivity (and specificity) values tend to be underreported in medical literature, especially with respect to symptoms and physical examinations. This is detrimental for Promedas, which relies on such sensitivity values. Hence, Promedas needs to make assumptions for unknown values in the literature.

As described in section 2.3.2.2, Promedas by default assumed sensitivities to be equal to 0.5, according to the zero-leg principle. This was an alternative to the one-leg model, which assumed a sensitivity of 0.99 instead. The reason for this was the observation that link values were not found to be of much relevance for the accuracy of Promedas. And since the zero-leg model outperformed the one-leg model, the zero-leg model was chosen by default instead.

6.1.1 Proposed values

The aforementioned assumption is quite blunt: one cannot be sure that the occurrence of some finding exists or doesn't exist with exactly equal chance for a diagnosis. Therefore, a new '0.1-leg model' was proposed, which would alter the assumed strength for this link by the type of finding, i.e. whether it is a symptom, finding from physical examination, lab test or different. Central to this model are the sensitivity values; specificity values were added for completeness but are not yet supported by Promedas.

Unfortunately there is no literature in existence that describes the sensitivity and specificity of diagnostic tests at large. An educated guess was made instead (Table 6.1). These numbers did not rely on some weighted average of the literature that does exist, but only from reasoning. A trial run was done on the old dataset of 37 patients.

6.1.2 Results and evaluation

The trial run only lead to a minor improvement. Out of 48 correct diagnoses within the dataset, 23 diagnoses received a probability over 0.5, as opposed to 22 in the default zero-leg configuration. This strategy would be adapted nevertheless in the future, because its performance was non-inferior to the

Type	Sens*	Spec**	Explanation
Symptom	0.5	0.5	Most results occur with multiple diagnoses; depends in a large part on what the patient experiences as troubling.
Physical exam	0.5	0.8	Findings from physical examination are less likely to be found in people with an unrelated diagnosis; findings may be overheard or overlooked.
Lab test	0.8	0.8	High certainty that abnormalities in blood relate to specific disrupted physiology, but factors exist that may skew these measurements.
Radiological finding	0.95	0.95	Abnormalities in radiology rarely ever occur without an underlying cause and diagnoses that can be detected through radiological findings rarely every don't present without these findings; small chance of being overlooked.
ECG	0.95	0.95	ECG at large is the golden standard for cardiac arrhythmias and suffers little from misinterpretations or distortions; small chance of interpretation error.
Vital parameters	0.5	0.5	May be abnormal with multiple diagnoses; might be within normal range in case the disease is not severe.
Targeted additional exam	0.8	0.8	These are tests that are employed in general for their high diagnostic value. The correction down to 0.8 comes from this group's definition as a 'remainder' of test types that do not yet have a separate group.

Table 6.1: A table of proposed best-guess link placeholders.
**=Sensitivity. **=Specificity.*

zero-leg strategy and this assumption is better-grounded than the zero-leg model assumption.

This 0.1-leg model may still be improved upon. One proposal was to further divide symptoms into hallmark symptoms and secondary symptoms, as was proposed by Herweijer, L. [38]. But additionally, it may be a treasure to perform a systematic review of the sensitivities and specificities of different diagnostic tests according to this list, both for Promedas and the medical field at large. Throughout the course of this thesis, no other modifications were made to improve the accuracy of this assumption.

6.2 Best case scenario

This thought experiment came up at the end of this thesis, as work was still on its way to close the optimisation cycle. The only step left after the analysis program was to verify whether the insights from it would improve the accuracy of Promedas in the long run. A clear-cut naive method was to determine a reliable strategy for this issue, before moving on to walk through this cycle until a stable point is reached. This thought experiment came up as an alternative.

6.2.1 Problem definition

Let's assume that Promedas was to be implemented instead in the world of Microsoft Copilot. Microsoft Copilot has a completely transparent emergency department, called the Emergency Response and Recovery Organisation Room (ERROR). The ERROR is limited only to cases with any of 58 predefined cardiovascular diagnoses. Moreover, the ERROR really likes to use forms instead of free-text notes for clinical registrations, but may make some mistakes in the stored data. Inside the ERROR, all sensitivity values have already been studied between any finding and any diagnosis, regardless of the reasoning leading up to these hypotheses. The ERROR wants to use these values to give its clinicians a second opinion and finds Promedas as a hypothetical model.

The ERROR staff decide to set up a study to test Promedas. They use their vast collection of sensitivity values to configure Promedas. The staff collects the data of 108 visitors, with at least 1 case for each diagnosis in their system. They will infer Promedas on each of these visitors at the end and analyse how well Promedas may assist them.

Through reasoning, one can identify two major conditions to achieve a high accuracy:

- All patients with the same disease must have fairly consistent test results. I.e. the probability values within the population remain relatively unchanged. If some finding F has a sensitivity of 0.6 for some disease D, then 60% of patients with disease D will now and in the future present with F. This also means that the base of literature in the ERROR must be representative of the patient population at all times.
- Promedas with its assumptions, architecture and link values, must adequately create a differential diagnosis given a set of findings.

6.2.2 Methods

This thought experiment was executed by calculating prior probabilities, sensitivity and risk ratio values based on the dataset of the optimisation cycle, i.e. the training set. These conditional values have been calculated naively, without considering the leak node in the diagnosis layer or maximum likelihood estimation.

These probabilistic values were then inserted in a new database representation of Promedas. This version could be inferred only on a subset of the training set and needed further debugging before it could be applied on the test set. The results were summarised by the nontransparent analysis code (Section 5.2.4).

6.2.3 Results

149 patients from the same training set could be used for analysis. The diagonal of the confusion matrix had become visibly more yellow, but still suffered from high accuracies outside the diagonal and biases in favour of some diagnoses. 6 out of 58 diagnoses were not included in this patient set and could therefore not be judged. Out of the remaining 52 diagnoses, only 6 had an average probability over 0.5. 38 diagnoses had an average probability over 0.1. The lowest average probability was 0.02.

The same occurred with the position in the list, with only 38 out of 52 diagnoses being on average in the top 10. 26 ended up in the top 5 on average. No diagnosis was in the first position on average.

When adjusted to filter out other correct diagnoses, these positions improved. 41 out of 52 reported diagnoses ended up at least in the top 10 on average, 30 ended up in the top 5. The worst average adjusted position overall was position 25.

This is compared to the results of the latest run before the experiment. Back then, 17 out of 58 diagnoses had an average probability of 0.5 or higher.

On the other hand, the worst-case probability was 0.000054. 10 diagnoses had a lower average probability than the lowest score in the experiment, i.e. 0.02.

The same diverse results exist for the average adjusted positions. In the latest run, 14 diagnoses ended up in the top 5 on average and 25 ended up in the top 10. The worst average adjusted position was position 51.

6.2.4 Discussion

Interpreting these results has lead to some difficulty. Firstly because it hasn't been run yet on the testing set. But it stands to reason that the performance of this network within its own dataset must be non-inferior to its performance in the testing set.

Considering that, the overall adequacy of the differential diagnosis positions have improved through this thought experiment, while the average probability of high-ranking diagnoses is lower overall than in the last run of the 'real' Promedas. On the other hand, this performance still isn't very good, given its ideal circumstances.

To reiterate, the experiment would only succeed on the conditions that the population would be consistent enough and that Promedas would be an adequate model. The first condition is difficult to judge with the results at hand. With respect to the latter condition, it is possible to speculate on a few potential shortcomings in Promedas. These are:

- Some diagnoses may be associated with findings that do not exist yet in Promedas. It should be noted from the translation step (Section 4.2) that the patient files are already covered in the most part by entities in Promedas. But some important examinations like ultrasound imaging still do not exist in the network.

- Cardiac arrhythmias still underperform at large. The current way in which ECG findings are represented through categories causes room for errors. The ECG findings still rely on the raw text alone and not on the actual ECG data. ECG descriptions may therefore be under-complete within the raw text files. The translation step also appears to have difficulty translating raw text into categories.

- Findings in Promedas may still be oversimplified in other ways. Chest pain, for example, is only represented in three different findings for under any circumstances, radiating and on exertion. But the patient's experience of chest pain could provide leads on its underlying diagnosis [39].

On top of these model shortcomings, the experimental setup itself also still leaves some room for additional experimentation. These could no longer been done within the scope of this thesis, but may provide more insight in the future. Two such suggestions are:

- The current dataset mostly contains cases with multiple diagnoses at play. Since links were created per diagnosis and finding per patient, some links were likely attributed to one diagnosis while only really existing in the other. Essential hypertension is a good example of this, which has consistently been assigned a high probability in the results set. Many patients in the training set had a prior history of hypertension, and essential hypertension now has over 200 findings attached from all these patients with different diagnoses. Two mitigations against this are to only use links that have been reported at least some x amount of times (relative to the number of patients with this diagnosis overall). Another may be to simply rerun this experiment with a patient set with fewer diagnoses per patient.

- To only set probabilistic values based on the population, but to use scientific literature to back up the existence of a link at all. In other words, to compromise between literature as an indication of the existence of a link and to using the population to guide the value on that link.

These additional experiments are obvious leads for future research. If through these experiments it is possible to find a highly accurate performance, then Promedas would reasonably be a good CDSS for the ERROR. The suitability of Promedas for the task of CDSS should not be dismissed before these experiments and any other leads have been conducted.

In case these additional experiments are successful, the focus should shift towards the optimisation cycle to discover to what extent relations can automatically be extracted from the literature.

This success would additionally be a good argument to repeat this experiment with real patient cases, in order to observe the ideal accuracy that can be expected from Promedas in the clinical setting.

Either way, finding appropriate probabilistic values for the network's configuration is bound to become a challenge, considering the underreporting of these probabilistic values in literature. With respect to the suggestion of Section 5.3.3, Promedas could use data from a population to approximate probabilistic values after a link's existence is supported by scientific literature.

7. General Discussion

7.1 Summary of results

Before moving onto the general discussion, let's summarize the stages that were walked through. Refer to Figure 3.1 for the overview in one figure.

- First, the dataset was expanded with 512 fictive cases in free text, imitating a letter to the patient's GP.
- The free text files were then translated by Microsoft Copilot into the same survey-like format that Promedas uses. The translation accuracy through this prompt was found to be reproducibly high, above 90%. The output from Copilot could be post-processed to match the input format of Promedas with <1% loss.
- The dataset was then split into a training set and a test set, at an 80-20 ratio. The test set would make confirmation bias measurable. The realm of the training set would provide ample data for future administrators (technical medicine interns) to continue improving the model.
- A program was written in Matlab to compare relations in the population to links within Promedas. Detected mismatches in this program would serve to inspire the administrators to propose changes.
- At the end of this term, there is still no highly reliable way to find potential links in the literature when prompted. Only on this condition can the group move on to optimise Promedas with acceptable manual labour.
- A thought experiment was conducted to validate the theoretical validity of this setup. Its results have not yet been compelling, but there are multiple routes to continue experimentation.

7.2 Interpretation of the results

To recapitulate, the main goal of this thesis could be summarized as to determine whether Promedas can exceed the diagnostic accuracy of clinicians.

As expected, it is premature to argue whether Promedas would be valid in clinical practice. Not only because of the current stage of development, but also because it has not yet been tested on real data. The findings in the fictive dataset are not without value, though.

First, large language models have turned out to be very reliable for the task of translating free text into discretised form. As discussed before, this strategy is not preferred, but it has turned out that no analytical alternatives might exceed this accuracy currently.

The proposed optimisation cycle as well as the analysis programs unfortunately have not been tested yet, but could reliably display discrepancies and forward these to an output file. The nontransparent analysis code provides good metrics of the model performance overall and could be utilised on the testing and validation datasets. Future administrators could easily reuse this script to analyse the impact of their work.

The final thought experiment has not provided compelling evidence in favour of or against this optimisation cycle. Section 6.2.4 lists multiple alterations to continue experimentation. In case the thought experiment turns out to be successful in the future, efforts should shift to close the loop and then to run through the optimisation cycle. It might in that case soon become relevant to investigate how Promedas would perform on real cases.

In case these experiments fail, the focus should shift to investigate the level of heterogeneity within the dataset, to discover to what extent each condition contributes to the model's failure. At this point, further paths to optimise Promedas structurally should already have been covered. If heterogeneity is not the issue, the suitability for Promedas for this population should be seriously reconsidered in its current form, which might be representative for the real population.

7.3 Limitations

The most dominant limitation within this thesis is the fact that the results have been acquired on the basis of a fictive, AI-generated patient set. Thus, any results within this report should not be considered to be representative of any expected results in the real world. The extent to which these steps can be expected to be representative for the real world has been explored throughout the intermediate discussions. That being said, it has proven to be an effective backbone for the rest of the development. And moreover, the entire pipeline has been set up to work from files that are similar to real doctor's notes. For that reason it should be straight-forward to repeat this experiment with real data. Importantly, the version of the database that

is going to be built using the fictive cases should not be used as a baseline model for the real dataset. A backup from January 2025 can be used as a baseline instead.

That limitation exists within a broader issue that the work of this thesis in general has relied a lot on AI. Especially so when Promedas was intended to undermine the reliance on AI, by explicitly setting scientific substantiation as a goal. Aside from generating patient cases, AI was used to translate cases and find scientific sources. For all these, it is important to capture and analyse their accuracies as much as possible, so long as they are in use. But even then, one must be wary that small error margins add up over multiple stages. Any analytical solution to these substeps should be preferred, so long as they prove to have non-inferior accuracy marks to LLMs. This shortcoming should be interpreted in light of the fact that a lot of transparency has been created by disassembling this process into its substeps, contrary to the LLMs that work like a black box.

Additionally, the implemented code files have been reviewed through ChatGPT and supervision and the results are consistent with manual calculations. Nevertheless, there is ever a chance of unforeseen bugs or mathematical errors that have not been resolved. In fact, none of the code has not been complemented by testing modules due to their currently still small scopes. These shortcomings exist among more issues related to technical debt, which are further elaborated in Section 7.4.3.

Lastly, analytics are still missing from many important stages in the code. The translation step would benefit from stronger statistical power and more expert opinions. The latter could both serve to find a more reliable 'average' expert opinion and to compare the LLM to inter-observer agreement in this step. The optimisation cycle would benefit at least from a metric for the rate of change over multiple iterations, but these iterations haven't been performed yet. The analysis code would benefit from metrics for its quality attributes and a survey on usability by other admins. It would have been preferred overall if the initial main hypothesis received a proper verdict based on statistical tests, but that stage has not yet been reached.

7.4 Future challenges and recommendations

This section describes future challenges for the Promedas research track and proposes recommendations for its further development.

7.4.1 Model structure and configuration

The development of Promedas is challenged to strike a balance between realism and computability. Through the results of this report and general experience with this network structure, it is possible to suggest a few changes to the data structure that might benefit its realism. These changes are not binding, as they may have implications that are computationally hardly or impossibly solvable. The proposals are as follows:

- Some concepts serve as a diagnosis and as a finding. Sinus tachycardia is an example of this: formally it is known as a diagnosis, but may be associated with any other diagnosis that increases someone's metabolism. The main objection against this change is that the BN structure must be composed of more than two layers. This means that the Quickscore algorithm is no longer possible and must be replaced by a slower alternative. A compromise might be to program relations in a layered approach and to develop an algorithm that 'flattens' the network back into two layers.

- The model currently does not employ specificity between any diagnosis and link. Specificity instead is estimated indirectly. But some problems arise from this estimation method: findings that are only linked to one diagnosis, whenever they are true, will always set the associated diagnosis' probability to 1. Additionally, the simplification does not account for the fact that some test may be more specific for one disease than another. Troponin measurement, for example, is highly specific for myocardial infarction, but may arise with lower specificity in diseases like myocarditis, heart failure and renal failure [40, 41].

- The model may benefit from additional features to approximate clinical realism. One suggestion is to distinguish symptoms by how telling they are for some diagnosis [38]. Hallmark symptoms of a diagnosis would deserve different treatment by the system than secondary symptoms, by assigning higher sensitivity values overall.

- Additionally with respect to clinical realism, symptoms in reality tend to be multidimensional. That is, symptoms have occurred with some time duration, have become better or worse and are located somewhere in the body. The ALECOBO framework in Dutch practice would be perfect for this task [42]. There are multiple ways to imagine an implementation of this, but this is generally not practical.

7.4.2 Reporting in medical literature

Another important challenge for the development of Promedas is the fact that conditional probabilities are widely unknown in medical literature, as was mentioned in Section 5.3.3. Additionally, where these probabilities are known, these tend to be measured in cohort studies, which do not reflect the dimensionality of the patient population that Promedas is tested on. To reiterate, these values add little value in clinical practice, because they may be very different across different environments. Sensitivity and specificity metrics are mostly important for tests that are explicitly intended to diagnose patients.

This also means that another important assumption is implicitly made in Promedas: every probability stands on one additional conditional variable. Thus, it should not be programmed with a $P(A|B)$, but with a $P(A|B \cap C)$. A link between a finding and a diagnosis must be the probability that a clinician can find a finding F given diagnosis D, given the fact that the patient is seen *here*.

In the face of the lack of sensitivities in the literature, a good compromise for Promedas would be to use literature only to substantiate the presence of a link. Promedas could use data from global or local populations to further fill in the probabilities on these links, to meet the needs of the end user with an appropriate configuration. An important future research point would be to investigate whether this strategy would provide Promedas with reasonable accuracy, by a similar approach as described in this report.

For clinical practice, it may be valuable to investigate these probabilistic values too for two reasons. The first being to educate clinicians in symptoms and tests that may have an insidiously high or low positive or negative predictive value for some diagnosis. The other being in an attempt to explore inter-centre differences and to identify causes for these differences.

7.4.3 Addressing technical debt

It must further be noted that some shortcuts had to be made throughout this methodology in order to achieve results within a reasonable time window. Let these shortcuts loosely be referred to as technical debt that must be corrected down the road. These have already been mentioned throughout the intermediate discussion sections, so let this section serve to reiterate and summarize the most important subjects.

The dataset would stand to benefit from more fictive patient cases. The current 512 cases have thus far been sufficient, but will not provide much statistical power nor high variety over 58 different diagnoses. The prompts

should also receive minor adjustments to encourage Copilot to be more verbose in its anamnesis and to include tract anamnesis. Copilot currently generates these in one or two sentences.

Within the translation step, the use of an LLM is currently sufficient. The prompt should receive minor adjustments to distinguish diagnoses from prior history from new diagnoses and the categorical findings should be better covered. It also, generally, calls for an analytic text processing approach down the line. But any engines for this purpose have not turned out to be feasible in their current states. Post-processing the data from the LLM also introduces some minor losses.

The analysis program within the optimisation cycle would benefit from some additional cleaning-up and should be expanded at large. Additional features should be to map out comorbidity and to include diagnoses from prior history. In face of the fact that future Technical Medicine interns will no longer be educated in Matlab, the best solution might be to rewrite the program from scratch in Python. Hopefully Python proves to be able to deal with the highly array-like logic as Matlab does.

All of these exist on top of the fact that Promedas still is limited to cardiological diseases, conditions and findings. After Promedas is found to be a viable approach in this field, it still must be expanded to other disciplines to prove its overall worth.

Lastly, the current setup still relies on fictive patient cases. Promedas should only be tested on real patient data when there is sufficient evidence of its efficacy. One proposal is to do this each time Promedas is expanded into a new discipline.

7.4.4 Place in clinical practice

Finally, Promedas requires a vision of its final implementation. The current digital infrastructure of Dutch medical practice fortunately has plenty of entry points for this application. ChipSoft's HiX system already manages unstructured and structured clinical data as it is [43]. SNOMED CT additionally aims to arrange interoperability between medical systems through a terminology database [44]. Converting structured data between HiX and Promedas should be fairly straight-forward and, regarding text processing, an LLM has proven to be a good stand-in for a translation step between free text and Promedas language.

Promedas would be a suitable plug-in for HiX in terms of data exchange. But it would be premature to speculate on the exact route of implementation. Promedas must be further investigated for its suitability before this point.

8. Conclusion

The goal of this thesis was to examine to what extent Promedas would be a suitable model for clinical decision support in the emergency department. Many steps were made over the course of this thesis to create an environment in which Promedas can be further improved upon.

The results of the experimental setup have been promising. A large fictive environment was made for experimentation. These free-text files could be translated into Promedas language with surprisingly high accuracy.

A train-test-validate strategy was proposed to speed up the network's development. This strategy still faces challenges in two forms: LLMs still fall short in automatically discovering links between findings and diagnoses; and its hypothetical best case scenario still requires further experimentation to confirm or rule out the cycle's overall effectiveness.

Promedas additionally suffers from underreporting of probabilistic values in medical literature. Promedas also still is labour-intensive to optimise, so long as no AI-driven search engines prove to reliably deliver appropriate sources for potential links.

Concludingly, Promedas might best be understood as a program that is ahead of its time. It seems like a promising technology for the future, but its development relies on search engines that still fall short in finding sources. As development of digital means continues, Promedas has an interesting future still ahead.

9. References

1. Newman-Toker DE, Peterson SM, Badihian S, Hassoon A, Nassery N, Parizadeh D, Wilson LM, Jia Y, Omron R, Tharmarajah S, Guerin L, Bastani PB, Fracica EA, Kotwal S, and Robinson KA. Diagnostic Errors in the Emergency Department: A Systematic Review [Internet]. Agency for Healthcare Research and Quality (US) 2022 Dec. Report No.: 22(23)-EHC043
2. Kong A, Barnett OG, Mosteller F, and Youtz C. How medical professionals evaluate expressions of probability. N Engl J Med 1986 Sep; 315:740–4. DOI: 10.1056/NEJM198609183151206
3. Lyman GH and Balducci L. Overestimation of test effects in clinical judgment. J Cancer Educ 1993; 8:297–307. DOI: 10.1080/08858199309528246
4. Heuer RJJ. Psychology of Intelligence Analysis. Langley, VA, USA: Center for the Study of Intelligence, Central Intelligence Agency, 1999 :51–63
5. Kappen B, Wiegerinck W, Akay E, Neijt J, and Beek A. Promedas: A clinical diagnostic decision support system. *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence*. 2003
6. Ben-Gal I. Bayesian Networks. *Encyclopedia of Statistics in Quality and Reliability*. Ed. by Ruggeri F, Kenett RS, and Faltin FW. Online: John Wiley & Sons, 2008
7. Diagnostic Error in Health Care C on, Health Care Services B on, Medicine I of, The National Academies of Sciences E, and Medicine. Improving Diagnosis in Health Care. Ed. by Balogh EP, Miller BT, and Ball JR. Washington (DC): National Academies Press (US), 2015 Dec
8. Hautz WE, Kämmer JE, Hautz SC, Sauter TC, Zwaan L, Exadaktylos AK, Birrenbach T, Maier V, Müller M, and Schauber SK. Diagnostic error increases mortality and length of hospital stay in patients presenting through the emergency room. Scand J Trauma Resusc Emerg Med 2019 May; 27:54. DOI: 10.1186/s13049-019-0629-z
9. Nuñez S, Hexdall A, and Aguirre-Jaime A. Unscheduled returns to the emergency department: an outcome of medical errors? Qual Saf Health Care 2006 Apr; 15:102–8. DOI: 10.1136/qshc.2005.016618

10. Maslow AH. The Psychology of Science: A Reconnaissance. Harper & Row, 1966 :15–6

11. Center BSL. Introduction to Bayesian Networks. Accessed: 12 Dec 2024. 2024. Available from: https://www.bayesserver.com/docs/introduction/bayesian-networks/

12. Goodrich MT, Tamassia R, and Goldwasser MH. Data Structures and Algorithms in Java. 6th. Wiley, 2014

13. Schilling RL. Measure, Integral, Probability & Processes - Probab(ilistical)ly the Theoretical Minimum. 1st. Technische Universität Dresden, Germany, 2021

14. MIT Department of Electrical Engineering and Computer Science. Lecture 16: Inference in Bayesian Networks (Final Part 1). Accessed: 2025-10-17; Course 6.825 Techniques in Artificial Intelligence, Fall 2002. 2002

15. Cooper GF. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. Artif Intell 1990; 42. Available as PDF:393–405. DOI: 10.1016/0004-3702(90)90060-D

16. Pearl J. Fusion, propagation, and structuring in belief networks. Artif Intell 1986; 29:241–88. DOI: https://doi.org/10.1016/0004-3702(86)90072-X. Available from: https://www.sciencedirect.com/science/article/pii/000437028690072X

17. Heckerman D. A Tractable Inference Algorithm for Diagnosing Multiple Diseases. *Uncertainty in Artificial Intelligence*. Ed. by Henrion M, Shachter RD, Kanal LN, and Lemmer JF. Vol. 10. Machine Intelligence and Pattern Recognition. North-Holland, 1990 :163–71. DOI: https://doi.org/10.1016/B978-0-444-88738-2.50020-8. Available from: https://www.sciencedirect.com/science/article/pii/B9780444887382500208

18. Gaebel J, Cypko MA, and Oeltze-Jafra S. Considering Information Up-to-Dateness to Increase the Accuracy of Therapy Decision Support Systems. Stud Health Technol Inform 2017; 243:217–21

19. Hao SR, Geng SC, Fan LX, Chen JJ, Zhang Q, and Li LJ. Intelligent diagnosis of jaundice with dynamic uncertain causality graph model. J Zhejiang Univ Sci B 2017 May; 18:393–401. DOI: 10.1631/jzus.B1600273

20. Sistrom CL and Garvan CW. Proportions, Odds, and Risk. Radiology 2004; 230:12–9. DOI: 10.1148/radiol.2301031028

21. Bakker G and Meijer J. Promedas: A Clinical Decision Support System in the Emergency Department. Unpublished internship report, accessible within the research group. 2024

22. Python Software Foundation. Python: A Programming Language. Version 3.x. 2024. Available from: https://www.python.org/

23. Sweigart A. PyAutoGUI: A cross-platform GUI automation module for Python. Accessed: 2024-03-05. 2024. Available from: https://github.com/asweigart/pyautogui

24. Sweigart A. Pyperclip: A cross-platform clipboard module for Python. Accessed: 2024-03-05. 2024. Available from: https://github.com/asweigart/pyperclip

25. Langens Q. Automatic Data Extraction To Create Patient Files. Unpublished internship report, accessible within the research group. 2025

26. Kuiper S. Evaluating Prompt Engineering Strategies for Clinical Information Extraction Using Microsoft Copilot. Unpublished internship report, accessible within the research group. 2025

27. Cypko MA and Stoehr M. Digital patient models based on Bayesian networks for clinical treatment decision support. Minim Invasive Ther Allied Technol 2019 Feb 27; 28:105–19. DOI: 10.1080/13645706.2019.1584572

28. Gaebel J, Wu HG, Oeser A, Cypko MA, Stoehr M, Dietz A, Neumuth T, Franke S, and Oeltze-Jafra S. Modeling and processing up-to-dateness of patient information in probabilistic therapy decision support. Artif Intell Med 2020 Mar 9; 104:101842. DOI: 10.1016/j.artmed.2020.101842

29. Cypko MA, Stoehr M, Kozniewski M, et al. Validation workflow for a clinical Bayesian network model in multidisciplinary decision making in head and neck oncology treatment. Int J CARS 2017; 12:1959–70. DOI: 10.1007/s11548-017-1531-7

30. Scikit-learn. Cross-validation: evaluating estimator performance. Accessed: 2025-09-08. 2025. Available from: https://scikit-learn.org/stable/modules/cross_validation.html

31. Inc. TM. MATLAB version: 9.11.0 (R2021b). Natick, Massachusetts, United States, 2021. Available from: https://www.mathworks.com

32. The MathWorks Inc. App Designer. Natick, Massachusetts, United States, 2021. Available from: https://www.mathworks.com

33. Kerr NL. HARKing: Hypothesizing After the Results are Known. Pers Soc Psychol Rev 1998; 2:196–217. DOI: 10.1207/s15327957pspr0203_4

34. Henning A and Krawiec C. Sinus Tachycardia. *StatPearls [Internet]*. [Updated 2023 Mar 5]. Treasure Island (FL): StatPearls Publishing, 2025 Jan. Available from: https://www.ncbi.nlm.nih.gov/books/NBK553128/

35. Marik PE. Don't miss the diagnosis of sepsis! Critical Care 2014 Sep; 18:529. DOI: 10.1186/s13054-014-0529-6

36. Hammink SFE. Improving Promedas: Clinical Decision Support System for the Emergency Department. Unpublished internship report, accessible within the research group. 2025

37. Mechanic OJ, Gavin M, and Grossman SA. Acute Myocardial Infarction. *StatPearls [Internet]*. [Updated 2023 Sep 3]. Treasure Island (FL): StatPearls Publishing, 2025 Jan. Available from: https://www.ncbi.nlm.nih.gov/books/NBK459269/

38. Herweijer L. Reliability Analysis of AI-Identified Links between Clinical Diagnoses and Findings in the Promedas Diagnostic Model. Unpublished internship report, accessible within the research group. 2025

39. Hollander JE and Chase M. Approach to the adult with nontraumatic chest pain in the emergency department. Accessed: 2025-10-17. Available from: https://www.uptodate.com/contents/approach-to-the-adult-with-nontraumatic-chest-pain-in-the-emergency-department

40. Ross G, Bever FN, Uddin Z, and Hockman EM. Troponin I sensitivity and specificity for the diagnosis of acute myocardial infarction. Journal of the American Osteopathic Association 2000 Jan; 100:29–32

41. Stark M and Sharma S. Troponin. *StatPearls [Internet]*. [Updated 2023 Apr 23]. Treasure Island (FL): StatPearls Publishing, 2025 Jan. Available from: https://www.ncbi.nlm.nih.gov/books/NBK507805/

42. Veening EP, Gans ROB, and Kuks JBM. Medische consultvoering. 1st ed. Dutch language eBook collection; Paragraph 2, Chapter 3. Houten: Bohn Stafleu van Loghum, 2009. Chap. 3:31. DOI: 10.1007/978-90-313-7995-8

43. ChipSoft. HiX. Accessed: 2025-10-17; EHR solution by ChipSoft (Amsterdam). Available from: https://www.chipsoft.com/nl-nl/oplossingen/hix/

44. Nictiz. SNOMED. © SNOMED International. All rights reserved. Accessed: 2025-10-16. Available from: https://nictiz.nl/wat-we-doen/activiteiten/terminologie/snomed/

45. Jacobson I, Booch G, and Rumbaugh J. The Unified Software Development Process. Addison Wesley Longman, 1998

46. Gleek Blog. Mastering inheritance in class diagrams. Accessed: 2025-09-10. 2024 Aug. Available from: https://www.gleek.io/blog/inheritance-class-diagrams

10. Appendices

A Case generation prompts

The MS Copilot prompts for data generation are supplied below. All prompts are in Dutch, since they were used to generate patient records in the Dutch language; to use English would have been likely to affect Copilot's performance.

A.1 Prompt 1

<template bestand>

Zie voorgaande template. Ik wil dat je een ontslagbrief genereert volgens dezelfde structuur. Ik wil dat je nu fictieve casussen genereert op basis van een diagnose die ik aan je doorgeef. Ik verwacht dat je het geslacht en leeftijd, de reden van verwijzing, hoofdklacht, klinische presentatie, symptomen, alle uitslagen en een voorgeschiedenis zelf verzint. Rangschik het dossier in dezelfde kopjes: samenvatting, relevante voorgeschiedenis, anamnese, actuele medicatie, lichamelijk onderzoek, aanvullend onderzoek, conclusie en beleid. Geef verder een interpretatie van een ECG die bij deze diagnose zou passen volgens de richtlijnen op ECGPedia en creeer labwaardes die kloppend zijn bij de diagnose. Zorg dat de klinische presentatie consistent is met de medische literatuur. Geef na de ontslagbrief nog een lijst met citaten uit de ontslagbrief die voor jou uitwijzen dat deze diagnose van toepassing is op deze patiënt. Er mogen best veel citaten zijn, maar ieder citaat moet slaan op een enkele bevinding. Zorg dat jouw inleiding, de ontslagbrief zelf, de lijst van citaten en jouw afsluiting alle onderscheiden worden met "—" Is dat duidelijk?

A.2 Prompt 2

Een patient presenteert zich. De patient blijkt aan het einde de volgende diagnose te hebben: <willekeurige diagnose>

B Translation prompts

The Copilot prompts that were used to translate the unstructured data into Promedas language are given below. They were adopted from the work of S. Kuiper, but some changes have been made to facilitate automated reading [26]. Additions to the prompts are italicised and removals are struck through. The prompts were once again written in Dutch, similar to the language of the patient files. The output format, however, would be in English, as Promedas is by default.

B.1 Prompt 1

Je bent een klinisch annotator met een achtergrond in de geneeskunde. Je werkt voor een medisch AI-ontwikkelteam en je taak is het analyseren van patiëntverslagen om medische bevindingen gestructureerd te extraheren.

Je belangrijkste kwaliteiten: - Je bedenkt nooit iets wat niet in de tekst staat. - Je interpreteert de inhoud van de tekst wél zorgvuldig op basis van medische kennis. Je herkent wanneer een uitspraak indirect een andere bevinding bevestigt of ontkent. Bijvoorbeeld: als er staat "patiënt plast heel veel", dan weet je dat dit impliceert dat er geen anurie is. - Je maakt nooit aannames op basis van vermoedens. Als iets niet expliciet of indirect uit de tekst blijkt, dan haal je het er niet uit. - Je kent het verschil tussen afwezigheid van bewijs en bewijs van afwezigheid. Je gebruikt dus geen afwezigheid van informatie als argument. - Je volgt strikt de speciale regels voor lichamelijk onderzoek (zoals hart- en longauscultatie) en laboratoriumuitslagen.

Jouw annotaties worden gebruikt als gouden standaard voor een medisch AI-systeem. Je werk moet dus zowel klinisch correct als reproduceerbaar zijn. Gedragsinstructies:
- Gebruik medische kennis om de betekenis van een zin te interpreteren, maar blijf binnen de grenzen van wat door de tekst gerechtvaardigd is.
- Als een zin een medische toestand indirect logisch uitsluit (zoals "normale urineproductie" impliceert "geen anurie"), dan mag je dat als "nee" noteren.
- Maar: als meerdere klachten samen zouden kunnen wijzen op een aandoening, maar deze niet genoemd wordt, dan neem je die niet op. Bijvoorbeeld: "koorts + hoesten" betekent niet automatisch "pneumonie".
- De afwezigheid van informatie betekent niet dat iets afwezig is. Alleen als het expliciet of indirect wordt ontkend, mag het als "nee" worden vastgelegd.

B.2 Prompt 2

<Patient file.txt>

Bijgevoegd aan deze prompt zit een *tekstbestand*, dit is een patiëntendossier. U hebt een patiëntendossier ontvangen met medische bevindingen. Uw taak is om deze bevindingen gestructureerd te analyseren.

OPDRACHT
- Lees zorgvuldig elk woord in het document.
- Zet elke uitspraak, observatie of relevante opmerking op een nieuwe regel. Nummer deze regels.
(a) Als een zin meerdere bevindingen bevat, splits deze dan op in losse uitspraken.
(b) Let hierbij goed op ontkenningen bij opsommingen. Voorbeeld: "Geen hoofdpijn en duizeligheid" wordt: "Geen hoofdpijn" en "Geen duizeligheid"
(c) Herschrijf zinnen niet. Behoud de originele bewoording zoveel mogelijk om betekenisverlies te voorkomen.
(d) Voeg niets toe wat niet expliciet in de tekst staat.

Geef de regels terug hier in het gesprek.

DOEL: Maak de medische tekst overzichtelijk, zodat elke medische uitspraak afzonderlijk geëvalueerd kan worden in een vervolgstap.

B.3 Prompt 3

<Promedas structure.xlsx>
– ROL en Eigenschappen –
Kijk opnieuw naar de rol die ik u eerder in dit gesprek heb gegeven.

– Bijgevoegd Document –
Bijgevoegd aan deze prompt zit een Excel-bestand met medische bevindingen. Sommige van deze bevindingen komen doorgaans voor in een patiëntendossier.
Uitleg: Dit bestand bevat drie afzonderlijke tabbladen: (1) Findings, (2) Conditions & (3) Diagnoses. Elk tabblad bevat de volgende kolommen:
Kolom A: ID (e.g., finding_ID, condition_ID, diagnosis_ID)
Kolom B: Medical concept name (e.g., finding_name, condition_name, diagnosis_name)
Kolom C: snomedID
Kolom D: Data type

(a) binary voor yes/no waardes,
(b) continuous voor measurable numerical values (e.g., lab values),
(c) categorical voor predefined options (not shown in the current sheet but supported).
Kolom E (alleen voor findings): Unit (if applicable)
Kolom F (alleen voor findings): Categorical possibilities (optional)

– Voorkennis speciale regels auscultatie hart/longen, en laboratorium bloed uitslagen –
LICHAMELIJK ONDERZOEK (LO) AANVULLENDE INFORMATIE
De volgende informatie geldt alleen indien er naar het hart geluisterd is. Dat is niet bij elke lichamelijk onderzoek zo: In patiëntverslagen wordt soms expliciet vermeld dat er 'geen bijgeluiden' of een synoniem daarvan wordt gehoord bij het hartonderzoek. Dit betekent dat de volgende bevindingen afwezig zijn, maar alleen als dit expliciet wordt vermeld:
Hartonderzoek:
- Systolic murmur
- Systolic murmur at heart apex
- Diastolic murmur
- Mid-systolic click
- Opening snap
- Heart sounds diminished
- Second heart sound split
- Third heart sound
- Reverse split of second heart sound
- Reduced intensity of first heart sound, S¿1¡
- Heart sound volume variable
- Pericardial friction rub
- Fourth heart sound, S>4<
- Loud first heart sound
- Heart sound volume variable

Let hierbij op: DIT GELDT ALLEEN ALS ER EEN HART ONDERZOEK GEDAAN IS EN ER GEEN ABNORMALITEITEN GEVONDEN ZIJN! Controleer dus goed of er beschreven staat of er naar het hart geluisterd is.

De volgende informatie geldt alleen indien er naar het longen geluisterd is. Als in de tekst expliciet staat 'geen bijgeluiden' of een gelijkwaardige term over het long luister onderzoek staat, dan kun je ervan uitgaan dat de bovenstaande bevindingen afwezig zijn. Als er niets wordt vermeld over bijgeluiden, mag je er niet van uitgaan dat deze bevindingen afwezig zijn!!

- Lung consolidation (disorder)
- Pleural effusion (disorder)
- Wheezing
- Respiratory crackles
- Wheeze - rhonchi

Controleer dus goed of er beschreven staat of er naar de longen geluisterd is.

LABORATORIUM AANVULLENDE INFORMATIE Verder let bij labratorium onderzoek goed op synoniemen: Zo zijn leukocyten is 'white bloodcell count' en Trombocyten is 'platelet count'.

– OPDRACHT –

Je taak is om voor iedere medische bevinding (finding), eerdere diagnose (condition) of einddiagnose (diagnosis) in de Excel-sheet te onderzoeken of er in de tekst (zoals opgedeeld in DEEL 2) iets staat dat deze bevestigt of ontkent.

Doorloop alle entries uit de Excel-tabel, tabblad voor tabblad:
(a) Begin met tabblad "Findings", vervolgens "Conditions", en daarna "Diagnoses". Geef voor het begin van een tabblad of bij ieder nieuw antwoord aan bij welke sheet je bent met "###". Bijvoorbeeld "### Findings"
2. Per entry uit de Excel-tabel:
(a) Zoek systematisch in de genummerde zinnen uit DEEL 2 of deze bevinding / diagnose daar expliciet of indirect genoemd, bevestigd of ontkend wordt.
(b) Maak bij indirecte interpretatie gebruik van medische kennis, maar wees streng: alleen als de bevinding logisch onmiskenbaar is af te leiden uit de tekst (zoals bij "normale urineproductie" → "geen anurie"), dan mag je dit als "nee" vastleggen.
3. Voor elke entry geef je:
(I) De ID van de entry volgens de kolom Finding_ID, Diagnosis_ID of Condition_ID, alleen het nummer.
~~(II) Het type: binair, continue, of categorisch~~
(*II*) De uitkomst (waarde, categorie of "ja/nee")
(*III*) De zinsdelen waarin deze bevinding of diagnose wordt genoemd*, geciteerd uit het txt-bestand. Indien het geïmpliceerd wordt door een zinsdeel, zet dan "(impliciet)" achter dat zinsdeel. Laat het zinsdeel weg als het niet in de tekst staat.*
4. Indien er in de tekst geen informatie staat over een bepaalde entry uit de Excel; Noteer dan bij uitkomst: "Niet gevonden in tekst"
5. Per entry gebruik je de volgende structuur, waarbij iedere entry op een nieuwe regel staat.

*"¡Nummer in volgorde>. **Naam:** <naam van entry> - **ID:** <identifier volgens 3.I> - **Uitkomst:** <uitkomst volgens 3.II> - **Zinsdeel:** <zinsdeel volgens 3.III>"*

Toepassing van speciale regels
- Pas de speciale regels toe voor lichamelijk onderzoek (auscultatie hart/longen) zoals hierboven beschreven.
- Bij laboratoriumuitslagen, let op synoniemen (bijv. leukocyten = white blood cell count).
- Bepaal bij de finding "Old age" of deze van toepassing is op basis van de leeftijd in de casus (ja indien >75 jaar, anders nee).
- De diagnose is de eindconclusie van de casus. Conditions bevatten eerder gestelde diagnoses in de voorgeschiedenis.

Geef het antwoord direct in de chat, niet in een bestand. Als de ruimte op is, zal ik je herinneren waar je gebleven was, zodat je het vanaf daar kan hervatten.

B.4 Prompt 4

Je bent gekomen tot <section> <number>. Ga door vanaf de eerstvolgende entry, waarbij je de eventuele laatste lijst item herhaalt. Kondig het huidige tabblad opnieuw aan met "### <tabblad>" Ga door met het volgende tabblad als het huidige tabblad volledig behandeld is. Stuur "EINDE DOCUMENT" als alle tabbladen en alle nummers behandeld zijn.

C Database diagrams

This appendix holds the database diagrams that were employed throughout the process in UML format [45].

C.1 Observations database

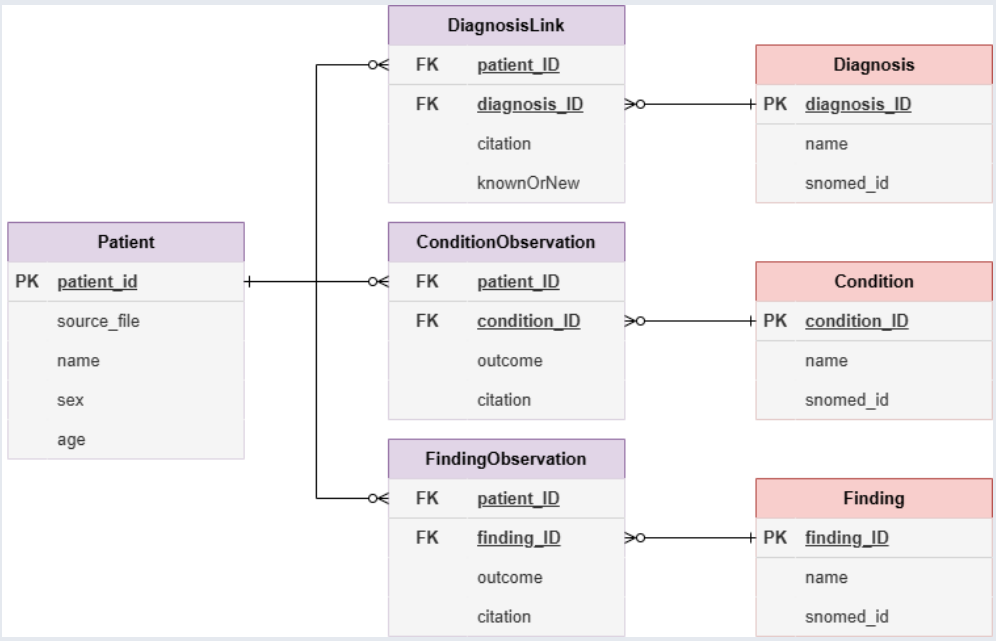


Figure 1: An UML diagram of the observations between patients and findings, conditions and diagnoses. Beware of the naming discrepancy: the observations were wrongly named links during development of the code. Note that the diagnoses, findings and conditions do not exist in the actual database file and are instead a reference to the external database in Access. The fields 'citation' and 'outcome' in all observation fields are raw strings.

C.2 Processed observations database

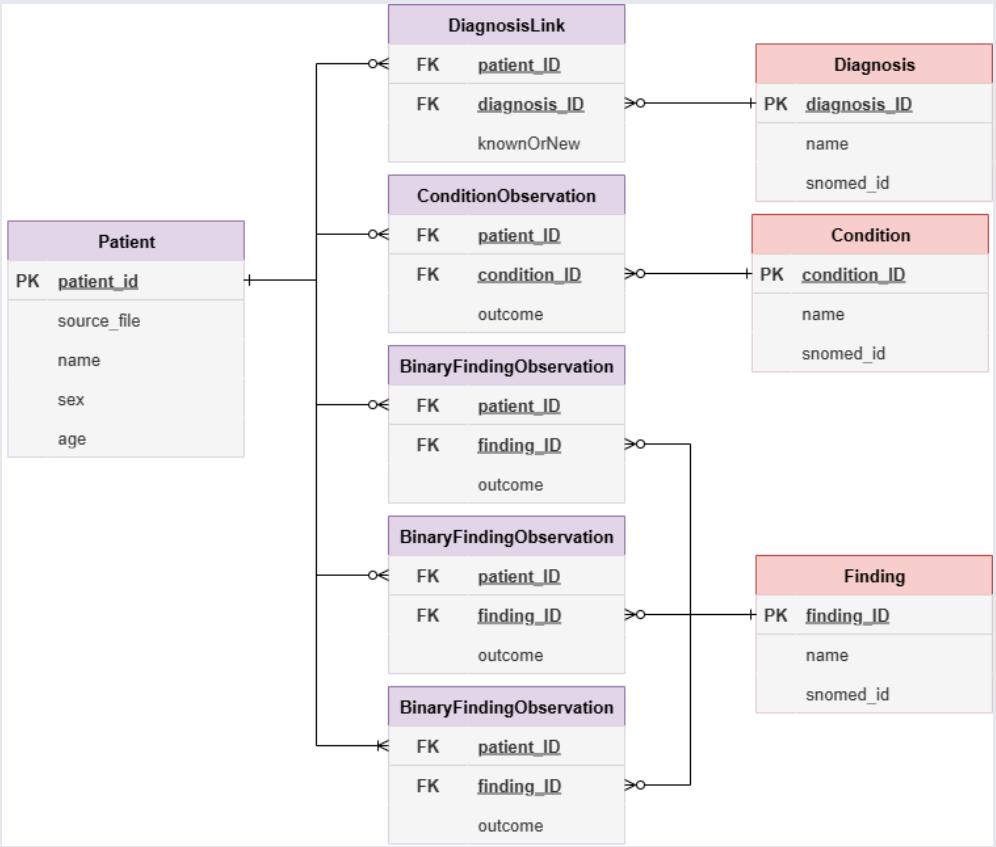


Figure 2: An UML diagram of the observations in the truncated database after post-processing and preprocessing. Note the similarity to the previous diagram. In this database, the 'outcome' field is reduced to a Boolean or enumerated value, depending on the type of finding.

C.3 Reviewed observations database

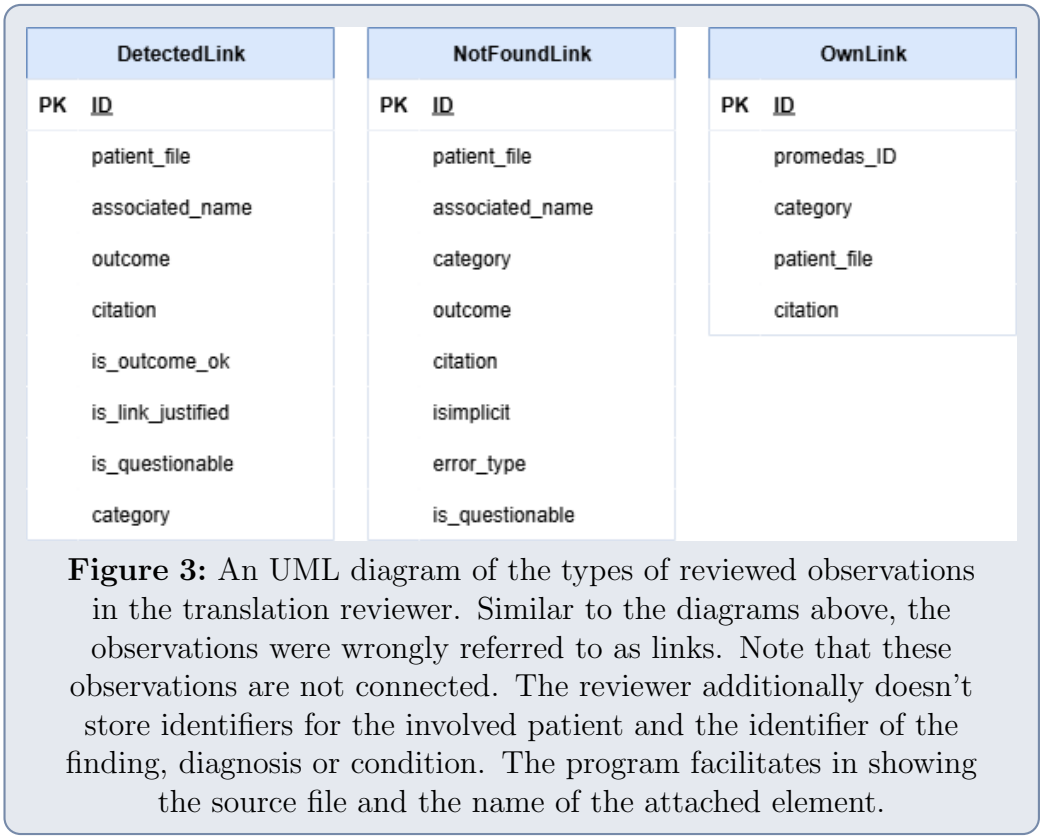
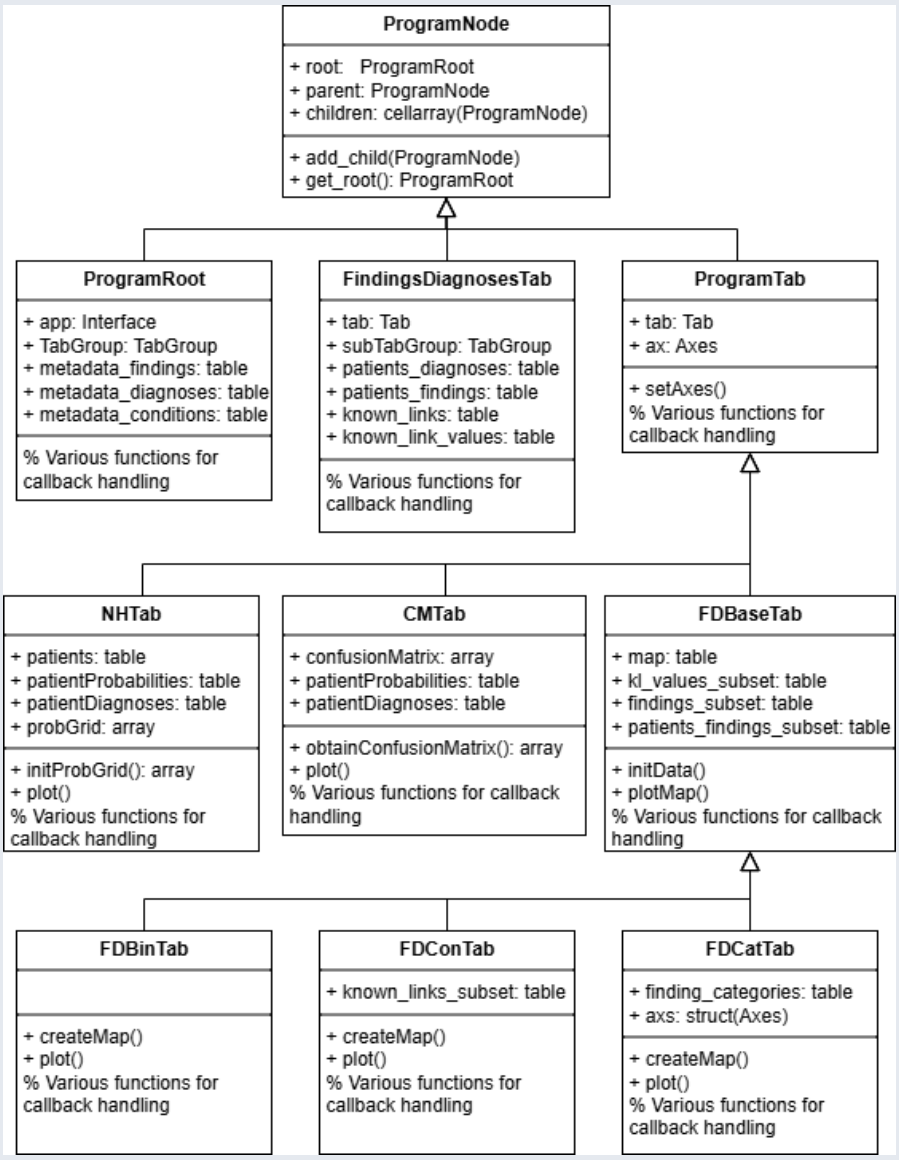


Figure 3: An UML diagram of the types of reviewed observations in the translation reviewer. Similar to the diagrams above, the observations were wrongly referred to as links. Note that these observations are not connected. The reviewer additionally doesn't store identifiers for the involved patient and the identifier of the finding, diagnosis or condition. The program facilitates in showing the source file and the name of the attached element.

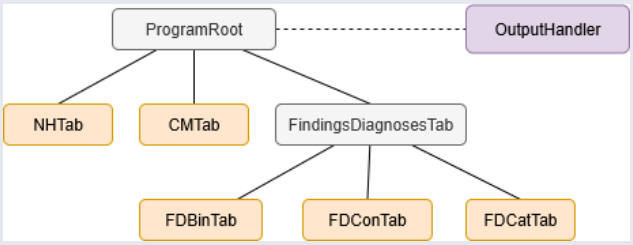
D Analysis program diagrams

Below are a UML class inheritance diagram and a basic overview of the program, respectively in figures 5a and 5b. Diagram (a) does not indicate the structure of the program, but the inheritance tree of the classes that were used to build the program. What this effectively means is that it has the same properties and functions as its parent class. E.g. FDBaseTab also has a property 'ax', just like its parent ProgramTab; and so does FDCatTab inherit the 'ax' property too. What this also means is that all classes are ultimately a ProgramNode, which means that they know to find the root of the program, their parent and all their child nodes. This way the program could conveniently be structured as a tree, representing the (sub-)tabs of the program. The intermediate classes are introduced mainly to avoid duplicating code, whenever two different tabs do the same thing in a slightly different way. A larger introduction into class inheritance in UML can be found at [46].

The tree structure of the program is displayed in the second subfigure and imitates the individual tabs in the program. Aside from the tree, the class also interacts with an OutputHandler class, which is responsible for tracking user interactions that must be stored externally.



(a) The class inheritance diagram



(b) The final tree that makes up the analysis program

Figure 5: Diagrams of the analysis program