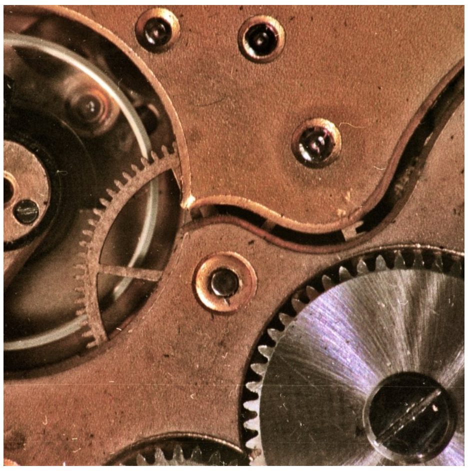


Computer, how long have I got left?

Merijn Beeksma

Computer, how long have I got left?

Predicting life expectancy with a long short-term memory to aid in early identification of the palliative phase

A thesis submitted in fulfillment of the requirements for the degree of Master.

September 11, 2017

Research Master Language and Communication
Faculty of Arts, Radboud University Nijmegen

Author
Merijn Beeksma, s4166809

Supervisor
Dr. S. Verberne

Second assessor
Prof. Dr. A.P.J. van den Bosch

ACKNOWLEDGMENTS

The interaction between experts from a broad range of backgrounds, including natural language processing, machine learning, and palliative care, provided me with an environment that functioned as an incubator for the ideas which lay at the foundation of this exploratory research. The highly interdisciplinary character of this research has truly led to a sum of parts which is greater than the whole.

I am grateful for my supervisor Suzan Verberne, whose advise, ideas and critical evaluation were around-the-clock available, and never more than an hour-wait for an email away. Your experience with the inevitable complications that are associated with this kind of research (noisy, missing, and inexplicable data, anyone?) ensured that we were never stuck on problems for long. Your practical take on complex issues provided me with the confidence that any bumps in the road could be overcome.

The help from Stef Groenewoud (Radboudumc), and André van der Veen (De Praktijk Index) have been of immeasurable value as well. Brainstorm sessions that continued for hours — and were still too short without exception! — were an incredible source of inspiration and motivation. While keeping each other's wild plans in line at best, translating ideas and insights from domain to domain was sometimes challenging, usually productive, and always exciting.

Indispensable has been the help of my dad, who contributed greatly to the process on many fronts, but most of all as my number #1 sparring partner. Many ideas have unfolded (and collapsed again) during our conversations, while possibly even more cups of coffee were being downed. Truly invaluable.

Essential has been the participation of the general practitioners of health consortium Nijmegen and their patients. It goes without saying that this research could not have existed without their contribution.

Finally, I would like to thank Paul for investing hours of time in proof reading, mental support, and allowing me to rattle in despair without judgment whenever I could not find bugs in the code or fix horrible lay-out issues.

Your help has advanced this work in many ways.

ABSTRACT

Life expectancy is a leading indicator when making decisions about end-of-life care, but good prognostication is notoriously challenging. Being overly optimistic about life expectancy, as doctors tend to be, greatly impedes the early identification of palliative patients and thereby delays appropriate care in the final phase of life. This research aimed to explore the feasibility of automatically predicting life expectancy based on electronic medical records, with the aid of machine learning and natural language processing techniques.

We trained a neural network (long short-term memory) with 1107 medical records, and validated the model with 127 medical records. Using identical evaluation criteria as were used to evaluate doctors' performance, our baseline model reached a level of accuracy similar to human accuracy. The inclusion of clinical narrative was enabled and optimized with the use of natural language processing techniques such as domain-specific spelling correction. The inclusion of keyword features improved the prediction accuracy with 9%, compared to both our baseline model and to the golden standard of human evaluation. Overall, we have shown that our approach for automatic prognostication is feasible and delivers promising results.

Her daughter and husband want euthanasia. They are angry that it is not possible anymore, because madam is no longer communicative. Nevertheless, her daughter is trying to force a reaction from her: "mom, do you want to die, do you want to go to heaven?" Madam is shaking her head, and says "no" eventually.

- Fragment from a doctor's note

TABLE OF CONTENTS

1 Introduction

1. INTRODUCTION

Electronic medical records (EMRs) contain a wealth of information. In addition to diagnoses, medication, lab test results and other structured information relating to medical care, EMRs contain large volumes of clinical narrative. Through textual data, physicians are able to capture information that would get lost if only predefined codes and descriptions would be available. Nuances, characterizations, specific details or rather the big picture, suspicions, and other salient observations that may or may not be obviously linked to the reason for encounter are described in clinical reports by the physician for each consultation. Although these texts offer an invaluable source of longitudinal, patient-specific information, they are an underused source in clinical decision support systems (Jensen et al. 2012).

This research aimed to explore the potential of systematic and automatic use of textual data to aid in Advance Care Planning (ACP). ACP is the process during which patients make decisions about the health care they wish to receive in the future in case the patient loses the capacity of making decisions or communicating about them in the future (Brinkman-Stoppelenburg et al. 2014). These decisions are made in consultation with the patient's loved ones and health care providers, the general practitioner (GP) in particular (Singer et al. 1996). Successful ACP enhances the quality of life and death for palliative patients, by preventing excessive treatment, providing timely palliative care, recording emergency contacts, appointing care takers, arranging pain management, organizing living arrangements, and documenting preferences regarding resuscitation and euthanasia, among other aspects. To ensure that it is not too late to influence processes that take place during the end of life according to a patient's wishes and preferences, timing is crucial.

Accurate prognosis of life expectancy is essential for general practitioners (GPs) to decide when to introduce the topic of ACP, and is a key determinant in end-of-life decisions, but it is notoriously difficult (Billings & Bernacki 2014; Weeks & Cook 1998; Frankl et al. 1989). Prognostication helps to predict when certain preparations and procedures should be started, how long the patient will be cognitively capable of making decisions, and when the first acts of end-of-life care need to be initiated. A large body of research shows that doctors' estimates are accurate in roughly 20% of all cases, and that predictions provided by different health care providers do not correlate with each other, due to their highly subjective nature (White et al. 2016; Higginson & Constantini 2002; Forster & Lynn 1988; Christakis et al. 2000). Life expectancy estimations tend to be overly optimistic (White et al. 2016; Higginson & Constantini 2002; Heyse-Moore & Johnson-Bell 1987; Christakis et al. 2000), leading to postponement of ACP, or not having conversations about ACP at all (Billings & Bernacki 2014).

Postponing the conversation about ACP is detrimental because this often leads to a situation in which the patient's preferences are discussed days, hours, or even just minutes before death. Physicians often struggle to determine the right moment to express concerns about the nearing end of life however, and to introduce the concept of ACP to the patient (Groenewoud 2015). Research shows that end of life decisions are often only discussed during emergency hospitalizations with junior clinicians who were not familiar with the patient up to that point (Szmuilowicz et al. 2010). Discussing ACP during a crisis situation leads to rushed decisions, made by patients that are not thoroughly informed. In times of distress, decisions tend to regard only issues of immediate concern, such as whether or not life-sustaining procedures should be applied. Such rushed, uninformed choices often lead to a poor representation of the patient's underlying values and ideas about life, and to frustration among loved ones (Billings & Bernacki 2014).

Good prognoses and proper timing are crucial for the intake of terminally ill patients in hospices as well. In the Netherlands, patients qualify for hospice care if their life expectancy is three months or less (Ministerie van Volksgezondheid, Welzijn en Sport (Dutch ministry of public health) 2015). While experts agree that patients ideally reside in a hospice for three months prior to death, patients typically receive only a month of hospice care (Integraal Kankercentrum Nederland, (Dutch quality institute for oncological and palliative care), 2015). Moreover, roughly half of all intakes is highly urgent due to a medical crisis, and more than a quarter of all patients die within seven days after the intake (Integraal Kankercentrum Nederland; Christakis & Escarce 1996). Postponing the conversation about ACP therefore excludes patients and their loved ones from important end-of-life decisions, and often results in indignation and dissatisfaction.

Confronting the patient too early is however detrimental as well. It may lead to unrealistic expectations, and to decisions that are not supported by the patient in a later stadium because they were built on hypothetical scenarios. Discussing ACP too early without repeatedly discussing it again later on, violates the main goal of ACP: ensuring that the actions that are executed or deliberately *not* executed during the last phases of life are in accordance with the patients wishes and preferences as much as possible (Billings & Bernacki 2014). In theory therefore, ACP cannot be discussed too early, but repeated discussion afterwards is required to ensure that the patient and the GP still stand behind decisions that were made earlier, or update them as necessary. Discussing ACP early on without repetition of the discussion when it is relevant, can therefore be seen as postponing ACP as well.

To support GPs in their decision-making processes, several disease-specific and general tools exist to encourage and help with prognostication, judging the patient's functional abilities, and determining the right moment to start the conversation about ACP. Tools designed to support the GP include the KPS (Karnofsky performance scale, Karnofsky & Burchenal 1949), the SUPPORT model (Lynn et al. 1995), Spitzer Quality of Life Index (Addington-Hall et al. 1990), the Seattle Heart Failure model (Levy et al. 2006), the MELD, (model for end stage liver disease, Wiesner et al. 2003), the Surprise Question (Moss et al. 2010), the RADPAC (Radboud indicators for palliative care needs, Thoonsen et al. 2012), and the SPICT (supportive and palliative care indicators tool, Highet et al. 2013).

These tools however rely heavily on the initiative of GPs, and their interpretation and evaluation of the situation. But the initiative is not always taken, and assessing the situation thoroughly and correctly is notoriously hard and time-consuming (Billings & Bernacki 2014; Robinson et al. 2013; Sudore & Fried 2010; Gott et al. 2009; Weiner & Cole 2004). Even if the initiative is taken once, the GP should continue to monitor the situation closely, to determine whether repeated discussion with the patient is needed. The combination of existing tools and human evaluation leads to suboptimal results: generalizing tools necessarily exclude patient-specific factors, thereby decreasing personalized care, while disease-specific tools can only be used for a subpopulation of patients, which make it harder for the GP to use the right tool at the right time. Additionally, one cannot recall and consider *all* information about a patient and similar cases all the time, and examining EMRs exhaustively in preparation of each consultation would be far too labor intensive. Therefore, human assessment leads to a fragmentary representation of the patient. Moreover, Christakis and Lamont (2000) showed that less experienced doctors make more and larger errors predicting life expectancy than doctors with many years of experience, and that the better a doctor knows a patient, the more likely the doctor is to overestimate the life expectancy of the patient.

Computers offer an interesting alternative to human assessment: they suffer less from memory limitations, and do not let their personal relation to the patient cloud their judgment. Perfect recall from memory is a challenge for a GP for *any* patient, and is increasingly compromised as the number of patients increases. As the Capaciteitsorgaan (Dutch organization for health care information provision) reports, GPs have over 2,000 patients on average. Because perfect recall is unproblematic for computers, they are capable of identifying complex patterns that may not be obvious to humans.

Complementary to memorizing all relevant information, learning to generalize is an effective strategy for evaluating new cases as well. As more variables are involved in the task however, less highly similar examples are available to learn the important cues from, and thus to properly generalize from. More fundamentally however, GPs are involved in only a limited amount of end of life cases to begin with: the average GP is expected to encounter 15-20 deaths annually[1]. Such low numbers cause a rather 'flat' learning curve for doctors to learn the important cues from. Because many different patient characteristics and medical factors influence the prognosis and the right moment for ACP, obtaining a concise but complete overview of the influence of all these variables and the interaction between them is incredibly complex, and due to the small amounts of heterogeneous data to learn from, it is hard for GPs to systematically learn the detailed and complex patterns from previous cases. Computers on the other hand detect complex patterns with relative ease, and can be trained on thousands of end-of-life cases.

Machine learning methods and data and text mining techniques have grown to be increasingly popular within the medical domain, and have been applied to a broad range of tasks, including medical decision-making (Walczak 2005; Mazurowski et al. 2008), automatic disease detection (Khemphila & Boonjing 2011; Al-Shayea 2011; Hazan et al. 2012), and automatic diagnostication (Khan et al. 2001; Kordylewski et al. 2001; Thangarasu & Dominic 2014; Lipton et al. 2015, Liu et al. 2016)[17-21]. Machine learning algorithms are characterized by the ability to learn without being explicitly programmed. Building models based on known cases allows machine learning algorithms to make predictions about new cases. Surprisingly, little research has been done exploring the use of self-learning algorithms for predicting patients' life expectancy. Successful prediction of life expectancy would aid in a decreased work load, increased confidence for GPs, more consistent treatment, less unnecessary treatment, earlier anticipation on palliative needs, and most importantly, has the potential to increase the quality of life and death for palliative patients. This research aims to explore of the potential of machine learning techniques for predicting life expectancy. The first question we address, is: *To what extent are self-learning algorithms trained on medical records able to detect the approaching end of life automatically?*

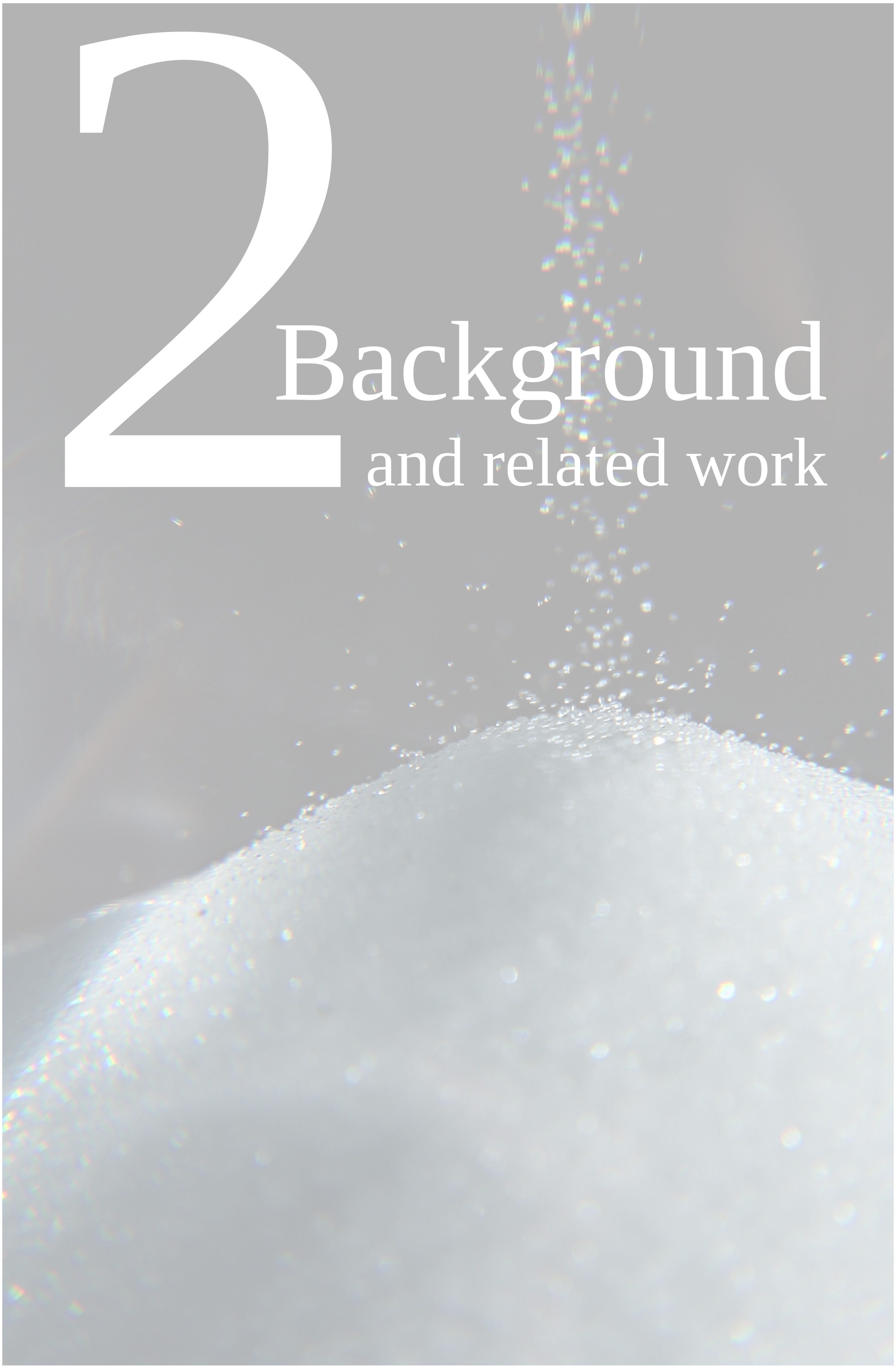
Medical records contain highly structured data, such as diagnostic codes and lab results, which are processed to train self-learning algorithms with relative ease. The situation changes drastically however, when unstructured data is presented to the computer, such as notes taken by the doctor during a consult. Textual data is incredibly noisy compared to well-structured medical data, but is available in large volumes and offers a rich source of information, complementing the structured data. It may contain important clues about the approaching end of life that cannot be

1 Based on population and mortality statistics as provided by the Centraal Bureau voor de Statistiek (national data center for statistics in the Netherlands).

inferred from the non-linguistic context. We expect that the detection and exploitation of linguistic clues enriches a model trained to make predictions about life expectancy. With this research we therefore aim to answer the following question additionally: *To what extent does the inclusion of textual data improve a prognostic model for detecting the approaching end of a patient's life?*

Chapter 2 provides background information about the standard approaches to identifying palliative patients, and discusses related work in terms of machine learning approaches in the medical domain. Chapter 3 outlines our overall method, and chapters 4-7 provide a detailed description of each research phase, ranging from initial data processing to validation on a held-out test set. In chapters 8 and 9 we discuss the results, draw conclusions about the research as a whole and offer suggestions for further research.

2 Background and related work



2. BACKGROUND AND RELATED WORK

2.1 Standard methods for identification of palliative patients

Little research has been done exploring the use of self-learning algorithms for predicting life-expectancy or the right moment for ACP, although some disease-specific work is carried out for predicting survival rates and clinical outcomes. Studies have been conducted for cardiovascular diseases (see Rumsfeld et al. 2016 for an extensive review), Alzheimer's disease (Ito et al. 2010; Zhou et al. 2012), diabetes mellitus (De Winter et al. 2006), and cancer (Burke et al. 1997; Wei et al. 2004), among other diseases. As Schwarzer and Schumacher pointed out in 2000, no real progression was made in prognosis and diagnosis up to that moment, which they mainly ascribed to methodological deficiencies. Many years later however, the techniques, availability and practical applications of self-learning algorithms have increased to a large extent in many areas, including disease-specific modeling — but not the area of predicting prognoses in general, or the right moment for ACP.

Reviews of tools for ACP as conducted by Quaseem et al. (2008), Maas et al. (2013), and Walsh et al. (2015) reveal that the tools that aid in early identification of palliative patients are not based on machine learning techniques. They do not automatically detect high risk patients, but are questionnaire-like frameworks designed to support the physician in detection of patients that require palliative care. Without exception, these tools are based on expert knowledge, but only a small number of tools is validated. Prominent tools such as the RADPAC, SPICT and NECPAL are intended for general use, and often provide disease-specific indicators additionally. The tools differ mainly in whether or not general indicators are included, which characterization of deterioration states is used, and the use of the Surprise Question. The Surprise Question is a much used method to identify palliative patients by answering the question: *Would you be surprised if the patient dies within the next few weeks / months / within a year?* Although the method is quick and easy to use, and highly intuitive (Moss et al. 2010; Moss et al 2008), it is also a very subjective measurement. Moreover, it has been shown that GPs feel uncomfortable — more than hospital doctors (Farquhar et al. 2002) — to commit to the negative answer (i.e. not being surprized) to the Surprise Question, which leads to overestimations of life expectancy (Thoonsen et al. 2011).

Virtually none of the existing tools are widely used (Maas et al. 2013). Regardless of the availability of these tools, research about their use in practice shows that GPs tend to avoid tools because they are afraid it will lead to an increase in workload, and because factors which directly or indirectly influence the palliative care process, but which are not directly of medical nature (e.g. psychological, social or spiritual factors) play only a marginal role in all these tools (Maas et al. 2013). When asked about their alternative approach for identifying palliative patients, GPs state that they rely mostly on discharge letters from the hospital, increased need for medical care, and decreased social contacts (Claessen et al. 2013). Maas et al. (2013) explored the role of the GP in the identification process through a questionnaire, which showed that identification is mainly based on subjective criteria, and therefore depends heavily on the experience of the GP with palliative patients.

There are currently no tools that make automatic predictions about life expectancy or the right moment for palliative care without relying heavily on the initiative, subjective judgment and time investment of the GP. To decrease the dependence on the initiative of the GP in considering palliative care, effort has been put into the development of computerized search algorithms which aim to identify patients with palliative needs by searching databases with EMRs globally for indicators according to existing tools such as the SPICT and the NECPAL (Mason et al. 2015; Thomas 2012). These systems are however not intended as stand-alone methods for identification of vulnerable patients, and suffer from limitations such as the inability to search systematically through textual data (Mason et al. 2015).

2.2 Deep learning approaches for clinical data

Because medical records are contained in large electronic databases, big data analyses and machine learning techniques potentially offer huge benefits for clinical practice, such as clinical decision support, modeling of clinical events and risks, and many other applications. To our knowledge, no research has been conducted for automatically predicting life expectancy regardless of the specific disease, and no research has been conducted for early detection of palliative patients. Some research has however been conducted for related subjects that are interesting for our current work because they shed light on methodological approaches to handling clinical data, and the corresponding challenges researchers face.

Deep learning and big data approaches have been applied both for general purpose applications and for disease-specific applications. General purpose applications for medical tasks often focus on automatic diagnosing of diseases. An example is DEEPR, a deep learning system which creates patient representations from EMRs and aims to predict future risks automatically, based on a convolutional neural network (Nguyen et al. 2016). DEEPR represents EMRs as a sequence of diagnostic and procedural codes for each encounter, separated by discretized time intervals to represent the sequential nature of the data. The authors address the issue of data sparseness: large collections of medical codes exist to refer to diagnoses, medication, and other types of medical information, but each patient is only associated with a small subset of such codes. Training a neural network with very sparse data is computationally expensive, increases the risk of overfitting, and requires large amounts of data. DEEPR uses word embedding techniques to represent the medical and procedural codes with dense vectors. Although this approach is able to capture long-term dependencies that characterize medical data, a disadvantage is the inability to fully capture time sensitivity: recent events should weigh more heavily than events in the distant past, but are regarded as similarly urgent by this approach, the authors explain.

Choi et al (2016) use a recurrent neural network in an application named DOCTOR AI for predicting diagnoses and medication for the next consultation for a patient, based on the patient's medical history. The authors note that the amount of data influences the results to a large extent: their model learns specific relations for *frequent* medical codes well, while it resorts to predicting the most frequent codes when given *infrequent* codes as input, because it is hard to learn about infrequent codes properly when not enough data is available. Additionally, they note that the model performs better when more medical data is available for a patient.

Disease-specific models are usually developed to predict the progression of a disease, or the survival rate associated with a combination of the disease and patient characteristics. Disease-specific approaches rely more on top-down knowledge than general purpose models, which can be explained from the observation of Choi et al.: datasets usually contain a couple of thousands of medical records, which tends to be insufficient for learning about infrequent phenomena well. When specific diseases are studied, much less cases are available, which increases the need for expert knowledge about the disease. Reliance on top-down knowledge is however insufficient for diseases that are poorly understood. Alternatively, these methods rely heavily on statistical data. Most of these specialized models focus on a limited set of possible outcomes, thereby only solving small parts of the puzzle. In reality however, patients usually show complex and heterogeneous combinations of symptoms, co-morbidity, personal characteristics and health care requests, which requires the combination and optimization of many models simultaneously, to handle the large amount of possible scenarios. Combining the output of several models to account for the interaction between variables is however not trivial, because due to the large number of possible scenarios, very specific domain knowledge or huge amounts of data are required, to properly base statistics on.

Although the approaches used for general purpose models suffer less from these issues, designing methods to handle EMR data requires solutions for several issues in general. Medical data is often very sparse and plagued by missing values. Some types of data, such as the reason for encounter, are obligatory to be recorded by the

GP during a consult, but many types of data can be provided optionally. Moreover, between consultations no data is available whatsoever. When data is missing, we cannot simply assume clinically normal values, because they depend on many factors. No contact between doctor and patient may indicate either that the patient is assumed to be healthy, or that the patient is stable, despite suffering from a disease. Which values are considered ‘normal’ is influenced by the medical history of a patient, as well as additional factors such as gender and age.

Moreover, health records contain variable amounts of data, ranging from as short as a week for new or young patients, to tens of years for older patients. To complicate the handling of time series data further, the intervals at which data is sampled are very irregular: aside from the occasional exception, patients do not visit the physician with a regular frequency, which impedes comparison of time-series data for different patients. Moreover, related medical events such as the first appearance of symptoms, consultation to assess the corresponding diagnosis, and treatment of the condition are often separated by large intervals of time. Medical episodes may span many weeks or years, especially in case of chronic diseases such as diabetes, or recurrent diseases such as cancer, requiring methods which are able to handle long distance dependencies. An additional request for handling long-term dependencies comes from regular check-ups: GPs sometimes schedule weekly or monthly consultations with a patient just to check if anything has changed regarding their condition, which is often not the case. Although these check-ups do not provide new information, they *are* documented and thereby introduce what could be considered as noise into the EMR: trivial information which is hard to separate from non-trivial data automatically.

Different approaches and algorithms have been applied to clinical time series data, including convolutional and recurrent neural networks, hidden Markov models, conditional random fields, and adopting discretely coded time gaps as features. The inability to exploit long-distance interactions and correlations however make these algorithms less suitable for learning long-distance dependencies (Dietterich 2002). Markovian methods for example are inherently unsuitable for clinical time series processing, because they have no memory. When routine admissions during which no relevant medical data is gathered interrupt an episode of a severe illness for example, the absence of recent important medical information adversely affects the model’s performance for the severe illness (Pham et al. 2017).

Hochreiter and Schmidhuber (1997) designed a type of recurrent neural network for modeling long-term dependencies: Long Short-Term Memory, or LSTM. LSTM cells are similar to the conventional, self-replicating RNN cells: at each time step, the LSTM receives new input for the current time step, and processed input from the previous time step. The memory cells of the current time step are updated, and information is passed to the next time step. The output per time step is therefore moderated by current and historical data. LSTM cells contain the addition of several gates: an input gate, an output gate, and a forget gate. The configurations of each gate are learned during the training process, and determine the information flow through the model. The forget gate instructs the cell to discard certain pieces of information it received from the previous time step. The input gate determines which pieces of information are updated as soon as new information comes in, and which are not.

The alternations of the information that result of the filters applied by the forget gate and input gate, determine the cell’s state, which can then be passed on to the next cell. Not all information is passed on to the next time step however: the output gate aims to only send through the relevant information. The addition of these gates to the cell architecture help the model to preserve relevant information over many time steps. LSTM models are therefore highly suitable for learning long-term dependencies. This research explores whether or not LSTM can be applied to automatic end of life detection with reasonable accuracy.

LSTMs have been shown to outperform other algorithms in many tasks, such as speech recognition, machine translation, signal processing, rhythm learning, handwriting recognition, weather forecasting, and other tasks concerning time series data. As a recent development, LSTMs models have received more attention in the medical domain. Lipton et al. (2016) employed an LSTM model to diagnose patients in a hospital setting based on sensor data such as blood pressure and temperature, and lab test results. Their results show that the LSTM model was able to successfully diagnose critical care patients overall, performing very good for some diagnoses such as diabetes mellitus, while other diagnoses were harder to predict. The authors characterize their approach as promising, especially considering that their approach did not make use of free-text notes and has access to much less information than doctors do when they consult with a patient.

Kim et al. (2016) used an LSTM model for conducting a similar task: the prediction of examination results given previous measurements. Their results show that their approach was feasible, and superior to other methods such as linear regression, especially for patients that display abnormal behavior from a clinical perspective. Pham et al. (2017) developed DEEPCARE, an LSTM-based approach to infer the current illness state and to predict future medical outcomes. Their model performs especially well for patients suffering from diabetes mellitus and mental health issues. They note that some natural language processing (NLP) methods, such as vector space representations, are therefore transferable to EMR processing.

LSTM models have additionally been applied specifically to aid in semantic understanding of unstructured textual data in EMRs. Research in this area mainly focuses on entity extraction and event detection. Jagannatha and Yu (2016) for instance used an LSTM to detect medical events in textual data. They showed that the model benefits from using more textual context: their document-based results outperform sentence-based results. Sadikin et al. (2016) showed that an LSTM model outperformed other models in drug name recognition in free-text data. Sahu and Anand (2017) used an LSTM to identify unknown interactions between drugs from free text. Similar to many other clinical text processing approaches, they represent the textual data with word embeddings. In contrast to Jagannatha and Yu (2016), the authors note that the model performs better for short than long sentences. In this line of research, authors without exception note that LSTM algorithms offer a feasible solution to classification and prediction tasks, while also noting that the noisy character of clinical data is challenging and adversely affects model performance. Similar to the previously discussed system DEEPR (Nguyen et al. 2016), the authors of DEEPCARE bring to light the similarity between EMR processing and NLP, comparing data such as diagnoses and interventions in EMRs to words and modifiers in natural language sentences.

3 Method

3. METHOD

3.1 General overview

In collaboration with Radboudumc, we gathered data from seven health care facilities that are part of the health care consortium of Nijmegen. The dataset contains a total of roughly 37,000 electronic medical records (EMRS). We approached the task of predicting life expectancy with the aid of an LSTM model.

This research was carried out in four consecutive phases. During Phase I, data were gathered from the EMRS and processed in order to create a structured format for each data type. Phase II prepared the feature selection process by ranking and aggregating features with a range of different approaches. In Phase III we developed and tested the LSTM model. We carried out many experiments to determine the optimal model architecture and feature set for two models: a baseline model, and a model that made use of additional keyword features. Both models were finally tested with a held-out set of validation data during Phase IV to determine the models' accuracy for unseen data. We compared the accuracy of the models to the accuracy of trained doctors as reported in literature. Figure 1 provides an overview of the processing steps.

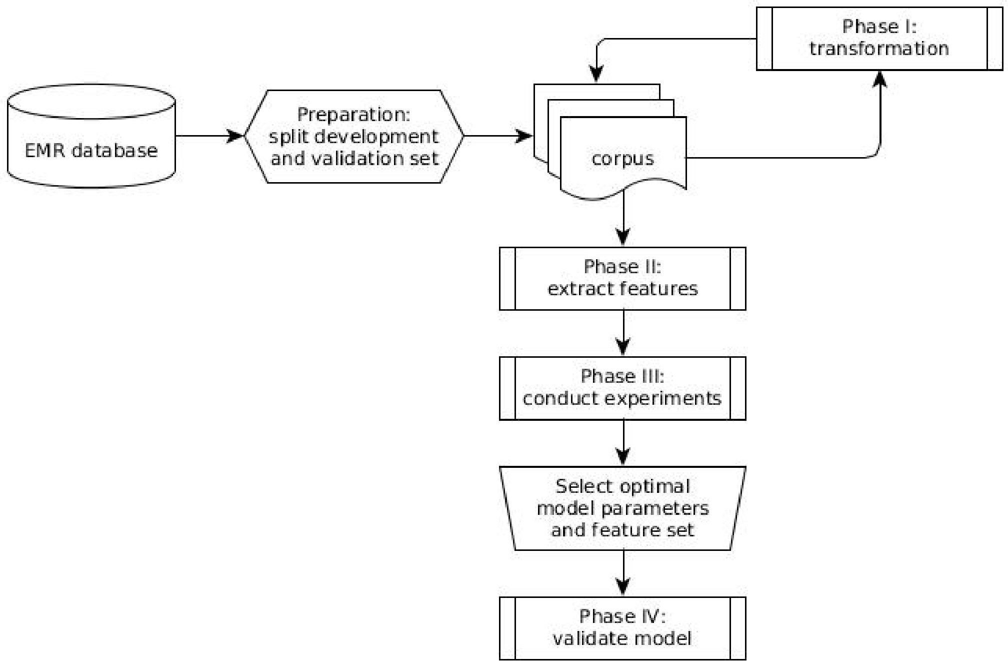


Figure 1: Project overview.

Prior to extracting data in Phase I, the corpus data were prepared to enable validation on unseen data during Phase IV. In order to enable validation while sacrificing as little of the corpus as possible, we split the data into two complementary parts: 90% of the data (1,107 patients) was used as development set for Phases I-III, and 10% of the data (127 patients[2]) was held out to use as external test set for Phase IV. We developed the model during Phase III with the use of a ten-fold cross-validation procedure to further minimize the loss of training data.

The held-out patients were chosen to be the 10% most recent patients of *each* health care facility, to avoid any bias as a result of the geographic area in which the patients lived, the health care practice they visited, and the general practitioner by whom they were treated. Additionally, the most *recent* patients were selected to simulate potential use of the model in practice — the model was trained on 'historical' data, and applied to 'new' patients.

3.2 Corpus description

The EMR corpus is used as input for the model to learn which features of the data are important indicators for estimating life expectancy. For training and evaluation purposes, the model requires a known date of death. Therefore, only the medical records from deceased patients were included, leading to a total 1,231 medical records (3.3% of the total number of patients). To avoid overfitting on the held-out test set, the test set is excluded from the descriptive statistics in this section — the statistics are based on the 90% development data only.

The development data consists of 578 (52%) records of female patients, and 530 (48%) records of male patients. The medical records span the five final years of life for each patient. The average age at the moment of death was 77; 80 for women and 75 for men. These averages correspond to the national averages as reported by the Centraal Bureau voor de Statistiek (national data center for statistics in the Netherlands).

3.2.1 Types of data

The EMRs contain both structured and unstructured data. Much of the information in the medical records is highly structured due to the use of standardized medical codes: ICD-10 codes (10th edition of the International Statistical Classification of Diseases and Related Health Problems) and ICPC-1 codes (1st edition of the International Classification of Primary Care). ICD-10 and ICPC codes are used to document multiple types of medical information during a consult, such as the reason for encounter and the diagnosis. The documentation for lab tests and the resulting

2 This is not exactly 10% due to rounding upward for each health care facility.

lab values follow predefined formats and are therefore structured as well. Finally, the textual data in the corpus used to describe the type of consult and the name of the medication are well-structured, due to the interface used for documentation (i.e. drop-down menu's). Table 1 provides an overview of all types of information that are documented during each medical consultation, with examples that illustrate the nature of the documented information. All feature types but the letters and notes are part of the structured data.

Table 1: Overview of documented information per medical consultation, illustrated with an example taken from the EMR corpus. 'S' and 'U' refer to 'structured' and 'unstructured' data, respectively. (Translation and code descriptions provided by author.)

Feature category	Data type	Unique items	Obligatory	Example
Contact type	textual data (S)	81	yes	regular 10-minute visit
Medical history	ICPC-1 code (S)	136	no	D10 (vomiting)
Reason for encounter	ICPC-1 code (S)	862	yes	A04 (general tiredness/weakness) N17 (vertigo/dizziness)
Diagnosis	ICPC-1 code (S)	572	yes	D73 (gastro-enteritis infection)
Medical intervention	ICPC-1 code (S)	380	no	A31 (directed physical examination) D45 (directives about health and diet)
ICD-10 reference	ICD-10 code (S)	857	no	A09 (infectious gastroenteritis and colitis)
Medication	textual data (S)	1,490	no	Paracetamol 500mg
Lab test	clinical code (S)	269	no	357 (patient's weight)
Letters / notes	textual data (U)	53,423	no	*cave ileus, drinkt te weinig, vandaag voor het eerst weer gegeten.* ('bowel obstruction, does not drink enough, ate today for the first time again.')

In addition to structured information, EMRs contain letters sent between specialists about the patient, and notes taken during the consult that are usually intended for personal use by the GP only. These feature types are illustrated by the bottom two rows in Table 1. Notes and letters are free texts written in highly variable formats, compared to the structured data. These types of data consist of natural language and are characterized by large amounts of noise, due to idiosyncratic use of language, many non-standardized abbreviations, spelling errors, ungrammatical sentences, and telegram-style writing. The linguistic quality of the texts depends heavily on whether the texts are personal notes, or meant for other readers as well. On average, 121 consultations were documented per patient for the five year period, and for roughly 75% of all consultations notes or letters were written. Of all textual data, 85% of the documents are notes, and 15% are letters.

Letter correspondence mostly functions to discharge a patient, to refer a patient to a different health care provider, to inform the GP about emergency hospitalization, or to obtain a second opinion. Research has shown that discharge letters are among the most important indicators for GPs when identifying patients with palliative needs (Claessen et al. 2013; Maas et al. 2013). Specialists may or may not elaborate about the context of the medical situation, but generally follow a strict format, allowing the GP to translate the textual data directly into medical codes without the loss of information. Letters therefore tend to provide little additional information that is not present in the form of structured data in the EMR, especially when compared to consultation notes.

Notes made during or after consultations often follow the so called SOEP method. The SOEP method allows GPs to make an explicit distinction in their documentation between *subjective* statements which focus on the patient's experience, *objective* statements from the GP's perspective, the *evaluation* of the situation, and the *plan of action*. Although the SOEP method stimulates a uniform documentation style among physicians, the distinction between the elements is not always clear-cut. GPs often use the *subjective-objective* distinction to separate the patient's words from their own observations (regardless of the objectivity of the statements), and the *evaluation* and *plan of action* are not easily distinguished in some cases either, especially if no specific treatment is prescribed. Table 2 illustrates notes written according to the SOEP method with a translated example.

Clinical notes tend to contain background and contextual information that cannot be inferred from medical codes and other structured information, such as information about a patient's social life, living arrangements, daily occupations, and functional and self-care abilities, among other things. Notes provide the GP with the opportunity to document changes which result in clinically normal behavior (and which are therefore not documented otherwise), but which are atypical, unexpected or suspicious for the specific patient, given additional background and contextual information about this patient. Additionally, notes generally contain nuances about the severity of the symptoms, descriptions of visible symptoms, information about the general physical and mental status of the patient, and may be used to document concerns, differential diagnoses, and other thoughts, ideas or observations.

Table 2: Example of clinical notes documented according to the SOEP method. (Translated by the author.)

SOEP **element**	**Clinical notes**
Subjective	"The patient's symptoms show roughly twice a week, on the nights his wife goes out to her hobby clubs. Patient then starts thinking about all kinds of things that could happen, such as a heart attack, when she is not around. The walls start closing in on him, he feels pressure on his chest and starts sweating."
Objective	"Seems like a respectful relationship to me. Patient wants his wife to enjoy her nights out but has much trouble being alone. Wife continues to go out despite of that. Patient does not undertake much activities since his retirement."
Evaluation	"Anxiety disorder. Hypochondriac?"
Plan of action	"I will collect some psycho-educational materials and hand them to the patient the next time."

3.2.2 Missing values

Medical records are characterized by missing values and irregular sampling. When data is absent, we could assume that the information is irrelevant, but we cannot simply decide whether the data is missing because it is *normal* or because it is *stable*. When a patient has a chronic disease, it would be redundant to report this constantly. No documentation in this case does not mean that the patient has been cured, but only that his status has not changed. In other cases, when the patient has the flu for example, we may assume that the patient recovered and is healthy again if no further consultation with the doctor was required after the initial visit. Because these scenarios have much impact on what the 'default' values should be for a patient, we decided not to provide default values to replace missing data.

3.2.3 Illness trajectories

In order to enable detailed evaluation of the results gathered during Phase IV, the patients were divided into three categories, by operationalizing three illness trajectories: a) a trajectory that is characterized by slow and prolonged deterioration, which is mostly associated with frail elderly and dementia patients, 2) a trajectory characterized by a short period of steady decline and a clear terminal phase, typical for cancer patients, and 3) a trajectory that is characterized a long period of functional limitations with intermittent ups and downs, for which death often seems sudden or unexpected, generally associated with respiratory disease and heart failure (Murray et al. 2005; Murtagh et al. 2004; Lunney et al. 2003). The differences between the trajectories are illustrated by Figure 2.

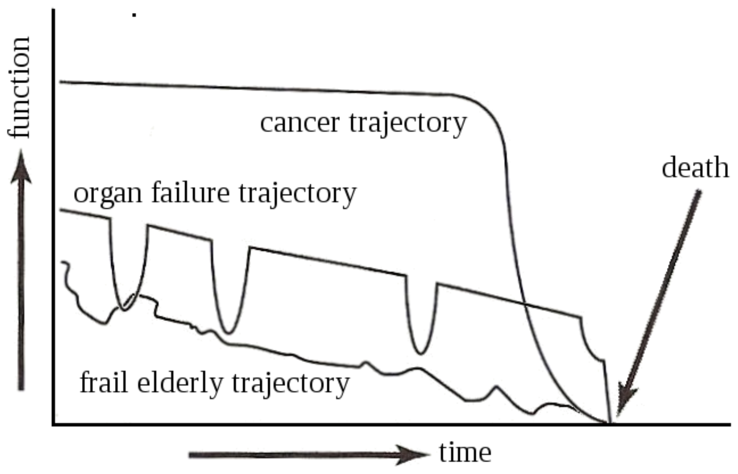


Figure 2: Trajectories of decline typical for the cancer, organ failure, and frail elderly trajectory. Figure adapted from Murray et al. (2005).

Although the existence of different illness trajectories is acknowledged in literature, clear, agreed upon criteria or definitions do not exist. In this research, patients are included in an illness trajectory category if at least one of the criteria of the trajectory applies to the patient:

- Frail elderly trajectory:
  - patient suffers from dementia or Parkinson's disease;
  - patient is 80 years old or older.
- Cancer trajectory:
  - patient has a type of malignant neoplasma.
- Organ failure trajectory[3]:
  - patient suffers from heart failure;
  - patient suffers from chronic obstructive pulmonary disease (COPD).

We do not determine the illness trajectory based on the cause of death, for two reasons: 1) in many cases the actual cause of death is not a direct result of a patient's most prominent medical condition (e.g. a patient may suffer from cancer, but die of something unrelated like pneumonia, due to overall health degradation), and

3 The organ failure trajectory may include many more diseases such as pulmonary heart disease, infection of the circulatory system, rheumatic fever, ischaemic heart disease, acute myocardial infarction, complicated hypertension, transient cerebral ischaemia, stroke, cerebrovascular disease, pulmonary embolism, and haemorrhoids. We decided however only to include heart failure and COPD in the organ failure trajectory, because 1) it increases the homogeneity of the group: the other illnesses are heterogeneous in terms of severity and impact on daily life, 2) heart failure and COPD cover a substantial part of the corpus, and 3) they lead to a less sudden death than the other conditions do, but do have a substantial impact on daily life. Therefore, ACP is more relevant for these diseases than for the other conditions.

2) because the actual cause of death is often unknown and therefore not documented. Alternatively, in order to see how the model's performance relates to different illness trajectories, patients are categorized into one or more trajectories based on the diagnoses they received during the five year period, regardless of the actual cause of death.

Figure 3 shows the distribution of patients over illness trajectories. More often than not, patients belong to multiple categories. A total of 146 patients did not fit any of the categories, and were left uncategorized. The average age for the group of patients that fall exclusively in the cancer trajectory was 64; 71 for the organ failure trajectory; and 87 or the frail elderly .

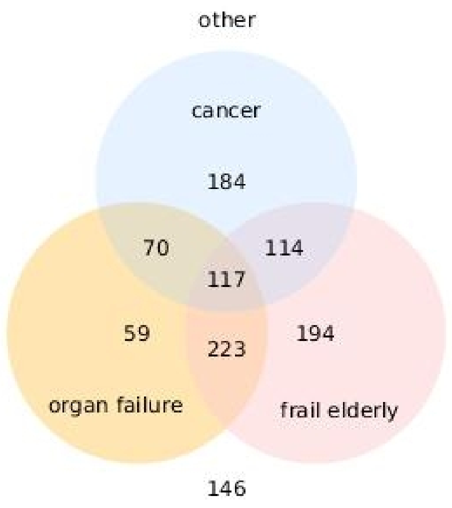


Figure 3: Distribution of patients over illness trajectories.

Figure 4 shows the relative cumulative number of patients per age at the moment of dying, for each illness trajectory and for the total set of patients. Because we aim to highlight the differences between the illness trajectories, we only included patients that are characterized by *one* illness trajectory in this figure. As an artifact of the categorization criteria, all patients aged 80 or higher are categorized as frail elderly. Therefore, no patients are added to the cancer and organ failure categories after the age of 80 in Figure 4. The data show that there is a clear difference between the different illness categories. Compared to the average, patients with cancer and organ failure tend to die at a younger age, while patients that fall into the frail elderly category tend to reach a higher age.

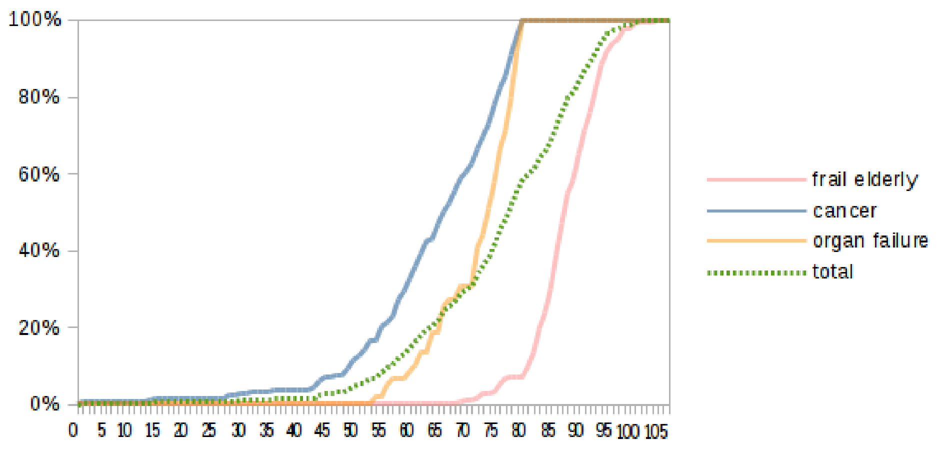
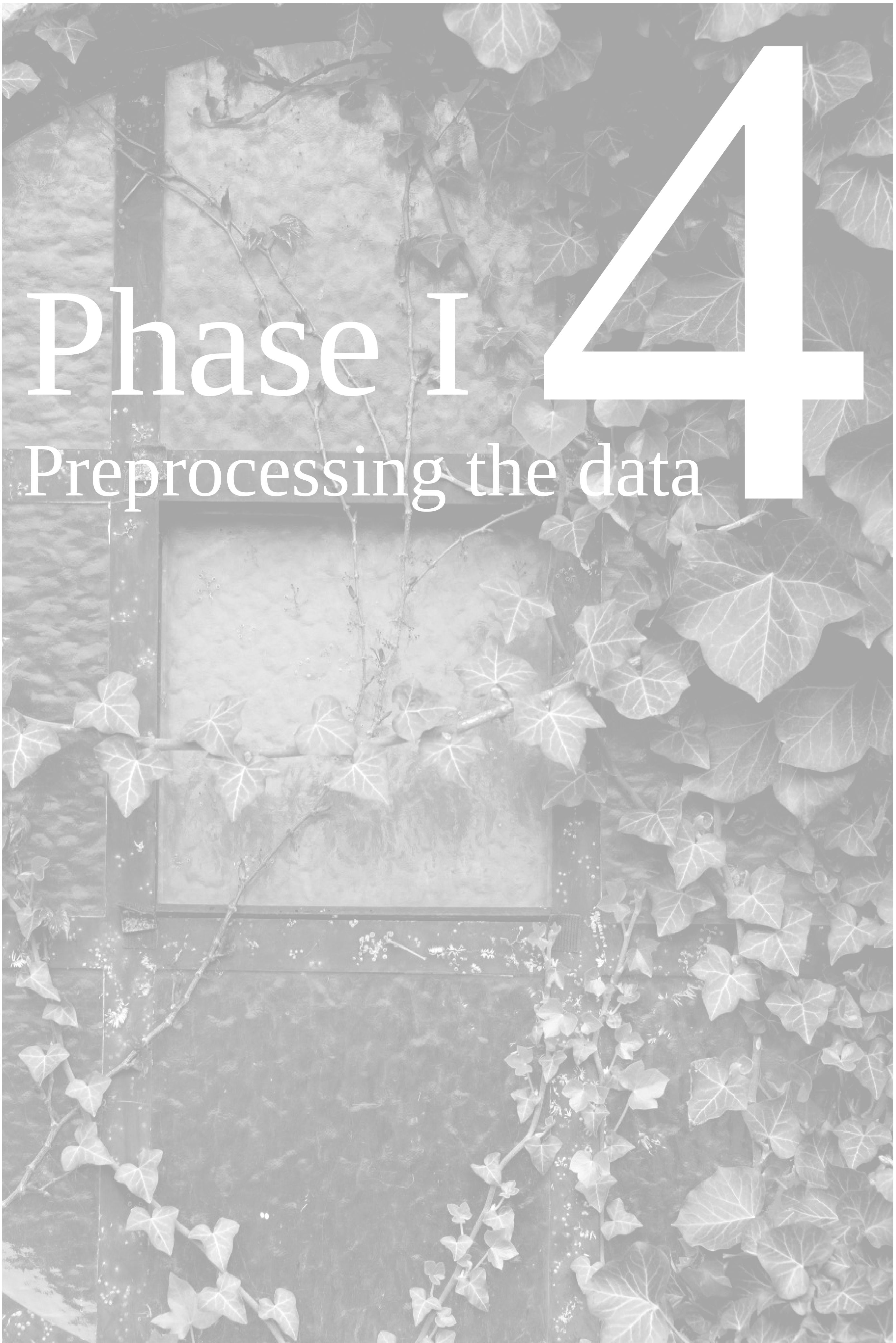


Figure 4: Relative cumulative number of patients per age at the moment of dying for each illness trajectory and for the total group of patients.



Phase I 4

Preprocessing the data

4. Phase I: Preprocessing the data

Phase I focuses on processing the corpus in such a way that it can be fed to the LSTM model. The input of Phase I consists of the raw corpus data. Different processing steps are applied to each feature category in order to make all features fit a strict predefined format. The output is a collection of all data per feature category for each patient. The unstructured data (free text) and the structured data (all other data) are processed in fundamentally different ways. The main goal of Phase I is to extract and process data from the corpus. Most of the processing steps during Phase I are directed towards extracting more useful information and less noise from the unstructured, textual data.

4.1 Processing structured data

The structured data consists of patient characteristics such as gender and the date of birth, and medical data which changes through time dynamically, such as diagnostic codes and prescribed medication. The main goal of processing the structured data during Phase I is to have all data of the same type fit the same mold. Some medical codes for example may show more detail than others, which impedes direct comparison. The medical codes are processed as follows: multiple codes that occur in one text field are split into separate codes, noise and white space are removed, and the codes are abstracted from the sub-code level (e.g. A23.4 is generalized to A23). Medication names are cleared from information regarding the dosage and use. Lab tests are only included when they resulted in irregular or abnormal values (as determined by the GP).

4.2 Natural language processing pipeline

The unstructured, textual data is of an entirely different nature than the structured data, and can therefore not be processed as such. The goal of processing textual data during Phase I is to normalize the text, and, to a certain extent, map different variations of a concept to the same word. Due to the nature of language and the high level of noise, textual data contains many more unique items than the structured data. The following issues are addressed as part of the natural language processing pipeline: 1) writing conventions, 2) language productivity, 3) phrases, 4) domain-specific corpus characteristics, and 5) noise.

Writing conventions in general, but their inconsistent use additionally, impedes recognizing two instances of a word as such. Examples include inconsistent application of upper- and lowercase letters sentence-initially, to proper names, to medication, and to chemical names, among other types of words. Furthermore, the comparison of words is inhibited by the inconsistent use of diacritics, and the use of non-lexical tokens, such as numbers, symbols, lab values, and dates.

Additionally, Dutch as a natural language is inherently **productive**. Dutch is characterized by inflectional, derivational and compounding processes, which obstruct the recognition of similar semantic entities. Both items within the following pairs refer to the same concept to a smaller or larger degree, but the members of each pair are not recognized as the same concept at all, without further processing: s*laap* ‘sleep’ (SG) - *slapen* ‘sleep’ (PL), *cva* (abbreviation of *cerebro vasculair accident*, ‘stroke’) - *cva’tje* (diminutive form), and *kanker* ‘cancer’ - *longkanker* ‘lung cancer’ for example.

Similar to compound words, the corpus contains multi-word units that are static **phrases** which refer to one concept, although the items may (infrequently) be used apart from each other as well. Such multi-word phrases lose some of their meaning when the items are analyzed separately. This is the case for many diagnostic terms, such as *diabetes mellitus*, *decompensatio cordis*, and *reumatoïde artritis*. Many of these terms rely heavily on loan words, and therefore do not adhere to the Dutch compounding rules. In some cases, words should be written as compound words, but because abbreviations tend to include the initials of all important *elements* of the phrase regardless of word boundaries, the elements tend to be split as a result when the full phrase is written as well. *Vesiculair ademgeruis* ‘vesicular breathing sound’, frequently abbreviated as *vag*, is often written as *vesiculair adem geruis* for example, thereby incorrectly splitting the elements of the compound *ademgeruis*.In many cases the abbreviated form is used more frequently than the original phrase, which may lead to further suppression of the original phrase structure.

Furthermore, natural language is characterized by synonymy and polysemy in general, but **domain-specific terminology** increases the problem of recognizing similar semantic concepts even further. On the one hand, the use of jargon alongside standard-language alternatives, and specifically the use of many Latin and Greek (derived) words alongside Dutch alternatives, greatly increase the amount of synonymy. Examples include *ruimte-innemend proces* - *kanker* (Dutch jargon versus the standard-language term for ‘cancer’), *letsel* - *laesie* - *trauma* (Dutch, Latin and Greek words for ‘injury’), and *bot* - *been* - *os* (two Dutch words and a Greek word for ‘bone’). On the other hand, many domain-specific abbreviations overlap fully with other corpus words, thereby increasing polysemy. Examples include *ca* for *carcinoom* ‘carcinoma’, which is used to abbreviate *circa* as well, and *mis* which is used to abbreviate *misschien* ‘maybe’, but which carries the meaning of ‘incorrect’ as well.

Finally, the noisy character of the data hinders the detection of semantically similar words. Because many of the texts are not written with the intention to be read by others, the writing conventions are often ignored or incorrectly applied, and many typographical errors, frequently used medical abbreviations, non-standardized and inconsistently used doctor-specific abbreviations, and other kinds of idiosyncrasies are used in the texts. Table 3 provides examples of such kinds of noise.

Table 3: Examples of noisy data.

Type of noise	Example	Correct form / error description
Ignored writing conventions	patient	*patiënt*
	alzheimer	*Alzheimer* (proper name used for disease)
	type ii	type II
Incorrectly applied writing conventions	U, Uw	*u, uw* ('you', 'your')
	jeuk-branderig-pijnlijk gevoel	enumeration should be linked with ','
	slagen/minuut	*slagen per minuut* ('beats per minute')
Typographical errors	op gelucht	should be concatenated
	konclusie	homophonic confusion
	knaker	letter shift (should be *kanker*)
(semi-)standardized medical abbreviations	*x-thorax*	*x* instead of *x-ray*
	1xdd	*éénmaal daags* ('once a day')
	re / r, li / l	*rechts* 'right', *links* 'left'
Doctor-specific abbreviations	ws, wrs, wsch, wrsch, waarsch	*waarschijnlijk* ('probably')
	zh, zhs, zkh, zkhs	*ziekenhuis* ('hospital')
	pat, pte, pa, pt, p	*patiënt(e)* ('(fe)male patient')
Other idiosyncrasies	re>li	'right side (*re*) more than left (*li*) side'
	o/rug, b/rug	*o/* indicates *onder* ('lower'), b/ *boven* ('upper')
	#	separates subjective notes from observations

The texts are processed by a natural language processing pipeline 1) to improve the quality of the texts by removing and correcting noise, 2) to improve the recognition of semantically similar words, and 3) to remove redundant information such as headers and footers from letters. As Figure 5 shows, the pipeline consists of processes to tokenize and normalize the text, remove headers and footers from letters, rephrase certain textual units, provide part-of-speech tags, and correct spelling errors. Each of the steps of the NLP pipeline after data extraction is discussed in more detail below.

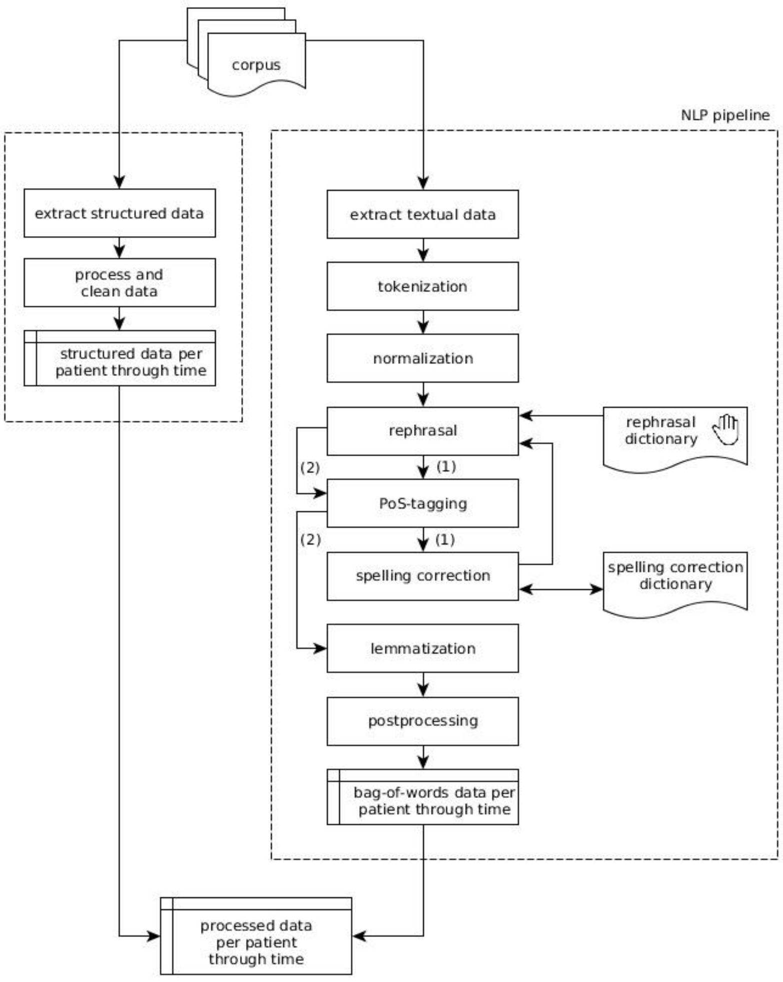


Figure 5: Data extraction and preprocessing pipeline. Processing steps 'rephrasal' and 'PoS-tagging' are passed twice. During the first pass, the arrows labeled (1) are indicative, during the second pass, the arrows labeled (2) are indicative.

4.2.1 Tokenization

Tokenization transforms the strings of text into a list of words for each document, by replacing white space and punctuation with word boundaries. Hyphens that occur within words (i.e., that are directly preceded and followed by letters) to create compound words are an exception, and are not replaced by word boundaries.

4.2.2 Normalization

Normalization is applied to neutralize differences between variants of the same word, e.g. 'patient', 'Patient', and 'patiënt' (*patient*). All tokens are lowercased, and diacritical symbols are removed ('ë' → 'e').

4.2.3 Rephrasal

Rephrasal is applied to 1) enable the recognition of synonyms, 2) map abbreviations to the corresponding word(s), 3) split certain compound words, and 4) avoid the separation of certain multi-word phrases. We replaced single- or multi-word units by other single- or multi-word units with the aid of a hand-made translation dictionary, that contains roughly 1,700 rephrased pairs.

The rephrasings may be one-to-one mappings for replacing synonyms (e.g. *fractuur* and *breuk* both refer to 'fracture', therefore *fractuur* → *breuk*), one-to-many mappings to partition certain compound words (e.g. *baarmoederhalsoperatie* represents 'cervix' + 'surgery', therefore *baarmoederhalsoperatie* → *baarmoeder-hals* + *operatie*), many-to-one mappings to retain fixed phrases (e.g. *diabetes mellitus* → *diabetes_mellitus*), and many-to-many mappings to perform several of the aforementioned steps simultaneously (*ruimte innemend proces hersenen* refers to the concept of 'cancer', and *hersenen* 'brain' is synonymous to *brein*, therefore *ruimte innemend proces hersenen* → *brein carcinoom*).

Whether or not compound words refer to one or more mental concepts for native speakers of Dutch, is complex to determine, especially automatically. Therefore, we hand-picked a set of frequent, domain-specific words instead of trying to determine automatically which compounds should be split. We picked the items from a list of the 5,000 most frequent unigrams, and 1,000 most frequent bi- and trigrams found in the development data.

Examples include *ooginfectie* 'eye infection', which is split into *oog* + *infectie* and *seniorenservice* 'senior service', which is split into *senioren* + *service*, while other compound words such as *netvlies* 'retina' (which consists of *net* + *vlies*) and *ziekenhuis* 'hospital' (which consists of *zieken* + *huis*) were left intact. Furthermore, we included words in the rephrasing dictionary if they were common non-standard words or had a notable amount of close semantic neighbors. The non-standardized abbreviations *p* and *pat* are frequently used to refer to *patiënt* 'patient' for example, and the words *ca*, *carc*, *carcinoom*, *gezwel*, *kanker*, *maligniteit*, *metastase*, *neoplasma*, *tumor*, and *ruimte innemend proces* — among many other words — all refer to the concept of 'cancer'. Because of the productive nature of language and the noisy character of the corpus, completeness of the rephrasal dictionary is not claimed.

To illustrate the effect of rephrasing, four examples are given below. The hypothetical example below shows text fragments for four patients, all suffering from cancer. The cancer-related words are underlined.

Patient A:	... *om de lymfekliertumor te behandelen....*
	'... to treat the lymph node tumor...'
Patient B:	... *dat er een r.i.p. li eierstok is vastgesteld...*
	'... that an expansion process in the left ovary is determined...'
Patient C:	... *is gebleken dat de ovariummetastasen...*
	'... has turned out that the metastases in the ovary...'
Patient D:	... *overlegd over de gevolgen van het eierstokca...*
	'... discussed the consequences of the ovary carcinoma...'

All patients have a form of cancer, although in all cases different words are used to describe the disease. Table 4 shows what the feature vector for these patients would look like, if no further processing was applied: the vectors for the four patients do not overlap at all, even though they all suffer from cancer.

Table 4: Feature vector for different hypothetical patients.

	lymfekliertumor	**r.i.p.**	**eierstok**	**ovariummetastasen**	**eierstokca**
Patient A	✓				
Patient B		✓	✓		
Patient C				✓	
Patient D					✓

The table only shows the features that are relevant for the corresponding examples.

Table 5 shows examples of different types of rephrasings that concern the concept of cancer. Example 2 in Table 5 shows the mapping of an abbreviation to a single-word phrase. Examples 1, 3 and 4 show the mapping of compound words into multi-word phrases. Such rephrasing is done because we interpret *lymfekliertumor* ('lymph node tumor') and *ovariummetastasen* ('ovary metastases') as two different types of the same disease (i.e. cancer), rather than two entirely different diseases. Separating compound words into separate parts enables storing the parts individually.

In examples 1 and 3 in Table 5, the compound words are split into two parts, but the result is not simply a partitioning of the compound. One or more parts are substituted by different words, in order to solve synonymy: *ovarium* is the Latin equivalent of the Dutch *eierstok* ('ovary'), and *metastasen* ('metastases') is one of many words that refer to the concept of cancer. If *ovarium* and *eierstok* would not be mapped to the same word, Patients B, C and D would be considered to have different diseases, as is illustrated by Table 4. Table 6 shows the result of the rephrasing procedure: in this representation, all patients have cancer, but not the same type.

Table 5: Examples of substitutions.

	Mapping		
1)	lymfekliertumor	→	lymfeklier carcinoom
2)	r.i.p.	→	carcinoom
3)	ovariummetastasen	→	eierstok carcinoom
4)	eierstokca	→	eierstok carcinoom

Table 6: Feature vector for different hypothetical patients after the texts are rephrased.

	lymfeklier	**eierstok**	**kanker**
Patient A	✓		✓
Patient B		✓	✓
Patient C		✓	✓
Patient D		✓	✓

The table only shows the features that are relevant for the corresponding examples.

The rephrasal dictionary was created on the basis of the development data, and applied to both the development data and validation data. After the following spelling correction step (described in Section 4.2.5) the rephrasal dictionary was applied once more, to ensure the rephrasing of words that were missed initially due to spelling errors.

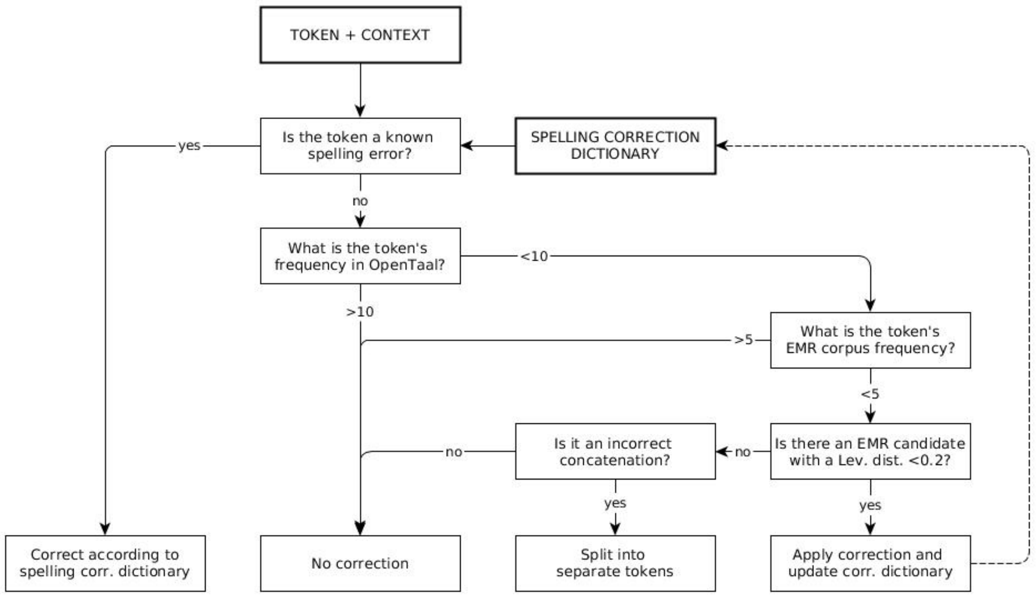
4.2.4 Part of speech tagging

Part of speech (PoS) tagging is applied to 1) enable word-sense disambiguation, 2) aid in spelling correction (see next section), and 3) enable feature selection in Phase II. Using Frog (version 4.2.2, Van den Bosch et al. 2007), the text is automatically provided with PoS tags. The tokens are substituted by '[token]:[PoS tag]' pairs, for example: *naald* ('needle') → *naald:N* ('needle:N'), in which the N refers to the PoS tag 'noun'. PoS tags help to a certain extent to discriminate between the different uses of words. Examples include polysemous words: *slaap*, which can refer to 'the side of the head' and to 'sleeping', which would be tagged as 'slaap:N' (noun) and 'slaap:WW' (verb) respectively, based on the linguistic context in which they appear, thereby enabling disambiguation. Abbreviations that correspond to more regular items, such as *ca* which is used to abbreviate *carcinoom* and *circa*, would be tagged 'ca:N' (noun) and 'ca:VZ' (adverb), respectively.

4.2.5 Spelling correction

Spelling correction is applied to normalize noisy data. Regular spelling correctors are sub-optimal for the EMR corpus. Because the corpus contains a large amount of domain-specific words and abbreviations, comparing the corpus words with a list of standard-Dutch words to detect non-words, leads to many false positives, and thus many incorrect substitutions. The noisy character of the textual data oftentimes leads to situations in which it is unclear whether an uncommon word is an actual error, e.g. **vag* instead of *vaag* ('vague'), a (non-standard) abbreviation with or without punctuation marks, e.g. *vag* instead of *vesiculair ademgeruis*, or simply jargon, e.g. *decompensatio*.

Moreover, it is possible that a word exists both in the medical domain as well as in general, but refers to different concepts depending on the domain, which may lead to different usage patterns. Examples include the use of *positief* ('positive') in everyday life as a synonym for *goed* ('good'), while in the medical domain it means that the phenomenon a patient is tested for, is indeed present. Another example is the word *mamma*, which in everyday use refers to 'mother', but in the medical context refers to 'breast'. The abbreviation *r.i.p.* is generally used for 'rest in peace', while in the medical domain it is used to abbreviate *ruimteinnemend proces*, which refers to cancer — 'rest in peace' will be intended way less frequently (if at all) in the medical domain than 'cancer', and vice versa in general language. Such differences in meaning lead to different semantic and syntactic embeddings. It is therefore



TOKEN + CONTEXT
Is the token a known spelling error?
SPELLING CORRECTION DICTIONARY
yes
no
What is the token's frequency in OpenTaal?
<10
>10
What is the token's EMR corpus frequency?
>5
<5
Is there an EMR candidate with a Lev. dist. <0.2?
no
Is it an incorrect concatenation?
no
yes
yes
Correct according to spelling corr. dictionary
No correction
Split into separate tokens
Apply correction and update corr. dictionary

Figure 6: Decision tree of the spelling corrector.

suboptimal to use general language as a foundation for a spelling checker that is intended to be applied to a specific domain.

To circumvent these issues, a spelling corrector was created specifically for this corpus. The spelling corrector follows a decision tree to detect possible spelling errors, as illustrated by Figure 6. The spelling corrector determines for each token in its specific syntactic context whether or not the token contains a spelling error. A dynamically growing list of spelling corrections that the spelling corrector has found by analyzing previous text, is consulted first. Each time the spelling corrector finds a new error, it is appended to this correction dictionary.

For each word that is not known to be incorrect (i.e., is not an item in the correction dictionary), the spelling corrector first decides whether or not the word contains an error. Based on the frequency of the word, it is decided whether or not a correction should be applied. Because the EMR corpus is relatively small, the frequency cut-off boundary used to identify possible errors within the EMR was 5. This means that a word is suspected to be an error if it occurs less than five times in the EMR corpus. By using a low frequency however, many false positives (infrequent but correct words that are often domain-specific, such as rare diseases) and false negatives (frequent spelling errors) are collected by the frequency filter, or slip through it, respectively. Therefore, the spelling corrector compares the tokens in the EMR corpus with background corpus OpenTaal first. The frequency boundary used for OpenTaal is 10. If a word occurs frequently enough in OpenTaal, it is not identified as a possible error. If a word is infrequent in OpenTaal, but frequent in the EMR corpus, it is assumed to be spelled correctly as well. If a word is infrequent in *both* corpora however, it is considered a possible error, and further processing steps apply.

The frequencies for the cut-off boundaries were determined by comparing the result of three different frequency boundaries: 5, 10 and 15. We determined the best boundaries for the EMR corpus and the OpenTaal corpus separately, because the corpora differ in size and domain. Higher boundaries than 5 for the EMR corpus led to detection errors that were actually false positives, such as correct but infrequent compound words (e.g. *littekenpijn*, ‘scar pain’) and names of relatively rare diseases and substances, among other things (e.g. the antigene ‘Galactomannan’). A cut-off boundary of 10 was chosen for the OpenTaal corpus, because similar to the EMR corpus a higher boundary led to too many false positives, and a lower boundary led to many false negatives, such as many non-Dutch words (e.g. *energy* and *obligation*) and frequent spelling errors (e.g. **produkt,* incorrect variant of *product*), which resulted in unwanted corrections.

When a spelling error is assumed, a plausible correction candidate is attempted to be found within the EMR corpus. A correction candidate is said to be plausible, if the relative Levenshtein edit-distance between the possible error and the correction candidate is less than 0.2, and if the correction candidate is more frequent than the frequency boundary. We used the relative distance between two words, because the absolute difference can either be considered as a large or as a small distance, which depends on the length of the incorrect word. When a short word like *meta's* (non-standardized abbreviation of *metastasen*, 'metastases') is corrected to *metaal* ('metal') for example, the mutation is relatively large. When a longer word such as **hersenvliesontstekign* is corrected to *hersenvliesontsteking* ('meningitis') however, the correction is more likely to be good because it is relatively small, and due to the fact that longer words have less near neighbors.

Based on the findings of Walasek (2016), who explored the effect of different edit distance thresholds for spelling correction in the medical domain, three values were tested to determine the optimal maximum edit-distance: 0.15, 0.175, and 0.2. In our case, the number of good corrections increased faster than the number of wrong corrections with an increasing maximum edit distance, therefore the maximum edit-distance of 0.2 yielded the best results. Tables 7 and 8 provide examples of correct and incorrect substitutions that resulted from using this value. Table 7 gives an overview of different types of corrections and explains the type of error that is corrected. Table 8 gives an overview of incorrect substitutions, and explains why the substitution is incorrect. We considered compound words that were split into separate correctly spelled parts as correct substitutions — these type of substitutions fit the spirit of splitting multi-word units into subparts, which is done during the rephrasing procedure as well to increase the recognition of multiple concepts from one word.

Table 7: Examples of correct substitutions and characterizations of the errors.

Correct substitutions	**Type of error that is corrected**
*contgrole → controle	letter insertion
*metastsen → metastasen	letter deletion
*behnadeling → behandeling	letter shift
*gekraagd → gevraagd	letter substitution
*onvermogenom → onvermogen om	incorrect concatenation
lymfeklierextirpatie → lymfeklier extirpatie	infrequent compound word

Table 8: Examples of incorrect substitutions and explanations for correction errors.

Incorrect substitutions	**Explanation**
snijwondje → snijwond je	*snijwondje* is correct but infrequent, better split would be *snij wondje*
*bloedgluc → bloed *gluc	*gluc* is an abbreviation, this problem is fixed by 2nd round of rephrasing however
*lipen → lopen	good substitution in some contexts, but should be *liepen* or *lippen* in other cases
*wgschl → *wrschl	substitution is a frequent non-standardized variant of *waarschijnlijk*
*weeiig → wee *iig	should be substituted by *weeïg*, but this word is absent from the corpus
kind(eren) → kinderen	brackets intended for optional word features are an unforeseen case.

The presumed error is substituted with the correction candidate, if the candidate meets the following criteria: 1) the relative edit-distance is maximally 0.2, 2) the correction candidate is more frequent than the frequency boundary, and 3) the correction candidate appears at least once in the same syntactic context as the suspected error. The context is defined as the two PoS tags of the words directly preceding and following the word. Some words appear only in a very limited number of contexts, while others are more flexible. Taking the context into consideration decreases the chance of incorrect substitutions. In case of **bloedind* for example, which is used in a noun position, we may have two equally plausible correction candidates based on the edit-distance, that are both more frequent than the frequency boundary: the noun *bloeding* ('bleeding') and the adjective *bloedend* ('bleeding'). It is hypothesized that the substitution that can occur within the same syntactic context, is more plausible than the substitution that is not; therefore **bloedind* is replaced by *bloeding* instead of *bloedend.*

If no plausible substitution is available, the spelling corrector checks whether the word is an incorrect concatenation of two existing words. The spelling corrector walks through the word letter by letter, splitting the word into two parts at each point, but only if the two parts both have a minimal length of two letters. Splits are considered plausible if both parts of the word occur in the corpus more frequently than the frequency boundary. If more than one plausible split is found, the optimal split is defined as the split for which the least frequent part has the highest frequency, compared to the least frequent part of other plausible splits.

If no good substitution can be found by either of the two methods, the word is left unchanged. If during the process a plausible spelling correction is found, the correction is added to the spelling correction dictionary. This way, when the spelling error is encountered again, the spelling corrector can simply retrieve the correct substitution from the correction dictionary and apply it directly.

After all words are checked and corrected if necessary, the rephrasing procedure is applied once more, to rephrase words and phrases that were skipped by the initial rephrasing procedure due to spelling errors. Additionally, the text is PoS tagged once more by Frog to enhance the quality of the assigned PoS labels, for the same reason: due to spelling errors, the wrong part of speech tags may have been assigned during the initial round of PoS tagging.

4.2.6 Lemmatization

Lemmatization is applied to all words in the corpus to neutralize the effect of inflection and derivation. Inflection and derivation hinder the recognition of conceptually similar words that are spelled slightly differently. Replacing all words by lemmas, increases the recognition of semantically similar words.

4.2.7 Postprocessing

Postprocessing is finally applied to remove non-words and redundant information, such as headers and footers of letters. This leads among other things to the exclusion of dates, phone numbers, urls, and e-mail addresses. Everything up to and including the actual opening of the letter, and everything from the closing of the letter onward is removed, because often medically irrelevant information such as contact information and the name of the health care facility are mentioned in the header and footer of the letter. Although we did not want to use this data as input to the model during Phases III and IV, we did not exclude it from the *start* of Phase I because the spelling corrector relies on word frequency counts. Excluding it from the start of Phase I would have caused the spelling corrector to make unnecessary and incorrect substitutions.

5 Phase II
Feature reduction

5. PHASE II: FEATURE REDUCTION

Phase II is dedicated to the preparation of the feature selection procedure which is executed during Phase III. During this phase, several approaches for feature reduction were applied to decrease the size of the feature categories. The definitive feature selection was executed based on experiments with different feature sets in Phase III — during Phase II the feature sets are merely prepared. This section discusses feature reduction methods for the structured data and unstructured data, respectively. Age and gender are not discussed in this section, because we did not reduce these feature categories in any way.

After merging similar feature categories, such as the categories of letters and notes, nine general features categories remained. Each of these feature categories however contains many unique features. While the algorithm needs no explicit instruction to learn which features are important for making correct predictions, introducing too much noise affects the learning process negatively, by requiring both more data to learn from and a higher model complexity.

The LSTM algorithm can only learn which (interactions between) values are relevant if 1) the values occur frequently enough overall, and 2) the values occur frequently enough *in significant contexts*. Influenza for example is frequently diagnosed, but is usually not an important indicator for life expectancy. In combination with old age and bronchitis however, the disease may be fatal. If the diagnosis influenza is frequent (thereby satisfying the first criterion), but combinations of influenza and other feature values that are indicative *together* are not frequent (thereby *not* satisfying the second criterion), the model is unlikely to learn the importance of influenza for predicting life expectancy. Therefore, the more values it needs to learn, the more data is needed to train the algorithm.

We define a good feature set as being maximally descriptive while minimal in size, and containing features that are relatively frequent. Including infrequent features in the feature set has drawbacks: as discussed, the algorithm needs much data to learn the predictive value of the feature, and even if it does, this knowledge is only applicable incidentally, while the model complexity is increased permanently. Phase II aims to reduce the size of the feature set while retaining as much good predictors as possible. Table 9 shows the number of features per feature category if no selection is made.

Using a total of almost 60,000 features would require a very complex, high-dimensional model and a lot of data to learn from. The development data consists of only 1107 patients, which is not a lot of data, considering the size of the feature set and the complexity of the task.

Table 9: Number of unique features per category.

Feature category	**# Features**
Medical history	136
Contact type	81
Diagnosis	572
ICD-10	857
Reason for encounter	862
Intervention	380
Medication	1,490
Lab tests	269
Keywords	53,423
Total for structured data	4,647
Total for unstructured data	53,423
Total amount of features	58,070

In order to relax the high demands in terms of data quantity and model complexity, methods for feature reduction were implemented to remove redundant, irrelevant and infrequent features. Depending on the type of feature, different methods were used. In the following sections, the techniques for processing structured and unstructured data are described, respectively.

5.1 Feature selection for structured data

5.1.1 Frequency-based feature reduction

As Table 9 shows, most feature categories are represented by many features in the EMR corpus. Not all values are equally frequent, however. Infrequent values are certainly not automatically irrelevant values: uncommon diseases for example may be very good predictors for life expectancy. However, because they are infrequent, including them in the model is expected to offer little to no increase in accuracy — it may even decrease the model’s performance. Because such features are uncommon in the training data, the model is likely to be unable to learn the relevance of such features properly. Moreover, because these features occur very infrequently in reality, the chance of encountering a new patient with the disease is small. Therefore, even if the model assigned the right value to the feature and predicts life expectancy well when a patient with that feature is encountered, the chances of encountering such a patient are very small in the first place. Excluding infrequent features is therefore expected to decrease the model complexity, without decreasing its accuracy. Three frequency-based feature reduction methods were applied to the structured data in Phase II, and tested during Phase III:

- absolute frequency cut-off boundary: only include values with an absolute frequency of at least 100;
- relative frequency cut-off boundary: only include values that make up at least 1% of data for a feature category;
- cumulative relative frequency cut-off boundary: sort values by their relative frequency (high to low), and only include the top *n* values that together cover at least 75% of the data for a feature category.

The amount of feature reduction for each reduction method is shown in Table 10.

Table 10: Remaining number of features per category after the application of different frequency-based feature reduction methods.

Feature	**Total**	**Abs. freq.**	**Rel. freq.**	**Rel. cum. freq.**
Medical history	136	0	24	38
Contact type	81	22	12	4
Diagnosis	572	218	21	83
ICD-10	857	231	18	103
Reason for encounter	862	179	17	56
Intervention	380	115	28	24
Medication	1,490	212	20	141
Lab tests	269	47	24	23
Total	4,647	1,025	164	473
Corpus coverage	100%	81%	60%	75%

5.1.2 Feature reduction for ICPC and ICD-10 features

In addition to the frequency-based reduction methods, one additional reduction method was tested for the ICPC and ICD-10 features: aggregation of codes by using different levels of detail. The ICD-10 and ICPC codes have a similar format: a letter followed by two digits, a decimal separator, and again one or two digits, e.g. ‘D84.02’. The letter indicates which system has been affected (e.g. ‘D’ refers to the digestive system), in this research referred to as the *main category level*. The two digits before the decimal separator indicate the illness or condition (e.g. ‘D84’ refers to an esophageal condition).

A combination of a letter and two digits is referred to as the *code level*, and is the level which is used as the default level throughout this research. The digits following the decimal separator specify the details (e.g. ‘D84.02’ refers to ‘esophageal reflux without esophagitis’). This level, the total of a letter and two digit pairs, is referred to as the *sub-code level*. This level is not used in the research because the codes reported in the EMRs rarely show this level of detail. Moreover, using this level of detail would only lead to *more* unique feature values instead of *less*.

In case of the ICPC codes, two additional levels are applicable. The ICPC cannot only be used for encoding the diagnostic components of a consult (in contrast to ICD-10 codes), but also for reporting the screening, complaint, treatment, test result, administrative, and referral components. For this reason, the ICPC system is used to encode different types of feature categories: it covers the medical history, reason for encounter, diagnosis, and intervention. We refer to this partitioning into components as the *component level*. Additionally, we refer to the combination of the main code and the component as the *subcategory level*.

ICD-10 codes cannot be categorized into such component classes. For the ICD-10 codes a subcategory level is also used, but it is not based on a combination of main category and component. The ICD-10 groups similar codes into thematically coherent sets, leading to compressed lists of codes. This list can be interpreted best as a list of primary disease patterns, such as ‘metabolism-related disease’ or ‘psychological conditions’. When we use the *subcategory level* for the ICD-10 codes, we refer to these primary disease patterns.

Table 11 shows the reduction that results from aggregation over different levels, for each frequency cut-off level. Features were reduced by using codes with higher levels of abstraction. To see which level provides the best results, each level was tested during Phase III.

Table 11: Amount of feature reduction as a result of different code levels and different frequency cut-off levels, compared to the default feature set (code level and no cut-off). The *retained* columns show the size of the feature set, the *reduced* columns show the amount of reduction.

	Code level		**Subcat. level**		**Main cat. level**		**Component level**	
	retained	*reduced*	*retained*	*reduced*	*retained*	*reduced*	*retained*	*reduced*
No cut-off	4,647	0%	2,145	54%	1,928	59%	1,882	60%
Abs. freq.	1,024	88%	467	90%	351	92%	314	93%
Rel. freq.	164	96%	158	97%	131	97%	88	98%
Rel. cum. freq.	472	90%	220	95%	200	96%	177	96%

Abbreviations: cat. = category, abs. = absolute, rel. = relative, cum. = cumulative, freq. = frequency.

5.2 Keyword selection

Compared to the structural data, there are more than ten times as much keyword features than all other features taken together, without further processing. Certain characteristics of textual data call for a different feature selection approach for textual data than the for the structured data. The meaning of a medical code is not variable and different codes cannot be used to refer to the same phenomenon, while the exact meaning of a word often depends on the context in which it is used, especially in case of polysemous words. Vice versa, many words can be used to describe the same concept, due to synonymy.

Three different approaches for keyword selection were prepared. The first method is a frequency-based ranking approach, which represents the textual data with a naive bag-of-words model. While it is true that words which are important in a document tend to occur relatively frequently in a document, plain frequency counts of unique words throughout the corpus tend to be bad indicators for the importance of words. Both the most frequent words (i.e. function words) and the most infrequent words (usually noise such as typographical errors) tend to be the least informative words. The frequency-based ranking approach, for which all unique words were ranked by their absolute corpus frequency, therefore functioned implicitly as a baseline to compare more advanced textual representations to.

Two additional approaches for keyword selection were prepared: 1) entropy-based ranking and 2) a vector space representation using word embeddings, created with WORD2VEC. The frequency and the entropy methods both deliver a list of keywords in decreasing quality, so when the top *n* of the list is taken during Phase III, it always includes the best keywords in terms of the ranking criteria. The entropy ranking and the WORD2VEC representation rely on distributional properties of the textual data, but in different ways, which is discussed in more detail in the following sections. Part of speech filtering was applied to all three selection methods: all parts of speech were used to determine the distributional properties of the words, but function words were removed from the three collections of keyword features after ranking the keywords.

5.2.1 Entropy-based ranking

The entropy-based ranking method aims to compare the actual keyword distributions throughout the corpus to a set of distributions which we regard as ‘optimally predictive’. We consider a distribution to be predictive if the word frequency increases or decreases through time. Figure 7 illustrates examples of predictive words, with different types of slopes. Although the frequency of the words in

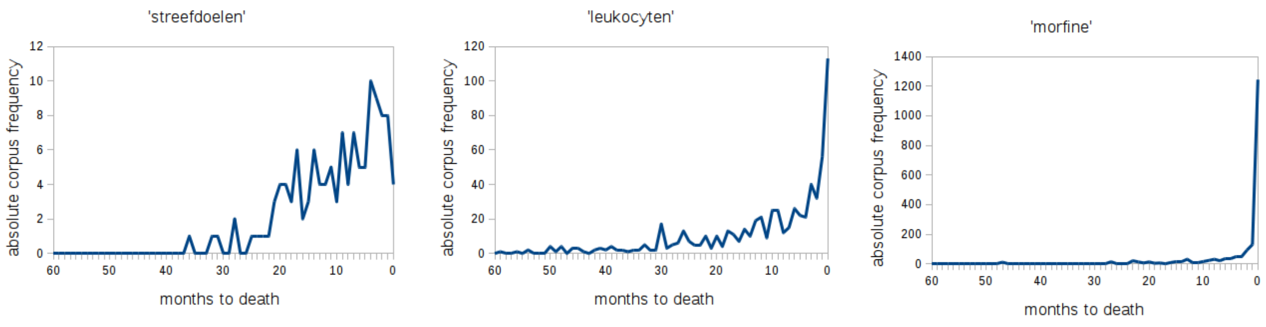


Figure 7: Three distributions of word frequencies through time.

Figure 7 changes at a different rate over time, we consider these distributions all to be predictive.

The Kullback-Leibler divergence is an entropy-based measure of similarity used to compare two probability distributions. In this research, we use the Kullback-Leibler divergence to compare the distribution of each word in the corpus to distributions which we regard to be optimal for predicting life expectancy. Which distribution we consider to be optimal, depends on the individual word distribution. Figure 7 shows three predictive words, but if we would compare them all to the same artificial 'optimal' distribution — a type of exponentially increasing or decreasing function — words with a relatively linear progression (such as *streefdoelen*) or a relatively steep increase or decrease in frequency (such as *morfine*) would be considered as a bad fit to such a fixed exponential function.

Therefore, we determined an optimal distribution for each individual word based on its actual corpus distribution. We use the relative corpus frequency to remove the effect of the *total* amount of textual data from the equation, because more texts tend to be produced as the end of life approaches. Using absolute frequencies would result in frequency increases through time for *most* words, complicating the search for words associated with the end of life.

The optimal distribution for a word is a fitted curve C. As Formula 1 shows, the function fits curve C by determining 1) amplitude *a*, indicating the rate of change, 2) background *b*, indicating the overall use of the word in the corpus regardless of time, and 3) time scale *c*, indicating the period from which the steady background frequency starts to change as a result from the approaching end of life.

Formula 1: $C_{\text{word}} = a \times -b^{x} + c$

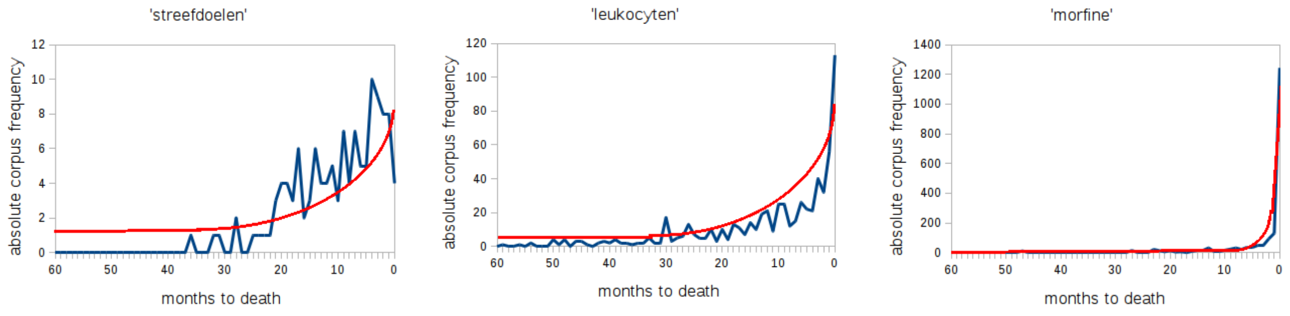


Figure 8: Optimal curve fitting for each individual word.

The fitted curve for a word functions as its *optimal* distribution, which can be compared to the *actual* distribution with the Kullback-Leibler divergence. The resulting entropy reflects to which degree the actual distribution adheres to the optimal distribution, or alternatively phrased: whether the frequency in- or decrease is relatively smooth and predictable, as for *morfine*, or whether it is relatively turbulent and unpredictable, as is the case for *streefdoelen*. Figure 8 illustrates that *morfine* adheres closely to its optimal distribution, while *streefdoelen* does not, due to the large amount of fluctuations. The distribution of each individual keyword is compared to its corresponding individual exponential, to provide an entropy score for each word. Words are regarded to be good predictors if their distribution throughout the corpus is characterized by a large amplitude *a*. Words however may also be good indicators if *a* is relatively small, while their actual distribution throughout the corpus adheres to the optimal curve tightly, without much fluctuations. Because the distribution of *morfine* fits the optimal curve better than *streefdoelen* fits its corresponding optimal curve, the entropy score for *morfine* is lower than for *streefdoelen.*

Figure 9 illustrates that a small part of the words occurs frequently, while the largest part of the words occurs infrequently (which is to be expected, according to Zipf’s law). Furthermore, many words have a relatively low entropy, while some words have a very high entropy. The most interesting words in the context of this research, are the words that reside in the upper-left quadrant of the graph, because they are both highly frequent and have a low entropy. Therefore, prior to ranking the words by their entropy score, a frequency cut-off and an entropy cut-off were applied, based on the data shown in Figure 9: words that occur in less than 200 patient records, words with an entropy score higher than 0.5, and words with a part of speech tag other than noun, adjective or verb are removed. The resulting words are ranked by increasing entropy.

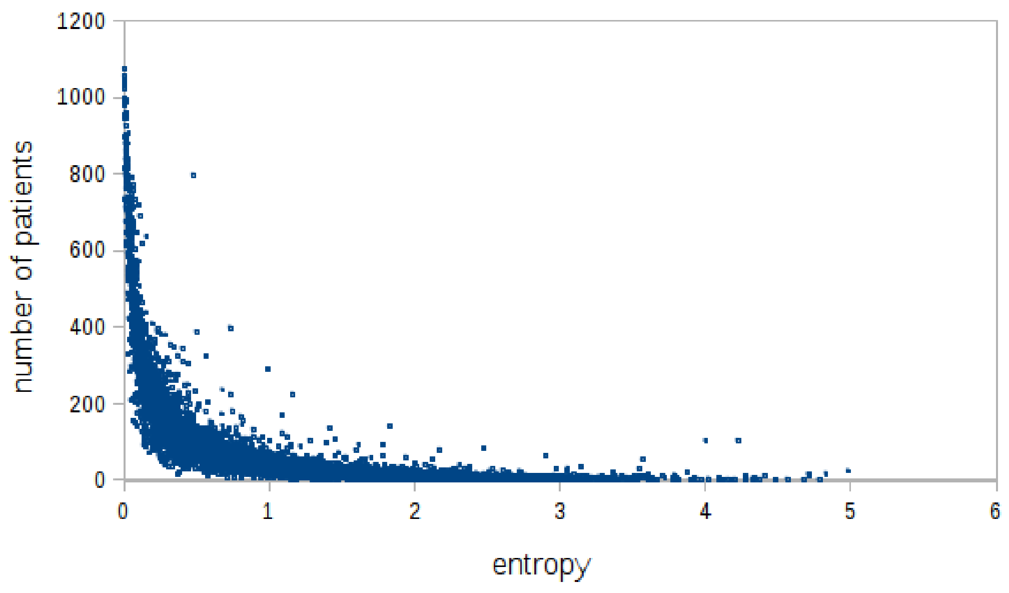


Figure 9: Distribution of words over the dimensions of entropy and corpus frequency (in terms of the number of EMRs in which the word is used).

To give an indication of the results, the list below contains the top 50 of most informative words, according to this ranking (*N*, *V*, and *A* refer to *noun*, *verb* and *adjective*, respectively).

The list contains roughly equal amounts of nouns, verbs and adjectives. Most of the nouns are clinically themed (e.g. *hypertensie* ‘hypertension’, *ademgeruis* ‘breathing sound’), and a category of nouns referring to time spans can be distinguished additionally (e.g. *dag* ‘day’, *week*, *jaar*, ‘year’). None of the verbs are obviously related to a clinical setting. We expected to find words along the lines of ‘to bleed’, ‘to suffer’ or ‘to vomit’, for example, but instead the list includes mainly general verbs and some verbs that are associated with functional abilities (e.g. *zien* ‘to see’, *lopen* ‘to walk’). The set of adjectives includes evaluation-related words (e.g. *goed* ‘good’, *normaal* ‘normal’), orientation-related words (e.g. *links* ‘left’, *rechts* ‘right’), and words that indicate a degree of certainty (e.g. *mogelijk* ‘possible’, *waarschijnlijk* ‘probable’), among others.

1. *patiënt* (N), ‘patient’
2. *goed* (A), ‘good’
3. *hebben* (V), ‘to have’
4. *onderzoek* (N), ‘research’
5. *rechts* (A), ‘right’
6. *zien* (V), ‘to see’
7. *klacht* (N), ‘complaint’
8. *worden* (V), ‘to become’
9. *anamnese* (N), ‘anamnesis’
10. *medicatie* (N), ‘medication’
11. *zullen* (V), ‘to shall’
12. *gaan* (V), ‘to go’
13. *kunnen* (V), ‘to be able’
14. *conclusie* (N), ‘conclusion’
15. *ver* (A), ‘far’
16. *ziekenhuis* (N), ‘hospital’
17. *controle* (N), ‘check-up’
18. *laat* (A), ‘late’
19. *gebruiken* (V), ‘to use’
20. *komen* (V), ‘to come’
21. *fysiek* (A), ‘physical’
22. *links* (A), ‘left’
23. *volgen* (V), ‘to follow’
24. *verband* (N), ‘bandage’
25. *pijn* (N), ‘pain’
26. *hart* (N), ‘heart’
27. *been* (N), ‘leg’
28. *doen* (V), ‘to do’
29. *bloeddruk* (N), ‘blood pressure’
30. *lopen* (V), ‘to walk’
31. *normaal* (A), ‘normal’
32. *dag* (N), ‘day’
33. *onderkant* (N), ‘bottom’
34. *ademgeruis* (N), ‘breathing sound’
35. *krijgen* (V), ‘to get’
36. *mogelijk* (A), ‘possible’
37. *last* (N), ‘bother’
38. *afwijking* (N), ‘deviation’
39. *week* (N), ‘week’
40. *waarschijnlijk* (A), ‘probable’
41. *blijven* (V), ‘to stay’
42. *diagnose* (N), ‘diagnosis’
43. *relateren* (V), ‘to relate’
44. *jaar* (N), ‘year’
45. *moeten* (V), ‘to be obliged’
46. *overleg* (N), ‘deliberation’
47. *maken* (V), ‘to make’
48. *hypertensie* (N), ‘hypertension’
49. *borst* (N), ‘chest’
50. *polikliniek* (N), ‘outpatient clinic’

5.2.2 Vector space representation using word embeddings

Using textual data generally results in very high-dimensional, but, at the same time, very sparse feature vectors. While the frequency-based and entropy-based ranking methods aim reduce the amount of data by delivering optimal subsets of words for any top *n*, these methods inevitably lead to the loss of potentially relevant information. Moreover, when applied to the texts in the EMR corpus, the feature vectors still only match the text documents sparsely, due to the short lengths of the documents and the large amount of unique words. Although it depends on the size of the text and the number of words that are included in the vector, the chance that words which occur in a given text are *also* selected as features, is small.

An alternative approach, which aims to address the issues of information loss and sparseness, makes use of word embeddings. This method employs WORD2VEC (Mikolov et al. 2013, Google's implementation, version 0.9.1, available via code.google.com) to create a vector space model, in order to reduce the dimensionality of the data without excluding potentially important indicators. In a vector space model, each word is represented as a point in a multidimensional vector space by a multidimensional vector, which is based on the word's distributional properties. All words are represented by the same *n* dimensions, but the exact position in the vector space is unique to each word.

The distance between words in the vector space is meaningful: words that reside in each others proximity (i.e., words with similar vectors), are semantically similar and tend to co-occur with each other, or occur in similar contexts. Figure 10 visualizes[4] a vector space of words. To illustrate how semantically similar words and thematically associated concepts are distributed over the vector space, we zoomed in on the word *dood* 'death'. Associated with the concept of death are family members and other loved ones, such as *echtgenoot* 'husband', *kleindochter* 'granddaughter' and *vriend* 'friend', as marked by hand in purple. Another thematic category of words related with death, are care-related words such as *zuster* 'nurse', *hospice* and *thuiszorg* 'home care', as marked by hand in blue.

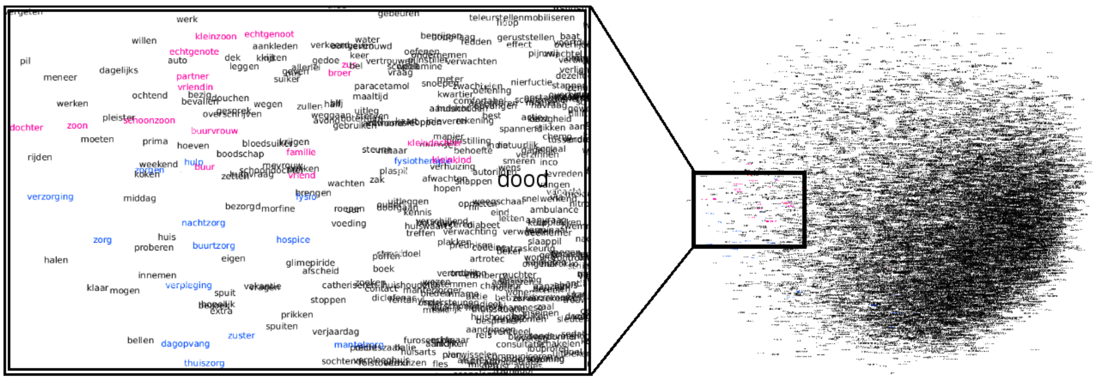


Figure 10: Visualization of the vector space representation created with WORD2VEC.

4 To enable visualization of the high-dimensional vector space, we applied a two-stage reduction procedure to map the model onto a two-dimensional vector space. We carried out the first reduction step with principal component analysis (PCA), to create a vector space with ten dimensions which emphasizes the variation in the vector space (Abdi & Williams 2010). Further reduction to a two-dimensional space was accomplished with the aid of a variation on Stochastic Neighbor Embedding (t-SNE). t-SNE has been proven to yield better results than PCA, but is computationally expensive (Maaten & Hinton 2008). Therefore, we preceded it with PCA reduction.

Instead of making a selection from all possible words in the corpus and thereby possibly missing important cues, the vector space representation allows the model to score each individual word that occurs in a given text. Similar vectors indicate similar words, therefore document representations can be created by calculating the mean of all feature vectors of the words in a text, which has proven to be effective for a broad range of tasks (Kenter et al. 2016).

To determine the optimal model architecture and parameter settings for WORD2VEC, we trained several WORD2VEC models with different architectures and parameter settings on the notes and letters from the development corpus. Inspired by Mikolov et al. (2013), we designed an analogy task to assess the quality of the model, because predefined tests to assess the quality of vector space models exist for English, but not for Dutch. We assigned WORD2VEC with the task of making an analogy to a word pair, given a third word. For example, given the pair ‘arm’ and ‘hand’ and given a third word such as ‘leg’, the model should return ‘foot’ in analogy to the given word pair. The test items (100 in total) were derived from IQ tests, collected through several online sources. Although these test items are not specific for the medical domain, the items included many general concepts such as body parts, animals, emotions, means of transportation, and family members. All items therefore occurred in the corpus.

As summarized in Table 12, we experimented with the model architecture (*continuous bag of words* or *skip-gram*), different cut-off boundaries for removing infrequent words from the model (10[5], 25, and 50), different window sizes (5, 7 and 10), and different numbers of dimensions (100, 200, and 300).

Table 12: Test settings to determine optimal WORD2VEC parameter settings, and accuracy per model.

Round	Architecture	Min. count	Window size	Dimensions	Accuracy
1a	bag of words	10	5	100	53%
1b	skip-gram	10	5	100	65%
2a	skip-gram	25	5	100	58%
2b	skip-gram	50	5	100	53%
3a	skip-gram	10	7	100	64%
3b	skip-gram	10	10	100	61%
4a	skip-gram	10	5	200	66%
4b	skip-gram	10	5	300	67%

For each model, the settings of interest are marked by a gray background.

5 The default cut-off boundary of WORD2VEC is 5, but due to the spelling correction procedure we applied, the number of words with a frequency <10 that remained in the corpus is almost none.

We used default settings for any other parameters. We conducted the optimization procedure in a stepwise fashion to decrease the number of experiments needed. During each round of testing, one parameter is varied upon. The setting that yielded the best result, which is either the initial default value or the value used for sub-experiments *a* or *b* (referring to *a* and *b* in column *round* of Table 12), is used for the following round of experiments.

Table 12 shows that the model that performed best, was the model tested during round 4b, which makes use of a skip-gram architecture, a cut-off frequency boundary of 10, a window size of 5, and 300 dimensions. Although the WORD2VEC model with 300 dimensions produced the best results on the analogy task, we tested the effect of using a WORD2VEC representation consisting of 100 and 200 dimensions (the other parameters kept equal to the best performing model) as well during Phase III, because we could not predict the outcome of the interaction between different amounts of keyword features and other features extracted from the structured data.

6 Phase III

Model development

6. Phase III: Model development

Phase III is dedicated to three goals: 1) transforming the data into a functional input format, 2) creating a baseline model, and 3) determining the effect of adding textual features. The following sections elaborate on the specific methods and results for each of these goals. Phase III resulted in an optimal model which was validated during Phase IV.

6.1 Transforming the data

Although the different feature categories have been processed during Phases I and II to ensure internal consistency for the feature format within each category, some additional processing steps were needed in order to create a viable input format for the LSTM model. This section describes how we represent the dynamically changing medical history of each patient through time.

6.1.1 Creating a timeline

When we created the input data for the model, the goal was to represent a patient by a series of feature vectors that change through time. The vectors are filled by scoring the frequency of each feature for each patient for each moment in time. We can however not simply feed the model only the days on which a patient visited the GP. Each patient has a medical history of five years, but patients do not visit the GP daily. The LSTM model expects fixed-length input sequences, while the sequences of data points for all patients are characterized by irregular sampling, and do not go back the full five years for some patients.

A possible solution is to pad the missing data points with empty feature vectors as described in literature that discusses the use types of recurrent neural networks in tasks concerning clinical data (see Lipton et al. 2016 for example), but this would lead to very sparse data, and very long sequences of data (5 years of data leads to more than 1,800 time steps), which greatly increases the model architecture and run time of the LSTM model. Alternatively, we chose to aggregate the data over periods of thirty days to create a timeline, to solve the issue of irregular sampling. Figure 11 illustrates a time line for a hypothetical patient that passed on May 19th, 2014. The date of death is time *t*, and the medical history of each patient spans from *t* to *t* - 60. The white dots on the timeline refer to documented activities in the EMR.

Each time step is represented by a feature vector. When a feature is present for a patient at a certain moment in time, the score for that feature increases with one for the one-month interval during which the feature occurred. Each patient is therefore

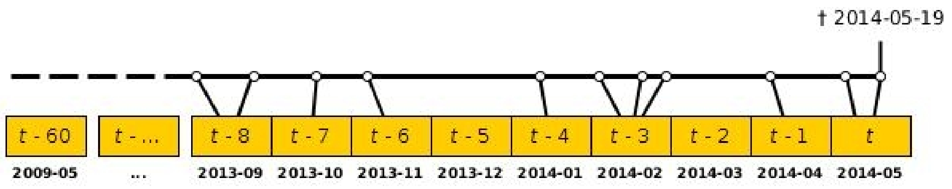


Figure 11: Timeline with data aggregated over intervals of thirty days.

represented by a medical history of 61 feature vectors (i.e. one feature vector for each month) which contain frequency counts for each feature that occurred during that month. If more than one consult takes place during a thirty-day period, the corresponding features are all mapped to the feature vector of that month (as illustrated by time step *t* - 3 in Figure 11).

Aggregating the data reduces the length of the entire timeline from over 1800 time steps to 61 time steps for a five-year period, and delivers more ‘dense’ feature vectors because multiple consults may take place in one month. Moreover, it reduces the need for padding to a great extent: occasionally no activity whatsoever takes place during a month, but in most cases the GP consults with the patient or with other health care providers at least once a month. Finally, medical episodes generally span a few weeks or months, which fits this level of aggregation well.

6.1.2 Sliding window

For all patients in the EMR corpus, the date of death is known. This is essential for training purposes: we can only validate the model’s predictions if we know the correct output. For each patient, a life expectancy is calculated for each time step. We cannot feed the entire medical history to the model at once when training the model, because nothing would be learned: the correct life expectancy would always be 0 months. Rather, we want to train the model to learn to predict the life expectancy for any given moment in time.

Ideally, we would feed the model all data up to the moment in time for which we want to predict the life expectancy, to provide the model with the maximal amount of information. We would need to divide the medical history in all possible sub-sequences, and use padding to create sequences of equal lengths, as shown in Figure 12.

Although this approach is theoretically viable, it has one major drawback: instead of learning which types of information are indicators of the approaching end of life, the model reaches a near perfect accuracy simply by learning to count the amount of

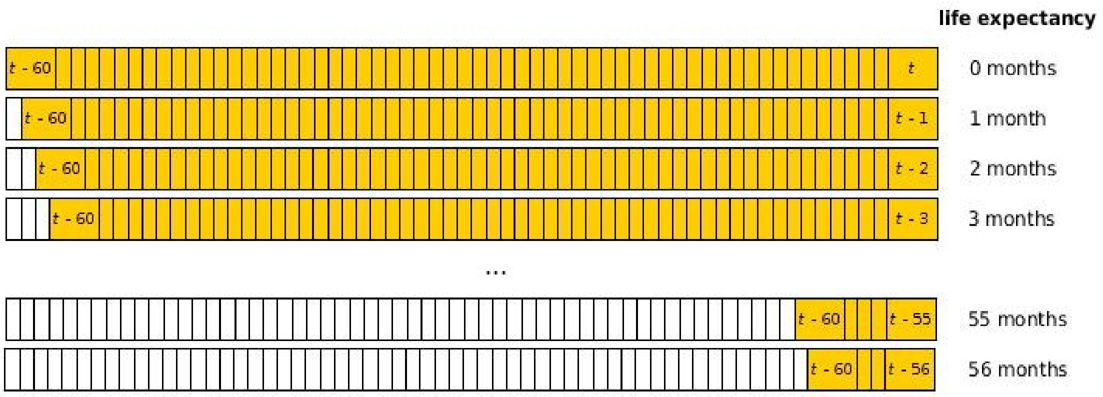


Figure 12: Division of medical history into sub-sequences by iteratively removing the most recent time step (from the right side of the sequence) and inserting padding at the start of the medical history (white boxes on the left side).

feature vectors used as padding. Alternatively, we use a sliding window to divide the complete medical history into sub-sequences of the history, as shown by Figure 13.

This way, when a window size of ten months is used, the patient data is divided into 50 subsequences of data, each of which differing one month from the preceding and following sequences in terms of life expectancy. The drawback of this approach compared to the previous approach, is that we can only use a fixed period of time as input for the model, thereby excluding potentially important cues that occurred further back in time. An overall advantage of dividing the complete medical history into such sub-sequences, was that the number of cases to train the model with expanded: each patient was transformed into fifty training examples, when using a window size of ten periods.

Finally, we exclude the month of death from the training data, for several reasons: 1) we are interested in predicting the moment of death in *advance*, 2) determination of the month of death is trivial due to the diagnostic codes: the diagnostic code for death is always applied in the month of death and never in a different month, and 3) the amount and character of the data in the month of death deviate significantly from the other data, due to many administrative events.

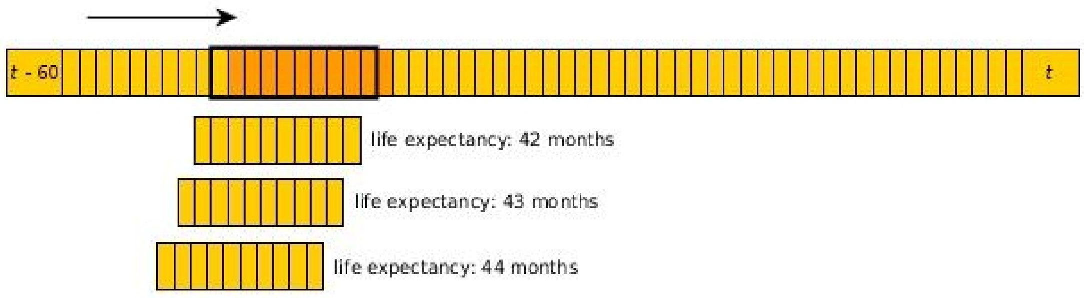

t - 60
t
life expectancy: 42 months
life expectancy: 43 months
life expectancy: 44 months

Figure 13: Window slides with steps of 1 along the 61-month sequence to create sub-sequences.

6.1.3 Normalization

The previous processing steps have led to the creation of input in the form of sequences of one-month time steps, each of which containing a feature vector that represents the frequency of occurrence of all features within that month. While these frequency counts reflect the medical history accurately, one additional processing step is needed to ensure that the input is processed properly by the LSTM model.

All layers of the LSTM model — input, hidden and output — are represented by *tensors*, or matrices, which are filled with rational numbers. When data flows through the model, it is transformed by operations such as multiplication: the input layer is processed by the model by multiplying it with the hidden layer, for example, and within the hidden cells the data are transformed by the input, output, and forget gates. The hidden layer is initialized with small random values, which get tuned during the training phase. The input layer is determined by our representation of the data.

Therefore, the raw frequency counts for features complicate the learning process. First of all, the feature categories are different of nature. A raw frequency of 5 in one month may be considered high for a feature category such as diagnoses: 5 heart attacks in one month would be a very high value. For other categories, a raw frequency of 5 may be considered low, such as for textual data: 5 encounters of the word *hartaanval* ‘heart attack’ in one month would not be considered high, because important events are usually referred to multiple times per document, and in multiple documents in a short time span. The more prevalent feature categories would therefore dominate the less prevalent categories during the learning process, if we would use raw frequencies.

A related issue, is the fact that some feature categories contain a large range of possible values, such as age, which ranges from roughly 0-105, while other feature categories are for example characterized by dichotomous values: a patient is either a male, or not.

Finally, some patients or GPs may be more easily inclined to schedule consultations than others. A higher consultation frequency however does not necessarily indicate more significance or more urgency, just as long text documents (which automatically leads to higher raw word frequencies) are not necessarily more important than short text documents.

Returning to the workings of the LSTM model, using raw frequencies is thus problematic, because the model will overrepresent certain feature categories while underrepresenting others, and will learn that more data indicates more urgency. When different feature values that are offered to the model may range from very small to very large, this leads to exploding and vanishing gradients, which impedes correct adaptation of the weights and biases of the hidden layer.

We address these issues by normalizing the data. We normalize the data per feature category because the categories differ in nature, and we normalize the data per month for each specific patient to annul the effect of the number of consultations in a month and the length of text documents. The frequency counts for the features are compressed to values between 0 and 1 by dividing all feature values of a feature category within a month by the highest value, and the data which do not have a natural lower limit of zero (such as the WORD2VEC dimensions), are normalized to values between -1 and 1 by dividing all feature values of a feature category within a month by the highest *absolute* value.

6.2 Conceptual baseline model

The task presented to the LSTM model is to formulate a hypothesis about the life expectancy of a patient at a certain moment in time, given a sub-sequence of the patient's medical history up to that moment. The training data for the model consists of time-series data for each patient in combination with corresponding life expectancies. Figure 14 shows a simplified overview of the model architecture.

Figure 14 illustrates a simplified version of the architecture for each time step. For each time step, each hidden layer (represented by a purple circle) consists of multiple units. Figure 15 zooms in on the final two time steps in Figure 14, and shows that the model is in fact a fully connected sequence of small networks. Each time step that is preceded by a previous time step, receives input from the feature vector and from the hidden cells in the previous time step. Information from all

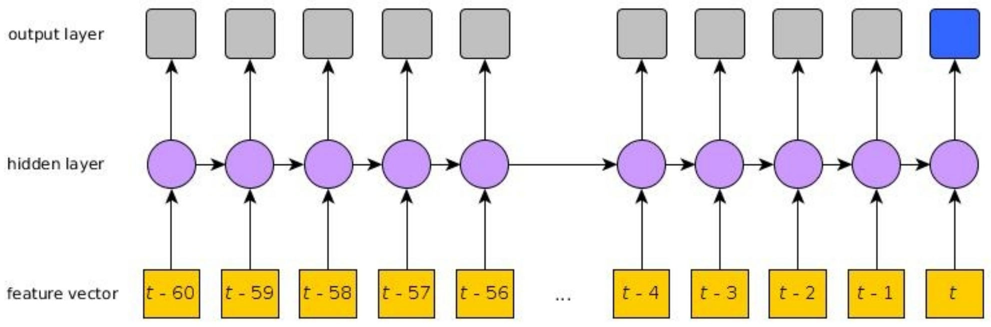


Figure 14: Simplified LSTM architecture. The yellow boxes represent a feature vector for each time step, the purple circles represent the hidden layer for each time step, the gray boxes represent the intermediate predicted life expectancies, and the blue box represents the output that is returned by the model: the life expectancy for time t in which we are interested.

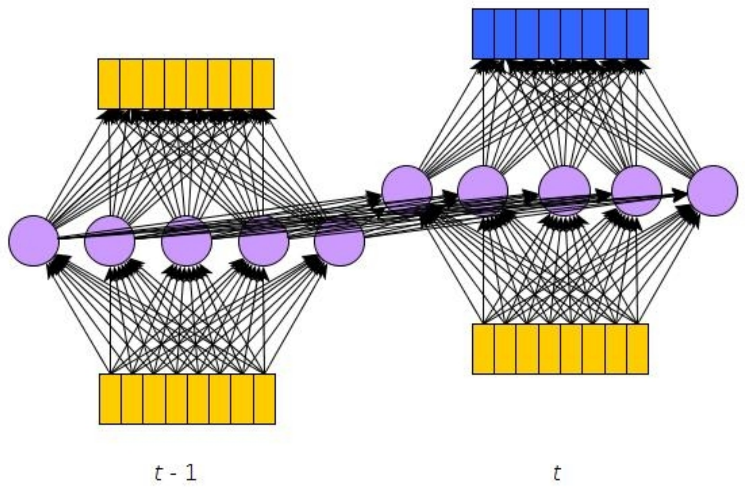


Figure 15: Detailed view of the final two time steps in Figure 14.

previous time steps in the sub-sequence is therefore employed by the model when the final prediction at time *t* is made.

Instead of a single-value prediction, we chose to model the probability that the end of life occurs at a certain moment in time by projecting life expectancy on a timeline with the aid of a softmax output layer. The output sequence represents a 61-month time line minus the size of the sliding window and minus the month of death. The model predicts the life expectancy by creating a probability distribution over time of the chance that the end of life will occur. The output sequence is transformed by a softmax function, to ensure that the probabilities for all months in the distribution together sum to 1. The result is illustrated by Figure 16, in which a predicted output sequence is plotted. The point for which the model predicts death with the highest relative certainty, can be interpreted as the point in time for which the model predicts the highest risk of dying for a patient. Based on the output in Figure 16, the model therefore predicts a life expectancy of ten months for the corresponding patient.

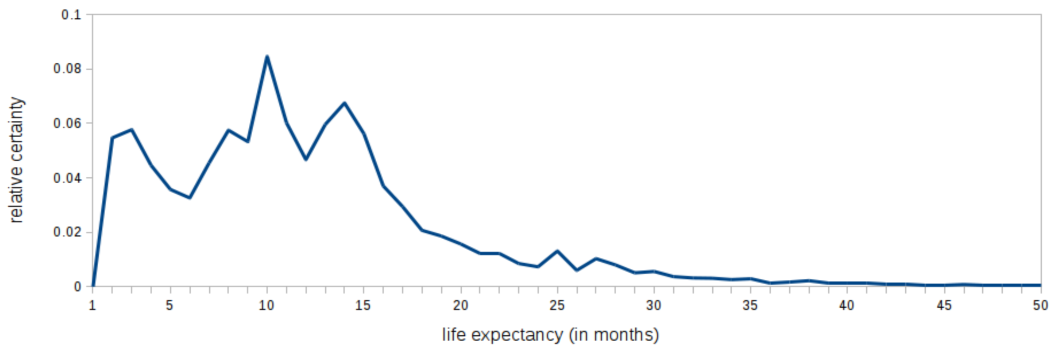


Figure 16: Example of a probability distribution created by the model to formulate a life expectancy.

6.3 Model optimization

The previous section described the conceptual model architecture. The model’s actual architecture was decided based on a series of experiments that were carried out in order to determine the optimal hyperparameter settings for the model, and the optimal sub-set of features. The set of included features and the model’s architecture are interdependent, and both affect the model’s performance. Varying the composition of the feature set affects the size of the feature set, and the size of the feature set directly influences the required model complexity.

Ideally, a grid search including all different combinations of feature categories, feature representations and model hyperparameters should be performed. Unfortunately, this leads to a combinatorial explosion — it would require several thousands of experiments to find both the optimal feature set and determine the optimal model architecture. Additionally, many hyperparameters are continuous scales with no clear boundaries, which complicates a full grid search further.

Alternatively, we explored the interaction between the feature set and the model architecture through a series of consecutive grid search steps, focusing on smaller grid searches. Each step builds on the results of the previous step.

To guide the grid searches, an indication of the model’s performance was reached through exhaustive ten-fold cross-validation. The development data (note: the validation data has been separated from the development data before Phase I already) was split randomly into ten non-overlapping parts that one by one functioned as a test set for each round of validation. During each round, the model was trained on the remaining 90% of the development data.

6.3.1 Tuning hyperparameters for the baseline model

We used a fixed feature set when searching for the optimal hyperparameter settings. To obtain this set, the following reduction methods were applied: for all features, the infrequent features were removed with the absolute frequency cut-off method, and the aggregation level for the ICD and ICPC codes was set to the code level. The keyword features were excluded from the feature set entirely, in order to create a baseline against which the effect of adding the keywords could be tested in a later stadium.

We explored the influence of the following parameters: batch size, learning rate, training epochs, model size in terms of number of hidden layers and hidden units per layer, peepholes, and dropout. Each parameter was initially set to a default value,

Table 13: default values during the process of hyperparameter optimization.

Hyperparameters	**Default value**	**Experimental settings**
Learning rate	--[6]	0.0001, 0.001, 0.01, 0.1, 1.0
Batch size	--[6]	1, 5, 10
Number of hidden units per layer	30	50, 75, 100
Number of hidden layers	1	1, 2
Peephole connections	not applied	not applied, applied
Dropout	not applied	0.4, 0.6, 0.8
Epochs	10	early stopping
Window size (in months)	10	6, 12, 18, 24

and adjusted when their effect on the model performance was tested specifically. When the optimal parameter setting improved upon the default setting, the default setting was replaced by the new value for the experiments that followed. Table 13 shows the tuned parameters, and the default and experimental settings per parameter.

6.3.1.1 Learning rate

Research carried out to explore the hyperparameter space for neural networks, has shown that the optimal parameter settings depend on the specific task at hand, and that some hyperparameters can be tuned independently of the other parameters, thereby decreasing the search space. Through an extensive study, Greff et al. (2016) show that the most important parameters for model performance are learning rate and model size, and that they can be tuned independently. In their experiments, the model size did not influence the optimal learning rate. They suggest therefore, that the learning rate can be tuned with a fairly small network before exploring the optimal model architecture in detail, which saves a lot of experimentation time. The first step in optimizing the hyperparameters, was therefore to experiment with the following learning rates: 10^{-5} - 1.0 in ten percent increments. The results showed that learning rates in the range 10^{-4} - 1.0 were too large. While the model performs increasingly well on the training data, the testing error only increases over time, as illustrated by Figure 17.

This was not the case for learning rate 10^{-5}. Figure 18 shows that for this learning rate, the training and testing error decrease over time, although very slowly. Compared to Figure 17, the gap between training and testing error has greatly decreased, and the rate of change for both training and testing error is much less explosive. Instead of opting for smaller learning rates which would significantly

6 This parameter has no default value, because it was among the first parameters to be tuned.

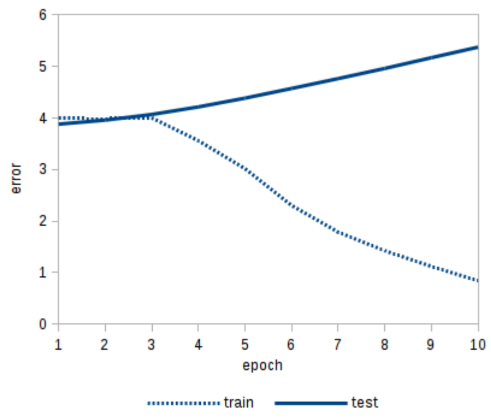


Figure 17: Train and test error at learning rate 10^{-4}.

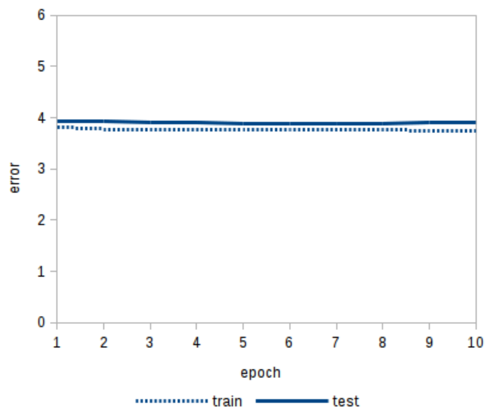


Figure 18: Train and test error at learning rate 10^{-5}.

increase the run time, we experimented with other variables to further lower the error on the testing data.

6.3.1.2 Batch size

Updating the weights of the hidden units in a neural network — i.e., the process of learning — can be done after each training example (stochastic gradient descent), in small or large batches, or after seeing the entire training set (full gradient descent). Batches are used to approximate the gradient of all training data, and thus batch size influences the training process. Large batches result in less weight updates, and are used with a larger learning rate, which may result in more computational speed per epoch. Larger batches however require more memory, and generally require more training epochs.

Small batches or stochastic learning require less training epochs because the weights are updated more frequently. Because small batches represent the entire training set worse than large batches, they result in noisier gradient updates, or in other words: they may lead the model into the wrong direction. Small batches however require a small learning rate, therefore taking a step in the wrong direction is less harmful than when larger batches and a larger learning rate are used — in fact, using small batches may even provide a regularization effect, thereby reducing overfitting.

Breuel (2015), who explored the hyperparameter space of neural networks as well, illustrates that optimizing the hyperparameters is more difficult as the batch size increases, because the range of learning rates which deliver good results decreases for larger batch sizes. Furthermore, he shows that larger batches generally decrease the model's performance. Based on these observations, two small batch sizes were tested during this research. We tested the effect of stochastic learning and

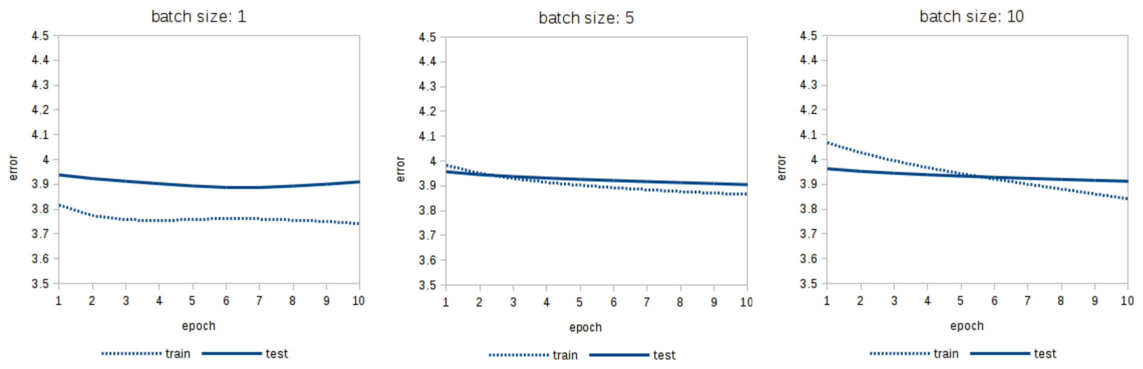


Figure 19: Train and test error for different batch sizes.

learning with minibatches of sizes 5 and 10. The effect of the batch size is shown by Figure 19.

The test errors for the larger batch sizes continue to decrease slightly over time, in contrast to the test error for stochastic learning. At a batch size of 10, the training error decreases at a faster rate than the test error, compared to the training error at a batch size of five. Because of the small gap between training and testing error for the batch size of five, we regarded this batch size to be optimal.

6.3.1.3 Model size: number of layers and hidden units

After the learning rate and batch size were optimized, we experimented with the model size. The default settings for the model size were chosen in order to create a model of medium complexity as a starting point. We used one layer with 50 hidden units as the default size for the previous experiments. The risk of *overfitting* increases along with the model complexity. Models which are too complex for the task impede the learning process: instead of detecting patterns and learning meaningful relations, the model simply learns to 'memorize' the data, resulting in poor generalization to new data. Overfitting can therefore be recognized by increasingly good performance on the training data, while performing increasingly bad on the testing data (Goodfellow et al. 2016).

If the model complexity is too low for the task however, the model will suffer from *underfitting*: it does not have enough capacity to capture complex relations between different types of information and will fail to learn which features are important indicators for the approaching end of life. In other words: the model is unable to reach a sufficiently low error on the training data (Goodfellow et al. 2016). When determining the model architecture, we want the model to 1) achieve a small training error to prevent underfitting, and 2) a small gap between training and testing error to prevent overfitting (Goodfellow et al. 2016).

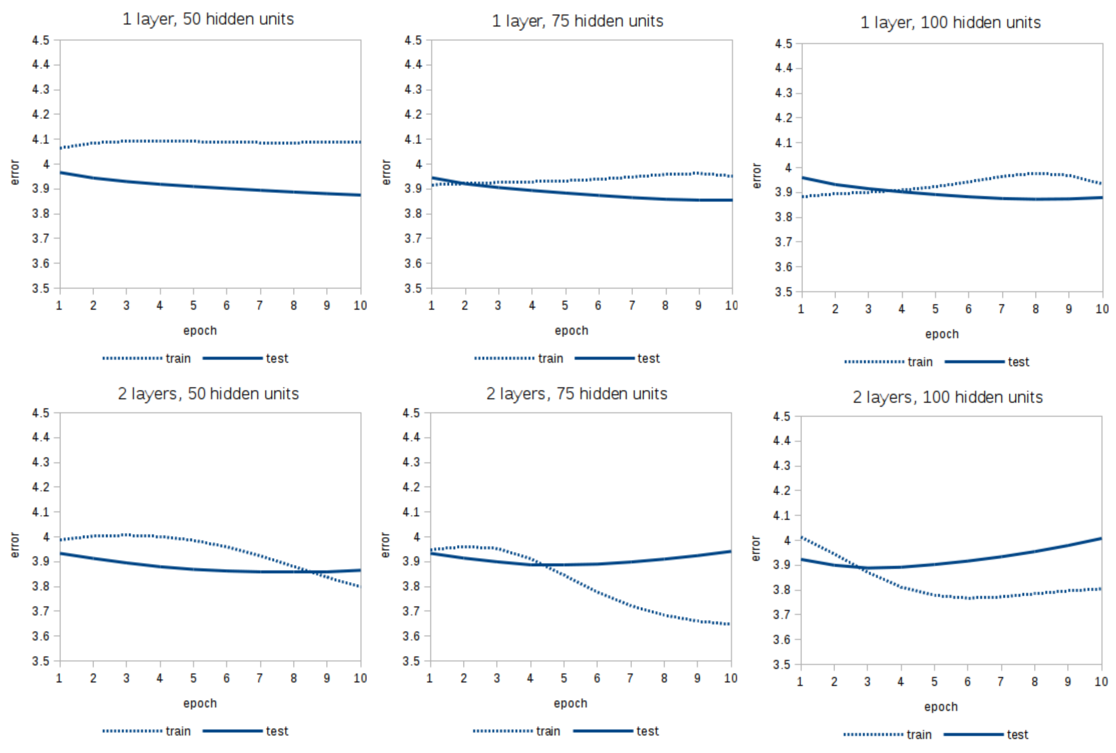


Figure 20: Effect of model size on training and testing error for each epoch. The training and testing error are represented by the dotted and the continuous line, respectively.

We experimented with 1 and 2 hidden layers, and with 50, 75, 100 and 125 hidden units per layer. The results are shown in Figure 20.

The training error fluctuates along with the model size. The model size does not impact the test error much, except for the largest models, in which the test error increases with each epoch. On average, the two-layer models perform better on the training data than the one-layer models, which show relatively large training errors, and therefore underfit the data. We conclude therefore, that one-layer models may not be sufficient for this complicated task, even when they consist of a relatively large number of cells. The two-layer models perform better on the training data, but the larger models suffer from overfitting: while the training error decreases rapidly, the testing error remains stable. The best performing models are the model with 1 layer of 100 hidden units, and the model with 2 layers of 50 hidden units. We preferred the two-layer model over the one-layer model in order to enable the exploration of the effect of using dropout.

6.3.1.4 Dropout

Complicated tasks ask for complicated networks, but such networks tend to be slow in use, hard to optimize, and run a high risk of overfitting. Multiple strategies exist to deal with the problem of overfitting. Solutions include the use of more training data, cross-validation, regularization, early stopping, and dropout. The best way to regularize the process is to combine the output of several models into a final prediction for each test case (Hinton et al. 2012), but this is computationally expensive both during training and testing, and requires a lot of run time, especially because we work with cross-validation. Therefore, this solution is sub-optimal for reducing the effect of overfitting in our task.

An alternative approach which *reduces* the run time instead of increasing it by large, while still approximating the concept of employing multiple differently trained models, is the use of dropout. Dropout refers to the process of randomly ‘dropping’ hidden units from the model during training. For each new training case, different units are dropped, thereby training many different ‘thinned out’ networks (Srivastava et al. 2014). Dropout prevents different hidden units to adapt to each others’ behavior, which has been shown to reduce overfitting in a range of tasks.

The use of dropout in recurrent networks however leads to varying results and is under discussion. Bayer et al. (2013) explain that the technique of dropout is unsuitable for recurrent networks, due to its negative impact on the recurrence by amplifying noise. Pham et al. (2013) and Zaremba et al. (2015) however demonstrated that dropout can in fact be successfully applied to recurrent networks such as LSTMs, by only applying it to the non-recurrent connections. Alternatively phrased, dropout may provide benefits for recurrent networks when it is only applied between *layers*, and not between *time steps*.

To test the effect of dropout in our model, we applied dropout only between the hidden layers and not between time steps, and experimented with 40%, 50%, and 60% dropout rates. The resulting models were applied to the test data *without* omitting any cells. The results are shown in Figure 21.

Other than expected, the different dropout rates do not seem to affect the error for the test data, but only affect the error for the training data, thereby *increasing* rather than *decreasing* overfitting, compared to the performance shown by the model to which no dropout was applied. One could argue that the use of dropout is beneficial even though it has no effect on testing error, because it decreases the run time slightly. We decided not to use dropout throughout the following experiments however, because we favor a simpler model above a slight decrease in run time.

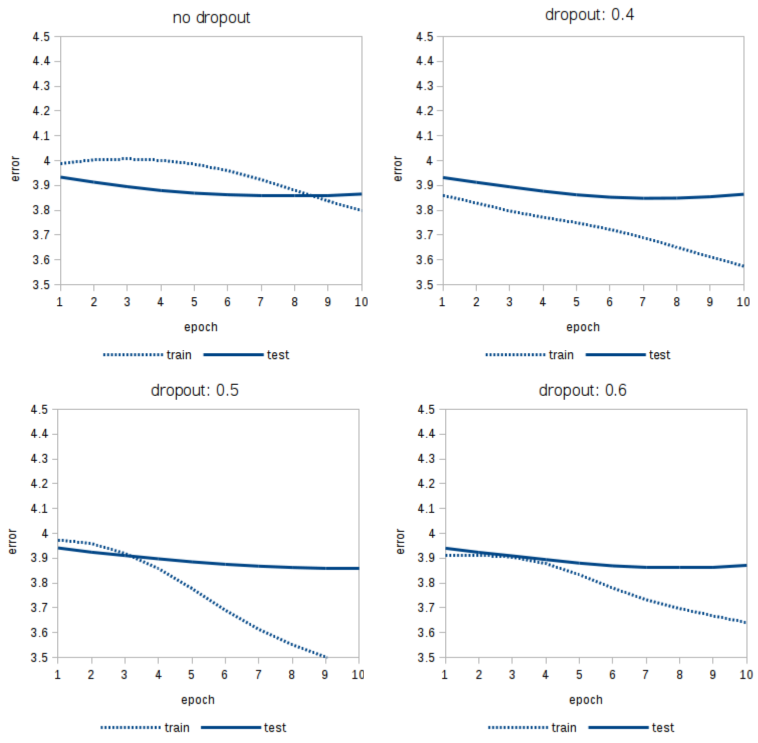


Figure 21: Effect of different dropout rates on the training and testing error. The percentages indicate the proportion of hidden cells that were omitted randomly for each new training case.

Hypothetically speaking however, we recommend further exploration of the use of dropout and its effect on a more extensive learning process if one were to use our model in a context that has stronger run time requirements.

6.3.1.5 Peepholes

Finally, we experimented with peephole connections. Peephole connections allow hidden cells to control their multiplicative input, output and forget gates. The implementation of peepholes has been shown to be beneficial in tasks that involve precise timing, because they improve a network's ability to learn from the time intervals between events (Gers et al. 2002). Although the implementation of peepholes may be helpful for certain timing-related tasks, research has shown that peepholes generally do not improve the error rate significantly (Breuel 2015, Greff et al. 2015). We tested the model with and without peephole connections. The result is illustrated in Figure 22.

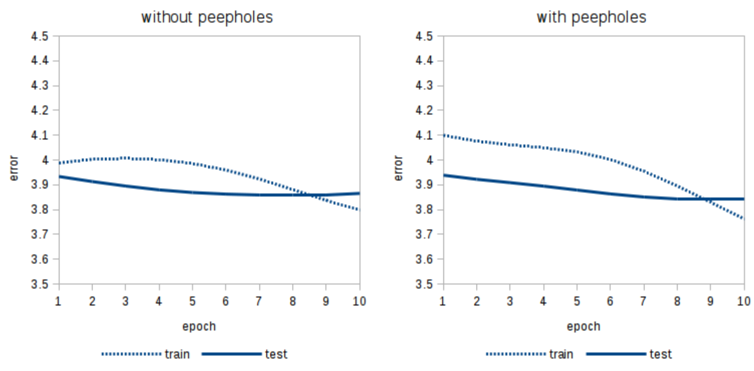


Figure 22: Effect of peephole connections on the training and testing error.

As the figure shows, peephole connections do not affect the error on the testing data much. It does affect the training error negatively initially, but after ten epochs this effect was negligible, compared to the training error for the model without the peephole connections. Peephole connections are therefore not useful for our task: they hardly — and only negatively — impact the results, while increasing the run time significantly. We therefore decided not to implement peephole connections in our final model.

6.3.2 Optimizing feature selection for the baseline model

In order to determine the optimal feature set, we tested several combinations of feature categories and the effect of different feature reduction methods. We first determined the representation which yielded the best results for each feature category individually, with regards to the different cut-off methods (no frequency cut-off, and absolute, relative, and relative cumulative frequency cut-offs) and level of abstraction for all codes (code, subcategory, main category, and component level). The absolute frequency cut-off performed best for each individual feature category.

The codes for diagnoses and reason for encounter, and the ICD-10 codes performed optimally when they were set to the code level. The codes for medical history and intervention performed best when they were aggregated to the sub-category level. For consultation type, medication and lab results no further aggregation was possible. Reducing the feature set by applying the absolute frequency cut-off boundary and aggregating over medical history and intervention codes, reduced the complete feature set, which includes 4,649 features, with 20% to a set of 931 features (including age and gender).

To determine whether any of the feature categories is redundant, we ran the model for each feature category separately. The single feature category that yielded the best results was used as a starting point, to which other feature categories were added one by one. The feature category that increased the model's performance most, was added to the feature set (as long it did not *de*crease the performance), after which the value of all remaining feature categories was determined again. The optimal feature set included all of the feature categories. The order in which the feature categories were added to the feature set, was: diagnose, medication, ICD-10, reason for encounter, lab results, intervention, medical history, consultation type.

6.3.3 Baseline model

The resulting model was used as a baseline model to compare the effect of adding textual features to. The window size of the baseline model was fixed to ten months; the effect of window size was determined after the final round of experiments during which we determined whether or not the inclusion of keywords improves the model.

After the keyword experiments, we tested the effect of window size for the baseline model, and for the keyword model which performed best. We approached the search for the optimal number of epochs similarly: after the keyword experiments, we optimized the number of epochs for the baseline model and for the keyword model which performed best. The effect of keywords is discussed in Section 6.4, and the effect of window size and the optimal number of epochs are discussed in Section 6.5.

6.4 Effect of adding textual features

The final experiment we conducted in order to determine the optimal model in terms of features, concerned the addition of keyword features. We compared the baseline model to models that include frequency-based, entropy-based and WORD2VEC-based features. We experimented with the addition of 100, 200 and 300 keyword features for each method. The model included a total of 931 non-keyword features. Adding up to 300 keywords therefore is a significant extension of the input layer, which may require a larger model. Therefore, we varied the size of the hidden layer (50, 100 or 200 cells): to get a rough indication of whether or not the model size should be fine tuned again after adding keyword features.

To enable comparison between the performance of the baseline model and the models to which keyword features were added, we needed a way to transform the probability distribution into a final prediction.

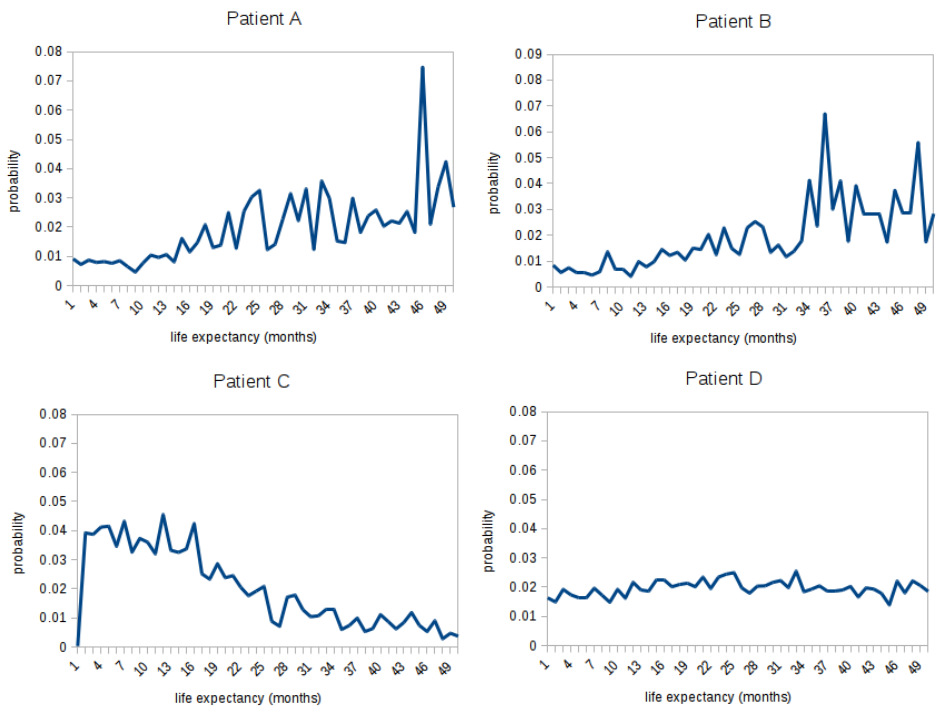


Figure 23: Examples of probability distributions produced by the baseline model.

Figure 23 shows four examples outputs produced by the baseline model. As the figure shows, it is not always obvious which criterion we should use to extract a prediction from the output. The probability distribution for Patient A shows *one* clear peak, enabling straightforward deduction of a life expectancy prediction. The distribution for Patient B however shows *two* peaks, while the distribution for Patient C points towards a longer period of heightened risk, and the distribution for patient D shows no clear high-risk period at all. Although many alternative approaches may be suitable for the deduction of a life expectancy from the probability distributions, we operationalized the distributions by taking the month with the highest probability for dying as the final prediction.

We compared the baseline model to the other models in terms of the root mean square deviation between the predicted and the actual life expectancy, and the mean deviation of the predicted life expectancy compared to the actual life expectancy. The results are shown in Tables 14 and 15.

The mean deviations between actual and predicted life expectancy are lower for all models including keywords, compared to the baseline. Interestingly, while the models (including the baseline model) on average tend to overestimate life expectancy, the models that include WORD2VEC features show the opposite pattern: they underestimate life expectancy. This is interesting from the perspective of ACP. Discussing ACP too late has serious consequences, while discussing it too early is unproblematic, as long as the discussion is repeated. To guarantee timely recognition of the palliative phase therefore, we rather want to underestimate life expectancy and closely monitor the situation, than overestimate life expectancy and postponing the conversation until it is too late.

Table 14: Deviation in months between actual life expectancy and model's predictions for the baseline model.

Root mean square	Mean deviation
17.6	6.4

Table 15: Deviation in months between actual life expectancy and predictions by different models which include keyword features. The models differ from each other in terms of selection method, model size, and number of included keywords. The best models are defined by two criteria: 1) having a relatively low root mean square, followed by 2) a relatively low mean deviation. The results corresponding to the best models based on these criteria, are marked with boldface.

Selection method	Hidden units	Root mean square			Mean deviation		
		100 words	200 words	300 words	100 words	200 words	300 words
Frequency	50	17.6	17.2	17.0	4.5	5.0	5.8
	100	17.5	17.4	**16.9**	2.1	1.2	**1.7**
	200	17.7	17.8	17.8	1.6	1.3	1.0
Entropy	50	17.4	17.8	17.8	5.1	5.6	5.4
	100	17.2	**16.9**	17.8	2.5	**2.3**	1.6
	200	17.7	17.5	17.7	2.3	2.0	1.3
WORD2VEC	50	**17.8**	18.2	18.2	**-3.4**	-4.3	-3.7
	100	18.1	17.8	17.8	-4.2	-4.1	-4.8
	200	18.3	18.3	18.4	-3.75	-4.4	-4.4

In order to determine the best model to optimize further, we looked into the results of the baseline model and the best performing model of each selection method in more detail. We selected the best models based on a relatively low root

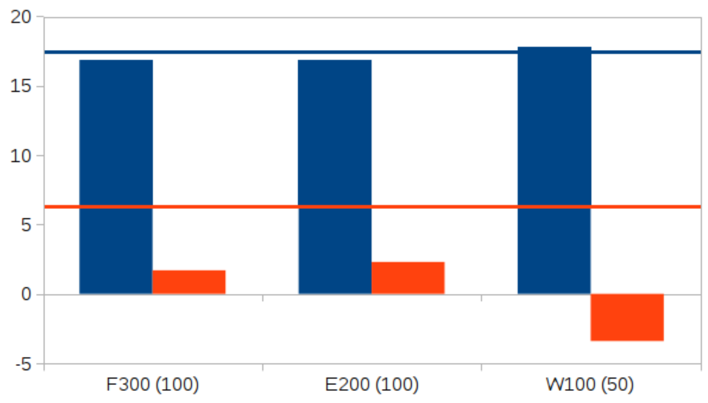


Figure 24: Root mean square deviance (blue) and mean deviance (red) of models with keyword features versus the baseline (indicated by the blue and red lines) for the best performing models per keyword selection method. In the labels on the *x* axis, the letter refers to the selection mechanism (F: frequency, E: entropy, W: WORD2VEC), the first number refers to the number of included keywords or dimensions, and the second number refers to the number of hidden cells per layer.

mean square as first criterion, and a relatively low mean deviation as second criterion. The best models were the models including 300 frequency-based keywords, 200 entropy-based keywords, and 100 WORD2VEC dimensions (results of these models are marked with boldface in Table 15). The results of the baseline model and the best performing models are visualized in Figure 24.

Figure 24 highlights the difference between the pessimistic character of the predictions provided by the WORD2VEC model (illustrated by the negative mean for the WORD2VEC model) and the optimistic character of the predictions for all other models (illustrated by the positive means for all other models). The root mean square and mean deviations are helpful to compare the models, but they do not offer a nuanced representation of the output.

In a large-scale, systematic review of literature on life expectancy prediction by doctors, White et al. (2016) evaluated a total of 42 research papers, in which a total of more than 12,000 prognoses were evaluated. To get a better indication of the quality of our model's predictions, we identified the research from the pool used by White et al. that was best comparable to our case, according to the following criteria: 1) the research was carried out anno 2000 or later to avoid outdated research, 2) prognoses were estimated on a continuous scale, 3) the doctors were no experts in palliative care, and 4) the patient group for which the predictions were made was not disease-specific.

The research carried out by Christakis and Lamont (2000) satisfied these criteria. They analyzed the prognostic accuracy of 343 doctors for 468 patients. The authors considered a prediction to be accurate if it fell within a window of 33% around the actual moment of death. According to their results, the predictions were accurate for 20% of the patients, overly optimistic in 63% of the cases, and overly pessimistic in 17% of the cases.

For the baseline model and the three best performing models that include keyword features, we evaluated the quality of the predictions with an approach identical to Christakis and Lamont (2000: 469-470): we divided the actual life expectancy by the predicted life expectancy, and regarded a prognosis as accurate if the quotient was a value between 0.67 and 1.33. Quotients smaller than 0.67 signify overly optimistic errors, while values larger than 1.33 signify overly pessimistic errors. Table 16 shows an overview of the quality of the doctors' predictions as adopted from Christakis and Lamont (2000: 470), predictions by the baseline model, and predictions by the three models that include keyword features.

Table 16: Evaluation of the quality of the predictions. Predictions are considered accurate if they deviate less than 33% from the actual life expectancy.

Assessor	**Accuracy**	**Overly pessimistic**	**Overly optimistic**
Human	20%	17%	63%
Baseline model	23%	58%	20%
Frequency-selected keywords	29%	27%	44%
Entropy-selected keywords	28%	46%	27%
WORD2VEC representation	38%	32%	31%

As the results indicate, the baseline model outperforms the doctors' estimates. The models that include keyword features further enhance the performance compared to the baseline, especially the model that includes the WORD2VEC embeddings. While doctors estimates are often overly optimistic, which harms early identification of palliative patients, the baseline and keyword models' predictions are rather pessimistic.

Interestingly, the amount of overly pessimistic and overly optimistic estimations does not correspond to what we expected to find: as Figure 24 shows, the baseline model and the models with frequency-selected and entropy-selected keywords tended to *over*estimate the life expectancy on average, while Table 16 shows that most of the inaccurate estimations of the baseline and entropy model are rather pessimistic. The incorrect predictions of the WORD2VEC model are divided over roughly equal numbers of overly optimistic and pessimistic predictions, where we expected a larger percentage of pessimistic predictions. Table 17 shows the ratio of predicted to actual life expectancy for the overly optimistic and overly pessimistic predictions[7]. We were unable to deduce these values for human assessors from Christakis and Lamont 2000.

Table 17: Ratio of predicted to actual life expectancy for the inaccurate predictions.

Assessor	**Overly pessimistic**	**Overly optimistic**
Baseline model	5.4	0.4
Frequency-selected keywords	4.9	0.4
Entropy-selected keywords	4.7	0.4
WORD2VEC representation	6.0	0.4

Taken together, the results from Tables 16 and 17 illustrate that the WORD2VEC model produced a larger amount of accurate predictions, but when it makes overly pessimistic predictions, it is *more* pessimistic than the other models (which explains the larger root mean square for the WORD2VEC model compared to the other models, as shown in Table 15). Because of its high accuracy compared to the other keyword models and its tendency toward relatively pessimistic estimations, we favored the model which included WORD2VEC-based features over the other keyword models. The next section describes the final steps we took to optimize the model performance.

6.5 Final parameter tuning

We finally tuned the number of epochs and the window size for the baseline and the WORD2VEC model. While the settings of these model parameters do influence the results, retaining their default values throughout the experiments did not hinder direct comparison between the different models, because these parameters are independent from the other parameters.

6.5.1 Epochs

We applied early stopping to determine the optimal number of epochs and to prevent overfitting. Training continued for a maximum of 100 epochs, or until no

7 Note that the ratio for overly pessimistic predictions is always larger than 1, while the ratio for overly optimistic predictions is always a value between 0-1. This results from the fact that pessimistic predictions are by definition smaller than the actual value, while optimistic predictions are always larger (and we divided the actual by the predicted life expectancy in both cases).

improvement was seen on the test data for 5 epochs. The optimal number of epochs for both the baseline and the WORD2VEC model turned out to be identical to the default value we used throughout the experiments: 10 epochs. Therefore, tuning the number of epochs did not lead to a change in accuracy.

6.5.2 Window size

Finally, we explored the effect of window size. The window size determines the number of time steps, or one-month periods over which data was aggregated, that are used as input for the model at once. We expected that increasing the window size would increase the accuracy, because the model has more data to base a prediction on. The window size was varied between values of 6 up to 24 months, in 6-month increments.

The output layer of the model has a fixed length of 61 months minus the final month *minus the window size*, as described in Section 6.1.2. The model learns the minimum and maximum output values during the training process, and never predicts larger values than 61 months minus the final month minus the window size, because it never encounters larger values. The size of the output layer decreases as the window size increases, and therefore the chance of predicting the life expectancy correctly increases automatically. Because we are not interested in a measure of accuracy that is influenced by these chance effects, we did not use the root mean square and mean difference between the actual and predicted life expectancy, but only evaluated the results according to the approach of Christakis and Lamont (2000: 469-470). The results for the baseline and the WORD2VEC model are displayed in Tables 18 and 19, respectively.

While the baseline model behaved as expected, the WORD2VEC model did not. The accuracy of the baseline model increased along with the window size, and while the vast majority of the incorrect predictions remains to be overly pessimistic, a shift from overly pessimistic to more overly optimistic predictions can be observed.

Table 18: Results per window size for the baseline model.

Window size	Accuracy	Overly pessimistic	Overly optimistic
6 months	20%	64%	16%
10 months (default)	23%	58%	20%
12 months	25%	55%	21%
18 months	26%	51%	24%
24 months	28%	46%	26%

Table 19: Results per window size for the WORD2VEC model.

Window size	Accuracy	Overly pessimistic	Overly optimistic
6 months	37%	32%	30%
10 months (default)	38%	32%	31%
12 months	37%	32%	31%
18 months	36%	32%	31%
24 months	34%	35%	30%

The accuracy of the WORD2VEC model is inversely proportional to the window size: as the window size increases, the model accuracy decreases, while the ratio of overly optimistic to overly pessimistic predictions remains roughly equal.

Additionally, the model's accuracy could have been increased or decreased in terms of the ratio of predicted to actual life expectancy. In other words, independent of the number of correct predictions, the model accuracy could have been improved by making predictions that are less 'off'. A closer look revealed however that the mean ratio of predicted to actual life expectancy remained steady for both the baseline and the WORD2VEC model, regardless of the window size.

Based on these results, for the baseline model it seems to be the case that the larger the window size is, the higher the accuracy is, while the accuracy for the WORD2VEC model reaches a peak when the default window size of 10 months is used. For our final model however, we decided to use window sizes of 10 months both for the baseline and the WORD2VEC model, for three reasons: 1) the run time of the model increases proportionally to the window size, 2) to maximize the comparability between both models during the validation test, and 3) a large window size decreases the size of the output layer, because the output layer has the size of 61 months minus the month of death *minus the window size*, due to the fixed amount of data (5 years) per patient. A smaller output layer technically means a higher chance of making a correct prediction.

6.6 Final model architectures

The final architectures of the baseline model and the WORD2VEC model are highly similar, and are illustrated by Figure 25. The models only differ with respect to the number of features that are fed to the model at each time step: the baseline model makes use of 931 features, while the WORD2VEC model includes 100 additional keyword features.

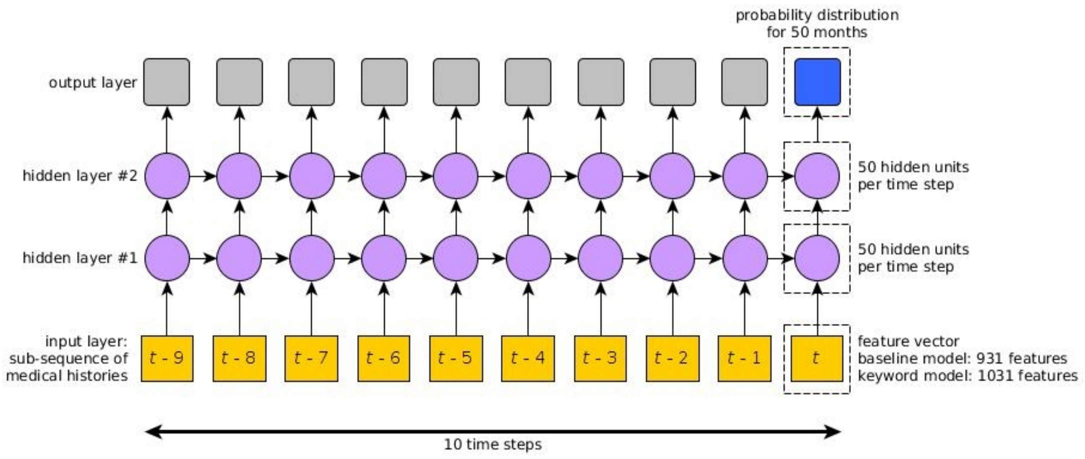

probability distribution for 50 months
output layer
hidden layer #2
50 hidden units per time step
hidden layer #1
50 hidden units per time step
input layer: sub-sequence of medical histories
t - 9
t - 8
t - 7
t - 6
t - 5
t - 4
t - 3
t - 2
t - 1
t
feature vector
baseline model: 931 features
keyword model: 1031 features
10 time steps

Figure 25: Final model architecture for the baseline and the WORD2VEC model.

For both models, the final architecture is a fully connected model consisting of one input layer, two hidden layers and an output layer, for each time step. The unrolled LSTM model consists of 10 time steps. For each time step, the input layer consists of a feature vector, and the hidden layers each contain 50 hidden units. The output layer for each time step consisted of 50 months (61 months minus the month of death, minus the window length), and is transformed by a softmax layer to create a probability distribution.

The weights are updated after each batch of five training cases, and the learning rate is 10^{-5}. No dropout or peephole connections are used. For the remaining parameters, we used standard settings: the weights were initialized randomly from a truncated normal distribution, we used a bias of 0.1, used AdamOptimizer to optimize the gradient descend procedure, and used cross-entropy to minimize the loss during the training process.

7 Phase IV Validation

7. PHASE IV: VALIDATION

7.1 General results

During Phase IV the optimal baseline and keyword model were validated with the held-out test set. Because the health care practices are not equally large, and to mimic real-life use of the model, we created a balanced test set by extracting the 10% most recent patients based on their date of death for each health care facility. The model was trained on all development data, which represented historical data, and was applied to the ‘new patients’. The results for the baseline model and the keyword model are presented in Table 20, along with the human baseline. Table 21 shows the ratio of over- and underestimation to actual life expectancy.

Table 20: Evaluation of the quality of the predictions. Predictions are considered accurate if they deviate less than 33% from the actual life expectancy.

Assessor	Accuracy	Overly pessimistic	Overly optimistic
Human	20%	17%	63%
Baseline model	20%	68%	12%
Keyword model	29%	52%	19%

Table 21: Mean ratio of predicted to actual life expectancy for the inaccurate predictions.

Assessor	Overly pessimistic	Overly optimistic
Baseline model	6.3	0.4
Keyword model	5.9	0.4

Compared to the results presented in Table 16, the models’ accuracy for the held-out validation set dropped: -3% for the baseline model and -9% for the keyword model. During Phase III the baseline model performed slightly better than the human standard, while the validation test shows that the performance of the baseline model equals human performance. We extracted features from the *entire* development set in Phase II, and applied cross-validation in Phase III, therefore the model was fine tuned on all possible features in the development data. The held-out test set however contained data that the model has not encountered before, and while this did not seem to affect the accuracy on the baseline model much, it notably affected the performance of the keyword model.

7.2 Relations between several output measures

We further analyzed the results of the best-performing model — the keyword model — in terms of Pearson's product-movement correlation coefficients. We expected to find a positive correlation between the actual and the predicted life expectancy. Additionally, we expected the model's certainty to both increase as the *actual* moment of death approached, and as the *predicted* moment of death approached. We therefore expected to find negative correlations between the relative certainty of the predictions, and both the actual and predicted life expectancies. Finally, we expected to find a higher level of certainty for predictions that are close to the actual life expectancies. Therefore, we expected the relation between the number of months between actual and predicted life expectancy, and the certainty of the predictions to be inversely proportional to each other. The tests, hypotheses, and results of the calculations are summarized in Table 22.

Table 22: Results for correlation calculations between several outcome measures.

Tested relations	**Hypotheses**	**Pearson's *r***	**Significance *p***
Actual vs. predicted life exp.	positive relation	.36	<.05
Certainty vs. actual life exp.	negative relation	-.35	<.05
Certainty vs. predicted life exp.	negative relation	-.61	<.05
Certainty vs. absolute difference between actual and predicted life exp.	negative relation	-.02	.12

As Table 22 shows, the calculations confirmed most of the hypotheses. The results show a moderately positive relation between the model's predictions and the actual life expectancy. The histogram in Figure 26 shows frequency counts of actual and predicted life expectancies. The actual life expectancies are evenly distributed: because the medical histories are divided in ten-month windows, every month in the range 1-50 is predicted 127 times, corresponding to the 127 test patients. The predictions are not as evenly distributed as the actual expectancies: the model shows a tendency to predict that death is either nearby or far away in time.

The results in Table 22 also show that the model is more certain about predictions in the near future, than predictions further away. Figures 27a and 27b show the model's certainty as a function of the actual (27a) and predicted (27b) life expectancy. The figures display the model's tendency to be more certain about short term life expectancies than about predictions that lay further away in time. The certainty of the model is however not a good indicator of the model's accuracy, as shown by the final test results in Table 22. Our expectation about a higher model certainty for more accurate predictions, was not reflected by the results.

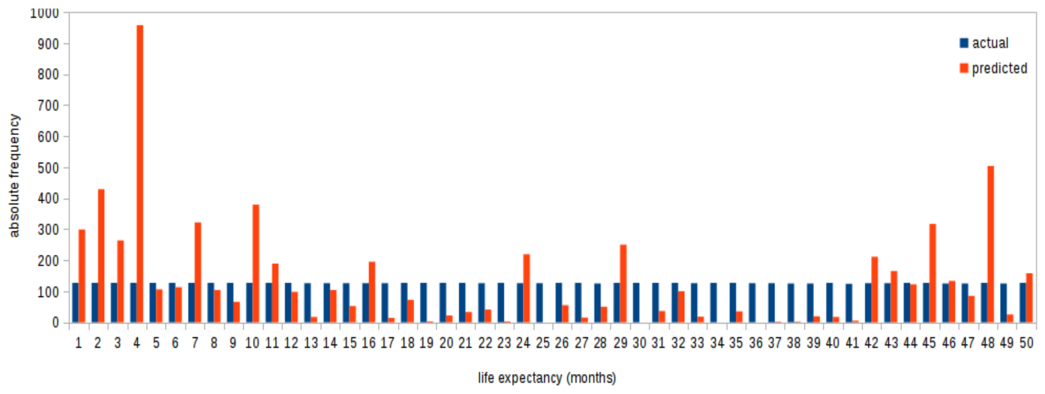
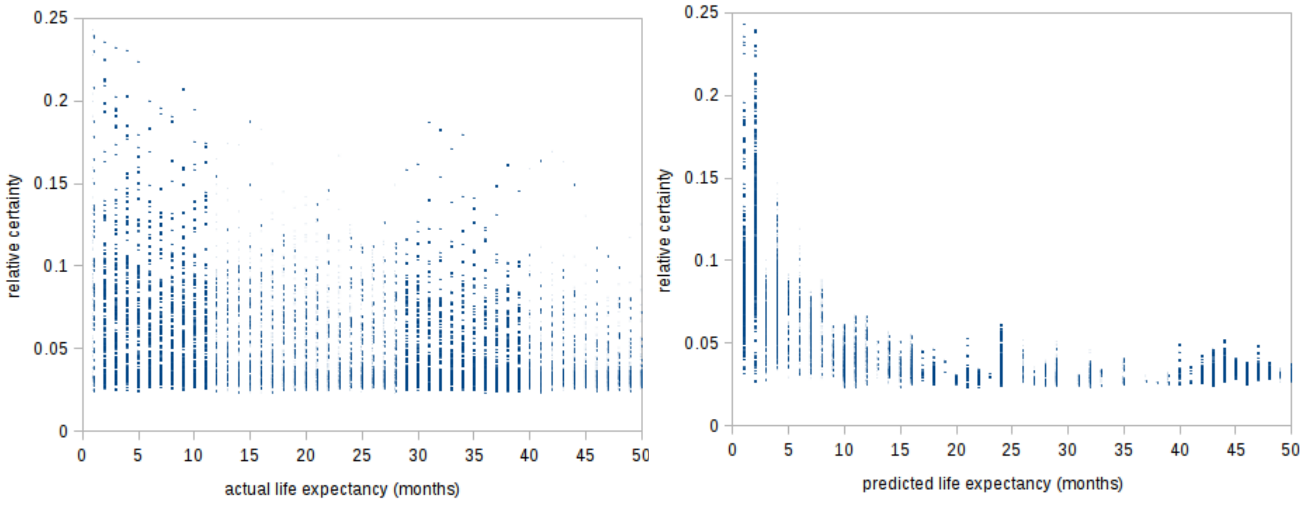


Figure 26: Histogram with absolute frequency counts for actual and predicted life expectancies, for each month in the range 1-50.



a *b*

Figure 27: Relative certainty versus actual (a) and predicted (b) life expectancy.

7.3 Results per illness trajectory

Finally, we disentangled the keyword model’s results for the different illness trajectories. To ensure clear-cut boundaries between illness trajectories, we separated the output predictions into three categories, according to the three illness trajectories as described in Section 3.1: a group of patients suffering *only* from cancer, a group of patients suffering *only* from organ failure, and a group of patients which can *only* be categorized as frail elderly.

In our analysis for each trajectory, we included the predictions for the ten-month windows of each patient from the onset of the disease that associates the patient with a specific trajectory, onward. We expected the model to perform better for the cancer trajectory than for the other trajectories and for the validation set as a whole, because of the steady and predictable decline associated with the cancer trajectory.

The analysis however showed unexpected results: the model performed worse for *each* of the three trajectories than for the validation dataset as a whole: the model scored between 17-20% accuracy for each of the three trajectories (compared to 29% accuracy for the entire validation set). This means that the model performed better for patients that cannot be categorized by these trajectories at all, and for patients that fall into several categories at once.

The model performed similarly for cancer patients and organ failure patients, with roughly two thirds of the incorrect predictions being overly pessimistic. For the frail elderly however, the largest part of the incorrect predictions was overly optimistic. Therefore, as we expected, there is a difference between the illness trajectories. Contrary to our expectations however, the trajectory that stands out most is not the cancer trajectory, but rather the frail elderly category.

These results may have been influenced by any of the following factors: 1) other trajectories that were not specified in this research are characterized by a more predictable pattern of degradation, 2) no clear definition of illness trajectories exists in literature, so our operationalization may have been flawed, 3) our model is trained and tuned to perform well *overall*, therefore the model's performance for specific diseases, illness trajectories, or any other categorization of patients, is bound to deliver worse results than the overall results. The results for different illness trajectories however show that the predictions for the cancer and the organ failure trajectories show similar tendencies as the performance for the validation data overall: the predictions tend to be overly pessimistic. In contrast, the performance of the model for the frail elderly category is notably different from the model's performance on the other trajectories and on the validation data as a whole: most of the incorrect predictions *overshoot* the life expectancy for the frail elderly. These results suggest that different training and fine tuning procedures may be optimal for different categories of patients.

8 Discussion

and ideas for further research

8. DISCUSSION AND IDEAS FOR FUTURE RESEARCH

The results of this research show that LSTMs are able to parallel the accuracy of doctors in predicting life expectancy, and show that the inclusion of features from unstructured clinical texts advances the quality of the predictions further.

8.1 Comparison to human prognostic estimates

The inclusion-exclusion criteria that were used by White et al. (2016) in the systematic review of doctors' estimates of life expectancy prohibited perfect comparison between doctors' performance and our model's performance: patients had to be defined as terminally ill, palliative, or otherwise non-curative for a research to be included, and research papers which included patients that were receiving artificial ventilation or that were admitted to an Intensive Care Unit, were excluded.

We, on the other hand, did not exclude any patients from the dataset. In contrast to the task presented to doctors, the model was therefore additionally employed to make predictions for patients that were still curative, and patients that received life support. Life expectancy is harder to predict for such patients, and the comparison between the doctors' performance and our model's performance was therefore bound to result in a pessimistic view of our model's accuracy. Regardless of the relatively pessimistic view on the results, the model already outperformed doctors, which leaves us optimistic about the capabilities of the model compared to doctors' prognoses when presented with identical data.

8.2 Amount of data

An issue which challenged the processing of all data types, was the amount of data available to us in this research. Our corpus consisted of roughly 1,200 patients which is a fair amount of data according to clinical standards, but is not considered to be a lot of data for training neural networks. As illustrated by the optimization process in Phase III and the drop in accuracy on the validation data in Phase IV, overfitting is a serious issue which we did not fully manage to tackle, even though we maximized the amount of training data, used cross-validation and early stopping, and explored the effects of drop-out. We expect that the use of more data in future research will aid in a better feature selection process, especially with regards to the textual features, and will help the model to generalize better to unseen cases.

8.3 Disease-specific training

An additional benefit of gathering more data is the potential benefit of creating disease-specific training sets. The results show that, contrary to what we expected, the model performs worse for specific categories of patients based on illness trajectories, than for the group of patients overall. The analyses per illness category did show however that the model handles different illness trajectories in different ways, being overly optimistic for some patients, while being overly pessimistic for others.

We expect therefore, that it may be beneficial to make predictions for a patient based on patients with similar characteristics. We could not explore this idea because we only had access to a limited amount of data, and division into several categories would further decrease the amount of training data. If enough data would be present to divide the dataset into separate categories however, it would be very interesting to explore additional disease-specific training of the model.

8.4 Vector space representation for structured data

One of the reasons for overfitting is the high dimensionality of the input data. Our input feature vectors for each time step contained roughly 1000 features. The more features the model needs to learn, the higher the requirements are for model complexity and amount of training cases. It is therefore important to minimize the size of the feature set, while retaining as much important information as possible.

Although we encountered many unique medical codes, medication names and lab codes in the EMR corpus, which required a feature selection process to reduce the dimensionality, textual data is even *less* restricted than structured data. Representing the textual data with a limited set of keyword features received much attention in this research due to the large amount of unique words textual data delivers.

A vector space representation proved to be an effective way of retaining as much information as possible, while keeping the feature set small. Because the nature of natural language data is vastly different from the nature of structured data in terms of unique items and noise, we did not consider transferring our approach for textual data to the processing of the structured data. In hindsight however, we think that a vector space representation would have been highly suitable for representing medical codes and medication names as well.

Compared to the approach we used, vector space modeling supersedes the need to disregard possibly informative features. Related research carried out by Nguyen et al. (2016) shows that the technique of using word embeddings to represent structured EMR data as dense vectors delivers a feasible representation for training neural networks. In future research, we would like to experiment with word embedding techniques to represent the structured data as well.

8.5 Learning from intermediate predictions

An approach to fight overfitting that we did not explore in this research, but will in the future, is the use of intermediate predictions: LSTMs are large, linked networks that contain a neural network for each time step, and therefore have an input layer, hidden layer(s) and an output layer for each time step. While we make active use of the input layer and hidden layers at each time step, we only used the output of the final time step to base the final prediction on. In future work, we will experiment with training the model on the intermediate outputs as well. This idea has been explored in related research by Lipton et al. (2016) as well with promising results.

8.6 Combining predictions through time

A different approach to fight overfitting, is inspired by the technique of combining the outputs of multiple trained networks into a single prediction. While this technique is known to increase the accuracy of the final predictions, it is very computationally expensive. In this research we used dropout instead to simulate the use of many small networks, but dropout did not work well in our case. What we did not explore in this research, but which we will in future research, is not to combine the output of many models to predict the output for one case, but to use our single model to make a predictions for a patient by combining predictions of the model *through time*.

This approach addresses a different issue simultaneously: our dissatisfaction of using a sliding window to overcome issues that are typically associated with the use of clinical time series data, such as irregular sampling, different amounts of data per patient, and data sequences of unequal length. To overcome these issues, we aggregated the data over one-month periods and used a sliding window to create sequences of equal lengths. The latter was unsatisfactory however, because historical data that falls outside of the window is completely ignored by our model, while we expect it to contain useful information. Making predictions for a patient at a certain moment in time by combining outputs from several windows further back in history therefore has two benefits: it enables the combination of multiple predictions into one which is expected to fight overfitting, and it enables making use of the full patient history.

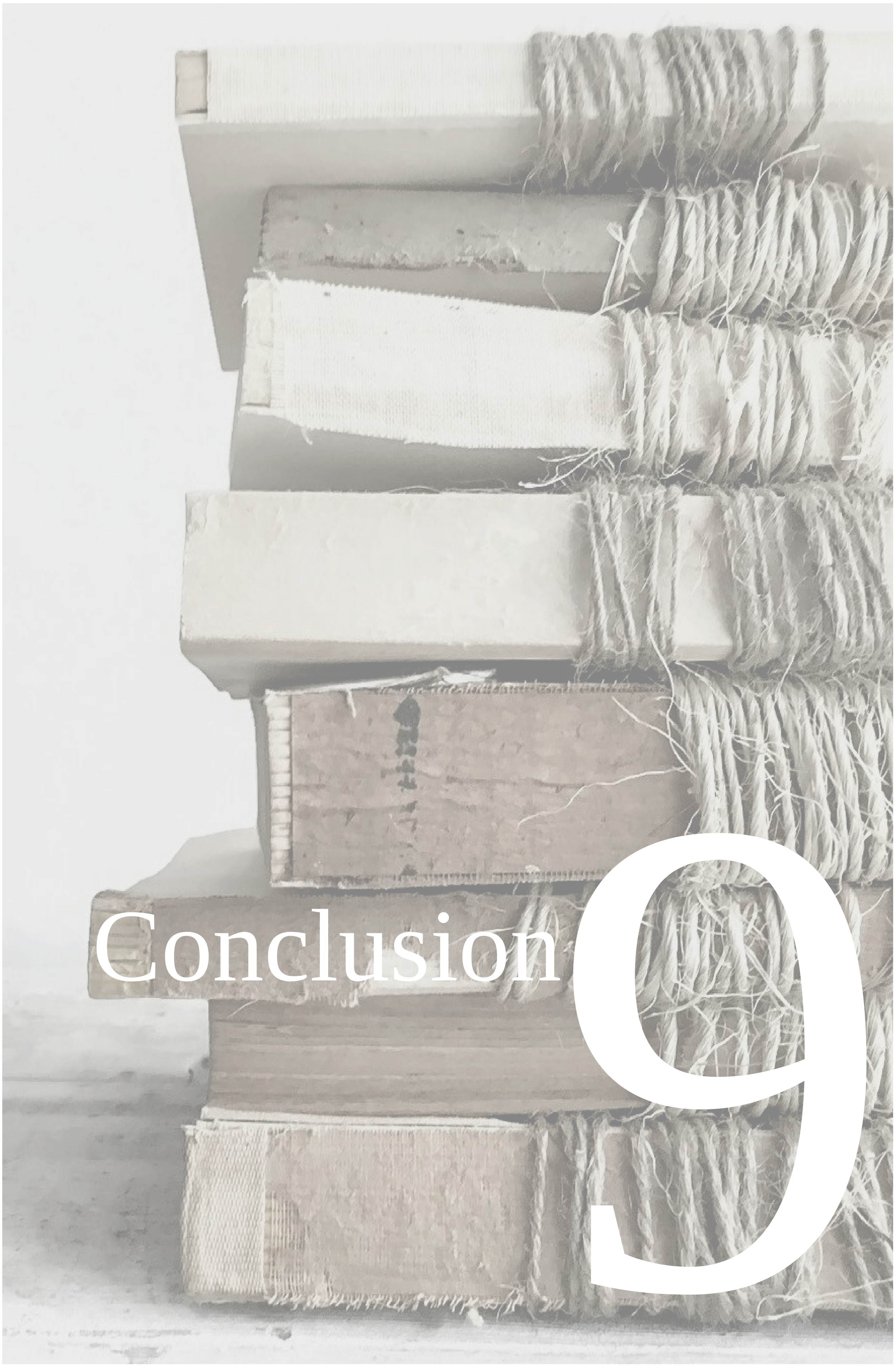
8.7 Textual data

The use of clinical narrative increased the model's ability to make correct predictions. The textual representations used in this research built on the assumption that documents are independent and self-contained units. The features we extracted from these documents were based on word counts and distributional properties of words. While such representations are sufficient for many machine learning tasks, potentially important information is lost by ignoring contextual embedding, intertextuality, and other types of dependencies between documents in the EMR corpus. We plan to exploit the highly informative nature of the clinical texts more in future work, by using more advanced NLP techniques to create a richer linguistic representation of the textual data.

8.8 Interpretation of the output

Next to data-related and processing-related ideas to improve the accuracy of the predictions in future work, different ways of interpreting the model's output may lead to more constructive predictions additionally. Instead of letting the model return one value as output, we wanted the model to return more informative output: a probability distribution for a large range of months. As output therefore, the model provides a prediction for each month in the range that states the probability that the patient is to die during that specific month.

While this method delivers very interesting results, as illustrated by Figure 22 for example, we also needed a way to operationalize these probability distributions in order to evaluate the model's performance. In this research, we used the month with the highest probability for dying as the final prediction. However, this is just one of many possible approaches for interpreting the model's output. Alternative methods include reporting the first, the last, or any peak above a certain probability threshold, and reporting sudden changes in life expectancy, among other approaches. Determining which measurement corresponds best to the actual life expectancy fell outside the scope of this research, but we recommend thorough exploration of output metrics for future work.



Conclusion 9

9. CONCLUSION

With this research we aimed to explore the feasibility of automatically predicting life expectancy based on electronic medical records with machine learning and natural language processing techniques. Life expectancy is a leading indicator when making decisions about end-of-life care, and prognoses influence the process of Advance Care Planning negatively when done inaccurately. End-of-life prognostication is a notoriously challenging, very time-consuming, highly subjective, and simply unpleasant task for most general practitioners. Being overly optimistic about life expectancy, as doctors tend to be, greatly impedes the early identification of palliative patients and thereby delays appropriate care in the final phase of a patient's life.

This research aimed to address these issues. Predicting life expectancy has not been attempted — let alone been solved successfully — before with techniques from the fields of machine learning and natural language processing. Due to the many differences between human assessment and computer processing, we expected machine learning techniques to have the potential to make predictions automatically, and to increase accuracy and objectivity of prognostication, to aid in early recognition of patients at risk. This research aimed to answer to following questions:

1. To what extent are self-learning algorithms trained on medical records able to detect the approaching end of life automatically?
2. To what extent does the inclusion of textual data improve a prognostic model for detecting the approaching end of a patient's life?

We aspired to approach the golden standard for prognostication as reported for doctors in literature. Additionally, we strived to increase the accuracy of the predictions by including linguistic features that were extracted from clinical notes written by the general practitioner, and letters between the general practitioner and other health care specialists.

The findings agree with and even advance the golden standard. Using identical evaluation criteria as were used to evaluate doctors' performance, our baseline model reached a level of accuracy similar to human accuracy. Our keyword model improves the prediction accuracy with 9%, compared to our baseline model and to the golden standard of human evaluation. Moreover, our model tends to make rather pessimistic predictions, while doctors tend to do the opposite. Pessimistic predictions as provided by our model, in contrast to optimistic predictions as provided by doctors, could aid in the early recognition of the palliative phase and timely discussion of ACP strategies, to ease the transition toward end-of-life care.

Our approach showed promising results. We consider it to be the first step in an interesting and important line of research, that has tremendous potential for real-life applications. While clinicians are challenged by the ever increasing amounts of data that are available for each patient, machine learning techniques are able to use the growing stack of big data to their advantage. Overall, we have shown that machine learning and natural language processing techniques are not only a *feasible*, but also a very *promising* approach for automatic prognostication. With this thesis, we hoped to contribute to this exciting line of research, to advance our understanding of what is needed for automatic processing of medical data, and to explore the use of highly unstructured clinical texts.

10 References

10. References

Abdi, H. & L.J. Williams (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics 2*(4): 433-459.

Addington-Hall, J.M., L.D. MacDonald & H.R. Anderson (1990). Can the Spitzer Quality of Life Index help to reduce prognostic uncertainty in terminal care?. *British journal of cancer 62*(4): 695.

Al-Shayea, Q.K. (2011). Artificial neural networks in medical diagnosis. *International Journal of Computer Science Issues 8*(2): 150-154.

Billings, J.A., & R. Bernacki (2014). Strategic targeting of advance care planning interventions: the Goldilocks phenomenon. *JAMA 174*(4): 620-624.

Bosch, A. van den, G.J. Busser, W. Daelemans & S. Canisius (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. F. van Eynde, P. Dirix, I. Schuurman & V. Vandeghinste (eds.), *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting*: 99-114 Leuven, Belgium.

Breuel, T.M. (2015). The effects of hyperparameters on SGD training of neural networks. *arXiv preprint arXiv:1508.02788*.

Brinkman-Stoppelenburg, A., J.A. Rietjens & A. van der Heide (2014). The effects of advance care planning on end-of-life care: a systematic review. *Palliative Medicine 28*(8): 1000-1025.

Burke, H.B., P.H. Goodman, D.B. Rosen, D.E. Henson, J.N. Weinstein et al. (1997). Artificial neural networks improve the accuracy of cancer survival prediction. *Cancer 79*(4): 857-862.

Christakis N.A. & J.J. Escarce (1996). Survival of Medicare patients after enrolment in hospice programs. *New England Journal of Medication 335*: 172-178.

Christakis, N.A., J.L. Smith, C.M. Parkes & E.B. Lamont (2000). Extent and determinants of error in doctors' prognoses in terminally ill patients: prospective cohort study. *British Medical Journal 320*(7233): 469-473.

Claessen, S., A. Francke, Y. Engels (2013). How do GPs identify a need for palliative care in their patients? An interview study. *BMC Family Practice 14*: 1471-2296.

Dietterich, T.G. (2002). Machine learning for sequential data: A review. *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*: 15-30. Springer Berlin Heidelberg.

Forster, L.E. & J. Lynn (1988). Predicting Life Span for Applicants to Inpatient Hospice. *Archives of internal medicine 148*(12): 2540-2543.

Frankl, D., R.K. Oye, & P.E. Bellamy (1989). Attitudes of hospitalized patients toward life support: a survey of 200 medical inpatients. *American Journal of Medicine 86*: 645-648.

Gers, F.A., N.N. Schraudolph & J. Schmidhuber (2002). Learning precise timing with LSTM recurrent networks. *Journal of machine learning research 3*: 115-143.

Goodfellow, I., Y. Bengio & A. Courville (2016). *Deep learning*. MIT press.

Gott, M., C. Gardiner, N. Small, S. Payne, D. Seamark et al. (2009). Barriers to advance care planning in chronic obstructive pulmonary disease. *Palliative Medicine 23*(7): 642-648.

Greff, K., R,K, Srivastava, J. Koutník, B.R. Steunebrink & J. Schmidhuber (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

Groenewoud, S. (2015). Advance care planning. Het levenseinde teruggeven aan de mensen. *Kwaliteit in zorg 3*: 14–18. Online available via : http://www.netwerk-palliatievezorg.nl.

Hazan, H., D. Hilu, L. Manevitz, L.O. Ramig & S. Sapir (2012). Early diagnosis of Parkinson's disease via machine learning on speech data. *Electrical & Electronics Engineers in Israel*: 1-4.

Heyse-Moore, L.H. & V.E. Johnson-Bell (1987). Can doctors accurately predict the life expectancy of patients with terminal cancer? *Palliative medicine 1*(2): 165-166.

Higginson, I.J. & M. Costantini (2002). Accuracy of prognosis estimates by four palliative care teams: a prospective cohort study. *BMC palliative care 1*(1)*:* 1.

Highet G, D. Crawford, S.A. Murray & K. Boyd (2013). Development and evaluation of the Supportive and Palliative Care Indicators Tool (SPICT): a mixed-methods study. *British Medical Journal of Supportive Palliative Care*.

Hinton, G.E., N. Srivastava, A. Krizhevsky, I. Sutskever & R.R. Salakhutdinov (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. & J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9*(8): 1735–1780.

Integraal Kankercentrum Nederland (2015). Jaarverslag hospices (REPAL): bijna gelijk aantal opnameverzoeken in 2015. Online available via iknl.nl.

Jagannatha, A.N & H. Yu (2016). Bidirectional RNN for medical event detection in electronic health records. *Proceedings of the conference. Association for*

Computational Linguistics. North American Chapter. Meeting: 473. NIH Public Access.

Jensen, P.B., L.J. Jensen & S. Brunak (2012). Mining electronic health records: towards better research applications and clinical care. *Nature reviews. Genetics 13*(6): 395.

Karnofsky D.A. & J.H. Burchenal (1949). The clinical evaluation of chemotherapeutic agents in cancer. C.M. MacLeod (ed.) *Evaluation of Chemotherapeutic Agents: 191–205.* New York: Columbia University Press.

Kenter, T., A. Borisov & M. de Rijke (2016). Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Khan, J., J.S. Wei, M. Ringner, H. Saal, M, Ladanyi et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine* 7(6): 673-679.

Khemphila, A. & V. Boonjing (2011). Heart disease classification using neural network and feature selection. *Systems Engineering*: 406-409.

Kim, H.G., G.J. Jang, H.J. Choi, M. Kim, Y. Kim et al. (2016). Medical examination data prediction using simple recurrent network and long short-term memory. *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory*: 26-34. ACM.

Kordylewski, H., D. Graupe & K. Liu (2001). A novel large-memory neural network as an aid in medical diagnosis applications. *Transactions on Information Technology in Biomedicine 5*(3): 202-209.

Levy W.C., D. Mozaffarian, D.T. Linker, S.C. Sutradhar, S.D. Anker et al. (2006). The Seattle Heart Failure Model: prediction of survival in heart failure. *Circulation 113*(11): 1424-1433.

Lipton, Z.C., D.C. Kale, C. Elkan & R. Wetzell (2015). Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677*.

Liu, C., H. Sun, N. Du, S. Tan, H. Fei et al. (2016). Augmented LSTM Framework to Construct Medical Self-diagnosis Android. *16th International conference on Data Mining*: 1-10.

Lunney J.R., J. Lynn, D.S. Foley, S. Lipson & J.M. Guralnik (2003). Patterns of functional decline at the end of life. *JAMA 289*: 2387-2432.

Lynn, J., J.M. Teno & F.E. Harrell jr. (1995). Accurate prognostications of death. Opportunities and challenges for clinicians. *Western Journal of Medicine 163*(3): 250.

Maaten, L.V.D. & G. Hinton (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*: 2579-2605.

Mazurowski, M.A., P.A. Habas, J.M. Zurada, J.Y. Lo, J.A. Baker & G.D. Tourassi (2008). Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks 21*(2): 427-436.

Mikolov, T., K. Chen, G. Corrado, & J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., I. Sutskever, K. Chen, G.S. Corrado & J. Dean (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*: 3111-3119.

Ministerie van Volksgezondheid (2015). Informatiekaart Palliatief Terminale Zorg (PTZ). Online available via www.ciz.nl.

Moss, A.H., J. Ganjoo, S. Sharma, J. Gansor, S. Senft et al. (2008). Utility of the "surprise" question to identify dialysis patients with high mortality. *Clinical Journal of the American Society of Nephrology 3*(5): 1379-1384.

Moss, A.H., J.R. Lunney, S. Culp, M. Auber, S. Kurian et al. (2010). Prognostic significance of the "surprise" question in cancer patients. *Journal of palliative medicine 13*(7): 837-840.

Murray, S.A., M. Kendall, K. Boyd & A. Sheikh (2005). Illness trajectories and palliative care. *International Perspectives on Public Health Palliative Care 30*: 2017-2019.

Murtagh F.E.M., M. Preston & I. Higginson (2004). Patterns of dying: palliative care for non-malignant disease. *Clinical Medicine 4*: 39-44.

Pham, T., T. Tran, D. Phung & S. Venkatesh (2017). DeepCare: A deep dynamic memory model for predictive medicine. *Pacific-Asia Conference on Knowledge Discovery and Data Mining:* 30-41. Springer International Publishing.

Robinson, L., C. Dickinson, C. Bamford, A. Clark, J. Hughes et al. (2013). A qualitative study: professionals' experiences of advance care planning in dementia and palliative care,'a good idea in theory but…'. *Palliative medicine 27*(5): 401-408.

Sadikin, M., M.I. Fanany & T. Basaruddin (2016). A New Data Representation Based on Training Data Characteristics to Extract Drug Name Entity in Medical Text. *Computational intelligence and neuroscience:* 6.

Schwarzer, G., W. Vach & M. Schumacher (2000). On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology. *Statistics in medicine 19*(4): 541-561.

Singer, P.A., G. Robertson & D.J. Roy (1996). Bioethics for clinicians: 6. Advance care planning. *CMAJ: Canadian Medical Association Journal 155*(12): 1689–1692.

Sudore, R.L., & T.R. Fried (2010). Redefining the “planning” in advance care planning: preparing for end-of-life decision making. *Annals of internal medicine 153*(4): 256.

Szmuilowicz, E., A. El-Jawahri, L. Chiappetta, M. Kamdar & S. Block (2010). Improving residents’ end-of-life communication skills with a short retreat: a randomized controlled trial. *Journal of Palliative Medicine 13*(4): 439-452.

Thangarasu, G. & P.D.D. Dominic (2014). Prediction of hidden knowledge from a clinical database using data mining techniques. *Computer and Information Sciences*: 1-5.

Thomas, K. (2012). Finding patients who may die: Electronic searching for patients with palliative care needs. *International Congress on Palliative Care.* McGill University.

Thoonsen, B., Y. Engels, E. van Rijswijk, S. Verhagen, C. van Weel et al. (2012). Early identification of palliative care patients in general practice: development of RADboud indicators for PAlliative Care Needs (RADPAC). *British Journal of General Practice 62*(602): 625-631.

Walasek, N. (unpublished). Medical entity extraction on Dutch forum data in the absence of labeled training data. Radboud University Nijmegen.

Walczak, S. (2005). Artificial neural network medical decision support tool: predicting transfusion requirements of ER patients. *IEEE Transactions on Information Technology in Biomedicine 9*(3): 468-474.

Weeks J.C., E.F. Cook, S.J. O’Day, L.M. Peterson, N. Wneger et al. (1998). Relationship between cancer patients’ predictions of prognosis and their treatment preferences. *JAMA 279*:1709-1714.

Wei, J.S., B.T. Greer, F. Westermann, S.M. Steinberg, C.G. Son et al. (2004). Prediction of clinical outcome using gene expression profiling and artificial neural networks for patients with neuroblastoma. *Cancer research 64*(19): 6883-6891.

Weiner, J.S. & S.A. Cole (2004). Three principles to improve clinician communication for advance care planning: overcoming emotional, cognitive, and skill barriers. *Journal of Palliative medicine 7*(6): 817-829.

White, N., F. Reid, A. Harris, P. Harries & P. Stone (2016). A Systematic Review of Predictions of Survival in Palliative Care: How Accurate Are Clinicians and Who Are the Experts? *Plos One 11*(8).

Wiesner R., E. Edwards, R. Freeman, R. Kim & P. Kamath (2003). United Network for Organ Sharing Liver Disease Severity Score Committee. Model for end-stage liver disease (MELD) and allocation of donor livers. *Gastroenterology 124*(1): 91-96.

Zaremba, W., I. Sutskever & O. Vinyals (2015). Recurrent neural network regularization. *ICLR*.