
Quality Assessment of Medical Health Records using Information Extraction

Master Thesis of Guido van der Zanden
Computer Science,
Track Information System Engineering
Enschede, August 19, 2010

Supervisors University of Twente:

Dr.ir. M. van Keulen
Dr.ir. A. de Keijzer

Supervisors Topicus Zorg:

V. Ivens, Msc.
D. van Berkel, Msc.

Abstract

The most important information in Electronic Health Records is in free text form. The result is that the quality of Electronic Health Records is hard to assess. Since Electronic Health Records are exchanged more and more, badly written or incomplete records can cause problems when other healthcare providers do not completely understand them. In this thesis we try to automatically assess the quality of Electronic Health Records using Information Extraction. Another advantage of the automated analysis of Electronic Health Records is to extract management information which can be used in order to increase efficiency and decrease cost, another popular subject in healthcare nowadays.

Our solution for automated assessment of Electronic Health Records consists out of two parts. In the first part we theoretically determine what the quality of Electronic Health Records is, based upon Data and Information Quality theory. Based upon this analysis we propose three quality metrics. The first two check whether an Electronic Health Record is written as prescribed by guidelines of the association of general practitioners. The first checks whether the SOEP methodology is used correctly, the second whether a treatment is carried out according to the guideline for that illness. The third metric is more general applicable and measures conciseness.

In the second part we designed and implemented a prototype system to execute the quality assessment. Due to time limitations we only implemented the SOEP methodology metric. This metric tests whether a piece of text is placed in the right place. The fields that can be used by a healthcare provider are (S)ubjective, (O)bjective, (E)valuation and (P)lan. We implemented a prototype based upon the 'General Architecture for Text Engineering'. Many generic Information Extraction tasks were available already, we implemented two domain specific tasks ourselves. The first looks up words in a thesaurus (the UMLS) in order to give meaning to the text, since to every word in the thesaurus one or more semantic types are assigned. The semantic types found in a sentence are then resolved to one of the four SOEP types. In a good Electronic Health Record, sentences are resolved to the SOEP field they are actually in.

To validate our prototype we annotated text from real Electronic Health Records with S,O,E and P and compared it to the output of our prototype. We found a Precision of roughly 50% and a recall of 20-25%. Although not perfect, because we had time nor resources to involve domain experts we think this result is encouraging for further research. Furthermore we shown that our other two metrics are sensible with use cases. Although no proof they are feasible in practice, they show that a whole set of different metrics can be used to assess the quality of Electronic Health Records.

Voorwoord

Voor u ligt de master thesis van Guido van der Zanden, ter afsluiting van mijn master Computer Science, track Information System Engineering, aan de Universiteit Twente. Deze is tot stand gekomen na iets meer dan zes maanden afstudeeronderzoek bij Topicus Zorg in Deventer. Het voorwoord is het eerste en enige stukje Nederlands dat u te lezen zult krijgen. De aard van een voorwoord staat mij toe iedereen aan te spreken in hun eigen taal, wel zo gemakkelijk.

Ten eerste wil ik Daan en Vincent, mijn begeleiders bij Topicus Zorg, bedanken voor hun steun. Beiden hebben ze op hun eigen manier bijgedragen aan het volbrengen van mijn afstudeeronderzoek. Vincent voornamelijk in de beantwoording van theoretische vraagstukken, Daan vooral bij het oplossen van technische problemen. Via hen wil ik ook heel Topicus Zorg bedanken voor het feit dat mijn afstuderen een zeer leuke periode was, beter kan ik me niet voorstellen.

Ten tweede wil ik ook Maurice en Ander bedanken voor de begeleiding vanuit Enschede. Alhoewel minder close betrokken bij mijn dagelijkse bezigheden hebben zij door het reviewen en feedback geven mij geholpen stappen te maken in mijn afstuderen.

Als laatste wil ik mijn ouders bedanken. Zij zijn degene die mij zowel de vrijheid als de mogelijkheid en kans hebben gegeven te kunnen doen wat ik doe en mij daar altijd en overal in te steunen.

Guido van der Zanden
Enschede, augustus 2010

Contents

Abstract	i
Voorwoord	iii
1 Introduction	1
1.1 Motivation and goals	2
1.2 Research questions	3
1.3 Outline	4
I Background	6
2 Electronic Health Records	7
2.1 Structure & Content	8
2.2 EHR Architecture	9
2.3 Guidelines	9
3 Information Extraction	11
3.1 Process	12
3.2 Typical IE tasks	12
3.3 Heuristics and Learning	13
3.4 General Architecture for Text Engineering	14
II Quality of Information	15
4 Information Quality Theory	16
4.1 Problems	16
4.2 Dimensions	18
4.3 Metrics	19
4.4 Information Quality of Unstructured Data	19
4.4.1 Quality of web data	19
4.4.2 Quality of requirement documents	21
4.4.3 Quality of Wikipedia pages	22
4.5 Summary	22

5	Quality of Electronic Health Records	24
5.1	EHR Problems	24
5.2	EHR Dimensions	26
5.3	EHR Metrics	27
5.3.1	SOEP Test indicator	27
5.3.2	Guideline compliance indicator	28
5.3.3	Conciseness indicator	28
5.4	Summary	29
III	System Design	31
6	Information Extraction Pipeline	32
6.1	Preprocessing	32
6.2	Specific Annotation	35
7	Global Design	37
7.1	Extraction Executor	37
7.2	Model	38
7.3	GATE components	39
7.3.1	UMLS MetaMap	40
7.3.2	NounChunk Classifier	41
IV	Evaluation	42
8	Experimental Evaluation	43
8.1	Goal	43
8.1.1	Example output	44
8.2	Experimental Set-up	45
8.2.1	Dataset	46
8.2.2	Experiment	47
8.3	Results	48
8.3.1	Machine Learning	48
8.3.2	Rule-based	48
9	Use Cases	50
9.1	Guideline compliance	50
9.2	Conciseness	51
10	Evaluation conclusions	53
V	Conclusions and Further Research	54
11	Conclusions	55

12 Further Research	57
12.1 Prototype improvement	57
12.2 Metric improvement	57
12.3 Metric implementation & Quality Scoring	58
12.4 Medical Support System	58

Nomenclature

CDM	Conceptual Dependency Model
DQ	Data Quality
EHR	Electronic Health Record
EOR	Episode-Oriented Registration
EPD	Elektronisch Patient Dossier
FT	Frame Theory
GATE	General Architecture for Text Engineering
GBZ	Goed Beheerd Zorgsysteem
GP	General Practitioner
HIS	Huisarts Information Systeem
ICPC	International Classification of Primary Care
IE	Information Extraction
IQ	Information Quality
LSP	Landelijk SchakelPunt
MMTx	MetaMap Transfer
MSS	Medical Support System
NHG	Nederlands Huisartsen Genootschap
NLP	Natural Language Processing
POR	Problem-Oriented Registration
POS	Part-Of-Speech
RFE	Reason For Encounter
UMLS	Unified Medical Language System

Chapter 1

Introduction

Topicus Zorg (Topicus Healthcare in English) provides Software-as-a-Service (SaaS) applications for various actors in the Dutch healthcare sector. In their products the information regarding patients is often the central pivot around which the applications are built, although other functionality such as planning or invoicing is provided as well. In The Netherlands, the whole set of medical data electronically available belonging to one patient is often referred to as the Elektronisch Patient Dossier (EPD). However, since internationally it is called the Electronic Health Record (EHR), we refer to it as EHR. In this research we investigate if and how we can automatically assess the quality of EHRs.

There are numerous reasons to investigate the automated assessment of EHRs. The two most important ones are the exchange of EHR information and the quality of healthcare in general. For exchanging information, intuitively understandability and readability are of great importance. Unclear written reports can be understood by the writer himself but not by other healthcare providers. EHR information that is not well understood can lead to problems in crucial situations, for example when conflicting medication or allergies are overlooked. The quality of healthcare argument means that EHRs are filled with valuable information which cannot be gathered at the current moment. The gathered and aggregated information can give insight in the effectiveness and efficiency of treatments.

The challenge in assessing EHRs is that much of the valuable information is in free text form. The basis of every EHR, next to some structured and predefined input, is a set of four free text fields in which the practitioner has to fill in all information concerning one appointment (called a contact) with the patient. Therefore traditional ways of analysing data such as Online Analytical Processing (OLAP) do not suffice, since they are incapable of dealing with unstructured data. In order to overcome these limitations, we apply information extraction (IE), which aim is to “analyse unrestricted text” and “get facts out of documents” [1]. The facts we extract from the EHRs form the basis for the quality assessment. However, we are not searching for random facts, but only the facts we can use in our quality assessment. The challenge is to create a IE process that extracts exactly the facts we want to know. Therefore we must investigate (a) what information we want to extract and (b) how we make the IE process extract exactly that information. Since the information extraction process is ‘dumb’ in the sense that it does not understand what we are looking

for, we must train it to extract the facts we are looking for.

1.1 Motivation and goals

As stated above there are two main reasons for automatised assessment of EHRs, which we will discuss here in more detail. First, in the near future all Dutch healthcare providers must exchange patient information. Furthermore, on regional level initiatives are carried out to exchange medical information, for example to streamline the care for diabetes patient in one region. One can say that the management of an EHR is becoming a collaborative undertaking by different healthcare providers. This trend has the advantage that healthcare providers can give care based on accurate information, which avoids the need for the patient to tell his story twice or preventing prescribing conflicting medication. However, this promise can only be made true when the information in the records is understandable for all healthcare providers working with it. Unclear abbreviations, ambiguous usage of terminology or other ambiguities undermine the seamless exchange of information between healthcare providers. Automated assessment of the quality of the records thus helps to achieve seamless exchange of information, since it can help healthcare providers to improve the quality of Electronic Health Record keeping.

The second motivation for assessing healthcare records is to provide ‘management information’. Already initiatives are taken implementing Routine Outcome Monitoring in healthcare, which is simply periodically measuring the outcome of treatments. By gathering this information the effectiveness and efficiency of a treatments can be measured and benchmarked. An example of the information gathered is filled in questionnaires regarding the seriousness of psychological complaints [2]. When over time more questionnaires are filled in we can see, both on high and low level, the course of complaints and thus also the effectiveness of treatment. If we can extract information from patient records automatically from patient records, we could continuously gather the information needed to investigate effectiveness and efficiency of healthcare.

There are other motivations for assessing EHR. For example, patient records are stored for at least 15 years. Guidelines for proper record keeping were not implemented back than and thus do not comply to current standards. Automated assessment could help filter out the useful information and store them as current guidelines prescribe. Concluding, there is a desire to extract the relevant information from the unstructured parts of EHRs. Therefore, the first goal of our research is thus formulated as follows:

Goal 1 describe the relevant facts that can be extracted from the unstructured parts of Electronic Health Records.

As stated before, the facts form the basis of our assessment. When these are formulated we must provide a manner to extract them from EHRs and use these to measure the quality of the EHR. Thus, our second goal is simple:

Goal 2 design and implement a prototype system that can extract the relevant facts from an Electronic Health Record and assess the quality of the Electronic Health Record.

1.2 Research questions

Our main research question follows logically out of the introduction and is stated as follows:

Main research question How to automatically assess the quality of Electronic Health Records using Information Extraction?

IE provides us with the methods and algorithms, we have to ‘feed’ it with our knowledge regarding what information we want to extract and how it should do so. Furthermore, we must interpret the results of information extraction ourselves in order to come to useful conclusions. Our first sub questions are therefore related to the quality of EHRs to provide a context for the IE process:

- I How to define the quality of Electronic Health Records?
 - a How can - in general - quality of data and information be defined and measured?
 - b How can we define the quality of Electronic Health Records using the general data and information theory?

First we investigate what exactly is meant by ‘the quality of Electronic Health Records’. We generalise this question by conducting research into the quality of data in general in order to provide theoretical background. This background is compared with expert opinions and guidelines regarding the quality of EHRs. This is done in order to see whether we can map the expert opinions and guidelines to the theoretical data quality background. This should simplify the assessment of EHR quality, since we can use former experience of (general) data quality. The second sub questions are formulated as follows:

- II How to extract the facts from the Electronic Health Records using Information Extraction?
 - a Which Information Extraction steps have to be undertaken?
 - b Which Information Extraction steps can be reused and which must be newly implemented?
 - c How to design a reusable software architecture that can execute the Information Extraction process?
 - d How to translate the extracted facts to a quality judgement?

The second step is to use IE to extract the facts that can answer the quality measures defined earlier. In order to do so we must investigate which steps are needed in the process. Since IE is applied many times before on different types of texts, we must reuse the experience of these works. However, we need to adapt the steps to fit our use and probably also implement new steps. As last, to proof that our concept works we need to design and implement it in a prototype. Since we will probably find multiple manners to assess the quality of EHRs, it is convenient to have a prototype that can easily be adapted to serve multiple tasks. As last we need to translate the extracted facts into a quality judgement as we have defined previously.

III How to validate our thesis?

- a What is the performance of the prototype?
- b How to validate our quality measurements?

The last question is a question present in every thesis, namely the validation of the work. First of all, since we have a prototype we must validate it. In IE validation is quantitative by three standard measurements described later in this thesis. IE never gives a 100% good result due to the complexity of unstructured data, thus we need to know how well it performs. The performance of our prototype tells us whether our concept is feasible: can we assess the quality of EHRs using IE. Secondly, we need to validate our quality measurements are right. If we can perfectly measure something, it does not mean the measure itself is proper. We can measure with a 100% accuracy the number of dots in a text, but it will not tell us anything about the quality of that text. A qualitative validation is needed here, for example by medical experts that can judge from experience and knowledge what a good EHR looks like.

1.3 Outline

The research is divided into five parts. In the first part the EHR is introduced in more detail, as well as the basics of IE. Part I thus introduces the building blocks of our research. Part II discusses both the theory and practice of quality of information, respectively based on a literature research and protocols and guidelines. Part III discusses the system design: what IE steps needs to be executed and how to fit it in a software architecture. Part IV then evaluates our prototype and Part V discusses our conclusions and further research.

Part I: Background In the background part we simply explain the structure and content of the EHR in Chapter 2. The structure defines on which parts we will perform IE, namely the SOEP entries which will be explained later, but also which extra data we can use to assess the quality of the EHR, for example unique treatment codes. Chapter 3 gives an overview of the general paradigm and common tasks of IE. We will show how we can use IE in order to extract the facts we want to find and finally an open-source IE initiative we will use in our prototype.

Part II: Quality of Information In Chapter 4 an overview is given of theory regarding quality of data and information. The theory is then combined in Chapter 5 with the demands from the medical area regarding EHR quality. In this chapter we try to relate the needs stated in protocols and guidelines to the theory. As a result we can define the quality requirements of EHRs from practice in theoretical terms which should make it easier to assess them as quantitative metrics.

Part III: System Design This part will form the technical heart of the research. In Chapter 6, based on the previous defined quality metric, we describe the IE pipeline: what steps do we take in order to extract the information we want. In Chapter 7, we describe the system in a broader sense. We discuss the data models used to store the results and how to evaluate the results to give a quality judgement about the EHR.

Part IV: Evaluation In Chapter 8 we both explain the methods of evaluation as the results of the evaluation. The quantitative evaluation of the system

will consists of the common IE evaluation measurements, namely precision and recall. In Chapter 9 we present use cases to validate our quality measurements qualitatively. In Chapter 10 the conclusions of the evaluation will be given.

Part IV: Conclusions and Further Research Chapter 11 will state our final conclusions and Chapter 12 will give some suggestions on how to continue the research on the quality of EHRs.

Part I
Background

Chapter 2

Electronic Health Records

Patient records are kept by every healthcare provider; general practitioners (GP), surgeons, physiotherapists, nurses and so on. Although for example GPs note different information than surgeons, the goal of keeping records is the same: to memorise all important events in the medical history of a patient for future use. In the case of GPs, on which this research is focused, the record will consist of reports of all contacts with a patient, all his deceases, medication and allergies. The information about former and present deceases is needed to make new diagnoses, medication and allergy information is needed in order to prevent conflicting prescriptions.

Historically, records were kept for internal use in the practice only. Thus, the manner and system in which records were kept was dependant on the practice. In 1990, the Nederlands Huisartsen Genootschap (NHG, Dutch GP Society) published a standard for GPs on patient record keeping [3]. Among other aspects readability, completeness, briefness and understandability were promoted. A review study in 1994 showed that record keeping of GPs did not satisfy those aspects. Most (hand-written) records were not readable and did not use the prescribed Problem-Oriented Registration (POR) system.

Not adhering to prescribed standards is not an issue if the writer is the only reader of a records: as long as he can understand it, everything is fine. However, when medical records are shared with colleagues in the same practice or with other institutions such as hospitals, problems can arise. This latter situation where information between healthcare practitioners is exchanged, is rapidly becoming reality. The Dutch government already set-up the infrastructural needs to facilitate exchange of patient records between different healthcare provider, and legislation is expected to follow soon. In this environment standardised, clear and understandable record keeping.

In the remainder of this Chapter we introduce the structure and content of patient records. This is done by demonstrating Promedico-ASP, a ‘Huisarts Informatie Systeem’ (HIS, GP Information System) which is sold by Promedico and (partially) implemented by Topicus Zorg. The structure we discuss adheres to the current standards of the NHG, namely the Episode-Oriented Registration (EOR) and the SOEP standard, which we will explain also. Furthermore, we discuss the global architecture of the ‘Elektronisch Patient Dossier’ called AORTA. Finally we discuss some guidelines and initiatives regarding the quality of EHRs.

Figure 2.1: An empty contact in Promedico-ASP

2.1 Structure & Content

The central pivot in Promedico-ASP, and many other HISs, are contacts with patients. In real life situations a patient comes to a GP for a consult, without the GP knowing exactly what the problem of the patient is. It is therefore sensible to organise the EHR on contacts. The basis of every contact are one or more sub contacts. The reason for sub contacts is to register different complaints under one physical contact. For example a patient who visits the GP for his regular diabetes check-up and simultaneously for pain in his wrist. This division is needed in order to aggregate all contacts of the same kind in one episode (see below). Furthermore, insurance regulation only permits to compensate one contact per day.

The basis of every sub contact is a SOEP journal (see Figure 2.1). SOEP is a method for registering contacts which is implemented in most Dutch HISs. The goal of a SOEP-journal is to make sure GPs follow a logical line of reasoning and do not forget steps in this process. A SOEP-journal consists out of four fields:

Subjective Feelings, observations and perception of the patient and thus the reason for the contact.

Objective Symptoms and signals the GP can observe and measure.

Evaluation Interpretation of complaints and symptoms resulting in one or more diagnoses.

Plan Activities to solve the problems, for example further examination or refer to a specialist.

Every contact is described in a SOEP-journal, although it is not obliged to fill in all journal lines (at least, it is not enforced by Promedico-ASP). In the E-line of a SOEP-journal, an International Classification of Primary Care (ICPC) code can be attached. The ICPC codes refer to standardised reasons for encounter (RFE), problems or diagnoses and primary care interventions. The attachment to the E-line is logical, since the E-line registers the diagnosis. Additionally, attached to the ICPC code one can add the label New, Again and Continuation to indicate whether the diagnosis is occurred for the first time, it occurred again or is a continuation of a previous occurrence.

In HISs it is common to apply EOR. Contacts are grouped into episodes which means the contacts relate to the same medical affection, and is basically done one ICPC code (see Figure 2.1). In this manner a GP has an list overview of all current affections and can quickly access all information regarding that affection in the follow-up contact for that affection. Additionally one can see directly whether the episode has ended or whether it deserves extra attention (for example diabetes or pregnancy). Episodes are another reason sub contacts are necessary, otherwise one could not distinguishably assign a contact to one episode.

2.2 EHR Architecture

The initiative to enable the exchange of patient records was taken in order to reduce the number of medical errors, give healthcare practitioners insight in all relevant information of a patient and thus reduce the need for patient to redundantly tell their story [4]. Due to legislation, all healthcare providers are obliged to open up their systems to others. Through the Landelijk Schakel Punt (LSP, National Switch) healthcare providers can fetch patient records from other healthcare providers. The LSP dictates the protocols over which the records are communicated and providers of healthcare applications may only use the LSP when they have the qualification ‘Goed Beheerd Zorgsysteem’ (GBZ, Properly Managed Care system). In short this means that a HIS must oblige to connectivity standards, security, availability, response time and so on. This implicates that all sorts of HISs can exist next to each other, which not by definition have to enforce the same input. In other words, the introduction of the EHR architecture does not enforce all HISs to use the same structure. Exchange of information is enforced in this way, but not a completely unambiguous way to structure, store and organise EHRs.

2.3 Guidelines

The NHG created a guideline for adequate patient record keeping called ADEPD [3]. The guideline describes the choice for EGR in contrast to POR, the content that should be in a HIS (personal, contact, medication and allergy information) and some general guidelines regarding registering contacts. Roughly, the guideline prescribes the use of the SOEP-system, but gives no further advice or guidelines to fill the SOEP-journal lines. However, the ADEPD document does

refer to the NHG-Standaard Medische verslaglegging (NHG-Standard Medical Reporting), which cannot be found any more on the NHG-website. However, another web source refers to the NHG-Standaard, stating it has six demands for proper reporting: availability, readability, completeness, brevity, reliability and understandability [5]. One must note that the standards was made in 1990 and refers to paper records. However, we think it still is applicable to digital patient records.

Next to ADEPD, two other projects related to EHR quality. First of all, the EHR-scan is a project to ‘benchmark’ ones HIS. Indicators that are tested are for example the number of episodes per patient or the percentage of contacts that is attached to an episode [6]. Two remarks must be made here, first that this scan only evaluates structured data and second that the indicators are more useful for comparison with national averages than direct quality measures. The last form of guidelines are the NHG-Standaarden [7]. For about 50 deceases the NHG created guidelines with the latest (scientific) insights. The guidelines contain information regarding the symptoms and complaints, the physical examination that the GP has to perform, information concerning the diagnoses (related to the physical examination, e.g. glucose values related to diabetes) and guidelines for a treatment plan (education, medication etc.). Although not directly related to the quality of EHRs, the guidelines follow the same line of reasoning as the SOEP-system. When investigating an episode for which a NHG-standard exists, one will expect to find similarities in the line of reasoning in the EHR and the NHG-standard.

Chapter 3

Information Extraction

As stated in the introduction, “information extraction gets facts out of documents” [1]. Turmo et al. [8] formulate a slightly more detailed definition, being that “the objective of IE is to extract certain pieces of information from text that are related to a prescribed set of related concepts, namely, an extraction scenario”. The extraction scenario is a notion of context to an IE system which without it cannot function. We first give the system a notion of what the facts are we want to know, only then the system can perform. A classic example of IE is the extraction of mergers between companies from newspaper articles. For example, from the sentence “Yesterday, New-York based Foo inc. announced their acquisition of Bar Corp.” the merger between Foo Inc. and Bar Corp. would be extracted in such an extraction scenario.

IE, but actually Natural Language Processing (NLP), originates from the Conceptual Dependency Model (CDM) introduced by Roger C. Schank [9]. The basic assumption is that there is an interlingual conceptual model onto which every language structure can be mapped. In other words, text can be analysed and mapped onto a conceptual model consisting of conceptual syntax and semantic rules, making text machine understandable. Although with a CDM a text can, in theory, be completely understood, mostly partial CDMs were implemented. Following Schank’s theory, Minsky introduced the Frame Theory (FT). A frame is a “data structure for representing a stereotyped situation” [9], for example the acquisition of Bar Corp. by Foo Inc. in the example above. Frames, also called slots, are defined before text analyses can be executed, together making an extraction scenario. Frames are simple key value pairs, but when frames have mutual relationships a semantic net is created. In the example above both Foo Inc. and Bar Corp. fill a slot (company1 and company2 for example), but the link between the slots defines their relation: company1 acquires company2.

CDM initiated lots of follow-up research into NLP, FT shaped IE as it is today: filling templates with values. Although sounding simple, a lot of steps have to be undertaken before the templates are filled. In this Chapter we will first introduce the basic IE process, and elaborate on some of the most common tasks in the process. Furthermore, we shortly introduce the concept of machine learning. Although text constructs can be described with rules, due to the diversity in language it was found impracticable to do this manually. With learning algorithms it was tried to relief the workload.

3.1 Process

The IE process can best be seen as an assembly line, called the pipeline, with text as input where at every step some information is added. At the beginning of the pipeline simple information is added such as word and sentence boundaries, but as the text goes further through the line, more complex information is added such as the annotation of persons, location or events. However, the assembly line on which the text is processed is only one half of the total picture. The different steps on the line need to know what to do, they must be trained. Training roughly comes in two flavours: manually learned heuristics, rules and patterns or by machine learning where rules are learned based on a number of examples.

Figure 3.1 depicts the typical process of training and deployment of IE steps. Next to the training phase, in which a step is learned how to annotate text, it is important to note that each step depends on the previous one. For example, a step that annotates noun groups (i.e. “the yellow bike”) needs a process called a POS-tagger to annotate words with properties like noun, verb or determiner. The philosophy behind the process is thus that richer information can be extracted by gradually adding more information.

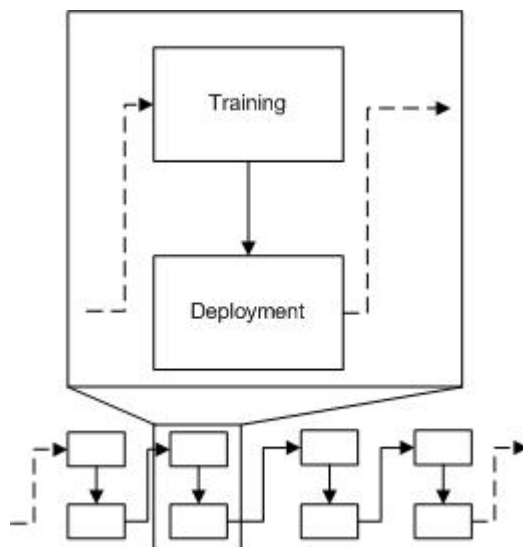
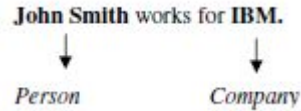


Figure 3.1: The typical IE process [9]

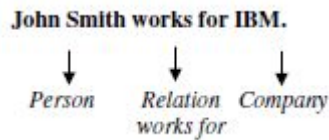
3.2 Typical IE tasks

In this Section we describe some typical (high-level) IE steps to illustrate the capacities of IE. Many low-level steps precede the example output we give, but we will not elaborate on those. In Chapter 6, our system design, we will elaborate on these low-level steps. All examples were taken from Moens [9].

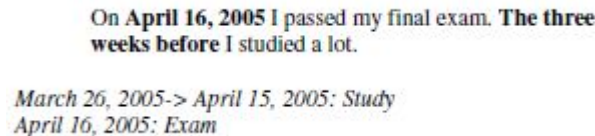
Named Entity recognition Named Entity recognition tries to extract entities such as persons or companies. Example:



Entity Relation recognition Relations can exist between named entities, as is shown in the following example:



Timex and Time line recognition Next to just recognising times and dates, both exact (Friday, July 1st) and relative (yesterday, next month) more complex time constructions can be extracted:



3.3 Heuristics and Learning

Two types of training exist, based on heuristics or on learning. Heuristics are no more than hand-made rules. It is evident that for simple tasks this is feasible, but for more complex tasks it is a tedious task. Complex tasks require many rules, and the rules itself can become very complex. Although heuristics can be powerful in IE, constructing them is a tedious, time consuming and complex task.

Machine learning is used in order to relief the burden of manually creating rules for each IE step. Machine learning comes roughly in two flavours: supervised and unsupervised. Supervised learning means the system needs a fully annotated text from which it can deduce rules. These rules can in the deployment phase be used to annotate new text. Unsupervised learning does not need a annotated text but tries to divide the text into separate groups (called clustering). Again, the learned model can be used in deployment phase. In both cases the machine learning algorithms need parameters to base there actions upon. A parameter could the part-of-speech type of a word previously annotated. The algorithm tries to map the parameters to what we want to annotate. For example, part-of-speech types are used to extract the word's semantic type (person, location etc.). The annotation we want to map to are called the classes.

3.4 General Architecture for Text Engineering

Since IE is a matured research topic, many problems have already been solved. Named Entity recognition systems for English text have almost human performance¹. Therefore, we do not need to re-invent the wheel but can reuse many work done before. Therefore we decided to build our prototype based on the General Architecture for Text Engineering, or GATE [1]. In order to make it easier to understand our prototype, we shortly introduce the working of GATE by shortly explaining its most important concepts.

First of all, GATE is a software package implemented in Java consisting out of a graphical user interface (GUI) that enables users to build an IE pipeline in a ‘click and point’ style. Next to the GUI GATE offers an API that enables programmers to embed GATE into their own software. In GATE, IE steps are called processing resources which are dynamically loaded into GATE. In order to use a processing resource we first need to specify where the processing resource is located and with which parameters it needs to be loaded. Secondly, we need to set the run-time parameters of the processing resource before we can run it. A run-time parameter is for example the text it takes as input. The design and process to use processing resources of GATE enables that new processing resources can be created and used in GATE easily.

Next to processing resources, GATE has language resources. Language resources are text combined with annotations. In GATE, annotations are divided into annotation sets. In an annotation set, multiple annotations can exist. An annotation contains a reference to a piece of text in the language resource. This piece of text can be of any size: A letter, word, sentence or any other piece of text. The annotation itself has a name to make it recognisable. For example the typical tokeniser in GATE annotates words and punctuations as ‘Token’. Annotations themselves contain features, which are used to specify properties of an annotation. In the ‘Token’ example a feature named ‘type’ is attached to the annotation, specifying the type of token: word, number or punctuation.

¹http://en.wikipedia.org/wiki/Named_entity_recognition

Part II

Quality of Information

Chapter 4

Information Quality Theory

The term information quality (IQ) is interchangeably used with data quality (DQ) [10], and has been referred to as “data quality in context” by Strong et al. [11]. Where traditionally Data Quality referred to intrinsic quality properties, general quality literature showed that data quality cannot be assessed independent of the people who use the data. Literature on data, information and knowledge claim a more fundamental difference between data and information, being that “data are pattern with no meaning” and “information is data with meaning” [12]. Since EHRs should consist out of data with meaning, we use the term information quality to emphasise this and to distinguish from early literature focused on solely intrinsic data quality.

Information is available in structured, semi-structured and unstructured form. However, because of the historical relation of Data Quality with database design, most research in Data Quality is related to structured data [13]. Since our focus is on SOEP journal lines, which is unstructured data, no or not much research is available that provide ready to use measures and methods to assess quality. However, in almost all Information Quality research, a distinction is made between abstract dimensions and concrete metrics. A dimension defines in general a quality aspect of information, whereas it can only be measured using one or more concrete metrics. Therefore, we can reuse the abstract dimensions to base our own EHR quality metrics upon.

In the remainder of this chapter we focus on the causes of Information Quality problems as well as Information Quality dimensions. Both investigations can provide us with the information needed to define our own Information Quality metrics. If we can link Information Quality problems to Information Quality dimensions, the problems encountered in the usage of EHRs can lead us to the dimensions we must assess. As last we review some existing metrics, which are all related to structured data, and discuss former attempts to assess the quality of unstructured data.

4.1 Problems

As stated above, an analysis of Information Quality problems can help in identifying the dimensions relevant to the quality of EHRs. We shortly discuss three investigations which mention Information Quality problems, although only one

explicitly links Information Quality problems to Information Quality dimensions.

First, Gackowski [14] bases the possible Information Quality problems on a schema of information. In this schema a distinction is made between informing entities and informed entities, and both are subsequently divided into active and passive entities. Entities can be robots, individuals, organisations or any other entity capable of sending and receiving information. In short, the difference between passive and active entities is that active entities can choose to inform or be informed, for example a teacher and a student. In between informers and informed entities direct or indirect informing takes place. Indirect informing is routed via for example databases and data warehouses.

All problems identified by Gackowski [14] are based on the schema of informing. To start, every informing entity has the problem of credibility and believability. Active informing and informed entities are also affected by intentionality: why do I inform/want to be informed? The communication channels between entities are the source of transmission quality problems, where transmission means psychical transmission through cables using network protocols. At the 'entry side' of informed entities, information presentation quality problems may arise, while at the exit side information utilisation problems may occur. As last, data delivery systems can be faced with information mapping quality problems, database, transmission and network problems, as well as accessibility problems.

Ge and Helfert [15] surveyed current Information Quality research and came with a categorisation of Information Quality problems. The categorisation was made by two axis: data perspective vs. user perspective and context-independent vs. context-dependent. All Information Quality problems found were divided into the quadrants. The context-independent problems consisted out of among others spelling errors, duplicate data, outdated data in the data perspective, and for example of inaccessible and insecure information in the user perspective. The context-dependent half has data problems such as domain constraint and government regulation violations, whereas the user perspective quadrant holds problems such as information that is hard to manipulate, aggregate, incompleteness or impartiality.

Stvilia et al. [16] conducted the only investigation linking Information Quality problems to Information Quality dimensions. They describe four causes of Information Quality problems which have affect on multiple dimensions. Also, dimensions can be affected by multiple problems. The problem causes mentioned are mapping, changes to the information entity, changes to the underlying entity and context changes. Mapping problems occur when there is incomplete, ambiguous, inaccurate, inconsistent or redundant mapping between the information and some state or event. The changes in information entities or the underlying entities speak for themselves, but can be caused in many ways. For example, malicious changes are something completely different than changes that are made to increase the Information Quality. This goes for changes in the information entity itself as well as to changes in the underlying entity. Regarding to a changing context, a context defines how information and Information Quality is assessed, because of underlying cultural or socio-technical assumptions. Any change in spacial or temporal context can have influence on the Information Quality.

Intrinsic	Contextual	Representational	Accessibility
Accuracy	Relevancy	Interpretability	Accessibility
Objectivity	Value-added	Concise representation	Access security
Believability	Timeliness	Consistent representation	
Reputation	Completeness	Understandability	
Consistency	Amount of data	Readable	
Precision			
Completeness			
Reliability			
Currency			

Table 4.1: Information Quality Dimensions [11, 16, 17, 20]

4.2 Dimensions

Information Quality dimensions are intuitively described quality aspects of data, which are not directly measurable. Instead of a single calculation or measurement, one must calculate one or more indicators or metrics. As a result, many dimensions have multiple metrics and due to overlapping dimension definitions, some dimensions may have the same metrics.

Since dimensions are quite easy to ‘make up’, researchers tried to categorise dimensions. One of the most adopted view is proposed by Strong et al. [11], who divided Information Quality dimensions into intrinsic, contextual, representational and accessibility. This categorisation is used in many other investigations, i.e. Lee et al. [17], Sonntag [18]. Intrinsic Information Quality dimensions focus on the quality of the information itself and to the ‘classical’ Data Quality dimensions such as accuracy, objectivity and believability. Contextual Information Quality dimensions take into account the circumstances in which the information is used, for example relevancy, timeliness and completeness. Representational Information Quality dimensions refer for example interpretability and easy of understanding, accessibility to how easy it is to access the information and how secure the access is.

Other views on Information Quality dimensions are available. For example, Information Quality dimensions can be categorised according to the sequence of usage. For information to be of good quality, it must be accessible, interpretable, relevant and integer [19]. However, since the first categorisation is widely accepted, we will use it to divide the dimensions. In Table 4.1 we en-listed all dimensions found in literature. We filtered some dimensions out in order to only keep the meaningful and proper documented dimensions. We filtered a dimension out when (a) the dimensions was only referred to in literature once (b) it was covered by another dimensions, for example reliability is covered by reputation and believability (c) the source of the dimension was too specific, for example only applicable in data warehousing.

As one can see, the meaning of the dimensions are intuitively quite clear. However, how to measure it does not follow automatically out of them. Therefore we have to look at the metrics in the following Section.

4.3 Metrics

As stated above, concrete Information Quality metrics ‘define’ the dimensions. In Table 4.2, 4.3, 4.4 and 4.5 a collection of possible metrics is summed up for all four Information Quality dimension categories. All metrics are taken from Batini et al. [13] unless otherwise cited. This overview gives some insight in how dimensions are more formally defined and what (general) measures are known now. If only a user survey, questionnaire or ‘-’ is filled in as metric, no direct calculation is available. A questionnaire or survey is also not really a metric, the questions in the survey/questionnaire are. However, no details were given regarding the content of these questionnaires and surveys.

What we can see from the tables is that many dimensions are approaches from a database perspective (as we have seen earlier). For example, a metric for completeness (number of null values / total number of values) can only be measured for structured data. Furthermore, some dimension metrics are defined quite vaguely: number of consistent values / number of total values. Although true, the most difficult problem remains how to define consistent. As last, many dimensions do not have any metric or can only be assessed by means of a user survey, which actually could be applied to every dimension.

Concluding, most of the metrics defined up until now are too vague, too database oriented or simply not investigated enough to be applied widely. In the following Section we discuss some research trying to assess the quality of unstructured data. What is roughly the case is that the type of unstructured data is narrowed down to specific data, resulting in metrics that can only be applied to that specific type of document.

4.4 Information Quality of Unstructured Data

A couple of investigations have been conducted into the assessment of unstructured data quality. We shortly describe three of them, with special focus on how the dimensions were measured and what metrics were used.

4.4.1 Quality of web data

As first example we present an investigation by Zhu and Gauch [21]. The goal of their research was to improve the retrieval of web pages, which is the technique making web search possible. This was done by adding data quality characteristics of pages into the search ranking algorithm next to term frequency. The expected result was that by adding quality characteristics in the ranking, higher quality web pages would appear higher in the result list. Of course, this expectation is only redeemed when the characteristics chosen are indeed indicators of quality. In their research, the following dimensions and metrics were used:

- Currency: the time stamp of the last modification of the document.
- Availability: number of broken links / total number of links.
- Information-to-Noise Ratio: total length of the tokens after preprocessing / size of the document.

Dimension	Metric
Accuracy	Number of correct values / number of total values
	Number of delivered accurate tuples
	User survey, questionnaire
Objectivity	User survey, questionnaire
Believability	-
Reputation	User survey, questionnaire
Consistency	Number of consistent values / number of total values
	Number of tuples violating constraints, number of coding differences
	Number of pages with style guide deviation
	User survey, questionnaire
	Semantic: Count of instances of the same elements having different values, structural: Count of instances of the same elements using different formatting [16]
Precision	-
Completeness	Number of not null values / number of total values
	Number of tuples delivered / number of tuples expected
	User survey, questionnaire
	Count of empty tags; count of incomplete values (circas); number of distinct elements [16]
	Missing elements from a recommended set of elements [16]
Reliability	-
Currency	Time in which data are stored in the system - time in which data are updated in the real world
	Time of last update
	Request time - last update
	Age + (Delivery time - Input time)
	User survey, questionnaire

Table 4.2: Intrinsic Information Quality metrics

Dimension	Metric
Relevancy	User survey, questionnaire
Value-added	-
Timeliness	$\max(0; 1 - \text{Currency} / \text{Volatility})$
	Percentage of process executions able to be performed within the required time frame
	User survey, questionnaire
Amount of data	$\min((\text{Number of data units provided} / \text{Number of data units needed}); (\text{Number of data units needed} / \text{Number of data units provided}))$ User survey, questionnaire

Table 4.3: Contextual Information Quality metrics

Dimension	Metric
Interpretability	Number of tuples with interpret data, documentation for key values User survey, questionnaire
Concise representation	Number of deep (highly hierarchical) pages User survey, questionnaire
Consistent representation	-
Understandability	-
Readability	-

Table 4.4: Representational Information Quality metrics

Dimension	Metric
Accessibility	$\max(0; 1 - (\text{Delivery time} - \text{Request time}) / (\text{Deadline time} - \text{Request time}))$ Number of broken links - Number of broken anchors User survey, questionnaire
Access security	Number of weak log-ins User survey, questionnaire

Table 4.5: Accessibility Information Quality metrics

- Authority: based on a score from 2 to 4 by Yahoo, 0 if not reviewed.
- Popularity: number of links pointing to the Web page.
- Cohesiveness: determined by how closely related the major topics in the Web page are.

The set of quality dimensions and metrics was validated by letting people review web pages whether they were appropriate for a certain search query. Then, the search queries were executed with the ranking algorithm including and excluding the extra properties. The conclusion was that the use of almost all metrics, certainly in combination with each other, resulted in a higher precision of the search algorithm.

4.4.2 Quality of requirement documents

Fabbrini et al. [22] have done multiple investigations into software requirements written in natural language. Requirements are known for their rigid format, a typical requirements could look like ‘The system must do action X with result Y’. Furthermore, in requirements engineering some heuristics exist in order to ensure the precision of the requirements. For example, the word ‘shall’ always is confusing and the sentence ‘must perform good’ is ambiguous. When ambiguous requirements are the basis for a software contract, it becomes impossible to check whether the requirements are met. Requirements should be clear, unambiguous and verifiable.

The rigid character of requirements was exploited by the authors. The requirements heuristics were translated into a Natural Language Processing system that tries to find violations of the heuristics. Furthermore, the heuristics

were coupled to four Information Quality dimensions, namely understandability, consistency, completeness and correctness. Understandability was subsequently defined by readability, uniguity and testability. Below we copied some examples of rules with their related Information Quality dimension.

- Testability: Use of vague words, for example “The C code must be *clearly* commented”. Clearly commented is not measurable. To detect this, a lexicon of vague words was created.
- Unambiguous: Sentences including ‘and’ and an adjective, for example “a young man and woman”. In this case it is not clear whether only the man is young.
- Correctness: Use of universally quantified statements, for example “Every person must have a social security number”, which in an immigration software system clearly is not the case. This kind of generalisations are very likely to be violated in the real world.
- Completeness: Use of underspecified words, such as “the system must works also during an *attack*”. An attack in this case can be a physical and digital attack, which are very different in nature. However, the clear distinction is not made.
- Readability: A readability index: $WS + 9 * SW$, where WS is the average number of words in a sentence and SW the average amount of tokens in a word.

4.4.3 Quality of Wikipedia pages

The last example was taken from a case study by Stvilia et al. [16]. Their subject in this case was Wikipedia. Since everyone can edit Wikipedia pages, Information Quality is easily endangered. So, next to peer reviews in Wikipedia itself, one may want to measure the quality of Wikipedia pages in another manner. To do so, meta data was used as metrics, for example the number of anonymous edits and the number of reverts. Also the data itself was used, for example the number of internal links and the number of images.

As in the previous example, the metrics were linked to Information Quality dimensions. To validate whether the Information Quality metrics were adequate, the function featured article (FA) of Wikipedia was used. Wikipedia users can vote that a page becomes a FA in order to make it appear on the frontpage of Wikipedia. It is reasonable to believe that FA have a higher quality than random articles (RA) since the system is ‘democratic’, whomever gets the most votes wins. In the validation, it was shown that using their Information Quality metrics they could classify Wikipedia pages into the category FA with a precision and recall of respectively 90 and 92%, for RA the numbers were 98 and 97%. In other words, just as in the first example, the indicators were usable to distinguish high quality from lower quality.

4.5 Summary

In this chapter we investigated Information Quality problems, dimensions and metrics, as well as some use cases regarding the quality of free text. The prob-

lems we discussed vary from very specific to very general. Ge and Helfert [15] mentions problems like spelling errors while Stvilia et al. [16] only names four high level problems. It is questionable whether the problems are useful in our research since many problems are very general (i.e. mapping problems) or relate directly to Information Quality dimensions (hard to understand relates directly to understandability). It may be in general the question whether we should compose a set of Information Quality problems rather than just relate dimensions directly to the problems we face in real applications.

From all the dimensions we have found, we selected all that were mentioned in literature at least twice. In the rest of the research, if we talk about dimensions, we only refer to the set of dimensions in Table 4.1. This selection was made because many dimensions were ‘made up’, but not all of them seemed to have proof from practice that validates their existence. Regarding metrics, we have shown an overview of existing metrics. However, this was done solely to show their existence. We do not think that they have any use for us, since most are strongly related to relational databases, thus structured data.

The examples that covered free text showed that some creativity must be used in order to assess it. Three categories of metrics can be distinguished:

1. Meta data metrics: using meta data, such as last update date or author, as indicator for quality.
2. External metrics: indicators that reside outside the content, for example the number of incoming links or appreciation rate of users captured using questionnaires.
3. Internal metrics: indicators in the text itself, such as the number of images.

Furthermore, the validation of metrics was done simply using a classification problem, and metrics were labelled useful whenever precision and recall of the classification was improved by them. Although no generic rules were extracted from the investigation, it gives some hints on how to solve our Information Quality problems.

Chapter 5

Quality of Electronic Health Records

In the previous chapter we have seen that measuring Information Quality is not a clear case. Although quality dimensions exist, the only way of measuring dimensions is by using indirect metrics. The quality of unstructured information, thus free text, is even more difficult and less investigated. In this chapter we analyse the Information Quality of EHRs by investigating the Information Quality problems that occur, based on the goals and problems indicated in literature regarding EHRs. Secondly we relate the Information Quality problems to Information Quality dimensions. Since few research initiatives relate Information Quality problems with Information Quality dimensions, we build our own reasoning regarding the relation between Information Quality problems and Information Quality dimensions. Thirdly we introduce the metrics that we will use to measure the Information Quality of EHRs. Of course, we base our metrics on the dimensions we found relevant to EHRs.

5.1 EHR Problems

As stated in the previous chapter, it is not always possible to distinguish clearly between Information Quality problems and Information Quality dimensions. Therefore it is not possible to map the problems related to the Information Quality of EHRs to Information Quality dimensions easily. Although we are not sure how to make a distinction between Information Quality problems and Information Quality dimensions, we still investigate the problems related to the Information Quality of EHRs. We do this based on our analysis on the current trends of the exchange of EHRs (see Chapter 2) and initiatives such as Routine Outcome Monitoring and the EPD-scan (Chapter 1). If appropriate we relate the EHR problems to the Information Quality problems identified before, but for the research the investigation of real practical problems is more interesting then relating them back to theory.

Although we do not reason with the Information Quality problems as basis, we can start by identifying some Information Quality problems we exclude from this research:

- Transmission, network, databases and accessibility. We assume the physical EHR architecture provides a secure, reliable and properly accessible manner for healthcare practitioners to interact with EHRs. Although all these problems are realistic in the exchange of EHRs, measuring these problems is not possible by investigating the EHRs itself. Rather, the whole architecture should be the scope of investigation.
- Problems such as credibility, believability, intentionality [14] and the containment of an impartial view [15] are not considered. Again, in the context of exchanging and measuring EHRs all those dimensions are important. However, we assume every healthcare practitioner to be upright and we see no reason for any healthcare practitioner to be on purpose not credible or believable. Furthermore, we think it is infeasible to determine the intentions of the experts using IE.

We do include the problems identified from assessing the practical situations of the EPD, based on the previous mentioned trends. To start with the exchange of EHRs, the practical situation occurs that healthcare practitioner A has to look into the EHR of a patient created by healthcare practitioner B, using the infrastructure of the EHR. This situation occurs in a wide range of settings: from sitting behind a desk when seeing the patient to an emergency situation. In both cases the healthcare practitioners must be able to use the EHR: (s)he must understand the content and must be able to understand it within a certain time frame, which is especially tight in emergency situations. Furthermore, the information presented to the healthcare provider in an EHR is used to base decisions upon, which could be a treatment plan, emergency surgery or anything in between. Problems can occur whenever the EPD is too long, written in unclear language, does not contain all information and so on. If we refer back to the identified Information Quality problems, the following problems can occur:

- Information that is *hard to understand*.
- Information that has a bad *presentation*.
- Information that contains *irrelevant* pieces.
- Information that contains *inconsistent* pieces.
- *Incompleteness* of information.

With regard to standardisation and analysis, in analysis of information the saying is “garbage in, garbage out”¹. Thus, if we want to gain knowledge from analysis of information, the ‘raw’ information should be of good quality. On the one hand this means that the information must be complete and correct. False or missing information pollutes the analysis result. On the other hand, it means that the information must be gathered before it can be analysed. Since we are talking about automated analysis, also the gathering of the input information is ideally fully automated. Whenever the text is not well structured it becomes harder to automatically analyse. Problems that can occur in the gathering and analysis of information:

¹http://en.wikipedia.org/wiki/Garbage_In,_Garbage_Out

- Information that is *hard to aggregate*.
- Information that is *hard to manipulate*.

Last, we found the problem of information *not being based on facts*. In this case we mean the absence of facts, since we assume that the healthcare provider is believable. This problem can be seen as a subset of incomplete information, but we mention it here explicitly since facts are most important in analysis of information. Furthermore, this is also a problem in the context of information exchange, since it becomes impossible to follow the line of reasoning if information is missing.

In short, the main problems are summarised in the lists above. These problems relate to the exchange and analysis context we described earlier. It is thus possible that not all problems are mentioned since they are not relevant to these contexts. However, since exchanging and analysing EHRs are currently important trends in the healthcare sector, we think these problems identified are sufficient.

5.2 EHR Dimensions

The problems in the previous section were mostly defined by ourselves, rather than extracted from theory. Furthermore, the theory does not seem to give us much help linking Information Quality problems to Information Quality dimensions, since problems were investigated separately from dimensions. Therefore, for all problems we identified in the previous section, we discuss which Information Quality dimensions are related to these problem.

Three problem areas were identified regarding the exchange of EHRs: understandable, timely understandable and completeness. The completeness relates directly to the (contextual) completeness dimension. Completeness simply implies that every (medical) fact that is known, *or should be known* by the healthcare provider, is present in the EHR. Understandable and timely understandable are related, but not the same. We can define understandable in terms of the following dimensions: consistent, interpretability, consistent representation and readability. Timely understandable can be defined by the amount of data, concise representation, relevancy, redundancy and understandability itself. The timely understandability is endangered when the content is hard to understand, but also when the information is long texts, a not concise representation and much irrelevant information is presented, independently of the information it's understandability. Based on the dimensions we have identified in theory, we created a hierarchy of dimensions which jointly define the dimension 'usability in an exchange context'. This hierarchy will be shown at the end of this chapter.

If we look at the problems regarding to information analysis, the same dimensions that affect understandability are relevant: interpretability, consistent representation and readability affect the way a human can read (and understand) information, but also how well an automated process can do. Since in IE we base our analysis on common constructs in the text (finding patterns), consistency in representation will improve interpretability and readability. Furthermore consistent, relevant, correct and on facts based information directly relates to the dimensions consistency, relevancy and correctness. What we see

is that for both exchanging EHRs and analysing EHRs, understandability and its sub-dimensions are important.

5.3 EHR Metrics

We have seen that a direct mapping between Information Quality problems and Information Quality dimension was not possible, or at least was not worked out in literature. The same goes for the mapping of Information Quality dimensions onto Information Quality metrics. Although for some Information Quality dimensions metrics have been proposed, most of them were related to traditional databases. Thus, to create Information Quality metrics for EHRs we must also use specific domain knowledge. In this section we propose three possible metrics.

The metrics defined are internal indicators. External indicators, such as popularity or incoming links, are not available for EHRs currently. The meta data in the EHRs are normal key value pairs, for example dates or ICPC codes. Medical specific meta data, such as ICPC codes, are filled in by the healthcare provider, while other information such as entering date, are generated by the system automatically. We consider this information meta data since it is primarily used to bring structure, for example ICPC codes are used to group contacts under an episode. Since we assume that no one deliberately enters wrong values, and the underlying system guarantees the correct values are stored in the database, we assume these values to be correct. Thus, we will not investigate this information on their Information Quality. Furthermore, we do not see any opportunity to use this information as indicator of quality.

5.3.1 SOEP Test indicator

As stated in Chapter 2, the NHG has stated in its guideline for adequate record keeping that GPs should use the SOEP-methodology [3]. The SOEP-methodology is meant for helping healthcare practitioners write down their analysis in a orderly manner, and to streamline the line of reasoning. First the patients complaints and symptoms (S or subjective), than physical examinations and lab experiments (O or objective), the diagnose (E or evaluation) and the treatment plan made and the information given (P or plan). If we want to investigate the understandability of EHRs, and especially the ‘consistent representation’ of an EHR, the first metric we can introduce is a SOEP-check: is every journal-line in the EHR placed in the right journal-line type? To make our metric more precise, we formulate this metric as follows:

What percentage of the stated facts are in the correct journal-line type?

In this case, we define a ‘fact’ as a statement that can occur in a journal-line. For example, a S-line can contain a symptom, complaint or question, an O-line contains measurement values and other findings. In order to measure the SOEP-test, we thus need a model of all possible statements coupled to their respective journal-line type, and a method to match statements and determine their type.

5.3.2 Guideline compliance indicator

The second metric we propose also comes forth out of the NHG. The NHG guidelines [7] are based upon most recent research and are created for around 75 illnesses. Each guidelines has the same basic outline, which corresponds to the SOEP-methodology. This means every guideline contains rules on how to recognise (S), how to investigate (O), how to diagnose (E) and what to do (P). However, it is not necessary to strictly follow the the guidelines. For example, sometimes multiple physical examinations are mentioned in the standard, but of course only the one that is sufficient in a particular case needs to be conducted. For GPs, the guidelines are a convenient reference in the diagnosis and treatment of deceases.

With the information in the guidelines, we can create two metrics for completeness and relevancy:

1. What percentage of the stated facts are prescribed in the NHG guideline?
2. What percentage of the NHG guideline can be traced back in the EHR?

The first metric measures relevancy: from all statements a GP makes, how much are (according to the NHG guideline) valuable? The NHG guideline we compare the EHR to is coupled on the ICPC code (almost) each contact has. The only problem that can occur when most of the text is not recognised as a stated fact. For example, if we only recognise two medical statements in a 100 sentences long text, the relevancy is low even if both statements can be traced to the corresponding guideline. However, the 98 lines were not recognised and thus not included in the measurement.

The second metric measures completeness: from all the steps we should take, how many have we actually conducted? Two side notes have to be made for this metric. Firstly, not every recommendation is mandatory. We can have a perfectly complete EHR, but still not include all information that is prescribed (see multiple examinations example above). Secondly, not all information has to be in one contact. For many diseases multiple contacts are needed. For example, someone's blood sugar is measured at a fixed time after a consult with a GP. Afterwards, the results of the examination are discussed. The results are thus spread over three contacts. Therefore, we should not evaluate single contacts, but all information from a whole episode.

Thirdly, one must notice in this case that we are not judging the work done by the healthcare professional. If the professional has done an excellent job treating a patient, there is no guarantee the EHR is of proper quality also, and vice versa. However, since we judge the content of an EHR based on a guideline, we must take into account that professional decisions can be made that deviate from the guidelines. Our relevancy and completeness metrics are therefore not by definition correct, they only indicates the possibility of irrelevant and incomplete information.

5.3.3 Conciseness indicator

The last metric we propose is one that measures the conciseness of a EHR. A concise representation means that, among other aspects, we have text that is relevant and without redundancy. For these dimensions, we want to measure two

indices the same way the EPD-scan did [6]: Instead of measuring a percentage or score, we produce an index that can be compared with results of other EHRs. To do so, we propose the following to metrics:

1. The number of stated facts of type A appears in a contact with ICPC number B.
2. The number of stated facts of type A with the same meaning in one journal line.

Both metrics measure how many statements (symptoms, complaints, diagnosis etc.) are made in a contact. The outcome of the first metric can be compared with outcomes from other EHRs. If for a simple disease a healthcare providers needs many more statements than others, one can assume it is not written very concisely. Since we compare by ICPC, one expects to see similar outcomes, just as is the case in the EPD-scan. The second metric is more clear, whenever we find the same type of statement with the same finding, we have redundancy. With the same finding we don't mean the exact same word, but a finding with a similar meaning. In order to do so, we need a ontology of medical terms in order to compare word and test them on similarity. The similar words must appear in distinct statements, since one statement can contain an enumeration of similar words. For example, the sentence 'has pain in the back, neck' would not contain redundancy while the sentence 'has pain in the back. Has pain in the neck' would.

5.4 Summary

In this chapter we have looked at the Information Quality problems, dimensions and (possible) metrics of EHRs. In Figure 5.1, both the hierarchy of dimensions as well as the link with their corresponding metrics is given. The problems were left out the image since their is no direct link between problems and dimensions. Instead, we formulated two main goals of the EHR on which the relevant quality dimensions were selected. Furthermore, we introduced some hierarchy into existing dimensions, something which barely has been done in literature. As we were writing this chapter, we found that the link between Information Quality and problem solving techniques was not explicitly mentioned anywhere. In problem solving techniques, problems are identified after which the aspects which influence the problem are identified (the dimensions). Dimensions are then related to metrics, which in their turn can exist out of multiple indicators. For Information Quality issues we think the same method should be followed, especially in order to find metrics. A predefined set of dimensions is convenient, but probably can always be extended with more domain, situation or context specific dimensions if needed, thus an attempt to enumerate all dimensions we think is possible nor useful.

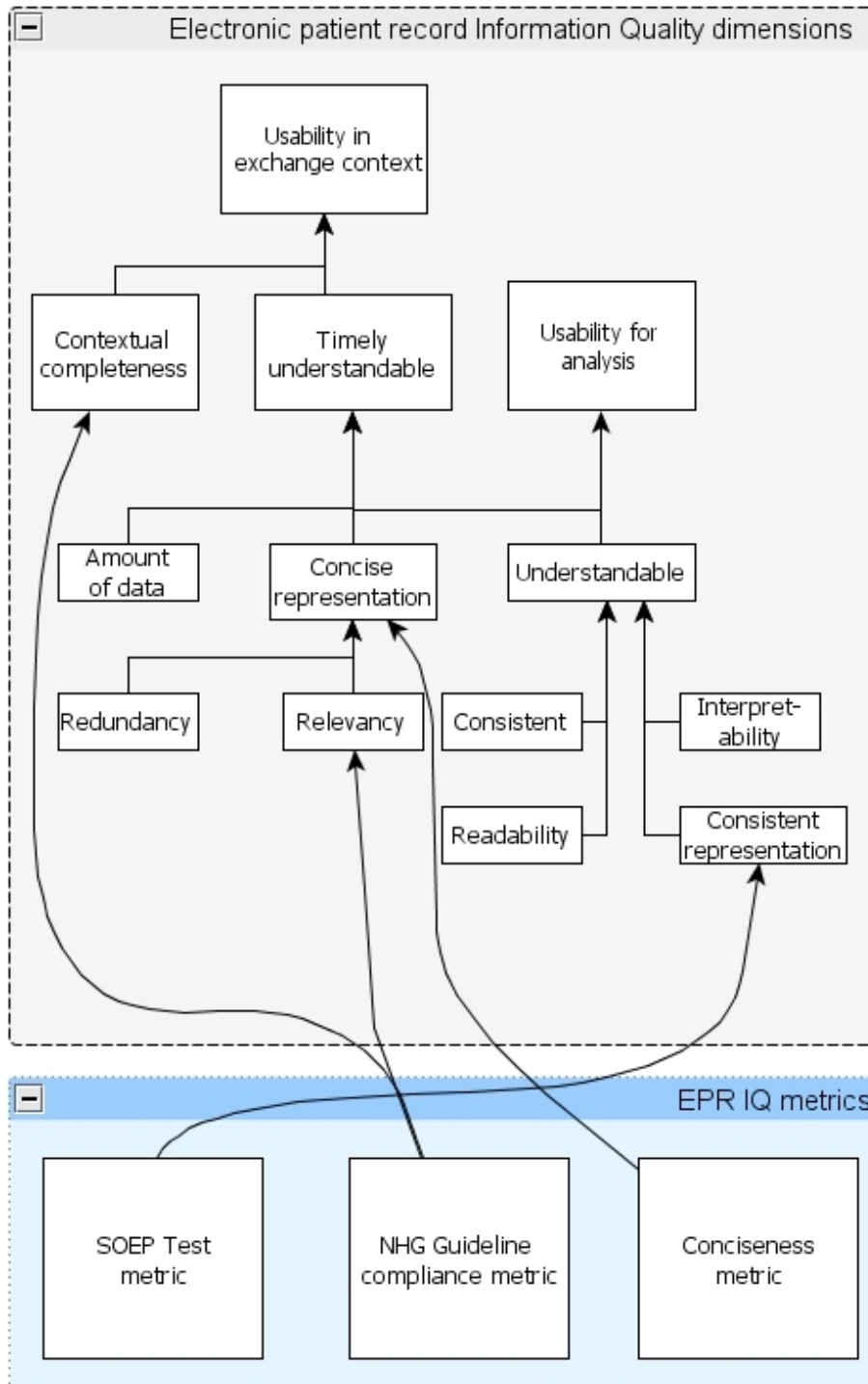


Figure 5.1: Graphical representation of link between dimensions and metrics

Part III
System Design

Chapter 6

Information Extraction Pipeline

In this chapter we elaborate on how we will apply the findings in previous parts. Since we implement an IE pipeline, a series of IE tasks, we first elaborate on the steps which we will implement. This is done in two sections. In the first, we elaborate on generic IE tasks which must be done in almost every IE application. Secondly, we elaborate on the domain specific IE task we implement. These tasks build upon the preprocessing steps and apply domain specific knowledge. Both these parts together form the pipeline, but we discuss them separately to emphasize the difference between general and domain specific steps. Since we use the GATE API some of the steps work ‘out of the box’. When this is the case, we will mention so. In Figure 6.1 the complete pipeline is presented in a diagram, and in the sections below each step is discussed.

6.1 Preprocessing

In this section the common process elements are explained. We say why we apply them, how we apply them and if necessary how we train them. In our preprocessing the following steps are carried out:

1. Tokeniser & Sentence Splitter
2. Dutch Gazetteer
3. Dutch Stemmer
4. Dutch POS-tagger
5. SVM trained POS-tagger
6. Noun Chunk Classifier

Tokeniser & Sentence Splitter The tokeniser and sentence splitter are two distinct processes, but because of their basic functionality discussed at once. The tokeniser simply annotates distinct tokens, which can be words, spaces,

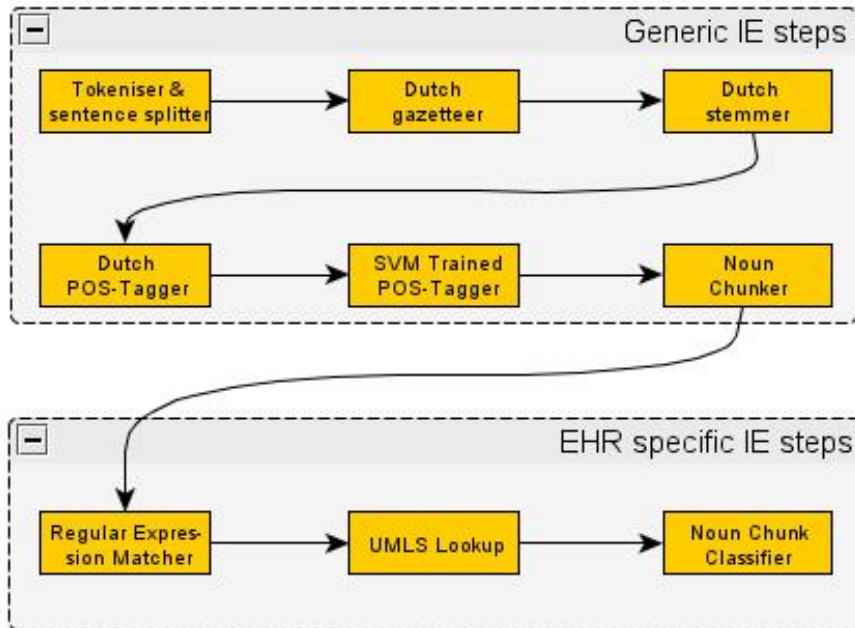


Figure 6.1: Pipeline diagram

numbers or punctuations. The sentence splitter simply annotates whole sentences. Although this task seems simple, some difficulties can be encountered. For example, a distinction has to be made between abbreviation with points and the end of sentences, indicated by a points. The tokenising and sentence splitting functionality is standard available in GATE. As a result of this process, sentences are annotated with ‘Sentence’, and all separate tokens with the annotation ‘Token’. Tokens have additional properties such as type, which holds information regarding whether a token is word, number of punctuation.

Gazetteer A gazetteer in GATE is simply a lookup of words. Using word lists, words are matched in the text and get annotated with the ‘Lookup’ annotation. Since we can match several word lists at once, we can assign a major- and minor type to them. For example, we added a gazetteer list that matches measurement units such as mg, kg, cm and so on. Each of these matched words get the major type unit, but with different minor types such as weight or length. Next to units, we created small lists in order to match months, days and several grammatical part-of-speech types such as conjunctions and determiners.

Stemmer A stemmer is an algorithm that, based on a set of rules, determines the base of words. For example, nouns can be in different plural forms or verbs in different conjugations. Since the stemmer brings all these differentiations in their basic form, two words in different forms can be recognised to have the same meaning. For example, the verb ‘to have’ can occur in I have or he has. Due to the stemmer we can more easily extract the fact that both I and he have

something. A Dutch stemmer was included in Gate, but one must keep in mind that it is far from being perfect. In our prototype, the stemmer adds an extra property to the token annotation, namely base.

POS-tagger A POS-tagger is a piece of software to annotate the part-of-speech (POS) of words; noun, verbs, adverbs and so on. In GATE an implementation of the Brill algorithm of Eric Brill is available. This algorithm uses a list of types (the lexicon) and a list of words plus their type in different situations (the context rules) to determine the POS of words in a text. Every word that is not in the context rule set is annotated with a special ‘not know’-tag. Since there is no Dutch lexicon and context rule set available in GATE, we used the lexicon and rules created by Geertzen [23], who based its set on the Eindhoven Corpus. As a result of this process, tokens in the text are expanded with a property that indicates their POS-type.

SVM POS-tagger The POS-tagger described above assumes well-written input. However, our data set contains spelling errors, unknown abbreviations and other impurities. Due to this, many words are not matched in the context rules. For this reason, we created a training set in order to create a State Vector Model (SVM), which is a method for (supervised) machine learning. Based on variables, namely the outcome of the previous steps (including the ‘normal’ POS-tagger), the SVM algorithm determines the POS-type of all words in the text. The most important variables were the Gazetteer results, stemmer results, POS-types and the word itself. The previous POS-tagger is thus used in this step to create even better POS annotations. This is reasonable: the normal POS-tagger captures all correctly spelled words, while our SVM POS-tagger relies more on surrounding words and annotations to find the POS-type of wrongly typed words.

Noun Chunker A noun chunker is a process that groups sentences called noun phrases. For example, the sentence “My eight year old neighbour has a yellow bike” would have two groups, namely “My eight year old neighbour” and “a yellow bike”. We do this since we expect that medical terms occur primarily in nouns, not in for example verbs. Many illnesses, medicine and treatments are nouns, thus filtering out the NounChunks should improve efficiency of the lookup of medical terms. NounChunks are recognised based on a rule set that takes the result of the POS-Tagger as its input. An simple example rule would be a determiner followed by a noun.

Our Noun Chunker is based on the GATE implementation. However, we did not adjust the rule set for Dutch grammar but simply used the English version. In practice this means that very large noun phrases are recognised. We did not found a Dutch grammar and we do not see the task of creating one ours. The consequence of not adjusting the Noun Chunker is that the lookup of domain specific terms (see Paragraph 6.2) will perform less efficient. The goal of chunking text is to limit the number of words that need to be looked up, thus efficiency is compromised. Since not efficiency but the concept of assessing the quality of EHRs is our goal, we do not see this as a problem. Furthermore it is possible that important domain specific words are not within a NounChunk, thus not looked up. However, we have seen that almost every sentence was

covered by one or more NounChunks, thus we do not think big problems arise from key words not captured within NounChunks.

6.2 Specific Annotation

This section elaborates on the specific additional annotations we add to the extraction process. Those additions have to do with medical terminology, in other words the extraction scenario. The following three steps will be carried out:

1. Regular Expression Matcher
2. UMLS Lookup
3. Noun Chunk Classifier

Regular Expression Matcher In GATE, JAPE is a ‘sort of’ regular expression language that can be used to match pieces of text. However, the difference is that we do not match words, but annotations and their properties in the text. For example, the sentence “10 mg” is annotated by a Token annotation that has the property Type with value ‘Number’ and is followed by the Lookup annotation (from the Gazetteer) with the majorType ‘unit’ and minorType value ‘weight’. The whole phrase is now annotated with the annotation Measurement, with the property Type is ‘weight’.

The benefit of JAPE is that we fairly easy can create a set of rules that groups together separate words. This results in higher level information on a domain level (opposed to i.e. the POS-tagger, which is domain-independent), and thus helps us understand the meaning of the sentence. We create rules to match the following ‘patterns’ in the text:

1. Addresses: postal codes, streets with house numbers, cities.
2. Dates: time and dates.
3. Numbers: simple numbers and numbers with units (measurements).

UMLS Lookup The Unified Medical Language System (UMLS [24]) is a metathesaurus that encapsulates many other medical thesauri, for example SNOMED [25]. In short, all terms in all the encapsulated thesauri are in the UMLS database. Each term is than related to a concept, which are unique. In this manner, different words in different thesauri can point to one and the same concept, making it possible to relate different terms which each other. Furthermore, each concept is categorised into a semantic type. The semantic network contains 133 semantic types and 54 relationships. The relationships are used to relate different concepts with each other.

The UMLS seems very adequate to use as lookup for medical terms due to its wide range of encapsulated thesauri (including Dutch translations of some thesauri), its unambiguous mapping to concepts and its free license (although for the included thesauri licenses are needed also).

To effectively use the UMLS we need to somehow connect GATE with the UMLS. To do so, we make use of Hitex [26] (downloaded from Openhealth-tools.org [27]) and MetaMap Transfer (MMTx) [28]. The details of these software projects will be discussed in more detail in the following Chapter, but important is that Hitex provides a manner to connect a GATE step with MetaMap. MetaMap basically is an efficient tool to search the UMLS data. We supply the NounChunks and we get all in the UMLS known words and their semantic types returned. We then save the concept and semantic type into a new annotation .

Noun Chunk Classifier The last step is the Noun Chunk classifier. For every NounChunk the semantic types of the found concepts is taken and for every NounChunk it is determined whether the combination of semantic types relate to a S, O, E or P journal line. Each journal line type is called a class in this context, hence the name classification. Thus, our Noun Chunk classifier is our SOEP-test (see Chapter 5), and therefore our EHR quality assessor. The other two indicators we defined were not implemented since implementing one indicator successfully would verify the concept of assessing the quality of EHRs using IE, and the SOEP indicator was the easiest to implement. By simply comparing the NounChunk classified journal line type with the journal line it is actually in, we can determine whether it is placed in the right place.

The problem we face is that we have to map every combination of semantic types to a journal line type. A NounChunk can hold any number of UMLS concepts, so basically we are mapping a list of semantic types to one journal line type. To do so, we first need to determine which semantic types point to which journal line type. Secondly, we need to know how to solve situations where a combination of semantic types points to multiple journal line types.

For every semantic type we determined ourselves to which journal line type it points. Not every semantic type refers to a journal line type, and sometimes semantic types refer to more than one journal line type. The consequence of this must be solved in the second step. Alternatively to our solution, a team of domain experts should map semantic types to journal line types, or a mapping could be determined based on data research. This would probably increase the accuracy of our classification. Furthermore, it is well possible that a semantic type maps to different journal line types, depending on the context in which it is used. To achieve this level of intelligence, a much more sophisticated IE process must be used and therefore we will not provide.

The second step, determining the final journal line type, is done in our prototype in the simplest way possible: to which journal line type is pointed by the UMLS concepts most is determines the final classification. However in further development, rules could be made to determine which combination of semantic types produces which journal line type. Furthermore, as stated above, one could include the context. Context means in this case other annotations in the surroundings of the NounChunk.

Chapter 7

Global Design

In this section we discuss the technical details of our prototype. As a basis we used HITEx [26], a Java based medical language NLP program. However, we stripped it down since the HITEx application was focussed on English texts and is too extensive. Thus, by keeping the architecture but loosing the details we managed to keep our prototype simple but still usable in a more complex environment if wanted.

A diagram of the architecture can be found in Figure 7.1. The architecture consists out of three components: In the main package the ‘ExtractionExecutor’ and ‘GatePipeline’ classes control the whole IE process, from data retrieval to data processing to data storage. The ‘extractionobject’ and ‘model’ packages are responsible for the actual retrieval and storage of data. As last, the ‘gate’ package holds our custom IE steps classes, which extend from GATE API classes. Furthermore, we make use of the MetaMap Transfer API [28] in order to lookup UMLS concepts.

We shall discuss the three components in the following sections in more detail. We will mention which parts were used from HITEx and which were implemented by ourselves.

7.1 Extraction Executor

The ExtractionExecutor is the ‘entry point’ of the system and has two stages: initialisation and execution. In the initialisation phase the ExtractionExecutor parses a XML-document into a Configuration object, instantiates a GatePipeline object with the Configuration object and initialises the GatePipeline object. The parsing of the XML configuration is done by the class XMLConfigurationParser. In the execution of the process, the ExtractionExecutor uses an object from the ‘extractionobject’ package to acquire the input data, feeds it into the GatePipeline and stores the result using the same object from the ‘extractionobject’ package. The classes from the ‘extractionobject’ must all implement standard functions for retrieving and storing data. The whole Configuration package is reused from HITEx, as well as the GatePipeline class. The ExtractionExecutor is instantiated with a URL to the XML-configuration file and an object from the ‘extractionobject’ package.

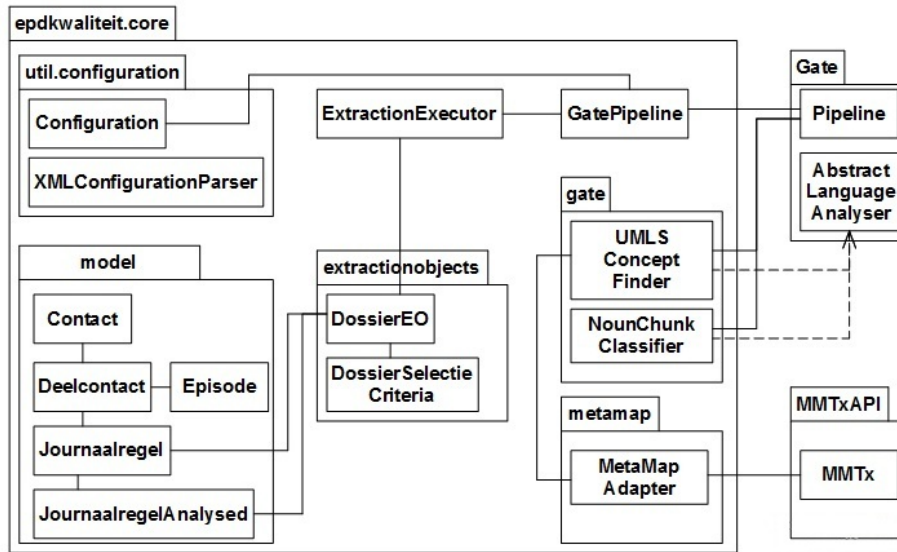


Figure 7.1: Global architecture

The GatePipeline is not an actual pipeline, but a wrapper for the Pipeline class in the GATE library. The responsibilities of the GatePipeline are to initialise the GATE API, convert input data into GATE readable format and execute the pipeline. In initialisation of GATE, all resources directories are set. Resource directories contain a XML-file that specifies where to find the IE steps we want to use, thus we need to set them in order for GATE to find them. Secondly in initialisation, we initialize the IE steps themselves. In the XML configuration the resource directories are specified, as well as the IE steps to execute along with their initialisation and run-time parameters. The parameters specify for example the name of the output annotation, i.e. that we want to annotate a NounChunk with the annotation name ‘NounChunk’.

7.2 Model

The model of our prototype consists out of two packages: the ‘model’ and ‘extractionobject’ (EO) package. The two packages together are responsible for the retrieval and storage of data. As stated above the classes in the EO package are used by the ExtractionExecutor to retrieve and store data. The classes in the ‘model’ package are used by the EO and form an Object-Relational mapping with the database (using Hibernate). We first discuss what the EO classes do, after which we explain the data model and their mapping to the model classes.

In the EO package two types of classes exist (see Figure 7.2): The ones that implement ExtractionObject and the ones that extend SelectieCriteria (SelectionCriteria in English). Classes implementing ExtractionObject need to specify the input, selection criteria and output class in order to be able to process any type of input and output data (resolving of generic types). Furthermore, the functions to retrieve the input data, store the processed data and retrieve the

processed data again must be implemented. The classes that extend `SelectieCriteria` are used by `ExtractionObject` to filter the input data. For example, we could specify to only retrieve the last 50 journal lines.

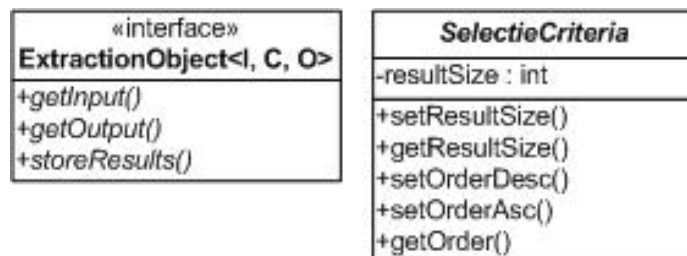


Figure 7.2: Model class diagram

The model package consist out of objects that directly relate to database tables, and the mapping between the database and objects is done using Hibernate. In our case, five classes are implemented to map with their respective database tables. The `Contact`, `Deelcontact`, `Episode` and `Journaalregel` represent the normal HIS structure in which every contact has one or more sub contacts and every sub contact has one episode and one or more journal lines. The `Journaalregel_Analysed` class is added to store the result of the IE process. This table holds an ID referring to the original journal line, thus we can always reconstruct to which sub contact, contact or episode the analysed journal line belongs.

Every class that wants to be the input class of `ExtractionObject` needs to implement `ExtractableObject`. `ExtractableObject` only enforces that a function `getInputString` is implemented. This function returns the string that must be processed, which can be another variable in every model class.

The result of the IE pipeline is not stored in a database. The result of the process is a `Document` object, which next to the text holds all annotations. GATE implemented a `SerialDataStore`, a manner to simply store `Document` objects in a serialised form on disk. Such a `SerialDataStore` is simply created by specifying a directory, after which `Document` objects can be synchronised with the data store. In the `Journaalregel_Analysed` table we thus do not store the actual result, but an identifier to retrieve a `Document` object from the data store.

As last, the `Journaalregel_Analysed` also calculates a (simple) score: number of correct classified `NounChunks` / total number of `NounChunks`. Although simple, if more metrics were implemented (for example those in Chapter 5.3), a much more complicated score could be calculated, or maybe multiple scores. The score of an EHR should be the final step in assessing EHRs, therefore we implemented this mechanism in a simple but illustrative way.

7.3 GATE components

The Gate package holds all classes that extend from `AbstractLanguageAnalyser`, a class in the GATE library. Every class that extends from `Abstract-`

LanguageAnalyser can be included in a GATE pipeline, thus be part of the IE process. In our prototype, we created two custom AbstractLanguageAnalysers: MetaMapUMLSConceptMapper and NounChunkClassifier, respectively responsible for matching NounChunks with the UMLS metathesaurus and classifying NounChunks to SOEP-lines. All the other IE steps (see previous chapter) we could directly use, since they are already available in the GATE library. Therefore, we here focus on our two custom gate classes.

7.3.1 UMLS MetaMap

As stated, the MetaMapUMLSConceptMapper is responsible for looking up UMLS concepts in NounChunks. However, it does so using the MetaMap library. As stated on their website, “MetaMap is a highly configurable program developed by Dr. Alan (Lan) Aronson at the National Library of Medicine (NLM) to map biomedical text to the UMLS Metathesaurus or, equivalently, to discover Metathesaurus concepts referred to in text” [29]. A perfect fit for our needs, and the HITEx package already had a class capable of communicating with MetaMap. However, we chose to use the Java equivalent of MetaMap, MetaMap Transfer (MMTx) [28], due to system requirements of MetaMap. However, in any other version we suggest to use the original MetaMap, since MMTx is not developed any more. For our needs, MMTx is still usable, but since the data files of the UMLS metathesaurus evolve, problems are to be expected with future use of MMTx.

In short, MMTx creates his own data and index files to quickly and efficiently search through all UMLS concepts. When MMTx gets input, it does some Natural Language Processing (NLP) of its own, of which the most important is variant generation. Of each word in the text among others synonyms, spelling variants and derivations are looked up, which are used to match UMLS concepts. All candidates are evaluated and assigned a score to, which indicates how well the term matches. As a result, MMTx return a list of possible mapping, each with a suitability score. The data and index files used by MMTx can be customised in order to improve performance, leave out unnecessary parts or apply to licences. For our prototype, we create a small subset containing only Dutch words.

For every SOEP-journaline we process, all NounChunks are passed to the MMTx API individually. This is done to lower the burden of finding UMLS matches. The HITEx package as well as MMTx itself limit their UMLS lookup to NounChunks, since most concepts in UMLS are nouns, rather than verbs or other POS-types (especially stop words). The result of MMTx is processed and the best mapping is annotated with an UMLSConcept annotation. The following properties are saved along with the annotation:

1. CUI: The unique concept ID
2. Concept: The concept string
3. Semantic type: A list of semantic types the concept refers to

7.3.2 NounChunk Classifier

The NounChunk classifier takes NounChunks and based on their UMLSConcept annotations classifies the chunk as a S,O,E or P-journal line, or as unknown if no UMLS concepts were found. This is done using the semantic types of the annotation. We determined for all semantic types to what journal line they refer. This can be any number of journal line, thus it is possible that a semantic type has no mapping, one mapping or multiple mappings. The mapping from semantic types to journal lines is done by our own reasoning. As stated in the previous chapter, a more thorough research should be done to create a well-established mapping.

In our prototype, for each NounChunk we enlist the semantic types of the concepts found in that NounChunk. Then, we retrieve for each semantic type to which journal line type it refers. For each semantic type referring to a journal line type, we give that journal line type one point divided by the total number of pointers the semantic type has. For example, if a semantic type refers to S, then S gets one point. If another semantic type refers to O and E, then both O and E get half a point. Finally, the NounChunk is classified the journal line type which has gathered the most points.

In one case the prototype that it cannot classify the NounChunk and annotates it with U for 'Unknown'. If when no semantic types with a relation to a journal line type are found. The second case is whenever there is a draw.

Part IV
Evaluation

Chapter 8

Experimental Evaluation

In this Chapter, we describe the process of evaluating our prototype. In this evaluation we want to discover how well our prototype performs, although we do not aim at a 100% accurate system. Firstly because this would be practically infeasible and secondly it goes beyond our main goal, which is to show we can assess the quality of EHRs using IE. Thus we must show that our prototype performs well enough to show automated quality control of EHRs, improved by more research, is feasible.

In Section 8.1 we present the goals of our evaluation, Section 8.2 we present the set-up of our experiments and in Section 8.3 we present the results.

8.1 Goal

Our main goal is to show that automated quality assessment of EHRs is feasible. However, for this evaluation the goal is more limited. Of the three proposed EHR quality indicators we selected the SOEP-test indicator to implement. In order for the SOEP test to work properly, we need the NounChunk Classifier to perform well. Since a journal line is graded based upon the classified NounChunks it contains, we need to determine the performance of the classification of NounChunks.

To evaluate the performance of our classification we must calculate three measures: Precision, recall and F-Measure. Precision is a measure that expresses the correctness of the classification, whereas recall expresses the completeness of the classification. Figure 8.1 depicts this difference. The oval represents the result of the classification and the dots in the left area of the oval represent the correctly classified elements, whereas those in the right area represent the incorrectly classified elements. The amount of correctly classified elements (left area in the oval) divided by the total amount of classified elements (all dots in the oval) is the measurement of precision. Recall is the amount of correctly classified elements divided by all elements that should be classified (left area in the oval and left area in the rectangle without the oval). The F-measure combines recall and precision into one measurement, using the formula $F\text{-measure} = \frac{(1+b) \cdot Precision \cdot Recall}{b \cdot (Precision + Recall)}$, where $0 < b \leq 1$ [8] (we show the calculation for precision and recall shortly). In all our evaluations, $b = 1$, which means that

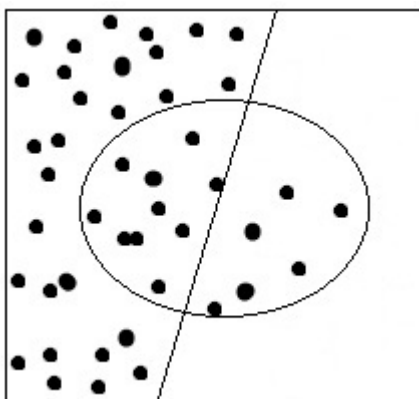


Figure 8.1: Recall and precision

whenever Precision and Recall are equal, the F-measure is equal to Precision and Recall.

In a classification context the terms true positive (tp), true negative (tn), false positive (fp) and false negative (fn) are used to name the outcome of the evaluation. If we refer back to Figure 8.1, the elements in the left oval area are true positives, the elements in the right oval area are false positives. In the areas outside the oval, the elements in the left area are false negatives (should have been classified), the elements in the right area are true negatives (are justly not classified). However, we must note that in our evaluation no true negatives will occur. Every NounChunk we cannot classify, we classify as ‘U’ for Unknown. With these terms defined, the formula to calculate the precision of a classification is as follows: $\text{Precision} = \frac{tp}{tp + fp}$. The recall is calculated as

$\text{Recall} = \frac{tp}{tp + fn}$. The following subsection will clarify this theory with some examples taken from our evaluation.

8.1.1 Example output

Figure 8.2 shows some example output of annotated documents. In the figure, the left text shows the reference set, the right set shows the text annotated by our prototype. The text between curly brackets (“soeptype=s” for example) is the property of the NounChunk annotation we compare. In the comparison between the reference and prototype set five cases can occur:

1. True positive: the five white lines are annotated correctly, thus a true positive.
2. False positive: the top three comparisons are false positives. We annotated a NounChunk falsely, it should not be annotated.
3. False positive & false negative: in a binary classification (classification with only two classes) we do or do not annotate a piece of text. However, since we annotate into more than two classes (S,O,E,P and U) more combinations are possible. In the second three lines we falsely annotated

		?-	hand.	{soeptype=p}
		?-	PRO: Weer	{soeptype=s}
		?-	keel,gb, longen,vag.	{soeptype=p}
Essentiele-hypertens...orgaanbeschadiging.	{soeptype=e}	<>	Essentiele-hypertens...orgaanbeschadiging.	{soeptype=s}
bursitis-arm.	{soeptype=e}	<>	bursitis-arm.	{soeptype=s}
bijbal-pijn-paar-uur;epidimitis.	{soeptype=e}	<>	bijbal-pijn-paar-uur;epidimitis.	{soeptype=p}
valt-al-langer-op-da...eczeem,fam:astma-	{soeptype=s}	=	valt-al-langer-op-da...eczeem,fam:astma-	{soeptype=s}
WD:AS:rugpijn,Sl.Tramadol-50-mg..	{soeptype=s}	=	WD:AS:rugpijn,Sl.Tramadol-50-mg..	{soeptype=s}
Droge-schilferige-huid-bovenooglid.	{soeptype=s}	=	Droge-schilferige-huid-bovenooglid.	{soeptype=s}
controle-eczeem,gaat-iets-beter.	{soeptype=s}	=	controle-eczeem,gaat-iets-beter.	{soeptype=s}
last-van-moeheid-aan...an-einde-van-de-dag.	{soeptype=s}	=	last-van-moeheid-aan...an-einde-van-de-dag.	{soeptype=s}
Ja..	{soeptype=s}	-?		
/kriebelhoest.	{soeptype=s}	-?		
nog-2-weken-geen-zocor;lab..	{soeptype=p}	-?		
reumafact.	{soeptype=o}	-?		

Figure 8.2: Annotation example

the NounChunk in a class, thus making it a false positive. Secondly, we also did not classify it the right class, making it a false negative. These mistakes are thus counted in two categories of errors. As we will show later, this plays a great deal in our evaluation.

4. False negative: the last four lines are falsely not annotated, thus making it a false negative. Please note that it is not a false positive since we not classified the NounChunk into any class.
5. True negative: we did not encounter any true negatives in our evaluation, which is a pure coincidence. A false positive simply means that we did not annotate a NounChunk and that doing so was correct. Please note again that this can only occur when we remove all ‘U’ NounChunks, because else every NounChunk is classified and we cannot have a true negative.

Now we labelled all comparisons as true and false positive and true and false negative we can calculate the precision, recall and F-measure, shown in the Equations 8.1 8.2 and 8.3.

$$Precision = \frac{tp}{tp + fp} = \frac{5}{5 + 6} = 0.45 \quad (8.1)$$

$$Recall = \frac{tp}{tp + fn} = \frac{5}{5 + 4} = 0.56 \quad (8.2)$$

$$F = \frac{(1 + b) * Precision * Recall}{b * (Precision + Recall)} = \frac{(1 + 1) * 0.45 * 0.56}{1 * (0.45 + 0.56)} = 0.50 \quad (8.3)$$

8.2 Experimental Set-up

In this section we discuss the set-up of our evaluation. We describe what data we use and how the experiment is carried out.

8.2.1 Dataset

Our dataset was extracted from an anonymous production database of Promedico HIS. The database was made anonymous by mixing patient, addresses and other personal information. In this manner there is no possibility that any (medical) information is traced back to the real person. From this database, we took 500 journal lines from one GP. Although we measure the performance of our NounChunk classifier, journal lines are the normal input of our prototype. Therefore we took for the evaluation whole journal lines. We took the lines from one GP since we did not want that our results were influenced by different writing styles of different GPs, because this would negatively influence the results of our prototype. Although important in a possible production version, this is beyond the scope of our research.

The dataset is used for two purposes. First of all, it is used to create a reference set. The reference set is a fully annotated set which is used to compare with the results of the system that is being evaluated. Secondly the dataset is used as input for the prototype. In this manner we can evaluate as many systems as we want since we can always compare it with the reference set.

The reference set was created as follows. Since we want to evaluate our NounChunk classifier we let our system annotate the text up until the UMLS lookup (see Section 6.2). The NounChunk classification was done manually by ourselves. Based on the text of the NounChunk we decided whether the NounChunk fitted best in a S,O,E or P journal line. If we could not decide what type of journal line the NounChunk should be in, we classified it as ‘U’ for Unknown. We want to emphasize that we did not classify the NounChunks based on the UMLS lookup data, but only upon our own intuition of in what journal line type the NounChunk should be in (see Chapter 2). We want to emphasise this because we also wrote the NounChunk classifier ourselves. If we did classify the NounChunks based upon the semantic types in the NounChunks, we could invalidate the results because of bias.

The statistics of the reference set can be found in Table 8.1. In the following subsection we discuss in more detail how we processed the dataset.

Type	Amount
journal lines	500
NounChunks	593
S NounChunks	165
O NounChunks	61
E NounChunks	81
P NounChunks	73
U NounChunks	213
Concepts found	333
NounChunks without concepts	385

Table 8.1: Reference set statistics

8.2.2 Experiment

In this subsection we describe how we executed our experiment. We did two types of experiments, one using machine learning and one using our own rule-based classifier.

Machine Learning We applied Machine Learning to our reference set in order to find patterns using a SVM trainer (similar as our POS-tagger). In GATE a SVM learning set-up was created that takes the semantic types of found UMLS concepts and tries to map it to journal line types. The evaluation is done by the k-fold evaluation method. This evaluation simply means the machine learning processor takes a portion of the whole set (in our case 75%) and trains a model based on that set. The remaining part of the set is then annotated using the learned set just created, after which the annotated set is compared with the reference set. This comparison then delivers the precision, recall and F-measure for each class apart, as well as for the whole set. This process is repeated k times, in our case 5. The repetition is done in order to make sure every piece of data is at least one time in the training part of the set.

Rule-based Our own prototype was evaluated by giving the plain text of our dataset to our prototype, resulting in an evaluation set which is fully annotated up until the Noun Chunk classifier. The evaluation set was then compared to the reference set using GATE. Since the reference set was annotated by our prototype up until the UMLS Lookup, the reference set and evaluation set are equal except for the NounChunk classification.

We compared the reference and evaluation set twice. First we simply compared the whole set including ‘U’ classification. Secondly, we removed all NounChunks in both the reference and evaluation set which were classified as ‘U’. We did this in order to remove rustle: A NounChunk classified a ‘U’ is a decision based on no knowledge, whereas a classification into a real journal line type is a decision based on knowledge. We are interested most in decisions based upon knowledge. For this reason we also did a ‘null test’. Because the high number of ‘U’ NounChunks, we asked the question what the result would be if our prototype classified every NounChunk as ‘U’.

Finally, we used three variants of our prototype. What was adjusted in the different tests was tie resolution, cases in which two or more journal line types had the same number of points. In our initial prototype simply the first in the list of ties was taken, secondly every tie resulted in a ‘U’ and thirdly the classification was random. We think tie resolution is useful since we found there were 14 ties, where ties with a maximal score of 0 (NounChunks having no mapping to any journal line type at all) were not included. Although this seems like a small number, we only found $593 - 385 = 208$ NounChunks that contain one or more concepts (see Table 8.1). Therefore $\frac{14}{208} * 100 = 6.7\%$ of all NounChunks containing a concept has a tie, thus leaving room for improvement. If we count the NounChunks that have no mapping (whether or not they contain concepts), we find that 439 of all NounChunks have no mapping to a journal line type. In other words, there are NounChunks that contained concepts, but from which no semantic type does map to any journal line type. In this case,

	TP	FP	TN	FN	Precision	Recall	F-Measure
S	14	6	0	53	0.70	0.21	0.32
O	1	0	0	21	1.00	0.05	0.10
E	5	4	0	28	0.55	0.15	0.24
P	1	1	0	25	0.50	0.04	0.07
U	0	1	0	77	0.00	0.00	0.00
Total	20	12	0	204	0.63	0.09	0.16

Table 8.2: Evaluation results SVM Machine learning

$\frac{14}{593 - 439} * 100 = 9.1\%$ of all NounChunks with a mapping is a tie.

8.3 Results

In this Section we discuss the results of both our Machine Learning and rule-based evaluation.

8.3.1 Machine Learning

The results of this evaluation are shown in in Table 8.2. What we see in the results are the averages of five runs. What one must notice immediately is the very small number of true positives. This causes both the recall to be very low (we did not find many results) and the precision to be high (when we find a result, it is likely to be a correct result). Furthermore, there are no true negatives. This is because we ask the SVM to classify every NounChunk, there are no NounChunks that are not classified. Therefore it is impossible to justly not classify a NounChunk.

8.3.2 Rule-based

The results of the three tests are in Table 8.3, 8.4 and 8.5. All three tables include the evaluation results for the test including and excluding ‘U’ NounChunks. Table 8.3 also includes the results for the null test.

In all results ‘Including U’, no true negatives can be found since it is not the task of the classifier to decide which pieces of text need to be classified. Furthermore, the number of false positives and false negatives is equal. To illustrate this the following example: If in the reference set a NounChunk is classified as ‘S’, and the classifier classifies it as ‘O’, than it is both a false negative (we say it is not a ‘S’) and a false positive (we say it is an ‘O’). If the ‘S’ part was missing, we would have a false positive only. If the ‘O’ part was not classified, we would have a false negative only. Since we by definition retrieve all NounChunks, recall is no adequate measure.

In the evaluation without ‘U’, recall is of importance. What one can see is that the number of false positives drops enormously (290 to 85 in the first test), which indicates that the prototype classified many journal lines as ‘U’ which are now removed. The same goes for the number of true positives. Apparently most of the correct classification by the prototype were ‘U’ journal lines. A correct classification as ‘U’ one must regard as a classified based upon having

no knowledge, whereas classifications into any other journal line type must be regarded as a choice based upon knowledge. Of course we want our prototype to base its decisions upon knowledge.

The null test results show that 39% of the NounChunks are classified correctly. Thus, only a precision greater than 39% would be a good result, otherwise we could better guess that all NounChunks are ‘U’. Remark that the results of the null test cannot be compared with the results of the evaluations without ‘U’. For these evaluations it is harder to decide when a result is good. Since the journal line types are not divided equally (see Table 8.1), we cannot say that it should perform better than a random classifier (which should guess about 20% right).

	Including U	Excluding U	Null test
TP	303	95	231
FP	290	85	362
TN	0	0	0
FN	290	306	362
Pr	0.51	0.53	0.39
Re	0.51	0.24	0.39
F	0.51	0.33	0.39

Table 8.3: Rule-based first run results

	Including U	Excluding U
TP	298	80
FP	295	61
TN	0	0
FN	295	300
Pr	0.50	0.57
Re	0.50	0.21
F	0.50	0.39

Table 8.4: Rule-based second run results

	Including U	Excluding U
TP	289	86
FP	304	122
TN	0	0
FN	304	294
Pr	0.49	0.41
Re	0.49	0.23
F	0.49	0.32

Table 8.5: Rule-based third run results

Chapter 9

Use Cases

In this Chapter we evaluate our other EHR IQ metrics. However, since we did not implement the other two metrics due to time restrictions, we will proof these concepts with use cases. The use cases describe a scenario how the system would behave. With some example data input we show our concepts.

9.1 Guideline compliance

Our Guideline compliance indicator compares a guideline from the NHG with an episode in the EHR. As stated in Chapter 5 each guideline has the outline corresponding to the SOEP-methodology, meaning it states how to recognise (S), how to investigate (O), how to diagnose (E) and what to do (P). However, many illnesses are treated over multiple contacts, therefore we cannot match one contact with a guideline, but need to match a whole episode with the guideline. This use case thus does exist out of a (simplified) guideline and a whole episode. With this (fictive) data we show how our metric works.

The guideline we choose was diabetes. Table 9.1 a simplified version of the guideline is presented. It describes a GP must suspect diabetes whenever someone has lost weight and is (abnormally) thirsty. To determine diabetes the GP must measure the glucose level and to determine extra risks measure the patients cholesterol. The patient has diabetes whenever the glucose levels are (sober and not sober respectively) above 6.0 or 11.0 mmol per liter. As last, diabetes should be treated using Metformine and if not proofing effective, with insulin shots.

Line	Guidelines
S	Symptoms are abnormal thirst and weight loss
O	Measure glucose level (HbA1c). Measure BMI and cholesterol for risk analyses.
E	Diabetes when glucose sober >6.0, glucose not sober >11.0 mmol/liter.
P	Start with Metformine (500mg, 1 tablet per day), if not effective use insulin shots. Advice to loose weight if BMI greater then >27.

Table 9.1: Diabetes guideline

In the EHR of a patient we come across the following three contacts (see

Table 9.2), all part of the same diabetes episode. As with our experimental evaluation we can define result categories.

1. Facts both in the NHG guideline and EHR (true positive)
2. Facts only in NHG guideline (false negative)
3. Facts only in EHR (false positive)

We can thus calculate the precision and recall. Please note these correspond to the metrics we defined in Chapter 5, but are formulated differently. The precision and recall values resulting out of this comparison evaluate the EHR quality, not the system performance as in our experimental evaluation. If we compare the guideline with the episode data we can simply count the occurrences of TP, FN and FP:

1. **S**: Both symptoms (thirst, weight loss) are mentioned (2 TP).
2. **O**: BMI and HbA1c sober measured (2 TP), cholesterol and HbA1C not sober not (2 FN), weight (1 FP).
3. **E**: Diabetes diagnosed (1 TP).
4. **P**: Insulin mentioned (1 TP), metformine not (1 FN).

We immediately see a flaw in our metric, namely that two exchangeable measurements are seen as two distinct measurements (glucose sober and not sober). Furthermore, weight (which is a measurement used for BMI calculation) is not recognised as ‘valid’ measurement. Nevertheless we can calculate the resulting precision and recall (see Equation 9.1 and 9.2).

$$Precision = \frac{tp}{tp + fp} = \frac{6}{6 + 1} = 0.86 \quad (9.1)$$

$$Recall = \frac{tp}{tp + fn} = \frac{6}{6 + 3} = 0.67 \quad (9.2)$$

9.2 Conciseness

Our conciseness indicator consists out of two indices, namely one that counts occurrences of one type of facts and one that counts redundant statements. We will show the working of this metric using three fictive S-lines from three different EHRs. This is done since the goal of this metric is to compare the results among different EHRs. Table 9.3 shows the content of the three lines. Again, the subject of the contacts is diabetes.

The first S-line clearly shows redundancy (thirsty and drinks a lot), while the others have no redundancy. The score on redundancy of the first line is thus lower than the other two lines. S-line two, in contrast to one and three, needs a lot of statements (4) to express the symptoms of the patient. This would result eventually in a lower conciseness score. Although both tests do not immediately give a final judgement regarding the quality of the EHR, in comparison to other they can. If the conciseness score of a GP is consistently higher than the average score of other GPs, the GP probably has a narrative writing style.

#	SOEP code	Content
1	S O E P	Sudden weight loss, patient did not noticed thirstiness, drinks a lot anyway. Weight: 75KG,BMI: 24 Take a glucose test, made appointment
2	S O E P	HbA1c sober: 6.1 Diabetes
3	S O E P	Persisting complaints related to diabetes condition Start using insulin shots

Table 9.2: Example contacts

#	Content
1	Patient is thirsty all the time, drinks a lot and goes to the bathroom numerously.
2	Thirsty, high blood pressure, mononeuropathie and polyurie
3	Goes to the bathroom a lot, weight loss

Table 9.3: Three example S-lines

Chapter 10

Evaluation conclusions

In the last chapter of Part IV we review the results of our benchmarking and case study evaluation. What can we draw up from our quantitative evaluation and how well do our case studies validate our EHR quality metrics?

What we can deduce from our machine learning evaluation is that there seems to be a quite accurate mapping from semantic types to journal line types, because of the relative high precision. However, due to the low number of positive examples we must not draw too strong conclusions: of all NounChunks it finds about 9% and accurately classifies 5%. The machine learning algorithm only found a few relations. In short these results prove nothing.

Our main evaluation has more promising results. We consider the results without ‘U’ our most important score since it leaves out the rustle of the ‘U’ classification and evaluates the classification based on knowledge, namely the mapping between semantic types and journal line types. The differences between different tie resolutions are almost equal, therefore we will not discuss these.

We can simply observe that the roughly 20-25% of all NounChunks are recognised (recall) with a precision of around 50%. These results are a strong indication that a more reliable SOEP-classifier is feasible because (a) the semantic type - journal line type mapping was created by a non-expert and (b) the whole IE process was not optimised.

We want to state the results of our prototype are not a proof that the SOEP-test is a sensible EHR quality metric. We have shown in Chapter 5 that the division of a contact using the SOEP-methodology is prescribed by the NHG, thus checking whether a healthcare provider does so is sensible by authority of the NHG.

The use cases we described do not proof anything. The only goal of the use cases was to point out that (a) more indicators are possible, probably much more than the three we presented and (b) the metrics we have presented do make sense. We see the results of our use case evaluation thus as an encouragement to investigate further into our metrics, as well as into new metrics.

Part V

Conclusions and Further Research

Chapter 11

Conclusions

In this chapter we answer our research question and discuss to what extent our goals were met. The main question of our research was “How to automatically assess the quality of Electronic Health Records?”. We will answer the three sub questions first in order to answer our main research question.

Firstly, we needed to define the quality of Electronic Health Records. We can conclude that not one answer can be given since quality is defined by the context in which the information is used. In the context of exchanging Electronic Health Records, a concise formulated record that can be read quickly means high quality. In the context of extracting management information from text, conciseness can conflict with completeness; due to conciseness concerns not all details are included. However, what is common is that all dimensions we defined relevant were deduced from the usability dimension (see Figure 5.1), whether being usable in a exchange or analysis context. The context of use is defined by the healthcare professionals themselves, they define how the Electronic Health Record is used. Since this context of use can change, the relevant dimensions and metrics can also.

The metrics we defined follow logically from NHG guidelines and standards. However, in the development of other metrics it must be made sure the metrics relate back to the dimensions defined for the context of use. A proper defined relation between metrics and dimensions validates the use of a metric.

Secondly, we investigated how to extract the relevant facts from Electronic Health Records. The answer is threefold. First of all, a proper preprocessing must take place in which word and sentence boundaries, tokens (words, punctuations, numbers etc.) and part-of-speech types are annotated. The preprocessing part is significant for almost every Information Extraction task. Secondly, the text needs to be mapped to a thesaurus, preferably one in which the thesaurus entries are related to semantic types. A mapping to a thesaurus gives meaning to the words and due to the semantic net we can find relations between the words. We used the UMLS, but also SNOMED could have been used and for other domains, other thesauri can be used. Thirdly, the annotated text must be used in one way or another to assess the quality, in our case a SOEP-test. The general pattern that can be extracted is that we need a generic preprocessing step, a domain specific thesaurus lookup step and a metric specific quality assessment step.

Thirdly, we needed to validate our prototype, which we have discussed in

the previous Chapter. In short we found that our Noun Chunk classifier, the Information Extraction step that classifies NounChunks into a journal line type S,O,E or P, performed reasonably well. Although the mapping between UMLS semantic types and journal line types was made by a non-expert and the whole Information Extraction process was far from optimised, we managed to achieve a precision of roughly 50% and a recall of 20-25%. Furthermore we showed in our use cases that (a) more indicators are possible, probably much more than the three we presented and (b) the metrics we have presented do make sense, and we see these results as an encouragement to investigate further into our metrics, as well as into new metrics.

In our Introduction we set two goals, namely to describe the relevant facts to extract from Electronic Health Records and to design and implement a prototype that can extract these facts and use them to measure the quality of Electronic Health Records. Although we did not precisely describe what facts to extract, we showed the most important to extract is the mapping to a thesaurus. The thesaurus mapping gives us a base to extract the knowledge in Electronic Health Records. Our second we achieved more literally, we designed and implemented a working prototype. Although not performing at its best, we proofed the concept of assessing the quality of Electronic Health Records works.

Chapter 12

Further Research

We have touched upon several subjects in this thesis and because of its explorative nature could not go into depth into every subject. Because of its explorative nature, we think this thesis can guide further research. We will discuss four further research proposals.

12.1 Prototype improvement

Our prototype can both be improved in the preprocessing and metric step. Since the preprocessing only consists out of generic steps, we only describe how to improve the Noun Chunk classifier. The biggest improvement can be made in the semantic type to journal line type mapping. Two options are possible. The first is to create a mapping by experts in the medical field. A expert mapping should be an improvement over a non-expert mapping.

Secondly, a mapping could be based upon empirical data. If we have a large enough set of annotated data, we can extract the mapping from this data. This brings other advantages as well. Currently, semantic types can map to any number of journal line types. If we have enough data we can also calculate probabilities between them. For example, if a semantic type in our prototypes now maps to two journal line types, both journal line types have a probability of 50%. However, it may well be possible that a semantic type maps in 70% of all cases to an E line, and 30% to a S line. With an extensive dataset we can calculate these probabilities.

12.2 Metric improvement

We deduced the IQ dimensions and IQ metrics from theoretical knowledge about IQ and knowledge from guidelines and protocols. However, we think there are three groups of people who could contribute better to the process of metric development:

1. End-users of EHRs.
2. Medical organisations such as the NHG.
3. Linguistic and communication experts or researchers.

The first two groups are from the medical field, but not by definition the same. It is well possible that end-users of EHRs such as nurses are not present in the NHG or alike, while people in medical organisation do not by definition still work ‘in the field’. It seems reasonable that medical organisation will organise the research needed to develop EHR quality metric, whereas end-users are involved in research, for example to find the biggest obstacles to understand EHR content in practice. The last group, linguistic and communication experts and research, we think are useful since Electronic Health Records are used to communicate information. Linguistics and communication studies both investigate how information is communicated best, thus they can help to develop metrics.

12.3 Metric implementation & Quality Scoring

We suggest the implementation of new metrics, preferably developed by the experts mentioned above. However, multiple metrics brings along the problem of giving a grade to an Electronic Health Record.

We suggest a few possibilities to expand the scoring mechanism in the case multiple metrics are implemented. First of all, we can simply combine all metrics into one score. We could average the scores of the different metrics and possibly assign weights to them. For example a final could consist out of 0.3 times the SOEP-test score and 0.7 times the NHG guidelines compliance score. However, in this manner overview is lost. We cannot see why an EHR gets a low or high score.

Better would be to average the scores into categories in order to preserve the origin of the grade. For example all metrics related to the exchange context can be aggregated into one score. In this manner a set of ‘key performance indicators’ is created, based upon the originally developed dimensions.

12.4 Medical Support System

As last suggestion we relate to more existing research into medical support systems (MSS). MSSs are meant to support healthcare providers in the healthcare process by for example giving suggestions for medication or diagnoses. Preferably it does so based upon the (unstructured) input of the healthcare provider and real-time.

For our system to be supportive it is necessary it is able to perform three tasks. First of all, it needs to point out that a quality error has been made. Secondly it needs to point where the error has been made and, most important, it must be able to tell the user what mistake has been made. In contrast to only judging about the Electronic Health Records by giving them a grade, the system helps to improve records.

One step further would be continuous quality control. In this scenario the system would continuously scan the input of the healthcare provider and alert him whenever a quality error is made. This is already possible in the form of spelling control, but goes further in this scenario. A simple example would be that the systems points out that a diagnose is made while some lab results are not written down in the EHR.

Bibliography

- [1] The University of Sheffield. Gate information extraction, Februari 2010. URL <http://gate.ac.uk/ie/>.
- [2] M. den Hollander-Gijsman and I. van Vliet. Routine outcome monitoring, 2008. URL http://www.nedkad.nl/docs/invitational_conference/v_Vliet_den_Hollander_ROM.ppt.
- [3] Nederlands Huisartsen Genootschap. Richtlijn adequate dossiervorming met het elektronisch patintdossier, 2010. URL <http://nhg.artsennet.nl/web/file?uuid=b9e61fa2-555d-44a7-9064-2687aa8ed900&owner=5a73f5c4-7ec8-4c34-8c93-851837e4b783>.
- [4] Informatiepunt BSN in de zorg en landelijk EPD. Informatiepunt bsn in de zorg en landelijk epd - wat het epd doet, 2010. URL http://www.infobsnzorg.nl/informatiepunt_com/patient_wat_het_epd_doet.php.
- [5] Enne Bouma. Categorijspreekuur, 2010. URL <http://www.dokterbouma.nl/Praktijkondersteuning/COPDprotocolframes/Categorijspreekuur.htm>.
- [6] L. Jabaaij, K. Njoo, S. Visscher, H. van den Hoog, W. Tiersma, H. Levelink, and R. Verheij. Verbeter uw verslaglegging, gebruik de epd-scan-h. *Huisarts & Wetenschap*, 52(5):240-246, 2009.
- [7] Nederlands Huisartsen Genootschap. Nhg-standaarden, 2010. URL http://nhg.artsennet.nl/kenniscentrum/k_richtlijnen/k_nhgstandaarden.htm.
- [8] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Computing Survey*, 38(2):4, 2006. ISSN 0360-0300. doi: <http://doi.acm.org/10.1145/1132956.1132957>.
- [9] M-F. Moens. *Information Extraction: Algorithms and Prospects in a Retrieval Context*, volume 21 of *The Information Retrieval Series*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 978-1-4020-4987-3.
- [10] M.B. Parker, V. Moleshe, R. De la Harpe, and G.B. Wills. An evaluation of information quality frameworks for the world wide web. In *8th Annual Conference on WWW Applications*, 2006. URL <http://eprints.ecs.soton.ac.uk/12908/>.

- [11] D.M. Strong, Y.W. Lee, and R.Y. Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997. ISSN 0001-0782. doi: <http://doi.acm.org/10.1145/253769.253804>.
- [12] A. Aamodt and M. Nygrd. Different roles and mutual dependencies of data, information, and knowledge – an ai perspective on their integration. *Data & Knowledge Engineering*, 16(3):191–222, 1995. ISSN 0169-023X. URL <http://www.sciencedirect.com/science/article/B6TYX-3Y5FPYP-1/2/20809c87dec08b33d661b97a8f420f24>.
- [13] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, 41(3):1–52, 2009. ISSN 0360-0300. doi: <http://doi.acm.org/10.1145/1541880.1541883>.
- [14] Z.J. Gackowski. Redefining information quality and its measuring: The operations management approach. In *Proceedings of the 2006 International Conference on Information Quality*, 2006.
- [15] M. Ge and M. Helfert. A review of information quality research - develop a research agenda. In *Proceedings of the 2007 International Conference on Information Quality*, 2007.
- [16] B. Stvilia, L. Gasser, M.B. Twidale, and L.C. Smith. A framework for information quality assessment. *J. Am. Soc. Inf. Sci. Technol.*, 58(12):1720–1733, 2007. ISSN 1532-2882. doi: <http://dx.doi.org/10.1002/asi.v58.12>.
- [17] Y.W. Lee, D.M. Strong, B.K. Kahn, and R.Y. Wang. Aimq: a methodology for information quality assessment. *Information & Management*, 40(2):133–146, 2002.
- [18] Daniel Sonntag. Assessing the quality of natural language text data. In *In: Proceedings of GI Jahrestagung*, 2004.
- [19] M. Bovee, R.P. Srivastava, and B. Mak. A conceptual framework and belief-function approach to assessing overall information quality. *International Journal of Intelligent Systems*, 18(1):51–74, 2003.
- [20] L.L. Pipino, Y.W. Lee, and R.Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002. doi: <http://doi.acm.org/10.1145/505248.506010>.
- [21] X. Zhu and S. Gauch. Incorporating quality metrics in centralized/distributed information retrieval on the world wide web. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 288–295, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3. doi: <http://doi.acm.org/10.1145/345508.345602>.
- [22] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. An automatic quality evaluation for natural language requirements. 2001. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.7525>.

-
- [23] J. Geertzen. Jeroen geertzen :: software & demos : Brill-nl, June 2010. URL http://cosmion.net/jeroen/software/brill_pos/index.php.
 - [24] U.S. National Library of Medicine. Unified medical language system (umls) - home, June 2010. URL <http://www.nlm.nih.gov/research/umls/>.
 - [25] IHTSDO. Ihtsdo: International health terminology standards development organisation, June 2010. URL <http://www.ihtsdo.org/>.
 - [26] S. Goryachev. Hitex manual, June 2010. URL https://www.i2b2.org/software/projects/hitex/hitex_manual.html.
 - [27] (Open Health Tools). hitex: Home, June 2010. URL <https://hitex.projects.openhealthtools.org/>.
 - [28] U.S. National Library of Medicine. Metamap transfer (mmtx), June 2010. URL <http://ii-public.nlm.nih.gov/MMTx/>.
 - [29] U.S. National Library of Medicine. Metamap, July 2010. URL <http://mmtx.nlm.nih.gov/>.