Universiteit
Leiden
The Netherlands

# Healthcare information system engineering: AI technologies and open source approaches

Shen, Z.

**Citation**
Shen, Z. (2025, December 3). *Healthcare information system engineering: AI technologies and open source approaches*. Retrieved from https://hdl.handle.net/1887/4284431

| | |
|---|---|
| Version: | Publisher's Version |
| License: | Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden |
| Downloaded from: | https://hdl.handle.net/1887/4284431 |

**Note:** To cite this publication please use the final published version (if applicable).

# Healthcare Information System Engineering: AI Technologies and Open Source Approaches

Zhengru Shen

# Healthcare Information System Engineering: AI Technologies and Open Source Approaches

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 3 december 2025
klokke 13:00 uur

door

Zhengru Shen
geboren te Anhui, China
in 1987

| | | |
|---|---|---|
| Promotores: | Prof.dr. M.R. Spruit | |
| | Prof.dr. S. Brinkkemper | Utrecht University |
| Promotiecommissie: | Prof.dr. M.M. Bonsangue | |
| | Prof.dr. S. Verberne | |
| | Prof.dr.ir. J.M.W. Visser | |
| | Prof.dr.ir. R.W. Helms | Open University |
| | Prof.dr. Y. Velegrakis | Utrecht University |

Copyright © 2025 Zhengru Shen. All rights reserved.

This thesis was accomplished with financial support from European Union's Horizon 2020 research and innovation programme under grant agreement No. 634238 (OPERAM).

# Contents

# Chapter 1

# Introduction

This dissertation focuses on the development of Healthcare Information Systems (HIS). The research designs and develops a number of HIS with various Machine Learning (ML) and Natural Language Processing (NLP) techniques that address a number of issues in healthcare. Further investigations are conducted on how to improve HIS engineering with Open Source methodology. All chapters together present an overview of HIS engineering in today's healthcare.

To provide a background of this dissertation, the introduction first presents the concepts of HIS engineering and various types of HIS related to this research in section 1.1. Then, Open Source Software and its application in HIS engineering are discussed, respectively in section 1.2 and section 1.3. Next, the main research question and related research questions are introduced, along with the main research methods used to investigate them in section 1.4 and section 1.5. Finally, the remaining chapters of this dissertation are outlined in section 1.6, and the outcomes listed in section 1.7.

## 1.1 Healthcare Information System Engineering

Healthcare engineering is defined as engineering involving all aspects of healthcare, from prevention to treatment. Its purpose is to improve human health and well-being through engineering approaches (Chyu et al., 2015). Chyu et al. (2015) lists Healthcare Information Systems (HIS) engineering as an important subject of healthcare engineering. A healthcare information system is defined as a system designed to manage healthcare data, including data collection, processing, reporting, and analytics of such data (Wager et al., 2021). These systems intend to provide better and timely

decision-making with respect to patient treatments and the provision of healthcare services (Fichman et al., 2011; Ljubicic et al., 2020). The scope of HIS engineering includes the following domains of interest: Electronic Health Record, E-Health, Data Mining and Big Data, M-Health, Telemedicine, Wireless Technology, and Information Security (Chyu et al., 2015). This dissertation addresses the first three domains in depth.

### 1.1.1 Electronic Health Records

As a prominent example of HIS, electronic health records (EHR) have moved beyond the simple digitization of healthcare data that are collected during routine delivery of health care. More and more resources are dedicated to utilize data in EHR to support clinical decision-making (Evans, 2016). For instance, the large amount of patient data stored in the EHR consists of a large variety of free-text documents, such as discharge summaries and medical notes. This spawned the need for the use of NLP to read, process and transform free-text data with standardized codes such as ATC and ICD10.

### 1.1.2 E-Health

An E-Health service refers to a service that uses information and communication technology to improve healthcare (Pagliari et al., 2005). EHR is often included as one of the common applications of E-Health. However, we consider it as a separate field because of its size and importance in the HIS domain. Blaya et al. (2010) presents a list of E-Health categories, including laboratory information management, pharmacy information systems, patient registration or scheduling, monitoring, evaluation, and patient tracking systems, clinical decision support systems, patient reminder systems and research data collection systems. In particular, clinical decision support systems (CDSSs) have received a lot of attention in recent years (Sutton et al., 2020). A CDSS supports clinicians in their complex decision-making processes with a combination of clinical knowledge, patient information, and other health information (Osheroff et al., 2012). CDSSs are often provided to clinicians as web or mobile applications.

### 1.1.3 Data Mining and Big Data

Big data and data mining are among the most important topics in today's IT world which are discussed and applied in almost all industries and research domains (Che

et al., 2013). Its importance in healthcare has grown significantly as a vast amount of the healthcare data is collected and stored (Yang et al., 2021). Data mining enables the discovery of hidden knowledge from big healthcare data so that healthcare professionals can use such knowledge to solve real-world problems. Applications that prove the benefits of data mining in healthcare can be found in a variety of healthcare research (Li et al., 2013; Menger et al., 2018a; Kavakiotis et al., 2017; Carchiolo et al., 2019; Albahri et al., 2020).

Besides the common structured data, healthcare data has a wide range of other data types, including semi-structured and unstructured data, such as images, texts, web pages and videos. Both researchers and practitioners in healthcare have developed or applied specific data mining techniques to cope with such data variability. Among them, NLP is an increasingly crucial component in facilitating healthcare services since textual data exists in almost all processes of healthcare, ranging from patient intake to discharge. Furthermore, a huge amount of healthcare literature has been accumulated over the decades, and the number is still growing on a daily basis. For example, in chapter 4 we investigate how to use NLP to encode patient health records, and in chapter 5 we extract knowledge from full-text literature.

### 1.1.4 Trends and Challenges

Ngafeeson (2015) discussed a number of important trends in the field of HIS. The scope of this dissertation focuses primarily on the following three trends:

**From Local to Global HIS:** With the development of Internet and information security, HIS has moved beyond a single hospital. There are many HIS that are built for national or international settings (e.g. chapter 3).

**From using patient data to all healthcare data:** Aside from patient data, a lot of other healthcare related data has been accumulated, such as healthcare literature. How to uncover knowledge from a huge amount of openly accessible full-text literature has gained more interest (e.g. chapter 6).

**From Numeric Data to More Complex Types:** As mentioned previously, there are a variety of data types of healthcare data, including text, images and videos. To fully unlock the potential of healthcare data, researchers are moving toward analyzing more complex types of healthcare data (e.g. chapter 5).

## 1.2 Open Source Software (OSS)

Open source software refers to software distributed with a license that guarantees free access to its source code, which allows users to freely redistribute the software, to create derived works, and to unrestrictedly use the software (Bretthauer, 2002). Its origin could be traced back to the 1950s. Decades of development have transformed OSS into a sophisticated engineering domain that has produced well-known open source projects such as Linux and Apache, among many others (Midha and Palvia, 2012). Therefore, OSS is an indispensable component of today's software industry. To facilitate the development of OSS, a number of IT tools have been created. Among them is GitHub, which hosts source codes of millions of OSS, and supports collaboration of software developers around the world (Shen and Spruit, 2019). Therefore, researchers have learnt more about OSS via studying open source projects hosted on GitHub (Kalliamvakou et al., 2015; Shen and Spruit, 2019).

Openly accessible OSS can benefit the IT development of both academia and industry in many ways. To begin with, the free accessibility of OSS allows us to develop our IT systems more efficiently by easily reusing existing tools or codes instead of creating everything from scratch. It, therefore, significantly reduces the IT development cost and time (McDonald et al., 2003). Secondly, the open source philosophy believes two heads are better than one, and a million heads can move mountains (Brabham, 2008). Therefore, OSS welcomes developers all around the world to contribute to the development and maintenance of software. This ensures its reliability and robustness due to the world-wide developer community. Lastly, the effective utilization of open-source software could significantly boost IT development in developing countries, especially in resource-poor areas where there are few IT resources (Shen et al., 2019a). Without OSS, developing countries might have difficulty building state of the art software, and their gap with developed nations would become wider and wider.

## 1.3 OSS in HIS Engineering

With the progress of healthcare IT in recent years, open source software has become an essential part of HIS engineering. More and more people in healthcare advocate OSS (de Abajo and Ballestero, 2012; Nolden et al., 2013; Shen and Spruit, 2019). Moreover, numerous open source software projects are created to solve healthcare problems (Athey et al., 2013; Hansen and Sørensen, 2013; Bankhead et al., 2017; Cavelaars et al., 2015; Gibson et al., 2018). Well-known and successful examples in-

clude openEHR (open Electronic Health Records), cTAKES (clinical Text Analysis and Knowledge Extraction System), and others. These projects have been used in multiple studies and some are even adopted in the real cinical settings. Table 1.1 shows more details about this selection of well-known open source healthcare software.

| OSS | Description | License | Year | Contributors | Dev Status |
|---|---|---|---|---|---|
| openEHR | As one of the most popular open source electronic health records and medical practice management solutions, it supports a variety of clinical practices. | GNU GPL | 2002 | 188 | Active |
| cTAKES | cTAKES is a NLP system for extraction of information from electronic medical record clinical free-text. | Apache License 2.0 | 2006 | 2 | Active |
| OpenClinica | OpenClinica is an open source software for Electronic Data Capture (EDC) and Clinical Data Management (CDM) used to optimize clinical trial workflow in a smart and secure fashion. | GNU LGPL | 2005 | 22 | Inactive |
| tranSMART | tranSMART is a knowledge management platform that enables scientists to develop and refine research hypotheses by investigating correlations between genetic and phenotypic data, and assessing their analytical results in the context of published literature and other work. | GNU GPL | 2013 | 45 | Inactive |
| Gadgetron | Gadgetron is an open source software for medical image reconstruction. | MIT license | 2013 | 48 | Active |
| NiftyNet | NiftyNet is an open-source convolutional neural networks platform for research in medical image analysis and image-guided therapy. | Apache License 2.0 | 2017 | 41 | Inactive |

**Table 1.1:** A selection of well-known OSS in healthcare

However, given the large amount of OSS in healthcare, how to choose the most appropriate OSS to solve a particular healthcare problem is challenging. For instance, there are numerous OSS for EHR: OpenEMR, OSEHRA VistA, OpenMRS, GNU Health, OSCAR, Hospital OS, and ClearHealth; each of them has its own strengths and weaknesses (Purkayastha et al., 2019). Which OSS to use in building an EHR system is a difficult question. As a challenging topic, reusing OSS still faces many

difficulties, particularly in the healthcare domain (Shen. and Spruit., 2019). The lack of resources to support the selection of OSS from tens of thousands of open source projects is one of them. Although systematic literature studies on clinical OSS could help us obtain a deeper understanding of the existing tools and their performances (Rehim et al., 2017; Guaitoli et al., 2014; Marien et al., 2017), they are not particularly useful in selecting OSS since they cover only a small proportion of the vast amount of clinical software. Furthermore, the rapid updates of OSS in healthcare are not reflected appropriately due to the slow scientific publication process.

## 1.4 Research Questions

As previously mentioned, HIS engineering plays a crucial role in addressing various clinical challenges, and facilitating the digitalization of healthcare. However, developing and implementing HIS in today's healthcare organizations still faces many challenges. Therefore, this dissertation poses the following main research question:

> **MRQ** – How can we employ artificial intelligence technologies – such as machine learning algorithms, knowledge systems and natural language processing techniques – based on open source principles to accelerate healthcare information system engineering in solving real-world clinical problems?

To answer the main research question, we formulate six research questions. The first three questions focus on a specific use case. First, we design an intelligent clinical decision support system for addressing polypharmacy reviews. Then, based on the use case, the next three research questions investigate the use of NLP for different biomedical/clinical challenges. The opportunities of open source clinical software for clinical research and practices are also explored in these questions.

> **RQ1** – How can we design a GDPR-compliant clinical decision support system that supports a multi-center multi-lingual clinical trial?

Polypharmacy, defined as the chronic use of multiple medications by a patient, often leads to severe clinical problems or accidents if it is inappropriate, including adverse drug effects, underprescribing, overtreatment, low patient compliance and decreased drug adherence (Björkman et al., 2002; Claxton et al., 2001; Munger, 2010; Steinman

et al., 2006; Wright et al., 2009). The Systematic Tool to Reduce Inappropriate Prescribing (STRIP) is a clinical intervention method crafted to deal with polypharmacy problems which are incurred by the concurrent use of multiple drugs. STRIP has been proven to be effective and is included in the Dutch national guideline for polypharmacy. To boost the usage of STRIP in clinical practices, we developed a web application called the STRIP Assistant (STRIPA) and a number of Dutch physicians further evaluated it in terms of user-friendliness, efficiency and effectiveness (Meulendijk et al., 2015a,b, 2016). However, to use the Dutch-based STRIPA tool in a large multinational randomized clinical trial (RCT), we need to address several issues, including multilingual support, clinical data security, data accessibility, consistency and GDPR-compatibility. Therefore, chapter 2 presents an overhauled STRIPA prototype with a lightweight data integration component that supports multinational implementations, ensures data consistency across countries, and maintains data accessibility and security.

**RQ2** – How can we improve rapid and cost-effective development of clinical NLP systems with external NLP APIs?

Natural language processing (NLP) has become essential for secondary use of clinical data. Over the last two decades, many clinical NLP systems were developed in both academia and industry. However, nearly all existing systems are restricted to specific clinical settings mainly because they were developed for and tested with specific datasets, and they often fail to scale up. Therefore, using existing NLP systems for one's own clinical purposes requires substantial resources and long-term time commitments for customization and testing. Moreover, the maintenance is troublesome and time-consuming. Therefore, chapter 2 presents a lightweight approach for building clinical NLP systems with limited resources. Following the design science research approach, we propose a lightweight architecture which is designed to be composable, extensible, and configurable. It takes NLP as an external component which can be accessed independently and orchestrated in a pipeline via web APIs. To validate its feasibility, we developed a web-based prototype for clinical concept extraction with six well-known NLP APIs and evaluated it on three clinical datasets.

**RQ3** – How can we utilize NLP techniques to automatically extract adverse drug reactions from the summary of product characteristics in European drug product labels?

Drug product labels are regulatory documents required as part of the marketing authorization of each medicine. They provide up-to-date and comprehensive information about the risks, benefits, and pharmacological properties of marketed medicines. As such, extracting the clinical knowledge stored in product labels and making it available in the form of computationally accessible knowledge bases would benefit several applications in the area of drug safety surveillance and assessment (Banda et al., 2016; Roberts et al., 2017). Therefore, in chapter 3 we investigate how to utilize NLP techniques to automatically extract adverse drug reactions (ADR) from standardized European product labels, namely SmPC. To answer the question, we develop an NLP pipeline to extract adverse drug reactions from SmPC.

**RQ4** – How can we design an open source big data framework to facilitate cost-effective, large-scale, biomedical literature mining?

A huge amount of biomedical literature has been produced over the decades. For example, as the leading biomedical literature database, PubMed Central (PMC) has archived over 5.3 million research papers, of which around 2.4 million full-text articles are easily accessible at the PMC Open Access Subset (PMC OA Subset) (Pubmed Central, 2019; Pubmed Open Access, 2019). The massive size of available biomedical literature requires researchers to utilize novel big data technologies in data storage and analysis. Among them is cloud computing which has become the most popular solution for big data applications in industry. Therefore, in chapter 4 we propose a cloud-based big data framework that enables researchers to perform reproducible and scalable large-scale biomedical literature mining in an efficient and cost-effective way. Additionally, a cloud agnostic platform was constructed and then evaluated on two open access corpora with millions of full-text biomedical articles. The results demonstrate that our framework indeed supports scalable and efficient large-scale biomedical literature mining.

**RQ5** – How can we support clinical developers' decisions in the software development process with a reproducible and scalable method for systematic studies on open source clinical software?

The plethora of open source projects offers great reuse opportunities for developers to build tools at lower cost and at a faster pace. Therefore, software reuse, also called code reuse, has become an essential topic in software development (Frakes and Fox, 1996; Zaimi et al., 2015). Zhang and Ho (2017) have recognized this importance and have called for more reuse of open source projects in clinical settings (Zhang and Ho, 2017). However, effective software reuse still faces many difficulties, particularly in clinical software development (de Oliveira, 2015). To address the problem, chapter 5 contributes to clinical software reuse research by conducting a large-scale analysis of open source repositories in the clinical domain. In particular, its purpose is to shed light on the following questions, so that clinical developers can make more informed decisions: 1). what is the current status of open source clinical software? 2). what are the impacts of various factors, such as the number of contributors and the number of commits, on the popularity of an open source clinical software? 3). what are the main focus areas of all the collected open source clinical software?

**RQ6** – How can we design an online platform where clinical developers can easily locate and confidently select appropriate open-source clinical software based on associated scientific literature?

Numerous open-source tools have been created across a variety of domains after decades of open source software advocacy (Anthes, 2016). With the accelerating popularity of open science, more and more tools will be added to open repositories. Open-source clinical software covers various research topics and clinical practices, including medical image analysis (McCormick et al., 2014), medical text processing (Cunningham et al., 2013), clinical trials management systems (Haak et al., 2016), and electronic health record systems (de Abajo and Ballestero, 2012). The plethora of tools offers great opportunities for both clinical researchers and practitioners to accelerate their work with available open-source tools and algorithms. However, it also leaves many people unable to easily locate the tools most suitable for their clinical research or practices. Therefore, chapter 6 addresses this challenge by designing a search tool that links open-source clinical software to their literature. With this LOCATE tool, clinical researchers and practitioners are able to find literature related to open-source

clinical software of their interest so that they can make better informed decisions.

## 1.5 Research Methods

The dissertation adopts the Design Science Research (DSR) framework. Different DSR approaches are used to answer the above research questions. The DSR framework is a well-known and widely used research method in information systems (Gregor and Hevner, 2013). Known for its strength and popularity in solving real-world problems by designing and building innovative IT artifacts (Hevner et al., 2008), this research follows the design science research methodology (DSRM) proposed by Peffers et al. (2007), which consists of six steps: problem identification and motivation, a definition of the objectives for a solution, design and development, demonstration, evaluation, and communication. Figure 1.1 explains how these steps are applied in the studies. Furthermore, the following methods have been applied in this dissertation: prototyping, computational experiment, and quantitative analysis.

### 1.5.1 Prototyping

In information systems development, prototyping is a quick and inexpensive way to improve requirements, commitment from end-users, and quality of code (Beynon-Davies et al., 1999). Prototypes can be built in the early stages, eliciting or validating requirements, in the middle stages, confirming the behavior of a system, or in the late stages, investigating the operational system. In addition to a development tool, prototyping can also be used as a research method, to increase knowledge and understanding of a system (O'Leary, 1988). Building such a prototype can additionally be used to quantify the performance of a system. In this research, we use prototyping in four chapters (chapters 2 to 4 and 7). In each study, a working prototype with all key functionalities of the system is constructed to validate its workings.

### 1.5.2 Computational experiments

An experiment is one of the common quantitative research methods in research, which investigates the effect of modifying an independent variable on the dependent variable. The focus of a computational experiment lies in the theoretical analysis and empirical testing of a computational method, such as approximation or optimization algorithms. Zelkowitz and Wallace (1998) for example categorize experimental

research in software engineering into observational, historical, and controlled methods. In this work, three chapters (chapters 3 to 5) use computational experiments to conduct their research.

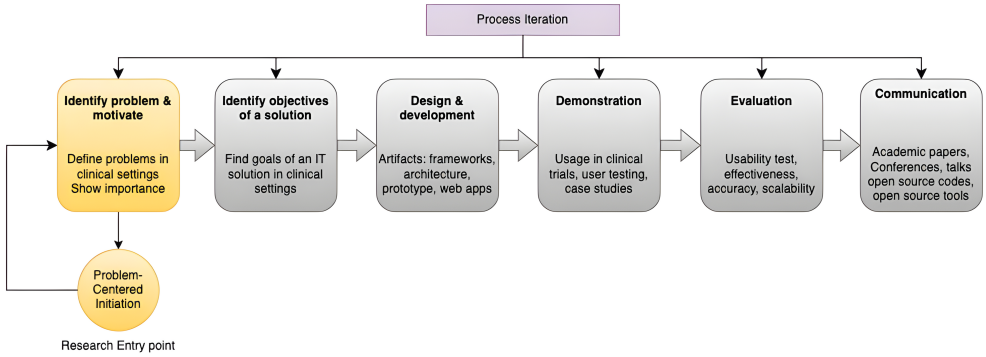



**Figure 1.1:** Process of design science research methodology

### 1.5.3 Quantitative analysis

Creswell et al. (2004) define quantitative analysis as a research method that involves the collection of quantified data and statistical analysis for supporting or refuting claims. Different statistical methods are applied when it comes to different collected data. Numerical and textual data are common data types collected in research. Researchers use statistical models to analyze numerical data to discover trends, correlations and other insights. For textual data, natural language processing techniques are applied before using any statistical models. For example, chapter 6 collects both numerical and textual data and performs quantitative analysis on them.

## 1.6 Dissertation Outline

Chapters 2-7 of the dissertation investigate the research questions RQ1-RQ6, with each chapter corresponding to one research question. All chapters are written as papers, published in proceedings of scientific conferences or in scientific journals. Table 1.2 offers an overview of Chapters 2-7 and their corresponding research question ID, research methods, and a summary of the datasets used in each chapter.

| Ch. | RQ | Research Methods | Data | Outcomes | Source codes |
|---|---|---|---|---|---|
| 2 | RQ1 | Prototyping | Table 2.1 in Chapter 2 | Architecture and Web Application running in production to support multinational clinical trials | Per request |
| 3 | RQ2 | Prototyping, computational experiment, Quantitative analysis | 100 EHRs in free-text and manual annotation of these documents | A prototype (web application) that processes free-text EHRs | Yes (GitHub) |
| 4 | RQ3 | Prototyping, computational experiment, Quantitative analysis | Adverse drug reactions sections for 647 common medicines scraped from the Electronic Medicines Compendium | A reusable data pipeline in Python and structured ADRs knowledge base | Yes (GitHub) |
| 5 | RQ4 | Computational experiment, Quantitative analysis | Over one million full-text biomedical articles from PubMed Open Access Subset; 29437 labeled full-text biomedical articles from PubMed | A framework supports big clinical data analysis | Yes (GitHub) |
| 6 | RQ5 | Computational experiment, Quantitative analysis | Metadata of 14971 clinical-related open-source GitHub repositories | A resuable data pipeline for conducting systematic studies on GitHub | Yes (GitHub) |
| 7 | RQ6 | Prototyping | Metadata of 5119 clinical-related open-source GitHub repositories; 8820 scientific papers that related to the selected repositories | A public web application as prototype | Yes (GitHub) |

**Table 1.2:** Overview per research question of the research methods and datasets used in Chapters 2-7

## 1.7 Dissertation outcomes

This section lists used and created datasets, available source codes, implemented prototypes and scientific publications of this thesis. Table 1.2 outlines the research questions, methods and outcomes from Chapter 2-7.

### 1.7.1 Data

**The i2b2 2008 Obesity Challenge** - consists of 611 discharge letters. The data provider (National Center for Biomedical Computing) annotated all discharge letters with 16 different medical condition terms in the context of obesity, including asthma, gastroesophageal disorder, and depression. Terms could be either annotated as being in the document, not being in the document, or undecided/unknown which was treated as not being in the document. The dataset is available at `https://www.i2b2.org/NLP/Obesity`.

**The i2b2 2009 Medication Challenge** - contains 1243 de-identified discharge letters with the gold standard annotations. Medication names in the annotations are used for the evaluation. The dataset is available at `https://www.i2b2.org/NLP/Medication`.

**Adverse drug reactions corpora** - contains section 4.8 Undesirable effects of the Summary of Product Characteristics (SmPC) of 647 medications scraped from the Electronic Medicines Compendium (EMC). This data is either free-text or tabular text. The datasets are available at `https://github.com/ianshan0915/ade-extraction/tree/master/data`.

**PubMed Open Access Subset** - has millions of journal articles and preprints that are made available under license terms that allow reuse. The dataset was obtained via the PMC FTP service as zip files and it contains 1010787 full-text biomedical articles (Shen et al., 2019b). This data could be found at `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist`.

**Labeled full-text biomedical texts from PubMed** - contains 29437 full-text biomedical articles from PubMed which are divided into three groups: breast cancer, lung cancer, and prostate cancer (Ye et al., 2016). This dataset was used for evaluating document classification models (Ye et al., 2016; Shen et al., 2019b) and is available at `https://figshare.com/articles/dataset/New_draft_item/3796302`.

### 1.7.2 Code and Prototypes

**The STRIP Assistant (STRIPA)** is a web application that was built to help physicians to perform medication reviews in the case of polypharmacy. It was used in a large-scale randomized clinical trial (RCT) across four European countries: the

Netherlands, Switzerland, Ireland and Belgium. A demo of the tool is available at https://youtu.be/GMhge8GxaVk.

**MAB-NLP** is a prototype that extracts clinical concepts from clinical free-texts, such as discharge summaries, clinical reports, etc. It was developed based on a scalable API-based architecture using Laravel. The code is available at: https://github.com/ianshan0915/MABNLP. A demo of the tool is available at https://youtu.be/dGk9NQGWYfI.

**Automatic Extraction of Adverse Drug Reactions** is a natural language processing pipeline that automatically scrapes the online summary of product characteristics of medications and then extracts structured adverse drug reactions from them. The code is available at: https://github.com/ianshan0915/ade-extraction.

**LDABiotext** is a spark application using a Latent Dirichlet Allocation (LDA) model to infer topics from a large collection of biomedical literature from the PMC OA Subset. It is written in Scala and can be deployed to a cloud infrastructure of your choice. The code is available at: https://github.com/ianshan0915/Spark-LDA-biomedical-text.

**GitHub Data Pipeline** is used to collect information about GitHub repositories via the GitHub API. Data analysis is conducted to uncover knowledge of the collected repositories. The code is available at: https://github.com/ianshan0915/clinical-opensource-projects.

**LOCATE** is a web application that is useful for both IT practitioners and researchers in the clinical community. It enables practitioners to explore related literature for a given open-source clinical software so that they could make an informed decision on which software to choose. Clinical researchers are provided with a list of potential useful open-source tools based on their research topics. Details of the prototype and its source codes are available at: https://github.com/ianshan0915/locate.

### 1.7.3 Publications

Publications that are related to the thesis:

1. Shen, Z., Meulendijk, M., and Spruit, M. (2016b). A federated information architecture for multinational clinical trials: Stripa revisited. *24th European Conference on Information Systems (ECIS)*. https://aisel.aisnet.org/ecis2016_prototypes/2

2. Shen, Z., van Krimpen, H., and Spruit, M. (2019a). A lightweight api-based approach for building flexible clinical nlp systems. *Journal of Healthcare Engineering*, 2019(1):3435609. https://onlinelibrary.wiley.com/doi/abs/10.1155/2019/3435609

3. Shen, Z. and Spruit, M. (2021). Automatic extraction of adverse drug reactions from summary of product characteristics. *Applied Sciences*, 11(6). https://www.mdpi.com/2076-3417/11/6/2663

4. Shen, Z., Wang, X., and Spruit, M. (2019b). Big data framework for scalable and efficient biomedical literature mining in the cloud. In *Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval*, NLPIR '19, page 80–86, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3342827.3342843

5. Shen, Z. and Spruit, M. (2019). A systematic review of open source clinical software on github for improving software reuse in smart healthcare. *Applied Sciences*, 9(1). https://www.mdpi.com/2076-3417/9/1/150

6. Shen., Z. and Spruit., M. (2019). Locate: A web application to link open-source clinical software with literature. In *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2019) - HEALTHINF*, pages 294–301. INSTICC, SciTePress. https://doi.org/10.5220/0007378702940301

### 1.7.4 Other Publications by the Author

1. van Tuijl, G., Leenen, W., **Z. Shen**, van de Weerd, I., and **S. Brinkkemper** (2011). Prioritizing requirements: An experiment to test the perceived reliability, usability and time consumption of bubblesort and the analytical hierarchy process. In *Proceedings of the International Requirements Engineering Efficiency Workshop (REEW 2011)*, pages 37–48. https://api.semanticscholar.org/CorpusID:53900285

2. Crowley, E. K., Sallevelt, B. T., Huibers, C. J., Murphy, K. D., **Spruit, M.**, **Shen, Z.**, Boland, B., Spinewine, A., Dalleur, O., Moutzouri, E., et al. (2020). Intervention protocol: Optimising therapy to prevent avoidable hospital admission in the multi-morbid elderly (operam): a structured medication review with sup-

port of a computerised decision support system. *BMC health services research*, 20(1):1–12. https://doi.org/10.1186/s12913-020-5056-3

3. Omta, W. A., van Heesbeen, R. G., **Z.Shen**, de Nobel, J., Robers, D., van der Velden, L. M., Medema, R. H., Siebes, A. P., Feelders, A. J., **S. Brinkkemper**, Klumperman, J. S., **M. Spruit**, Brinkhuis, M. J., and Egan, D. A. (2020b). Combining supervised and unsupervised machine learning methods for phenotypic functional genomics screening. *SLAS Discovery*, 25(6):655–664. https://doi.org/10.1177/2472555220919345

4. Omta, W. A., van Heesbeen, R. G., **Shen, Z.**, Feelders, A. J., Brinkhuis, M., Egan, D. A., and **Spruit, M.** (2020a). Purifyr: An r package for highly automated, reproducible variable extraction and standardization. *Systems Medicine*, 3(1):1–7. https://api.semanticscholar.org/CorpusID:215898718

5. Blum, M. R., Sallevelt, B. T. G. M., Spinewine, A., O'Mahony, D., Moutzouri, E., Feller, M., Baumgartner, C., Roumet, M., Jungo, K. T., Schwab, N., Bretagne, L., Beglinger, S., Aubert, C. E., Wilting, I., Thevelin, S., Murphy, K., Huibers, C. J. A., Drenth-van Maanen, A. C., Boland, B., Crowley, E., Eichenberger, A., Meulendijk, M., Jennings, E., Adam, L., Roos, M. J., Gleeson, L., **Shen, Z.**, Marien, S., Meinders, A.-J., Baretella, O., Netzer, S., de Montmollin, M., Fournier, A., Mouzon, A., O'Mahony, C., Aujesky, D., Mavridis, D., Byrne, S., Jansen, P. A. F., Schwenkglenks, M., **Spruit, M.**, Dalleur, O., Knol, W., Trelle, S., and Rodondi, N. (2021b). Optimizing therapy to prevent avoidable hospital admissions in multimorbid older adults (operam): cluster randomised controlled trial. *BMJ*, 374

6. Sallevelt, B., Huibers, L., Op Heij, J., Egberts, T., Puijenbroek, E., **Shen, Z.**, **Spruit, M.**, Jungo, K., Rodondi, N., Dalleur, O., Spinewine, A., Jennings, E., O'Mahony, D., Wilting, I., and Knol, W. (2021). Frequency and acceptance of clinical decision support system-generated stopp/start signals for hospitalised older patients with polypharmacy and multimorbidity. *Drugs & Aging*, 39. https://doi.org/10.1007/s40266-021-00904-z

7. Shen, Z., Meulendijk, M., Knol, W., Huibers, L., Wilting, I., Jansen, P., and Spruit, M. (2016a). Stripa investigational medical device dossier (imdd). Technical report, UU/UMCU. https://marcospruit.nl/pub/2016%20-%20Shen%20et%20al%20-%20IMDD.pdf

# Chapter 2

# Federated Architecture for Multinational Clinical Trials

The Systematic Tool to Reduce Inappropriate Prescribing (STRIP) is a clinical intervention method crafted to deal with polypharmacy problems which are incurred by the concurrent use of multiple drugs. STRIP has been proven to be effective and is included in the Dutch national guideline for polypharmacy. To boost the usage of STRIP in clinical practices, a web application called the STRIP Assistant (STRIPA) was developed and further evaluated as user-friendly, efficient and effective by Dutch physicians. STRIPA has now evolved into a software tool that supports a large multinational randomized clinical trial (RCT). However, in order to successfully implement and use the application in such an RCT, several issues, including multilingual support, clinical data security, data accessibility and consistency, need to be addressed. In this chapter, we present an overhauled STRIPA prototype with a lightweight data integration component that supports multinational implementations, ensures data consistency across countries, and maintains data accessibility and security. The component includes a high-level information architecture, data models redesigned to generalize data entities from all countries, and the ETL processes that integrate diverse data sources and transfer data between databases.

This work was originally published as: Shen Z., Meulendijk, M. and Spruit, M. (2016). A Federated Information Architecture for Multinational Clinical Trials: STRIPA Revisted. *Proceedings of the 24th European Conference on Information Systems (ECIS)*, Prototypes. Paper 2. https://aisel.aisnet.org/ecis2016_prototypes/2

## 2.1 Introduction

Polypharmacy, defined as the chronic use of multiple medications by a patient, often leads to severe clinical problems or accidents if it is inappropriate, including adverse drug effects, under prescribing, overtreatment, low patient compliance and decreased drug adherence (Björkman et al., 2002; Claxton et al., 2001; Munger, 2010; Steinman et al., 2006; Wright et al., 2009). The elderly, generally adults aged over 65 years, are the main sufferers of polypharmacy (Munger, 2010). The demographic shift towards an elderly population among developed countries brings necessity and urgency to the implementation of an effective solution to reduce the risks and maximize the benefits of polypharmacy.

The STRIP assistant (STRIPA) is a web application that was built to help physicians to perform medication reviews in the case of polypharmacy. A standardized medication review method named Systematic Tool to Reduce Inappropriate Prescribing (STRIP) is incorporated into the application. Meulendijk et al. (2015b) presented the main components of the application, including functional architecture, user interface, decision rule engine, and data formats. It has gained recognition as a user-friendly, effective and efficient tool among Dutch physicians (Meulendijk et al., 2015a, 2016). Now, it is a part of a large-scale randomized clinical trial (RCT) that aims at investigating impacts of medication optimization in the multi-morbid elderly among four European countries: the Netherlands, Switzerland, Ireland and Belgium.

In order to implement and use the application across countries, relevant data from each country need to be collected and properly managed in the first place. These data contain medical data on medications, drug interactions and standardized medical terminology, and patient health records. Besides, usage data recorded in each country should be accumulated for further analysis. As the number of data sources grows, the complexity and diversity also increases. How to properly integrate and manage such diverse data sources becomes a challenge.

In this chapter we present an overhauled prototype that deals with the above data integration issues so as to support conducting medication reviews in multiple countries. The prototype, named as STRIPA.EU, expands the previous version of STRIPA with a data integration component which mainly consists of a number of databases and ETL processes that populate the databases. The component is able to integrate necessary heterogeneous data sources from different countries, easily update the data as data sources renew, and gather country-specific usage data together into a centralized database. In general, the application works independently as a cus-

tomized application for systematic medication reviews in each country. However, it also has a centralized database that maintains data consistency across countries and subsequently keeps research integrity and consistency of the large international RCT.

The remainder of the chapter is organized as follows. At first, related work and the detailed design objectives of the present research are discussed. Second, the data integration component that extends STRIPA to multiple countries is elaborated from three aspects: data representation, integration architecture and the ETL implementation. Preliminary results of the implementation of STRIPA.EU are briefly discussed in the following. The final section presents contributions of this research and possible future research aspects.

## 2.2 Background

### 2.2.1 Status Quo of Data Integration

Data integration aims at providing a unified view of heterogeneous data residing at different sources (Calì et al., 2002). There are three steps in data integration: extracting data from homogeneous or heterogeneous data sources, transforming data into a common structure and loading it into the final target (Muilu et al., 2007). These steps are commonly shortened as ETL. Data integration has become of great importance in the medical/clinical domain because of the tremendous increasing volume and complexity of data gathered at the medical community (Branson et al., 2008; Brazhnik and Jones, 2007). Many studies regarding integrating heterogeneous data sources in the areas of biomedical informatics and information technology have been conducted over the last decades (Karasavvas et al., 2004; Louie et al., 2007; A Seoane et al., 2013). However, there is little literature focusing on integrating clinical data sources, especially pharmaceutical data from various countries. But the abundance of information about drugs and its heterogeneous nature among countries does pose a challenge to data integration (Meulendijk et al., 2015b).

Louie et al. (2007) divided data integration into two axes: integration architecture, and data representation. Integration architecture refers to the final destination into which data and metadata are loaded. There are two main types of integration architecture: data warehouse and federated database system (FDB) (Louie et al., 2007). The need for combining various independent data sources into a secure, but easily accessible data architecture has given rise to the development of federated database systems (Calì et al., 2002; Ziegler and Dittrich, 2004). A FDB system has a reliable

infrastructure for data sharing and collaboration across disparate data sources while maintaining security and privacy locally. A number of aspects of such a system, such as privacy, security, access control and availability, have been well-studied both in research and in industry (Ansper et al., 2013).

In relational databases, data representation refers to a conceptual data model which represents the common schema of diverse data sources (Louie et al., 2007). For data integration, data models give a unified and structured view of the integrated and reconciled data sources, and also offer the elements for expressing the queries over integrated systems (Calì et al., 2002). Identifying such a data representation from heterogeneous data sources is mostly time consuming and requires extensive human interactions (Zhao and Ram, 2007). Over the last three decades numerous methods have been proposed to facilitate this process (Batini et al., 1986; Clifton et al., 1998; Passi1 et al., 2002; Ramesh and Ram, 1995; Zhao and Ram, 2007). Data models for STRIPA.EU are created by detecting schematic correspondences among data sources at both structural and semantic levels.

### 2.2.2 Design Objectives

As stated by Meulendijk et al. (2015b), data exchange and integration for STRIPA are difficult because of incompatible information systems and customized international classification standards for drugs and diseases in each country. In particular, since most countries use medical coding standards modified on the basis of international standards, or even completely disconnected ones, data encoded with country-specific codes are not fully interoperable. But decision rules of STRIPA are crafted with international codes, and they only recognize input encoded in such codes. Hence, semantic matching of data sources at instance level is necessary in data integration. Moreover, data of drugs and drug interactions from third-party companies vary a lot in terms of both schemas and formats. Last but not least, multilingualism of data sources adds another dimension of complexity to data integration. Table 2.1 demonstrates the diversity and complexity of data sources from different countries.

To overcome these issues, we developed a data integration component that is presented in three pillars: data representation, ETL processes and integration architecture. The component is added into STRIPA as a built-in tool. The design objectives when developing the new prototype were as follows:

(1) *to create a federated database system with a unified schema that supports implementation of the application in multiple countries*

(2) *to develop ETL processes that populate both a federated database and multiple localized databases in the predefined schema*

(3) *to regularly update and maintain medical data, including drugs and drug interactions, in each country*

(4) *to have an architecture which makes the application running in each country on customized local data sources and all the locally collected data remain under the control of the local authorities for the sake of privacy and security*

(5) *to create a unique identifier for each data item in the system*

| Supplier | Database / Standard | Scope | Format | Country | Languages |
|---|---|---|---|---|---|
| Z-Index | G-Standard | Medications, Clinical Interactions | Fixed Width | Netherlands | Dutch |
| RIVM | ICD-10 | Episodes | XML | Netherlands | Dutch |
| APB | Delphi-Care | Medications, Clinical Interactions | Fixed Width | Belgium | French, Dutch |
| FOD | ICD-10 | Episodes | CSV | Belgium | French, Dutch |
| HCI Solutions | INDEX | Medications, Clinical Interactions | XML | Switzerland | German, French |
| DIMDI | ICD-10 | Episodes | XML | Switzerland | German |
| HelixHealth | Safescript | Medications, Clinical Interactions | Fixed Width | Ireland | English |
| WHO | ICD-10 | Episodes | XML | Ireland | English |
| MedDRA MSSO | MedDRA | Adverse Events | CSV | Netherlands, Belgium, Switzerland, Ireland | Dutch, French, German, English |
| Regenstrief | LOINC | Measures, Laboratory Tests | CSV | Netherlands, Belgium, Switzerland, Ireland | Dutch, French, German, English |

**Table 2.1:** Data sources from different countries

## 2.3 Data Integration Component

Details of the data integration component that supports and realizes the design objectives are described in continuation. It is elaborated from three main aspects: data representation, integration architecture, and the ETL processes.

### 2.3.1 Data Representation

As discussed in the previous section, determining data models that represent a common schema of all heterogeneous data sources is crucial for data extraction and trans-

formation at the beginning of an ETL process. There are a great amount of methods or tools that have been developed for detecting schema-level correspondences. In this chapter we identified common schema elements (i.e., data entities, relations and attributes) by matching schema elements of all the available data sources at both structural and semantic level. Meanwhile, unnecessary elements are filtered out from the identified common schema according to the system requirements.

To begin with, data required for the application are summarized into two categories: medical knowledge and patient data. Medication data, medical codes and decision rules constitute the knowledge base of STRIPA. As the core data component, medication data contain all available drugs, drug interactions and adverse events in a country. Decision rules in the application are implemented based on established clinical guidelines. Hospitals register a variety of information about a patient on patient health records, but STRIPA only requires patients' medical data on treatments, diagnoses and measurements. Moreover, patient health records need to be encoded with standardized medical codes because both decision rules and drug interactions are composed with standardized codes and they can only recognize encoded patient's information.

As shown in Table 2.1, medication data are provided by many different data suppliers. By matching data entities of all data sources, four common data entities are derived: medication, product, substance and interactions. Medication stores generic drugs and their substances are saved in substance. Besides, each generic drug might have a list of products in the market and this information is recorded in product. Interactions between two generic drugs are put into interactions. Figure 2.1 shows the relationship between these data entities and critical attributes of each data entity are also listed.

Patient health records are represented by three data entities: diagnoses, measurements and treatments. Standardized medical codes that are utilized to encode patient health records include LOINC, SNOMED, modified and translated versions of ICD-10 in each country. They are represented with four data entities: episodes, dosage-units, dosage-intervals and measures. Episodes is used to instantiate diagnoses with ICD-10 codes. For instance, one record of patients' diagnoses refers to an episode identifier given the patient's number and a timestamp. Measures is extracted from LOINC and it relates to patients' measurements the same as episodes to diagnoses. Together with medications, dosage-intervals and dosage-units generate instances of treatments of patients. Figure 2.1 demonstrates how patient health records are modelled with three layers of data entities. Splitting patient health records into a set of

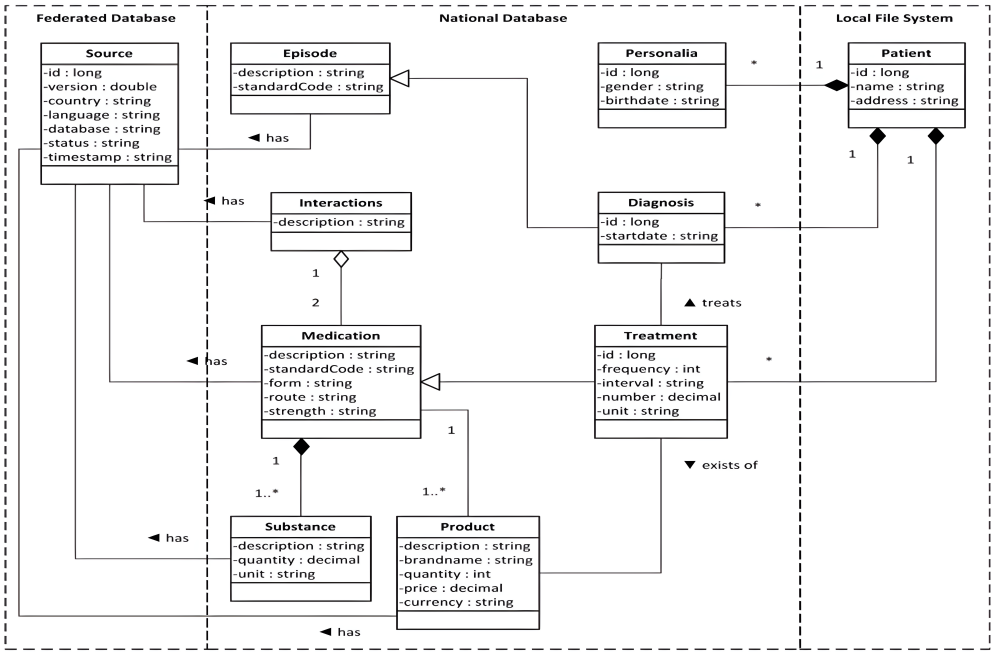



**Figure 2.1:** Class Diagram depicting main data models of STRIPA

data entities reduces data redundancy in the first place because of database normalization. It also provides an easy way of updating medical codes in each country. For example, adding new ICD-10 codes into episodes requires no modification for diagnoses and the rest of database.

Since each country regularly renews its medication data by adding new medications or removing old ones, medication data in the application need to be updated accordingly. In order to keep track of medication data during a series of updates, a unique identifier for each data item is created and recorded in sources before data items are added into relevant databases. Table sources also stores other metadata which keeps medical knowledge data manageable. Figure 2.1 shows attributes of sources and relationships between sources and other data entities. In addition, to maintain the privacy of patients patients' identification information, such as name, address, is stored as patient in local file system within hospitals, and the rest are put into personalia in national databases. They are linked with a unique patient identifier.

### 2.3.2 Integration Architecture

Integration architecture provides an outline of databases in a system from a data integration perspective. Data flows are also included to give a dynamic view of how data are transferred in-between databases. Building a reliable integration architecture is crucial for any system with distributed and autonomous databases in different locations. For STRIPA.EU a federated database system was developed instead of a data warehouse for a number of reasons. At first, unlike a data warehouse, the database in each country is not a subset of a centralized database, whereas each contains all data necessary for the application and takes nothing from the centralized one. Secondly, patient health records collected locally need to remain under the control of the local authorities for the sake of privacy and security. Only collected research data in each country are allowed to be transferred into the centralized database. Finally, a federated database system grants autonomy to the database in each country, which guarantees national databases having power to adapt themselves in response to changes.

Figure 2.2 depicts the integration architecture that was designed for STRIPA.EU. It is a federated database system which is composed of a federated database and a number of autonomous national databases. As shown in the figure, each country has its own national database, and private patient data is extracted and saved in the local file system within hospitals. Physical independence of the local file system prohibits unauthorized access and use of locally collected private patient data in each country. National databases are autonomous and independent of each other. The autonomy of national databases ensures that countries could customize their databases from various aspects such as schema and data instances. Because of the high independence of national databases, the application in each country has a high system availability. In particular, database breakdown in any country does not affect other countries. The functions of the federated database are twofold: storing and managing metadata for data in all countries, and gathering user log data for further user behaviour analysis (Pachidi et al., 2014). There are two ETL processes for the implementation of STRIPA.EU in each country. The first ETL process takes the aforementioned external data sources and transfers them into a national database while writing metadata into the federated database. Another ETL is designed to accumulate application usage data from all countries. A detailed description of ETL processes will be given in the following.

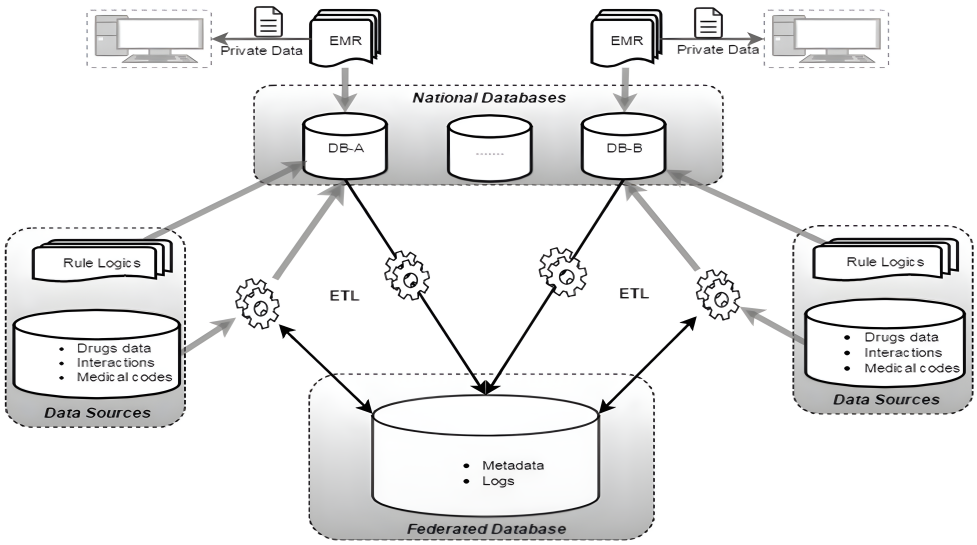



**Figure 2.2:** Information Architecture for data integration in STRIPA.EU

### 2.3.3 ETL

Determining data models and designing integration architecture only lay the groundwork for the federated database system. In order to provide the client side with a reliable data service, all relevant data need to be imported and heterogeneous data sources, including the most important medication data, need to be transferred using well-designed ETL processes. Therefore, ETL plays a decisive role in the implementation of STRIPA.EU.

There are a number of challenges ahead when it comes to constructing a successful ETL implementation: 1) significant dissimilarity between external data sources and target data structure; 2) external data sources are encoded in diverse formats, including CSV, fixed width and XML; 3) ETL needs to be autonomous, preferably just a click away. A powerful open source tool (Talend Open Studio for Data Integration) has been used to operationalize the ETL processes. It is capable of tackling a variety of data formats. And its graphic interfaces make ETL easier to follow and evaluate by non-IT experts. In the following paragraph an example of the ETL processes developed for medication data in the Netherlands is elaborated to give a glimpse of the complexity.

Medication data in the Netherlands are provided in positional flat files by the G-Standaard. There is a significant structural difference between data sources and the

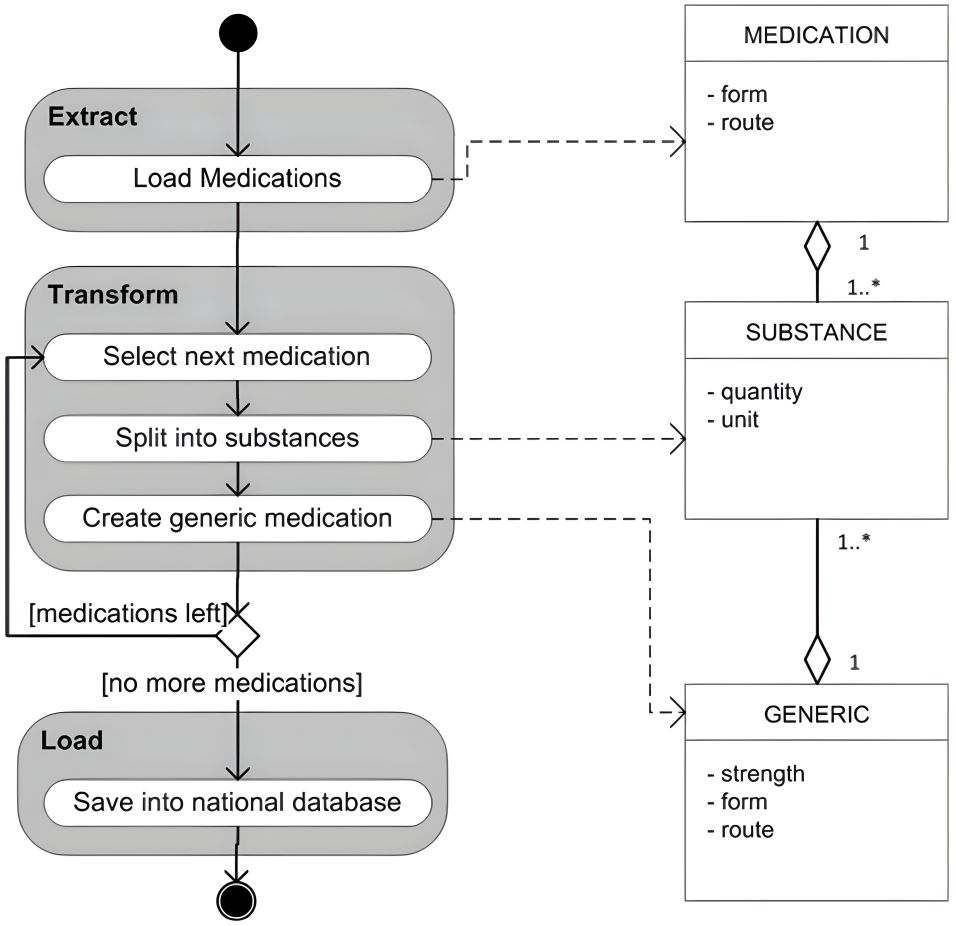



**Figure 2.3:** The ETL process of extracting medication data in the Netherlands

above data model. In specific, generic medications needed in the national database do not exist in data sources, whereas only medicinal products, coupled with product relevant information, such as product name, form, route, etc. are stored separately in different files. ETL should generate generic medications by grouping medicinal products based on their substances. Figure 2.3 conceptualizes the ETL process with the Process-Deliverables Diagram (van de Weerd and Brinkkemper, 2009).

As mentioned above, the regular updates of medication data require us to renew drugs and drug interactions within the system accordingly. And the updating process has been built as a part of the ETL implementation. The following process deliverable diagram illustrates the process of updating a database with its new released external

data sources. It is coded in a python script and automated with a batch file that loads and directly runs the script.

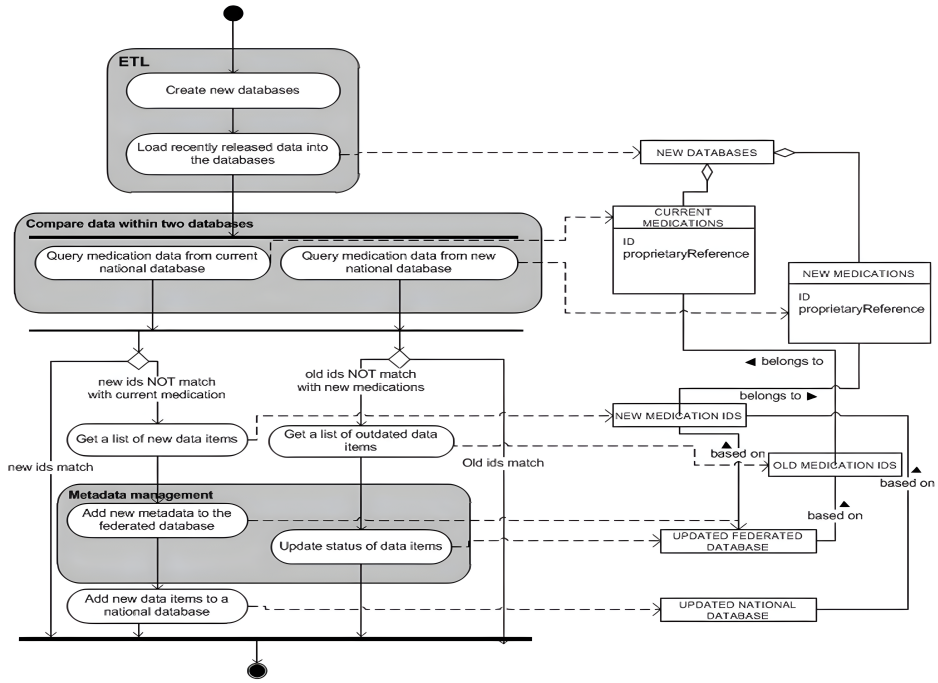



**Figure 2.4:** Process Deliverable Diagram depicting the data updating process

## 2.4 Evaluation & Contribution

Evaluation of the application has been carried out or is currently planned in multiple steps. At first, ETL processes are tested on data completeness, data consistency and data integrity. During the development of ETL processes, random manual comparisons of data between databases and data sources have been performed. More systematic testing tools and methods will be developed to assess ETL performance in terms of efficiency and maintenance before the implementation of the software. The next step of the evaluation focuses on the integration architecture. How well this architecture supports the implementation and use of the application in multiple countries is investigated. Security and privacy issues of the clinical trials are properly addressed with the architecture. Moreover, having integrated data sources from four

data sources in the given architecture shows its technical feasibility and suitability for such multinational clinical trials.

The successful development of STRIPA.EU and initial feedback collected from physicians suggest that the proposed data integration component fulfils the requirements of multinational clinical trials. These requirements, such as healthcare interoperability, privacy and security, are very common among many other large-scale clinical trials. Therefore, this leads us to believe that the federated information architecture and data integration methods have a good potential to be reused in other clinical trials.

## 2.5 Conclusion

This chapter presented a federated architecture for multinational clinical trials through the case study of STRIPA.EU. The proposed solution addresses key challenges in multinational clinical data integration, including multilingual support, data security, and consistency across countries. By implementing a federated database system with carefully designed data models and ETL processes, we achieved a balance between local autonomy and centralized coordination. The architecture successfully supports a clinical trail across four European countries while maintaining data privacy and security requirements.

# Chapter 3

# A Lightweight API-Based Approach for Clinical NLP

Natural language processing (NLP) has become essential for secondary use of clinical data. Over the last two decades, many clinical NLP systems were developed in both academia and industry. However, nearly all existing systems are restricted to specific clinical settings mainly because they were developed for and tested with specific datasets, and they often fail to scale up. Therefore, using existing NLP systems for one's own clinical purposes requires substantial resources and long-term time commitments for customization and testing. Moreover, the maintenance is also troublesome and time-consuming. This research presents a lightweight approach for building clinical NLP systems with limited resources. Following the design science research approach, we propose a lightweight architecture which is designed to be composable, extensible, and configurable. It takes NLP as an external component which can be accessed independently and orchestrated in a pipeline via web APIs. To validate its feasibility, we developed a web-based prototype for clinical concept extraction with six well-known NLP APIs and evaluated it on three clinical datasets. In comparison with available benchmarks for the datasets, three high F1 scores (0.861, 0.724, and 0.805) were obtained from the evaluation. It also gained a low F1 score (0.373) on one of the tests, which probably is due to the small size of the test dataset. The development and evaluation of the prototype demonstrates that our approach has a great potential for building effective clinical NLP systems with limited resources.

This work was originally published as: Shen, Z., van Krimpen, H., & Spruit, M. (2019). A Lightweight API-Based Approach for Building Flexible Clinical NLP Systems. *Journal of Healthcare Engineering*, 2019(1), 3435609. https://doi.org/10.1155/2019/3435609

## 3.1 Introduction

Today's technologies allow the accumulation of vast textual data, which consequently has boosted the popularity of NLP research. There has been a huge amount of papers published and a variety of NLP systems or toolkits crafted in multiple domains over the last two decades. Among them, clinical NLP occupies a large portion. There are clinical NLP systems, such as Apache cTAKES, that integrate different NLP tools to process clinical documents (Davis, 2012; Savova et al., 2010). There are also NLP tools which target certain specific clinical needs, including extracting medication information (Patrick and Li, 2010), identifying locations of pulmonary embolism from radiology reports (Cai et al., 2016), and categorizing pain status (Kreuzthaler and Schulz, 2015).

Figure 3.1 presents a general architecture of a clinical NLP system that contains two main components: background knowledge and framework (Davis, 2012). Background knowledge contains ontologies, domain models, domain knowledge, and trained corpora. The widely used clinical domain knowledge is the Unified Medical Language System (UMLS) (Bodenreider, 2004). Framework refers to a software platform that integrates various NLP tasks or modules either sequentially or hierarchically into NLP pipelines. GATE and UIMA are the leading open-source frameworks (Cunningham et al., 2013; Ferrucci and Lally, 2004). There are two levels of NLP tasks: low-level tasks and high-level tasks. Low-level tasks include tokenization, part of speech tagging, sentence boundary detection, and so on. High-level tasks refer to the semantic level processing such as named entity recognition, relation extraction, and sentiment analysis.

History has shown that building a successful clinical NLP system requires a tremendous amount of resources. For instance, it took a team from Columbia University 14 years to commercialize the MedLEE system (Chiang et al., 2010). The development of cTAKES started at the Mayo Clinic in 2006, and further external collaborations with four other universities in 2010 resulted in the first release of the current Apache project (Savova et al., 2010). Therefore, creating reusable NLP pipelines based on open-source modular frameworks like GATE and UIMA becomes more reasonable (Chiang et al., 2010; De Castilho and Gurevych, 2014). Although it dramatically reduces resources and level of expertise, we argue that it is not an efficient and effective solution for two main reasons. Firstly, nearly every NLP pipeline that is created to address a single specific clinical need, either rule or machine-learning based, has been proven to be useful for only its designated purposes (Kreimeyer et al., 2017).

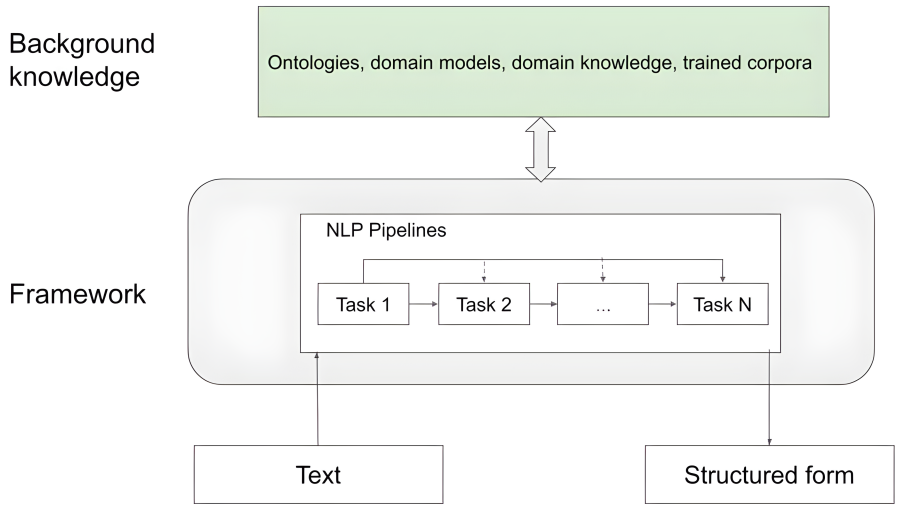



**Figure 3.1:** A general architecture of clinical NLP systems.

Thus, reusability is difficult given the properties. Secondly, deploying cTAKES-based NLP pipelines implies a high cost of operation which requires installation and configuration of multiple components by NLP experts (Carrell, 2011). Besides, maintenance of a deployed NLP system requires a continuous investment.

With the purpose of simplifying and outsourcing the NLP implementation, software as a service, or SaaS, has been introduced to the NLP world during recent years (Dale, 2015). SaaS generally refers to the mode of software delivery where end-users are charged with a monthly or annual subscription fee to utilize a set of functionalities over the Internet (Currie et al., 2004). NLP systems distributed in the SaaS model are often available through web application programming interfaces (APIs) and named as NLP APIs or cloud-based NLP APIs (Dale, 2015; Rago et al., 2016). Many NLP APIs have emerged from both companies and universities and are growing popularly (Cunningham et al., 2013). A few prominent examples are IBM Watson, Aylien, Lexalytics, and TextRazor (Dale, 2015). From the cost-benefit perspective, these NLP APIs allow developers to rapidly create NLP-enabled tools without investing abundant resources on implementing necessary NLP techniques in codes and on regular maintenance. A number of applications based on NLP APIs were built (Haffari et al., 2017; Hellmann et al., 2013; Martínez et al., 2016).

To utilize NLP APIs, API-based frameworks have been produced (Rago et al.,

2016; Abdallah et al., 2017; Chard et al., 2011; Rizzo and Troncy, 2012). API-based systems, also known as cloud-based, refer to tools that are built on external web APIs and having their functionalities partially or fully accomplished with one or a pipeline of APIs. Due to the growing popularity of web APIs in the software industry, API-based tools are abundant in companies. For instance, an API-based CMS (content management system) is utilized to save development resources and follow-up maintenance (APIs, 2017). Furthermore, researchers have also investigated the approach in recent years. Rizzo and Troncy proposed the Named Entity Recognition and Disambiguation (NERD) framework that incorporates the result of ten different public APIs-based NLP extractors (Rizzo and Troncy, 2012). A web-based tool called TeXTracT was devised to support the setup and deployment of NLP techniques on demand (Rago et al., 2016). Abdallah et al. developed a flexible and extensible framework for integrating named entity recognition (NER) web APIs and assessed it across multiple domains (Abdallah et al., 2017). Although these tools exhibit promising results, few were built for clinical NLP or evaluated on clinical datasets. Therefore, it is safe to say that adopting these tools in clinical settings would be problematic due to the unique characteristics of the clinical domain. For example, privacy is considered to be of the utmost importance, but none of the above tools have taken it into consideration.

This paper thus proposes a lightweight framework which enables a rapid development of clinical NLP systems with external NLP APIs. The approach has the following advantages compared to traditional NLP frameworks: (1) fast development; (2) lower costs; (3) flexibility; and (4) programming language independent. The deployment is minimized by outsourcing both NLP tasks and background knowledge to external API services. Thus, NLP systems can be quickly and cost-efficiently developed based on the proposed framework. The framework is flexible in many aspects. To begin with, it supports the flexible combination of different NLP tasks from external APIs. Secondly, users have the freedom of choosing their preferred NLP API vendors, and multiple APIs can be integrated to achieve better results. To evaluate the framework, we have built a web-based open-source clinical NLP application.

## 3.2 Methods

### 3.2.1 Design Science

Our research followed the design science research as we built and evaluated the framework because of its strength and popularity in solving a real-world problem by designing and building an innovative IT artifact (Hevner et al., 2008). In our case, the artifact is a lightweight framework that facilitates clinical NLP systems development. We follow the design science research methodology (DSRM) proposed by Peffers et al., which consists of six steps: problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication (Peffers et al., 2007).

The DSRM is initiated by the (I) problem identification and motivation, which we addressed by literature study. Previous studies have described the general architecture of clinical NLP systems and how expensive it is to build them. Even though the introduction of modular NLP frameworks reduced the complexity of NLP systems, it is still challenging to create clinical NLP systems for many healthcare institutions due to limited resources. Based on the identified problem, we inferred the (II) objectives for a solution: creating a lightweight NLP framework that enables a rapid development of an API-based clinical NLP system. In the (III) design and development, we developed the framework based on the general architecture we identified, after which each of its components is explained in detail. To (IV) demonstrate and (V) evaluate the framework, a web-based open-source clinical NLP application was developed. Moreover, experiments were carried out with three clinical datasets to primarily examine whether external NLP APIs would deliver the state-of-the-art performance. The final step of the DSRM is the communication. The paper serves as the start of our (VI) communication on this topic.

### 3.2.2 Evaluation Design

Three English anonymized clinical datasets were used in our evaluation. Two of the datasets are obtained from the Informatics for Integrating Biology and the Bedside (i2b2) center: 2008 Obesity Challenge and 2009 Medication Challenge. The third dataset comes from a European clinical trial called OPERAM. Since the primary goal of our evaluation is to prove that external general-purpose NLP APIs can yield good performance on clinical data, we only used a subset of the two large i2b2 datasets.

(I) *2008 Obesity Challenge.* This dataset consists of 611 discharge letters. All dis-

charge letters are annotated with 16 different medical condition terms in the context of obesity, including asthma, gastroesophageal disorder, and depression. Terms could be either annotated as being in the document, not being in the document, or undecided/unknown which was treated as a not being in the document. The strength of this dataset, concerning the aim of these tests, is that there are a lot of documents, and its weakness is that it is only annotated for 16 abstract terms in the context of obesity. To simplify the experiment, we randomly selected 100 discharge letters and labeled each document with the medical conditions that are annotated as "present."

(II) *2009 Medication Challenge.* 947 out of 1243 in total deidentified discharge letters have the gold standard annotations. Medication names in the annotations are used for the evaluation. By comparing the annotated medication names with those generated from our application, we calculate the evaluation metrics. We also randomly select 100 out of the 947 documents.

(III) *OPERAM Dataset.* The dataset consists of five discharge letters that have been used during the pilot of the OPERAM clinical trial (Shen et al., 2016b). Medical experts of the trial annotated these letters by both medical conditions and pharmaceutical drugs. Moreover, standardized clinical codes for each annotation are included. With this dataset, we aim to demonstrate the performance of our NLP application with clinical documents from practices, even though it is clear that the small size limits our findings.

We extracted entities of "medical condition" or "pharmaceutical drug" from the, in total, 205 clinical documents and then encoded them with UMLS. Based on the encodings, extracted entities were filtered so that distinct entities were extracted for each clinical document. In order to measure the performance of our extraction, we have used well-known metrics: precision, recall, and F1 score. They are computed from true positives (TP), false positives (FP), and false negatives (FN) for each document. As stated above, annotations of the 2008 Obesity Challenge are different from the other two datasets. To simplify the identification of positives and negatives, we divided annotations into two groups: positives that are in the text and negatives which are not mentioned. Therefore, comparing clinical entities extracted by our application to the ground truth, we calculate the following:

(I) TP: entities that were both extracted and annotated as positives

(II) FP: entities that were extracted as positives but were annotated as negatives

(III) FN: entities that were not extracted but were annotated as positives

Precision Equation 3.1 represents the proportion of extracted positives that are annotated positives. On the contrary, recall Equation 3.2 is the proportion of annotated positives that were correctly extracted as such. F1 score Equation 3.3 is the harmonic mean of precision and recall:

$$\text{precision} = \frac{TP}{TP + FP} \tag{3.1}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.3}$$

## 3.3 Results

The section presents results in two parts: the framework and a web-based open-source clinical NLP application. The architecture lays down the technical groundwork, upon which the application was constructed. The following explains each of them in details.

### 3.3.1 A Lightweight NLP Architecture for Clinical NLP

The architecture addresses the issues of existing clinical NLP applications, including interoperability, flexibility, and specific restrictions within the clinical field, such as privacy and security. The strength of our proposed architecture is shown in its capabilities: (1) freedom of assembling suitable NLP APIs either sequentially or hierarchically based on scenarios; (2) encoding clinical terms with comprehensive and standardized clinical codes; (3) the built-in deidentification function to anonymize clinical documents. Figure 3.2 depicts its four main components: external APIs, infrastructure, NLP pipelines, and Apps.

#### External APIs

In this architecture, two types of APIs, namely, an NLP API and a domain knowledge API, are included to parse unstructured medical text and map parsed terms against a medical metathesaurus, respectively. The NLP API provides various cloud-based

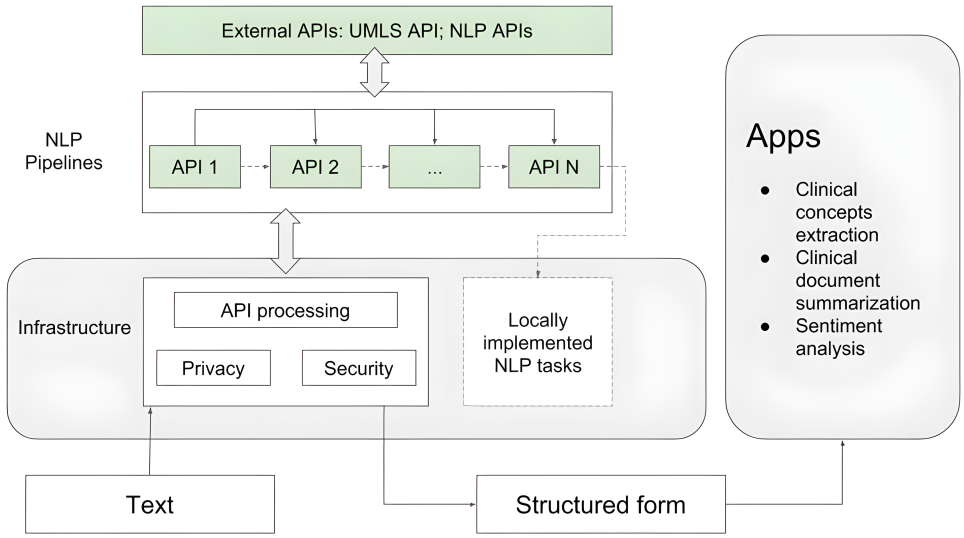



**Figure 3.2:** A lightweight NLP architecture for clinical NLP.

NLP services that parse unstructured text for different purposes, including entity recognition and document classification. The domain knowledge API supports the mapping of medical text to concepts from the UMLS metathesaurus. As the most used biomedical database, UMLS contains millions of biomedical concept names and their relations. In addition, domain models and training corpora are available for specific clinical documents such as radiology reports, pathology reports, and discharge summaries (Davis, 2012). The UMLS is a major part of the solution for standardization and interoperability as it maps terms extracted by multiple APIs to standardized codes such as ATC and ICD10.

### Infrastructure

The infrastructure layer prepares clinical data before sending them to external APIs by deidentification and adding authentications. Furthermore, it processes results received from external APIs for later integration. An optional component, locally implemented NLP techniques, is also incorporated.

(I) *API Processing.* The purposes of API processing are two-fold: (1) prepare clinical text before sending them to external APIs and (2) process results returned from external APIs. Given the difference between multiple APIs, data processing

is inevitable to achieve interoperability. Specific API processing tasks include formatting clinical text for APIs requests, filtering results returned from APIs, and data conversion.

(II) *Privacy.* Privacy protection is a critical issue in clinical data sharing for both research and clinical practices, and privacy violations often incur legal problems with substantial consequences. The privacy component embedded in the infrastructure offers technical solutions to deidentify or anonymize patient-level data, such as CRATE (Cardinal, 2017) and DEDUCE (Menger et al., 2018b). CRATE is an open-source software system that anonymizes an electronic health records database to create a research database with anonymized patients' data. With CRATE implemented, our approach can directly use patients' data. In comparison with CRATE, DEDUCE is more lightweight. As a Python package, it processes sensitive patient information with commands like *"deduce.deidentify_annotations()."*

(III) *Security.* The security component controls the access of clinical data and all external APIs. Authentication and encryption are added to safeguard data sharing via the Internet.

(IV) *(Optional) Local NLP Tasks.* As discussed previously, an external NLP API grants no control of what NLP techniques to employ. In case some specific NLP techniques are required, our local NLP technique component provides a choice of implementing your own NLP techniques locally in a preferred language.

**NLP Pipelines**

This layer provides a list of NLP services from which clinical applications can select the most suitable ones on demand. First of all, differences among NLP API providers in terms of their available NLP services are apparent. However, as shown in Table 3.1, there are also a number of common NLP services. Secondly, systematic studies have summarized some commonly used NLP techniques in clinical NLP applications (Kreimeyer et al., 2017). By combining the common NLP services of various APIs and the useful NLP techniques in clinical settings, a shortlist of NLP services is selected for the architecture.

Moreover, multiple NLP services from different APIs can be integrated either sequentially or hierarchically for a single clinical NLP task. This enables clinical NLP applications to address the limitations of individual APIs caused by particular NLP

| NLP API | Available NLP Services |
|---|---|
| IBM Watson NLU | Entity extraction, concept extraction, relation extraction, text classification, language detection, and sentiment analysis |
| Aylien | Article extraction, entity extraction, concept extraction, summarization, text classification, language detection, semantic labeling, sentiment analysis, hashtag suggestion, image tagging, and microformat extraction |
| Lexalytics | Sentiment analysis, concept extraction, categorization, named entity extraction, theme extraction, and summarization |
| Meaning Cloud | Topic extraction, text classification, sentiment analysis, language detection, and linguistic analysis (POS tagging, parsing, and lemmatization) |
| Alchemy API | Entity extraction, concept tagging, keywords extraction, relation extraction, text classification, language detection, sentiment analysis, microformat extraction, feed detection, and linked data |
| TextRazor | Entity extraction, disambiguation, linking, keywords extraction, topic tagging, and classification |
| Developer Cloud | Concept extraction, translation, personality insights, and classification |
| Open Calais | Entity extraction, relation extraction, and sentiment analysis |
| Dandelion API | Entity extraction, text classification, language detection, sentiment analysis, and text similarity |
| Haven OnDemand | Autocomplete, concept extraction, document categorization, entity extraction, language detection, sentiment analysis, and text tokenization |

**Table 3.1:** NLP services of common NLP API providers.

techniques implemented and data employed to build it. More importantly, having a configurable NLP pipeline brings scalability and flexibility. For instance, a clinical concepts extraction enabled application can support combining entity extraction service from two or more of the NLP APIs in Table 1. However, interoperability between

different NLP APIs becomes a challenge as both their inputs and outputs might vary considerably. Therefore, the NLP pipelines contain an integration component which facilitates the interoperability by implementing a proper integration strategy.

#### Apps

In the application layer, clinical NLP-enabled applications for various needs can be created. They are produced either for performing a specific NLP task such as extracting diagnoses from discharge summaries and identifying drugs and dosage information from medical records or with a general purpose of processing unstructured clinical text. Existing NLP applications in clinical domains are categorized into the following groups:

(I) *Concept Extraction.* Kreimeyer et al. conducted a systematic literature review of NLP systems constructed to extract terms from clinical documents and map them to standardized clinical codes (Kreimeyer et al., 2017).

(II) *Text Classification.* Classification of free text in electronic health record (EHR) has surfaced as a popular topic in clinical NLP research. Koopman et al. devised a binary classifier to detect whether or not death is related to cancer using free texts of death certificates (Koopman et al., 2015). Other text classification examples in clinical settings cover classifying a complete patient record with respect to its eligibility for a clinical trial (Ni et al., 2015), categorizing ICU risk stratification from nursing notes (Marafino et al., 2015), assessing inpatient violence risk using routinely collected clinical notes (Menger et al., 2019), and among others.

(III) *Sentiment Analysis.* Unlocking the subjective meaning of clinical text is particularly helpful in psychology. A shared task for sentiment analysis of suicide notes was carried out as an i2b2 challenge (Pestian et al., 2012).

### 3.3.2 Prototype: API-Based Clinical Concept Extraction

To evaluate the architecture, a prototype that extracts clinical concepts from clinical free texts has been developed. This section first illustrates the design of its main components. Then, the prototype itself is presented.

### External NLP APIs

As described above, web NLP APIs have gained wide popularity over the last few years. Both academics and companies recognized the importance and extended their NLP systems with web APIs. As shown in Table 3.2, the prototype incorporates six leading NLP APIs from both academia and industry in its implementation. The selection is based on three criteria: (1) free or free trial available; (2) industrial APIs supported by big companies/teams; (3) academic APIs verified by peers.

| API | Fee | Company/team | References |
|---|---|---|---|
| IBM Watson NLU | Free trial | IBM | https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/ |
| MeaningCloud | Free trial | MeaningCloud LLC | https://www.meaningcloud.com/developer/documentation |
| Open Calais | Free trial | Thomson Reuters | http://www.opencalais.com/opencalais-api |
| Haven OnDemand | Free trial | Hewlett Packard | https://dev.havenondemand.com/apis |
| TextRazor | Free trial | TextRazor Ltd. | https://www.textrazor.com/docs/rest |
| Dandelion API | Free trial | Spaziodati | https://dandelion.eu/docs/ |

**Table 3.2:** NLP APIs selected for the prototype.

### NLP Technique Implemented Locally

Studies have revealed that negation is very common in clinical reports (Chapman et al., 2001; Mehrabi et al., 2015). For instance, "no fracture," "patient denies a headache," and "he has no smoking history" often appear in clinical texts. In order to correctly extract clinical terms, negation detection becomes inevitable. However, given that most of the selected NLP APIs are tools for text processing and analysis in the general domain, the negation issue of clinical documents is not properly tackled, and they cannot filter out irrelevant information. Therefore, negation detection is implemented locally for the prototype. As the most well-known negation detection algorithm, NegEx has been adopted by a number of biomedical applications (Chapman et al., 2003; Meystre and Haug, 2006; Mitchell et al., 2004). We implemented the algorithm to handle negation in this prototype.

### API Processing

NLP APIs first extract clinical terms which will be filtered by the local negator. Then the UMLS API transforms the filtered clinical terms to the standardized codes, such as ATC codes, ICD-10, or SNOMED, which ensures that the extracted clinical terms are interoperable after integration.

For each extracted term, the UMLS API returns its top 10 matched codes. These top matches are ranked on their similarity to the extracted term, with the first as the most similar one. The prototype captures the unique identifier of each matched code for later use.

As discussed above, when multiple APIs are applied for one task, results need to be integrated. The prototype employs a double weight system to integrate multiple APIs. The first weight system determines whether an extracted term is similar to another extracted term from the same document. The weight of a pair of two extracted terms is calculated based on their top 10 matches from the UMLS API and then is compared with the similarity threshold $\gamma$ ; if the weight is higher than the threshold, we consider it to be an equal term. The weight formula is shown as follows:

$$\frac{\alpha}{4} + \frac{3\beta}{4} > \gamma$$

where $\alpha$ refers to the percentage of equal terms over all 10 terms and $\beta$ is the percentage of equal terms over the top 3 terms. $\alpha$ and $\beta$ are calculated based on the UMLS API matches of two extracted terms. The weight is a value between 0 and 1, 0 being that the terms are not similar at all and 1 being exactly the same. For a given NLP task, an initial value of $\gamma$ = 0.1 is recommended, and then according to the number of false positives and false negatives, we adjust the value of $\gamma$ to achieve optimal output. The strategy of tuning these parameters is discussed further in Section 3.3.

The second weight system determines whether an extracted clinical term has enough cumulative weight from all NLP APIs. Since the performance of NLP APIs varies, a weight for each individual API is estimated by using the F1 scores calculated after testing each API on a small subset of clinical documents. The F1 score for each API is normalized to an extractor-weight $\omega$ . For each clinical term extracted, we sum the weights of the extractors the term was extracted by. If the weight is over the extractor threshold $\theta$, it is considered to be actually extracted. If it is less, it is considered to be a false extraction. The weight is computed as follows:

$$\sum_{i=1}^{n} \omega_i \tag{3.4}$$

where $\omega$ is the weight of an NLP API and n refers to the number of API used. The pseudocode of the integration process is shown in Figure 3.3.

**Algorithm 1** Multiple APIs Integration Algorithm

**Input:** $X = [X_1, X_2, ..., X_n]$: returns of n APIs;
$W = [\omega_1, \omega_2, ..., \omega_n]$: weights of n APIs;
$\gamma$: similarity threshold;
$\theta$: extractor threshold;
**Output:** $T$: a list of clinical terms
*Initialisation* : $\omega_\alpha = 0.25$ and $\omega_\beta = 0.75$
*Filter out same/similar terms extracted by one API*
**for** $i = 1$ to $n$ **do**
  **for** $x_a$ in $X_i$ **do**
    Get the rest of terms: $X_j = X_i - X_a$
    **for** $x_b$ in $X_j$ **do**
      calculate the percentage of equal terms over all 10 terms: $\alpha$
      calculate the percentage of equal terms over top 3 terms: $\beta$
      calculate the pairwise similarity: $\delta = \omega_\alpha * \alpha + \omega_\beta * \beta$
      **if** $\delta \geq \gamma$ **then**
        discard same/similar term: $X_i = X_i - X_b$
      **end if**
    **end for**
  **end for**
**end for**
Get filtered arrays of terms: $X_\delta = [X_{1\delta}, X_{2\delta}, ..., X_{n\delta}]$
*Filter out extracted terms by the weights over all APIs*
Compute weights over all APIs: $X_w = \sum_{i=1}^{n} X_\delta W$
**for** $w_{sum}$, $x$ in $X_w$ **do**
  **if** $w_{sum} \geq \theta$ **then**
    Add the term the final list: $T+ = [x]$
  **end if**
**end for**
**return** $T$

**Figure 3.3:** The pseudocode of the integration process.

### Prototype

Figure 3.4 shows the overall functional components of the prototype, which is an instantiation of the proposed architecture. The prototype is a web application with a minimalistic user interface, developed with HTML5, CSS, JavaScript, and PHP for the back end. Given that many existing NLP APIs use JSON as the default format, JSON is the chosen format for data transferring between different components. Figure 3.5 presents a screenshot of the application. Users need to provide clinical documents they want to process in the upper input field and then select APIs and coding stan-

dards. After clicking the Extract button, the results will be displayed in the table at the bottom. "Diseases Remote" lists the extractions of external NLP APIs, while "Diseases Local" represents results of combining external NLP APIs, the local negation handler, and the UMLS API. Unfortunately, the application is not accessible online due to a lack of API token management. Sharing our tokens online might incur a charge when there are a large number of API requests. Nevertheless, researchers are able to deploy their own version of the system with the source codes we share on GitHub at https://github.com/ianshan0915/MABNLP. A demo video is also available at https://youtu.be/dGk9NQGWYfI.

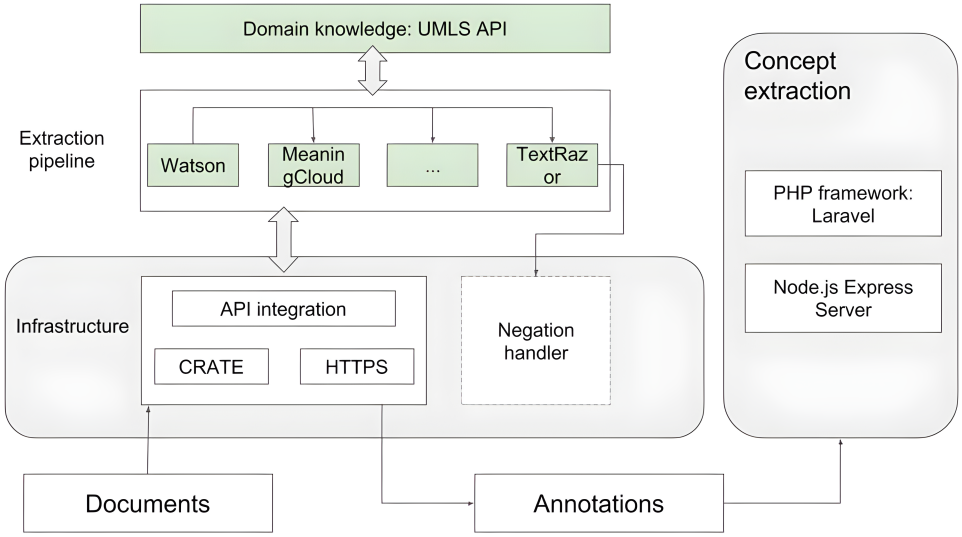



**Figure 3.4:** Prototype architecture.

### 3.3.3 Evaluation Results

As explained before, the prototype comes with three hyperparameters that adjust the extraction outputs: negation $\kappa$, term similarity threshold ( $\gamma$ ), and extractor threshold $\theta$. The hyperparameter tuning was manually conducted by the researchers in the experiments.

The impacts of the controlling hyperparameters on the outputs of our experiments vary. First of all, negation surprisingly shows little positive influence as shown in Table 3.3. Its main reason probably lies in the fact that the implemented negation algorithm, NegEx, only uses negation cue words without considering the semantics

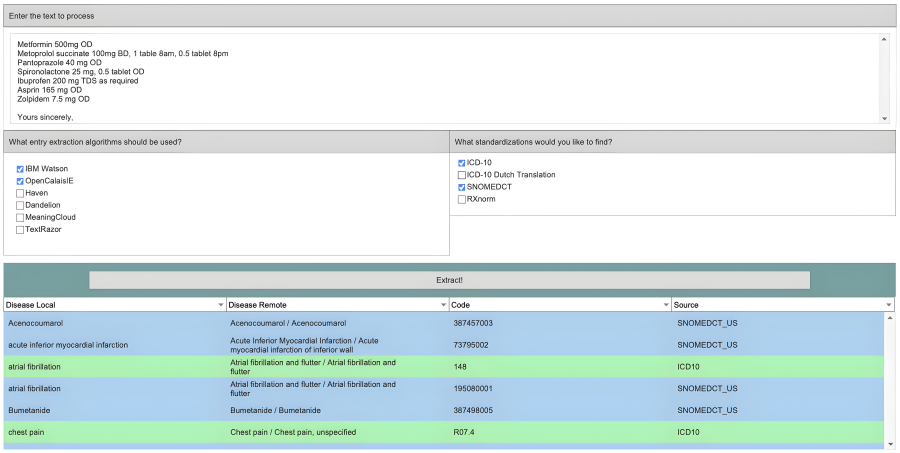



**Figure 3.5:** Prototype user interface of the multiple NLP API extraction pipeline. A demo video and source code are available online.

of a sentence (Mehrabi et al., 2015). Implementation of more advanced algorithms, such as DEEPEN and ConText, will be conducted in future research. The higher $\gamma$ value means a higher similarity threshold for entities to be merged, which results in a lower false positive and higher false negative numbers. By increasing the $\theta$ value, we want entities to be extracted by more APIs, and subsequently lower the number of false positives and increase the number of false negatives. However, higher values bring down the number of true positives. The aim is to strive for the best combination of these hyperparameters for each specific NLP task. The experiments suggested that the values of $\gamma = 0.1$ and $\theta = 0.35$ are a decent starting point for further exploration.

Results have shown that the performance of the prototype is not consistent. Datasets like the obesity challenge can rely on our approach, but its reliability on datasets, such as the medication challenge and OPERAM dataset, need further improvement and evaluation.

Many NLP systems have been tested on the two i2b2 datasets, and there are benchmark performance metrics being published in the literature (Uzuner, 2009; Uzuner et al., 2010). We calculated the averages of top 5 best systems as the baselines. As displayed in Table 3.4, the prototype performs well and has great potential of being adopted for clinical concept extraction. In case of the OPERAM dataset, there is no benchmark. Therefore, its performance is evaluated from an expert intervention perspective. By comparing the automated extracted clinical concepts with the anno-

| Dataset | Negation ($\kappa$) | Recall | Precision | F1 score |
|---|---|---|---|---|
| Obesity challenge | True | 0.733 | 0.939 | 0.823 |
| | False | 0.805 | 0.925 | 0.861 |
| Medication challenge | True | 0.62 | 0.835 | 0.712 |
| | False | 0.636 | 0.838 | 0.724 |
| OPERAM medical conditions | True | 0.594 | 0.271 | 0.373 |
| | False | 0.594 | 0.271 | 0.373 |
| OPERAM medications | True | 0.795 | 0.816 | 0.805 |
| | False | 0.795 | 0.816 | 0.805 |

**Table 3.3:** Impact of negation from the experiments.

tations, we estimate how well the prototype can be used to assist physicians during their manual extraction process. Unfortunately, feedback from physicians indicate that the prototype is not yet considered practically useful. Firstly, its poor performance in extracting medical conditions requires physicians to spend more time filtering out incorrect extractions. Secondly, the prototype fails to identify the associated dosages and frequencies of medications.

## 3.4 Discussion

We argue that outsourcing NLP tasks offers efficient NLP solutions for processing unstructured clinical documents. To begin with, outsourcing often leads to a reduction of both IT development and maintenance costs. Furthermore, a lower level of NLP expertise is required when external NLP services are used. A developer with limited knowledge of NLP could develop a clinical NLP application such as our prototype. Lastly, the architecture supports NLP services beyond clinical concept extraction. By adding a sentiment analysis NLP pipeline constructed by external NLP APIs, our prototype can perform sentiment analysis on clinical documents. For instance, changing from concept extraction to sentiment analysis can be accomplished by adjusting the

| Dataset | $\kappa$ | $\gamma$ | $\theta$ | Recall | Precision | F1 score |
|---|---|---|---|---|---|---|
| Obesity challenge | False | 0.1 | 0.2 | **0.805** | **0.925** | **0.861** |
| Baseline* | | | | 0.771 | 0.815 | 0.787 |
| Medication challenge | False | 0.1 | 0.35 | 0.636 | 0.838 | 0.724 |
| Baseline* | | | | **0.794** | **0.845** | **0.818** |
| OPERAM medical conditions | True | 0.1 | 0.5 | 0.594 | 0.271 | 0.373 |
| OPERAM medications | False | 0 | 0.35 | 0.795 | 0.816 | 0.805 |

* Average of the top 5 best systems from the challenge.

**Table 3.4:** Overall results on three datasets.

API request parameters from ""features": "entities"" to ""features": "sentiment"."

### 3.4.1 Evaluation Results

In comparison with the popular biomedical NLP component collections listed in (Przybyła et al., 2016), the main advantage of our proposed approach is its lightweight nature. The popular component collections, such as cTAKES, Bluima, and JCoRe, require an intensive IT resources investment including Java developers, NLP specialists with experience in the UIMA framework, and local hardware support. On the contrary, clinical institutions could start to process unstructured text with as little resources as possible due to the fact that our cloud-based approach outsources NLP to external NLP services. Moreover, Bluima has not been updated for four years. Instead of replacing the popular NLP tools, our approach should be considered as an alternative approach in the face of time and resource constraints.

### 3.4.2 Error Analysis

An error analysis has been carried out in order to better understand the performance of the prototype. As explained in Section 2.2, there are two types of errors, namely, FPs and FNs. Figure 3.6 shows the percentage of FP and FN errors in all experiments. First of all, one major source of errors in the two i2b2 datasets is false negatives, which means many annotated terms in the datasets are not extracted by our prototype. The high proportion of FNs is in great part attributed to the entity-type detection errors.

Since some NLP APIs (MeaningCloud and Open Calais) are unable to extract pharmaceutical drug entities, it results in a lower amount of extracted entities and higher false negatives. Therefore, to enhance the performance, NLP APIs such as MeaningCloud and Open Calais might as well be excluded.

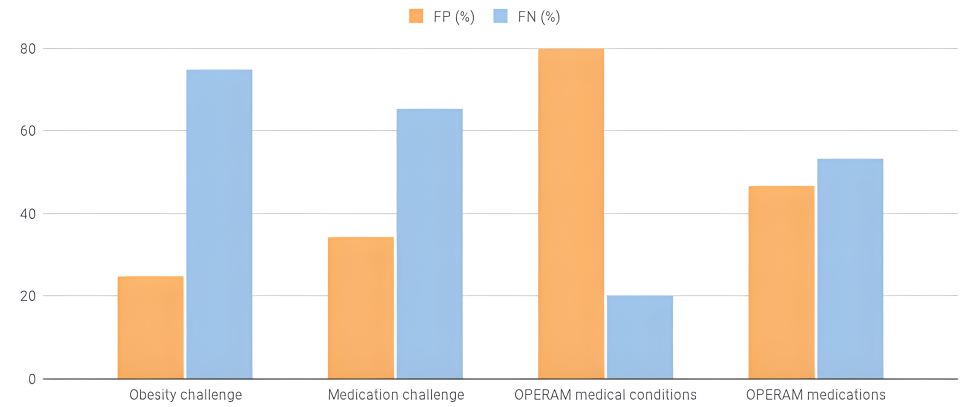



**Figure 3.6:** Error distribution of all the experiments, false positives vs false negatives.

Nevertheless, the higher number of false positives led to an overall performance loss in the OPERAM medical conditions extraction. We found out that the problem lies in the annotation. For example, the sentence "Fall during the night, multiple hematomas. Orthostatic hypotension proven." contains two medical conditions: hematoma and orthostatic hypotension. Hematoma was found by two out of six extractors; orthostatic hypotension was found by five out of six. However, neither of these two was annotated, most likely because the context of the sentence was in past tense and potentially not applicable to the current state of the patient.

### 3.4.3 Limitations and Future Research

There are a number of hurdles that prevent the adoption of our approach in daily practice. Further research is necessary to sufficiently address these concerns. First of all, practical implementation requires a more thorough privacy and security component. The privacy and security component is part of the proposed architecture and currently implemented in the prototype using CRATE (Cardinal, 2017) and HTTPS. However, since only anonymized datasets are used in the evaluation, the deidentification toolkit, CRATE, was not validated. Before the practical adoption, we need to

first evaluate the performance of the privacy and security component with real-world clinical data.

Another concern lies in the computational efficiency of our approach, namely, execution time. As shown in the demo video, it takes about 20 seconds to process a discharge letter. In specific, the majority of time (15 seconds) goes to annotation in which extracted terms are first encoded with UMLS and then pairwise similarity between them is calculated. Since the prototype was running locally on a laptop with 8 GB RAM, we think it would become faster if we implement it on a larger server.

In practice, clinical NLP is employed to solve various clinical problems, ranging from entity extraction to cohort detection. Our research demonstrates that the proposed approach performs well on clinical concept extraction. It is crucial to conduct further evaluation on other tasks, such as cohort detection and sentiment analysis before adopting the approach in practice.

Last but not least, due to the wide adoption of health information systems (HIS) in healthcare institutions, developing a simple method that supports the integration of our approach with HIS would facilitate its implementation.

## 3.5 Conclusion

The proposed NLP architecture offers an efficient solution to develop tools that are capable of processing unstructured clinical data in the healthcare industry. With our approach, less time and resources are required to create and maintain NLP-enabled clinical tools given that all NLP tasks are outsourced. Moreover, the prototype built upon the approach produces satisfactory overall results, and its performance on certain datasets indicates that its practical application in clinical text processing, particularly clinical concept extraction, is promising. Nevertheless, high variance among different datasets brings concerns on its generalization and practicability.

# Chapter 4

# Automatic Extraction of Adverse Drug Reactions from SmPC

The summary of product characteristics from the European Medicines Agency is a reference document on medicines in the EU. It contains textual information for clinical experts on how to safely use medicines, including adverse drug reactions. Using natural language processing (NLP) techniques to automatically extract adverse drug reactions from such unstructured textual information helps clinical experts to effectively and efficiently use them in daily practices. Such techniques have been developed for Structured Product Labels from the Food and Drug Administration (FDA), but there is no research focusing on extracting from the Summary of Product Characteristics. In this work, we built a natural language processing pipeline that automatically scrapes the summary of product characteristics online and then extracts adverse drug reactions from them. Besides, we have made the method and its output publicly available so that it can be reused and further evaluated in clinical practices. In total, we extracted 32,797 common adverse drug reactions for 647 common medicines scraped from the Electronic Medicines Compendium. A manual review of 37 commonly used medicines has indicated a good performance, with a recall and precision of 0.99 and 0.934, respectively.

This work was originally published as: Shen Z, Spruit M. Automatic Extraction of Adverse Drug Reactions from Summary of Product Characteristics. *Applied Sciences*. 2021; 11(6):2663. https://doi.org/10.3390/app11062663

## 4.1 Introduction

Drug product labels are regulatory documents required as part of the marketing authorization of each medicine. They provide up-to-date and comprehensive information about the risks, benefits, and pharmacological properties of marketed medicines. As such, extracting the clinical knowledge stored in product labels and making it available in the form of computationally accessible knowledge bases would benefit several applications in the area of drug safety surveillance and assessment. For example, during post-marketing safety assessments, it is crucial to determine whether an investigated adverse drug reaction (ADR) is already labeled (Banda et al., 2016; Roberts et al., 2017).

There are two main versions of drug product labels around the world: Structured Product Labels (SPL) introduced by the Food and Drug Administration (FDA) in the US, and the Summary of Product Characteristics (SmPC) supervised by the European Medicines Agency (EMA) in the EU (Shekhani et al., 2020). Both product labels provide information in sections, and the content of each section is in free narrative texts. This study focuses on the undesirable side effect section of product labels which describes adverse drug reactions (ADRs).

To date, there are no structured machine-readable ADRs. Therefore, extracting ADRs from unstructured product labels becomes an interesting research topic. There are many studies focusing on transforming unstructured ADRs data into structured machine-readable data (Ly et al., 2018; Fung et al., 2013; Wu et al., 2019; Demner-Fushman et al., 2018; Kuhn et al., 2016). However, the majority of current studies focus on the US version of product labels. For example, systems such as SPLICER and SPL-X were developed to extract ADR terms from particular sections using various natural language processing (NLP) techniques, including named entity recognition, rule-based parsing, and NegEx (Ly et al., 2018). Fung et al. (2013) proposed to use open-source NLP tools to extract drug indication information from SPL. Wu et al. managed to fetch ADR terms using Oracle Text search from 1164 single-ingredient medicines (Wu et al., 2019). A variety of NLP techniques were designed to extract ADRs from SPL in the 2017 Text Analysis Conference track (Roberts et al., 2017).

However, the FDA SPL is the main data source used in such studies. Until now, the SmPC, as the European equivalent to SPL, has barely been investigated and no method has been developed solely for extracting ADRs from the SmPC. This study contributes to the field as a starting point.

In this study, the main research question we are addressing is how to utilize NLP

techniques to automatically extract ADRs from standardized European product labels, namely SmPC. To answer the question, we first develop an NLP pipeline to extract adverse drug reactions from SmPC. A knowledge base is created from the extracted terms. The main characteristics of the NLP pipeline and the knowledge base are summarized as follows: (a) open-source and reproducible; (b) customizable for related ad hoc tasks; (c) knowledge base applicable for clinical studies.

The paper is structured as follows. section 4.2 defines a number of methods and materials used in our study. Then the results are presented in section 4.3. Discussions in section 4.4 summarize the main findings of this research and some limitations. section 4.5 concludes the study.

## 4.2 Materials and Methods

In this section, we describe the SmPC data source, elaborate on the various NLP techniques that are applied to extract ADRs from SmPC, and briefly explain the evaluation setup for validating our automatic ADR extraction approach. A conceptual overview of the ADR extraction pipeline is illustrated in Figure 4.1. The first step is to scrape the side effects data from the Electronic Medicines Compendium (EMC). Then the ADRs are extracted accordingly. Finally, the performance of our NLP method is evaluated by both NLP and clinical experts.

### 4.2.1 Dataset

All SmPCs were scraped from the EMC which provides the most recent and openly accessible regulated and approved information about medicines licensed in the UK (Electronic Medicines Compendium (EMC), 2020). The EMC was selected as our data source for two main reasons: (1) all information is in English; (2) it has more than 14,000 documents, all of which have been checked and approved by the European government agencies. Figure 4.2 shows an example of the side effects section of SmPC.

### 4.2.2 Scraping the Side Effects Section of SmPC

Since this study focuses on commonly used medications, we only scrape the SmPCs of a limited number of medicines in the EMC. As shown in Figure 4.1, the process of scraping side effects data starts with identifying active substances. Then, for

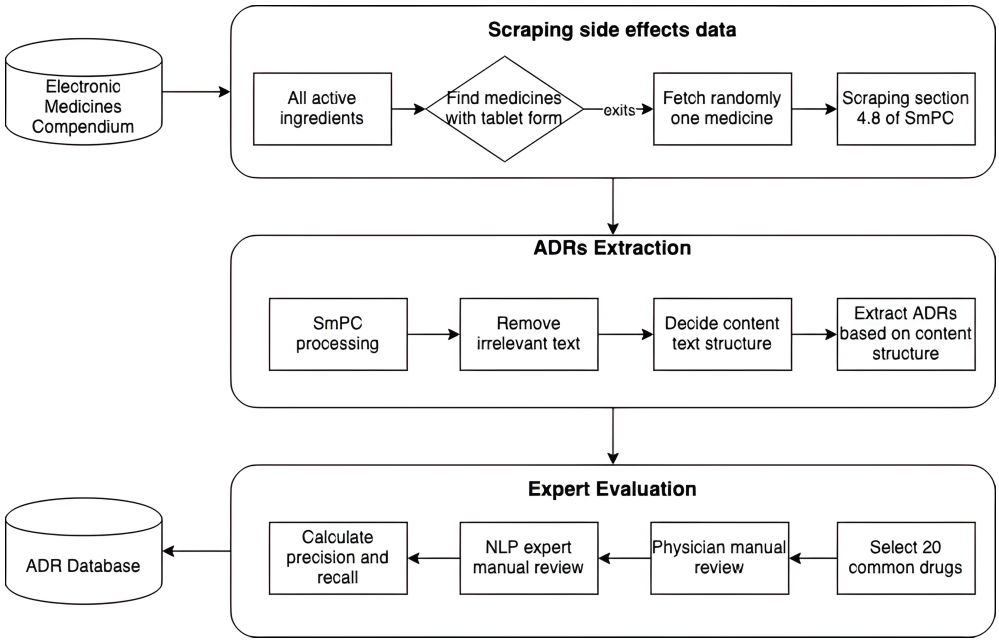



**Figure 4.1:** A conceptual overview of the automatic Adverse Drug Reactions (ADRs) extraction pipeline.

each active substance, only one medication in tablet form is included in the final list which contains 647 medicines, because the tablet form is the most common medication route. When there is no medicine in tablet form for a given active substance, none is selected. At last, section 4.8 Undesirable effects of the SmPC is obtained for all 647 medications in the final list. The 4.8 section consists of ADR information in both plain text and structured tables, we scraped the available information in HTML format so that the data structure is kept. All data is stored as a JSON file, which is used in the later steps.

### 4.2.3 ADR Extraction

#### SmPC Processing

In the second phase of our NLP pipeline is ADR extraction. It starts with processing the above-mentioned ADR relevant HTML excerpts. As we described before, the section 4.8 in the SmPC contains ADR information in diverse formats: plain text, structured text, and tables. Tables used in describing ADR information also come

**4.8 Undesirable effects**

Summary of the safety profile

Headache, abdominal pain, diarrhoea and nausea are among those adverse reactions that have been most commonly reported in clinical trials (and also from post-marketing use). In addition, the safety profile is similar for different formulations, treatment indications, age groups and patient populations. No dose-related adverse reactions have been identified.

*Tabulated list of adverse reactions*

The following adverse drug reactions have been identified or suspected in the clinical trials programme for esomeprazole and post-marketing. None was found to be dose-related. The reactions are classified according to frequency (very common > 1/10; common ≥1/100 to <1/10; uncommon ≥1/1000 to <1/100; rare ≥1/10000 to <1/1000; very rare <1/10000); not known (cannot be estimated from the available data).

**Blood and lymphatic system disorders**

Rare: Leukopenia, thrombocytopenia

Very rare: Agranulocytosis, pancytopenia

**Immune system disorders**

Rare: Hypersensitivity reactions e.g. fever, angioedema and anaphylactic reaction/shock

**Metabolism and nutrition disorders**

Uncommon: Peripheral oedema

Rare: Hyponatraemia

Not known: Hypomagnesaemia (see section 4.4); severe hypomagnesaemia can correlate with hypocalcaemia. Hypomagnesaemia may also be associated with hypokalaemia

**Psychiatric disorders**

Uncommon: Insomnia

Rare: Agitation, confusion, depression

Very rare: Aggression, hallucinations

**Figure 4.2:** An example of a Summary of Product Characteristics (SmPC) excerpt.

with different structural styles. Therefore, we classify the 4.8 section into separate categories in terms of its content structure. Figure 4.3 elaborates on the process of categorization. Each category has its own ADR extraction technique.

First, we identify a list of structural features in the HTML excerpts, for instance, the counts of '<table>', the counts and positions of some MedDRA terms (Mozzicato, 2007), such as the frequency and SOC (System Organ Classes) (MedDRA, 2020). Table 4.1 shows some specific examples of such words. Secondly, the features are engineered from the scraped HTML files. Based on the features, we obtain three main structural categories: structured text, free text, and tabular.

- Free-text: ADRs are explained in free-text, as shown in Figure 4.4a. ADRs are hidden in sentences, which thus requires named entity recognition (NER) to extract them.
- Structured text: ADRs are described in a structured text, as shown in Fig-

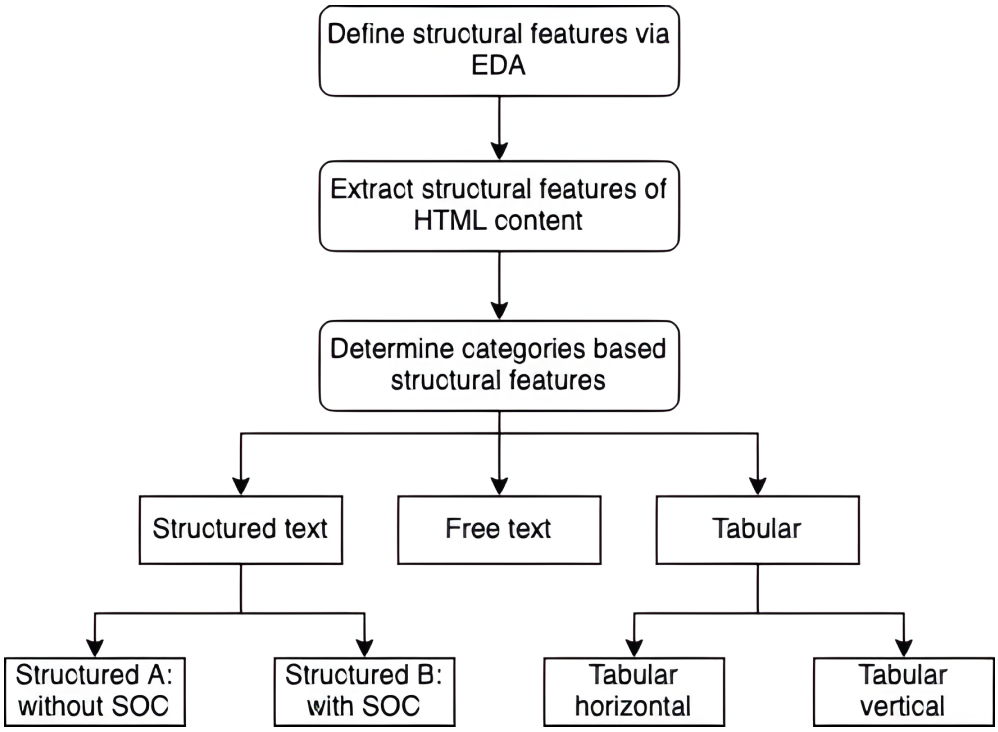



**Figure 4.3:** A decision tree for classifying the ADR content during SmPC processing.

ure 4.4b. It depicts a list of ADRs and their frequencies in an organized format.

- Tabular: ADRs are presented in a very structured way, namely a table. Figure 4.4c shows an example of such tables. These tables are further split into two groups: tabular horizontal and tabular vertical.

### ADR Extraction

In the previous section, we identify three main structural categories that require different ADR extraction techniques, respectively. An overview of the extraction pipeline for each category is illustrated in Figure 4.5.

To uncover the hidden ADRs in the free-text, entity extraction techniques are applied. Specifically, we simplify the extraction process by using the IBM Watson Natural Language Understanding API that offers sophisticated NLP techniques in extracting meta-data from content such as concepts, entities, keywords, and others.

| Term Type | Description | Examples |
|---|---|---|
| Frequency | Terms that describe how often an ADR happens | Very common<br>Common<br>Uncommon<br>Rare<br>Very rare<br>Not known |
| SOC | The highest level of groupings in MedDRA. There are 27 groups which are defined by etiology (Electronic Medicines Compendium (EMC), 2020). | Immune system disorders<br>Metabolism and nutrition disorders<br>Nervous system disorders<br>Eye disorders<br>Cardiac disorders<br>Gastrointestinal disorders<br>Skin and subcutaneous tissue disorders<br>General disorders and administration site conditions<br>Investigations |

**Table 4.1:** Examples of ADR-specific terminologies.

Benchmarking studies have shown that IBM Watson Natural Language Understanding API is a simple and useful NLP tool in solving various clinical NLP problems (Canonico and De Russis, 2018; Shen et al., 2019a). A demonstration of how the API works is available (IBM, 2021b). Since the number of ADR terms is relatively limited, we can use the API for free. It offers a lite account which can process 30, 000 items per month free of charge (IBM, 2021a). Frequencies of the extracted ADRs are assigned as unknown due to their absence in the free texts.

For the structured text, we extract ADR terms and their corresponding frequencies with syntax parsing. The syntactic structure of a structured HTML text is depicted by the occurrences and positions of SOC (System Organ Class) and frequency terms. With the identified positions of SOC and frequency terms, we can obtain the HTML elements that contain only ADR terms. In most cases, ADRs terms in an HTML element are separated by a comma. Thus, by splitting the string by comma, we extract several ADRs for a given SOC and frequency. An example of such ADR extraction is illustrated in Figure 4.6. The left is the structured text in HTML and the right side shows the extracted ADRs in JSON.

The tabular text refers to HTML tables that describe ADRs of a given medicine in a two-dimensional grid format. The first step of information extraction for HTML tables is to detect the structure. As mentioned above, we identify two types of struc-

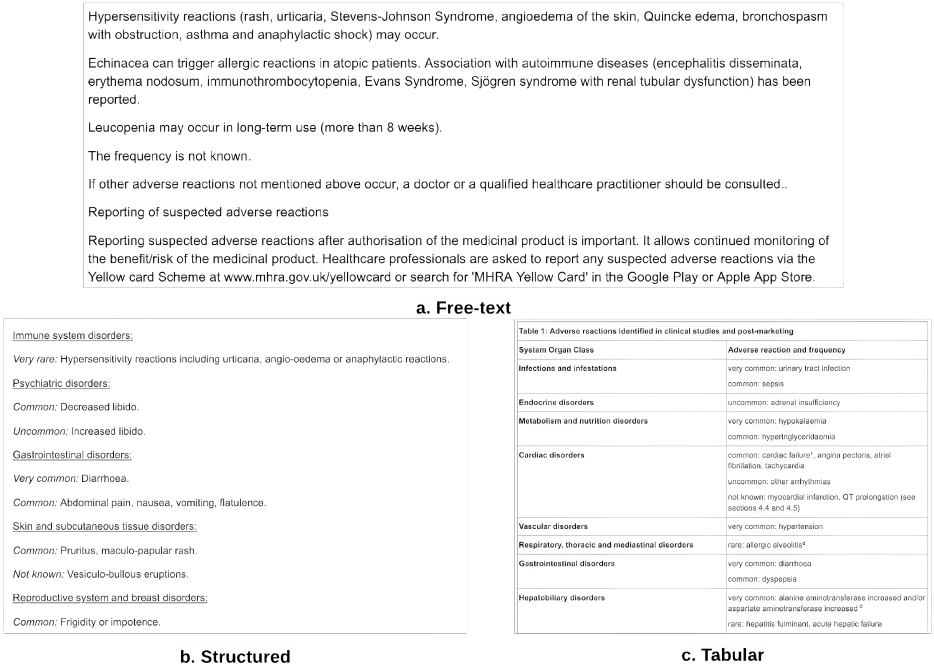



**Figure 4.4:** Example fragments of the three main structural categories: **(a)** free text, **(b)** structured text, and **(c)** tabular text.

tures: vertical and horizontal. Similar to structured text, ADR extraction starts with getting the occurrences and positions of SOC and frequency terms. Then, with the position indexes of SOC and frequency terms, HTML table cells that contain ADR terms are fetched and processed. ADR terms in an HTML table cell are usually separated by a comma. An example is given in Figure 4.7.

### 4.2.4 Evaluation

An expert manual review is performed in the evaluation of ADRs extracted from the SmPCs in this study. To alleviate the burden of manual reviewing for our clinical experts, we only sample a small subset of common medicines. Moreover, an NLP expert is also involved in the manual review. The NLP expert specializes in processing the clinical text and has a sufficient clinical knowledge to properly review the extracted ADRS. In total, 37 medications were selected by the clinical expert for manual review.

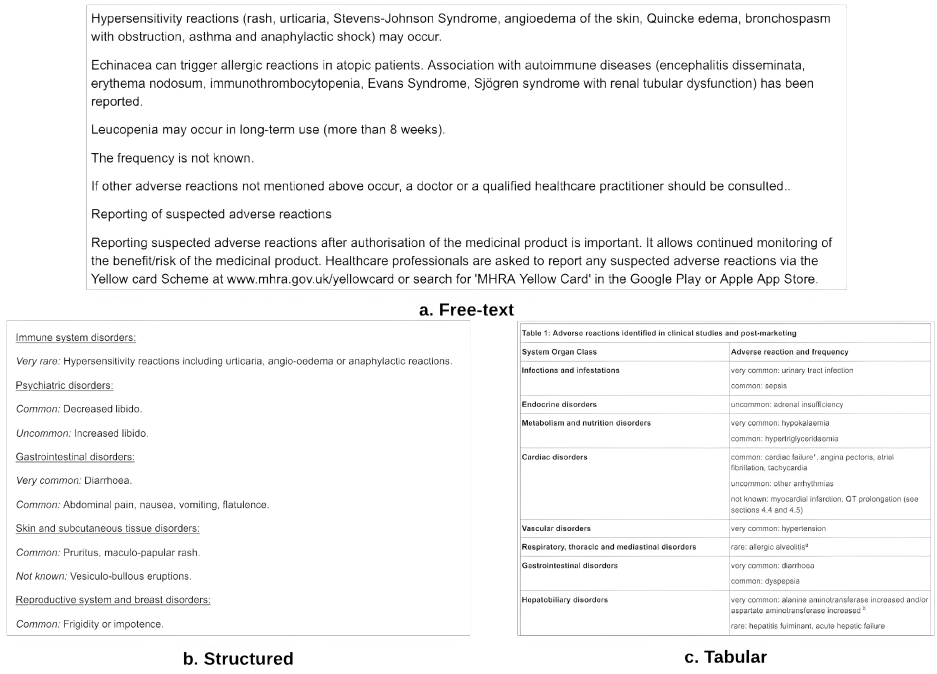



**Figure 4.5:** An Overview of the ADR Extraction Pipeline.

The clinical expert provided a list of top 50 commonly used medicines in Dutch primary care, and 37 of 50 were available in our scraped drug list. At the beginning of the evaluation, both clinical and NLP experts are requested to review five identical medicines to align the manual reviews of different experts. Following that, due to time constraints, the clinical expert is asked to review five more medicines, whereas the NLP expert continues to review 32 more medicines.

For each medicine, all extracted ADRs are examined and labeled as 'correct', 'near', or 'incorrect'. Furthermore, the number of missing ADRs is counted by comparing the extracted ADRs with the original SmPC. Subsequently, we obtain the numbers of true positives (TP), false positives, and false negatives (FN). The ADR terms with only 'correct' label are considered as TP, both 'near' and 'incorrect' are FP, and the missing ADRs are FN. The standard metrics of recall, precision, and F-score are calculated to measure the performance of the proposed automated approach.

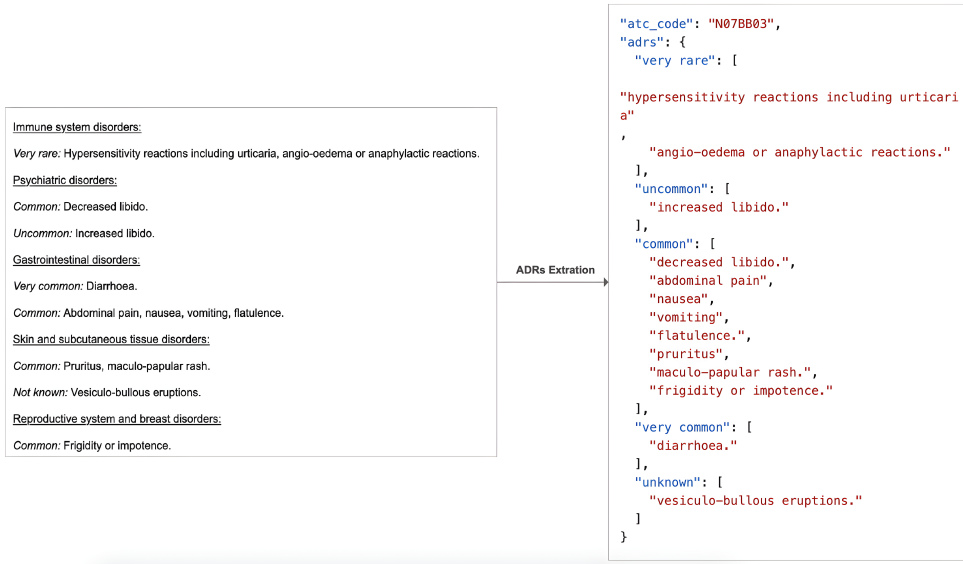



**Figure 4.6:** An example of ADR extraction for the structured text.

## 4.3 Results

This section presents the results of our proposed ADR extraction method and the manual evaluation results.

### 4.3.1 Overview

This study scrapes the SmPC documents of a total number of 647 marketed medicines, from which 32,797 ADR terms and their frequencies are extracted. Among all extracted ADR terms, 8069 are unique. The average number of ADR terms per medicine is 51. Table 4.2 offers a statistical overview of the extraction results from our selected medicines.

### 4.3.2 The Manual Evaluation Results

Clinical experts chose 37 commonly prescribed medicines for manual reviews. Appendix A in the section 4.6 provides a complete list of the 37 reviewed medicines. Among these chosen medicines, there are 28 tabular texts, 7 structured texts, and 2 free texts. As explained before, two experts participated in the manual evaluation. Table 4.3 shows the results of the manual review, including some important perfor-

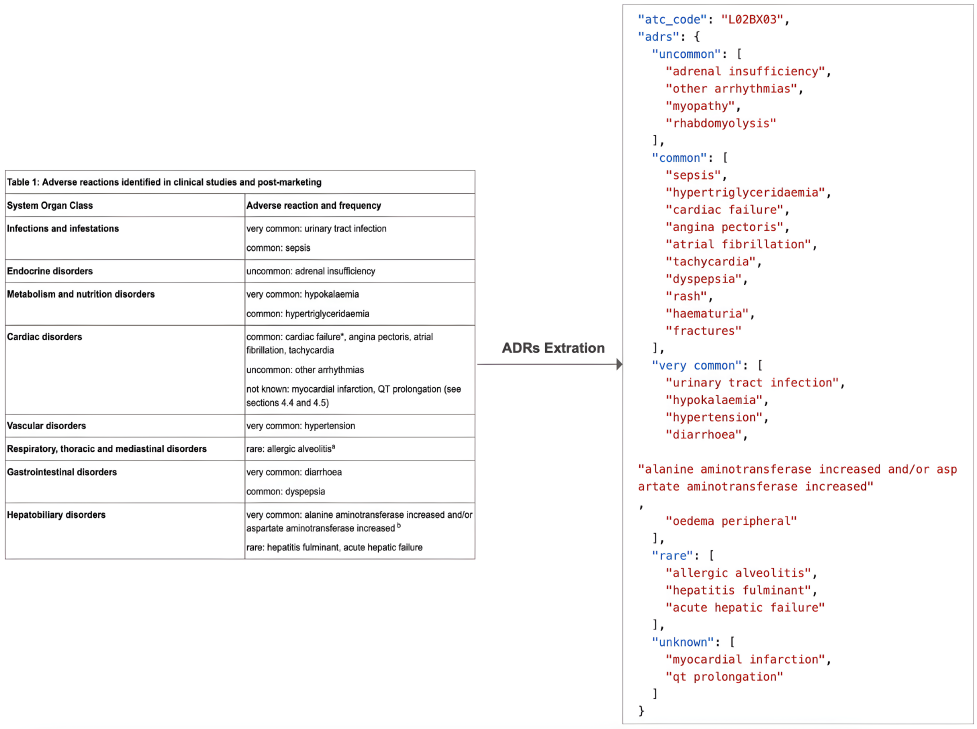


Table 1: Adverse reactions identified in clinical studies and post-marketing

| System Organ Class | Adverse reaction and frequency |
|---|---|
| Infections and infestations | very common: urinary tract infection<br>common: sepsis |
| Endocrine disorders | uncommon: adrenal insufficiency |
| Metabolism and nutrition disorders | very common: hypokalaemia<br>common: hypertriglyceridaemia |
| Cardiac disorders | common: cardiac failure*, angina pectoris, atrial fibrillation, tachycardia<br>uncommon: other arrhythmias<br>not known: myocardial infarction, QT prolongation (see sections 4.4 and 4.5) |
| Vascular disorders | very common: hypertension |
| Respiratory, thoracic and mediastinal disorders | rare: allergic alveolitis[a] |
| Gastrointestinal disorders | very common: diarrhoea<br>common: dyspepsia |
| Hepatobiliary disorders | very common: alanine aminotransferase increased and/or aspartate aminotransferase increased [b]<br>rare: hepatitis fulminant, acute hepatic failure |

ADRs Extration


```
"atc_code": "L02BX03",
"adrs": {
  "uncommon": [
    "adrenal insufficiency",
    "other arrhythmias",
    "myopathy",
    "rhabdomyolysis"
  ],
  "common": [
    "sepsis",
    "hypertriglyceridaemia",
    "cardiac failure",
    "angina pectoris",
    "atrial fibrillation",
    "tachycardia",
    "dyspepsia",
    "rash",
    "haematuria",
    "fractures"
  ],
  "very common": [
    "urinary tract infection",
    "hypokalaemia",
    "hypertension",
    "diarrhoea",

"alanine aminotransferase increased and/or asp
artate aminotransferase increased"
,
    "oedema peripheral"
  ],
  "rare": [
    "allergic alveolitis",
    "hepatitis fulminant",
    "acute hepatic failure"
  ],
  "unknown": [
    "myocardial infarction",
    "qt prolongation"
  ]
}
```


**Figure 4.7:** An example of ADR extraction for the tabular text.

mance scores, such as recall and precision. For each medicine, the number of true positives, false positives, and false negatives are counted. An extracted term is considered as true positive only when it is reviewed as a correct ADR compared with the original text. False positives refer to the extracted terms that are not ADRs. The missing ADRs are labeled as false negatives. The overall recall and precision of all reviewed medicines are 0.99 and 0.932, respectively, which shows the effectiveness of our ADR extraction approach and the reliability of its outputs.

### 4.3.3 Error Analysis

Three main types of errors are identified among the 116 incorrect extracted ADRs of Reviewer 1 (false positives). Table 4.4 summarizes the error analysis of incorrectly extracted ADRs. Our approach encountered issues with splitting multiple ADR terms that are joined by a colon or semicolon. For instance, two ADR terms (agitation and aggression) could be extracted from 'Agitation, aggression'. However, if multiple

| Characteristics | Statistics |
|---|---|
| # of selected marketed medicines | 647 |
| # of medicines with structured text | 141 |
| # of medicines with tabular text | 419 |
| # of medicines with free text | 87 |
| Average ADRs per medicine | 51 |
| 25% percentile ADRs per medicine | 21 |
| 50% percentile ADRs per medicine | 38 |
| 75% percentile ADRs per medicine | 67 |
| Top medicines in terms of extracted ADRs | 1.Topamax 100 mg Tablets (269)<br>2. Revolade 25 mg film-coated tablets (255)<br>3.Capecitabine Accord 150 mg film coated tablets (240)<br>4.Glivec 100 mg film-coated tablets (232)<br>5.Risperdal 0.5 mg Film-Coated Tablets (218)<br>6.Xadago 50 mg film-coated tablets (215)<br>7.Invega 12 mg prolonged-release tablets (207)<br>8.LUSTRAL 100 mg film coated tablets (207)<br>9.Isentress 100 mg chewable tablets (191) |

**Table 4.2:** Overview of the ADR extraction results.

ADR terms are joined by 'or', 'and', or semicolon, such as the examples shown in Table 4.4, our extraction method fails to properly address this. The second type of false-positive results is related to the data cleaning of our approach. The extracted ADRs contain noises such as unrelated words or special characters. As shown in the examples, to get the correct ADRs, we need to clean up noises like 'Ž2020', and 'in combination with insulin or sulphonylurea'. Lastly, a small number of extracted terms are not ADRs at all. For example, 'skin and subcutaneous tissue disorders' is an SoC term

| | Reviewer 1 (NLP Expert) | Reviewer 2 (Clinical Expert) | Totals |
|---|---|---|---|
| # of reviewed medicines | 32 | 5 | 37 |
| # of extracted ADRs terms | 1700 | 118 | 1824 |
| # of correct ADRs (TP) | 1590 | 110 | 1703 |
| # of incorrect ADRs (FP) | 116 | 8 | 124 |
| # of missing ADRs (FN) | 7 | 11 | 18 |
| Recall | 0.996 | 0.909 | 0.99 |
| Precision | 0.932 | 0.932 | 0.932 |

**Table 4.3:** Final results of the manual expert reviews.

that should be excluded from our final ADR lists. Since other incorrect extractions do not belong to any specific group, it is difficult to exclude them. However, the number of such errors is so small that its impact is limited.

| Type of Error | Examples | Extracted Terms | Counts |
|---|---|---|---|
| Unsplit multiple ADTs | "angio-oedema or anaphylactic reactions" | - angio-oedema<br>- anaphylactic reactions | 81 |
| ADR with noise | - 'retinal detachment'<br>- 'hypoglycaemia in combination with insulin or sulphonylurea' | - retinal detachment<br>- hypoglycaemia | 22 |
| Non-ADR | - 'all causality frequency'<br>- 'general disorders:'<br>- 'skin and subcutaneous tissue disorders' | Not ADR | 13 |

**Table 4.4:** Error analysis for false positives.

Since there are only seven missing ADRs out of 1706 extracted ADRs, false-negative (missing ADR) is not a common error. However, we found that such an error is more likely to happen to medicines using free text to describe their ADRs. Since only 87 out of the selected 647 medicines summarize their ADRs in free-text, and the number of unstructured reported medicines has been steadily decreasing with recent SmPC updates, the false-negative error has a relatively limited and decreasing impact on the performance of our approach.

## 4.4 Discussion

Many studies investigated the extraction of ADRs from SPLs from the FDA in the USA, and some effective methods have been developed. However, as an equivalent in Europe, the SmPC from the European Medicines Agency receives very little attention in the field of NLP. In this study, we fill the research gap with the development of an automated ADRs extraction method for the SmPC.

The manual experts review results demonstrate that our proposed method is effective and has the potential of being used to solve ADR related clinical problems. Specifically, our method achieves an overall recall of 0.990 and a precision of 0.932. Such high performance has never been reported in previous studies focusing on extracting ADRs from SPLs (Wu et al., 2019; Pandey et al., 2019).

As discussed in the error analysis, there are a few ways to further improve the performance of our method. For example, noise in the extracted terms can be cleaned with regular expressions. When it comes to the unsplit multiple ADRs, we can extend the split function with more options so that it allows strings to be split by 'or', 'and', or semicolon. Furthermore, encoding extracted ADR terms into MedDRA Preferred Terms could reduce the non-ADR errors, thus further improving the performance.

Our study has some limitations. First, there are no standardized ADR annotations based on the SmPC available to benchmark ADR extraction methods. It is difficult to reproduce the performance of this study. Manual reviews from different experts might present different performance scores.

Another limitation lies in the manual review process of this study. Involving clinical experts in the manual review process is always challenging given their busy schedule. The coronavirus has made the situation even worse. Therefore, the manual process in this study only included one clinical expert. To compensate for this, we added one clinical NLP expert as another reviewer. The NLP expert is familiar with this topic and was trained for the task by the clinical expert. To address this, we plan to expand the manual review to a larger size of samples and a bigger group of clinical experts in a post-COVID world.

The proposed ADR extraction method is developed based on data from the EMC in UK. However, due to Brexit, product labels in UK might changes. Then the method will not perform the same as shown in this study. Further development of this method should focus on using the SmPCs from the EMA.

## 4.5 Conclusions

The contributions of this study are two-fold. First, it contributes to the field of clinical NLP by introducing an open-source and reproducible method that extracts ADR terms from the SmPC. The high-performance scores (recall: 0.99 and precision: 0.934) indicate our approach is very effective, which leads us to believe that the method could be useful in the processing and coding of ADRs in the SmPC. The second contribution lies in the clinical field. The results of our extraction method are structured data that describes marketed medicines and their recorded ADRs and frequencies. Such a knowledge base could be applied to address practical clinical problems, ranging from ADR assessment in clinical trials to assisting the detection of ADRs in patients.

## 4.6 Supplementary Materials

The following materials are available online at https://github.com/ianshan0915/ade-extraction/paper, Appendix A: Appendix A-manual review results.xlsx. Source code of the project is available on https://github.com/ianshan0915/ade-extraction.

# Chapter 5

# Big Data Framework for Biomedical Literature Mining

The massive size of available biomedical literature requires researchers to utilize novel big data technologies in data storage and analysis. Among them is cloud computing which has become the most popular solution for big data applications in industry. However, many bioinformaticians still rely on expensive and inefficient in-house infrastructure to discover knowledge from biomedical literature. Although some cloud-based solutions were constructed recently, they failed to sufficiently address a few key issues including scalability, flexibility, and reusability. Moreover, no study has taken computational cost into consideration. To fill the gap, we proposed a cloud-based big data framework that enables researchers to perform reproducible and scalable large-scale biomedical literature mining in an efficient and cost-effective way. Additionally, a cloud agnostic platform was constructed and then evaluated on two open access corpora with millions of full-text biomedical articles. The results indicate that our framework supports scalable and efficient large-scale biomedical literature mining.

This work was originally published as: Shen, Z., Wang, X., & Spruit, M. (2019, June). Big Data Framework for Scalable and Efficient Biomedical Literature Mining in the Cloud. *In Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval* (pp. 80-86). https://doi.org/10.1145/3342827.3342843

## 5.1 Introduction

A huge amount of biomedical literature has been produced over the decades, and the number is still growing on a daily basis. Furthermore, with the advocacy of open science, an increasing number of publications become openly accessible. For example, as the leading biomedical literature database, PubMed Central (PMC) has archived over 5.3 million research papers, of which around 2.4 million full-text articles are easily accessible at the PMC Open Access Subset (PMC OA Subset) (Pubmed Central, 2019; Pubmed Open Access, 2019). The size of the latest update of full-text articles in the PMC OA Subset is around 100 GB, so storing, processing, and analyzing such massive data is not exactly trivial.

The difficulty of extracting hidden useful knowledge from literature increases as the number of literature arise. Analyzing today's massive biomedical literature, especially full-text articles, is a challenging task. The huge amount of available full-text articles results in very high computational and storage requirements. However, existing biomedical text mining tools that run on single commodity hardware cannot meet the requirements. For instance, a full run of BioContext on 20 million MEDLINE abstracts and 234 thousand PMC full-texts took 2-3 months (Gerner et al., 2012). To tackle this infrastructure barrier, cloud-based solutions that integrate multiple off-the-shelf big data technologies including parallel computing and cloud computing were introduced (Luo et al., 2016). Parallel computing is a fundamental infrastructure which enables the execution of data analysis tasks simultaneously on a cluster of machines or supercomputers (Luo et al., 2016). A prominent example is Hadoop, an open-source MapReduce package for distributed data management (Dean and Ghemawat, 2004). Cloud computing is a new paradigm for sharing computational resources across the Internet. It offers a scalable and cost-effective solution for big data analysis (Assunção et al., 2015). Amazon and Google are today's two biggest cloud infrastructure providers (Rajdho and Biba, 2013).

Nevertheless, the development of cloud-based solutions for large-scale literature mining is complicated, requiring IT experts to integrate a number of specific technologies, such as massive parallel processing, distributed databases, scalable storage systems, and a variety of third-party text mining libraries (Assunção et al., 2015; Yang et al., 2017b). Additionally, the configuration and management of cloud infrastructure are notoriously difficult and resource-intensive (Hendrickson et al., 2016). Since bioinformaticians specialized in text mining or natural language processing often lack the experience of managing cloud infrastructures, it is of great necessity that

there is an integrated cloud-based platform which relieves them from the burden of technical details, such as cloud setups and configurations. On the other hand, faced with various text mining needs, biomedical scientists need a great deal of freedom when it comes to selecting natural language processing (NLP) and text mining techniques (Lamurias and Couto, 2019).

In this work, we first proposed a big data framework named SELM to facilitate large-scale biomedical literature mining by utilizing cloud computing (Bahrami and Singhal, 2015) and Apache Spark (Zaharia et al., 2016). In specific, SELM incorporates a storage layer that supports cost-effective and scalable data storage in the cloud, and a management layer which manages both cloud infrastructure and text mining applications. An implementation of the framework was then presented as a cloud-based computational platform. The main characteristics of the platform are as follows: (1) scalable, (2) flexible, (3) efficient, (4) affordable, (5) reusable analysis to improve research reproducibility. Furthermore, the code of our implementation is open source and available on GitHub (Shen and Wang, 2019).

## 5.2 Related literature

Over the last few years, a number of platforms or systems have been produced to perform text mining tasks on large-scale literature corpora. For instance, Labropoulou et al. devised the OpenMinTed platform to support text mining of open access scholarly content (Labropoulou et al., 2018). Users can conduct text mining on self-defined corpora with graphic user interfaces. Textpresso Central is a similar tool crafted specifically for the biomedical domain (Müller et al., 2018). Although both tools are proven to be effective and efficient for researchers to discover useful insights from literature, they do not address big data challenges at all. To fill the gap, studies on large-scale biomedical text mining introduced software with the capacity for handling big data. SparkText, a text mining framework based on Apache Spark, was created and evaluated in (Ye et al., 2016). Tafti et al. proposed their big data analytics system to identify adverse drug events (ADEs) from literature and social media posts (Tafti et al., 2017). Besides, high performance computers were also employed in large-scale biomedical text mining (Ide et al., 2018; Xing et al., 2018). Nonetheless, given that supercomputers, such as Tianhe-2 (Liao et al., 2014), XSEDE (Towns et al., 2014), are not reachable to most researchers, the supercomputer dependent frameworks lack practical implications.

Table 5.1 summarizes the main characteristics of the existing platforms or systems,

including open source, cloud support, scalability, flexibility, and reproducibility.

- **Open source**: Releasing the code of a software under open source licenses becomes a popular practice in software industry and academia. It enables people to freely reuse the software for their own purposes (McKiernan et al., 2016). Whether a platform is open source is usually explicitly described.

- **Cloud support**: As discussed previously, cloud-based solutions are scalable and cost-effective. We examined the existing tools on whether they support large-scale text mining in the cloud.

- **Scalability**: It refers to how well a platform can scale up to large-scale literature mining. The size of corpora in the evaluation reflects the scalability of a system. According to the evaluation results stated in (Labropoulou et al., 2018; Müller et al., 2018; Ye et al., 2016; Tafti et al., 2017; Ide et al., 2018; Xing et al., 2018), we divided into high, medium and low.

- **Flexibility**: Given that a complex collection of text mining techniques are required for biomedical literature mining tasks (Lamurias and Couto, 2019), platforms or systems are not able to include all text mining techniques for users. Therefore, a certain degree of freedom should be granted to users in customizing text mining methods according to their specific needs. Flexibility reflects the level of freedom.

- **Reproducibility**: Reproducing computational experiments with a high degree of certainty is becoming a very important factor in assessing the quality of one's research (Korolev and Joshi, 2014). How well a platform supports the development of reproducible text mining pipelines determines the level of its reproducibility.

As shown in Table 5.1, not all existing platforms have cloud support. But there is a consensus among researchers about the benefits of utilizing cloud-based solutions for big data (Alharthi et al., 2017). Healthcare system developed on the cloud has shown its capacity to collect, store, and analyze big health data (Zhang et al., 2017). Geospatial scientists also utilized cloud computing to tackle big geospatial data challenges (Yang et al., 2017a).

| Paper/Platforms | Description | Open source | Cloud support | Scalability | Flexibility | Reproducibility |
|---|---|---|---|---|---|---|
| OpenMinTed Labropoulou et al. (2018) | A platform facilitating text mining of scholarly content | Yes | Yes | Low | Medium | High |
| Textpresso Central Müller et al. (2018) | A customizable platform for searching, text mining, viewing, and curating biomedical literature | No | Yes | Low | Medium | Medium |
| bigNN Tafti et al. (2017) | A scalable framework to analyze ADEs from large-scale biomedical text | No | No | Medium | Low | Low |
| SparkText Ye et al. (2016) | an efficient text mining framework built on big data infrastructure and a Cassandra NoSQL database | No | No | Medium | Medium | Low |
| LAPPS Grid Ide et al. (2018); Towns et al. (2014) | An open, interoperable web service platform for natural language processing (NLP) research and development | Yes | Yes | Medium | Medium | High |
| ParaBTM Xing et al. (2018); Liao et al. (2014) | A runnable framework that enables parallel text mining on the Tianhe-2 supercomputer | Yes | No | High | Low | Low |

**Table 5.1:** Summary of Existing Tools for Large-scale Biomedical Literature Mining

## 5.3 Framework

In this section, we present a framework which is composed of three main components for storing, analyzing massive biomedical literature and a underlying cloud infrastructure layer.

As displayed in Figure 5.1, the framework provides a high-level representation of our large-scale biomedical literature mining solution which is cloud-based and built upon Infrastructure-as-a-Service (IaaS). Instead of being limited to IaaS from a specific provider, it is compatible with all major public cloud infrastructures.

The storage layer utilizes the distributed file system from the cloud providers. It

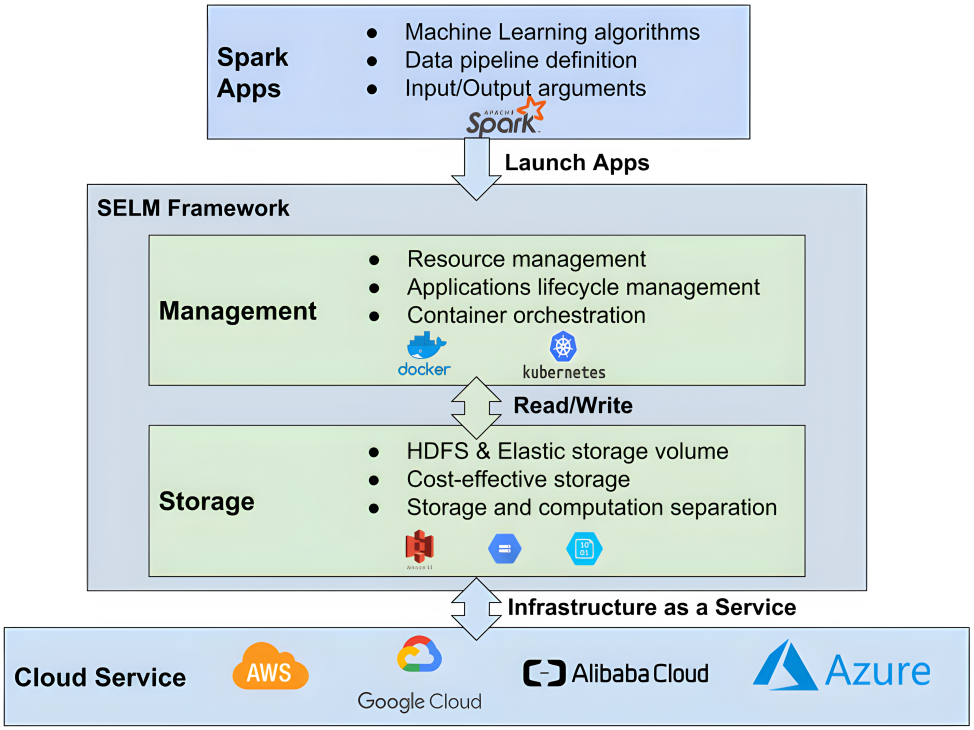



**Figure 5.1:** SELM: the Big Data Framework for Scalable and Efficient Biomedical Literature Mining in the Cloud

is extremely cost-effective and scalable due to the separation of storage and computation. High availability and data loss prevention are also secured at the same time. In addition, the accessibility of the storage layer is further extended to various use cases that are not bounded with particular machines or networks. In practice, Amazon Web Services (AWS) S3, Google cloud storage and Azure blob storage are employed for the reason that they are scalable for unstructured data with the vendor-variant Hadoop Distributed File System (HDFS) (Shvachko et al., 2010).

In the management layer, Kubernetes and Docker are employed to manage computational and storage resource, and Spark applications at scale. Spark applications written by bioinformaticians in the Spark Apps layer are built as Docker images which is an executable package. Firstly, it allows users to easily manage the lifecycles of Spark applications with simple configuration files. Secondly, it facilitates the computation resources to interact with the storage layer in an orderly fashion which is based on user-defined requirements. To scale up to run Spark applications on a

large cluster, Kubernetes is introduced to orchestrate the Docker containerized Spark applications.

As the external component of the framework, the Spark apps layer allows biomedical researchers to easily created reusable text mining modules with Apache Spark based on their specific needs. To begin with, biomedical researchers can create a Spark application with some widely used programming languages, including Python, Java, Scala, and R. Moreover, the built-in libraries in Apache Spark and third-party NLP libraries make it simpler. Before running a Spark application, Kubernetes deployment configurations should be added so that the application could be scaled up to running on a large number of servers. As mentioned before, Spark applications are assembled as Docker images in this layer which could be easily distributed and deployed across platforms (Boettiger, 2014). Therefore, biomedical literature mining tasks created for the framework are reusable.

## 5.4 Evaluation

To evaluate the performance of the framework, a cloud platform was constructed and evaluated on two large-scale full-text articles datasets. This section briefly discusses the datasets, the details of the experiment setup and evaluation metrics.

| Dataset | Year range | Size | Task | Model | Cloud Specifications | Evaluation metrics |
|---|---|---|---|---|---|---|
| PMC OA Subset | 1918-2019 | 1,010,787 | Vilidate the scalability with large biomedical literature corpora | LDA | 190G RAM, 64 cores | Execution time<br>Estimated cost |
| SparkText | 2009-2016 | 29,437 | Classifying full-text articles into breast lung or prostate cancer | Naive Bayes | 80G RAM, 16 cores | Execution time<br>Estimated cost<br>Accuracy<br>Precision<br>Recall |

**Table 5.2:** Details about Datasets, Experiment Setup, and Evaluation Metrics

### 5.4.1 Datasets

The first dataset contains over one million full-text biomedical articles from PubMed Open Access Subset. The articles were obtained via the PMC FTP service as zip files (Pubmed Open Access, 2019). With this dataset, we intend to demonstrate the powerful scalability of our approach by conducting topic modelling on three big samples. The biggest sample contains 1010787 full-text articles, and the smallest one comprises

267471 articles. The second dataset devised by Ye et al. in (Ye et al., 2016) was a collection of labeled biomedical articles from PubMed. More specifically, it contains 29437 full-text articles which are divided into three groups: breast cancer, lung cancer, and prostate cancer. we employed 80% of the entire dataset to train a classification model while the remaining 20% was used for testing.

### 5.4.2 Experiment Setup

An implementation of SELM has been created to verify our design and to evaluate its performance on analyzing big biomedical literature corpora. The implementation was constructed on top of the Google Cloud Platform. In particular, the cloud service in our implementation is Google Cloud. The storage layer relied on Google Cloud Storage. An open source software, Kubernetes Operator for Apache Spark (GoogleCloudPlatform, 2019), was utilized for the management layer. These tools were integrated into a cloud platform which supports running Spark applications in a scalable and efficient manner. The specifications of the cloud infrastructure in each experiment are in Table Table 5.2.

Spark applications were developed in Scala 2.11. Each Spark application consists of two main components: NLP pipeline and data modeling. In the NLP pipeline, raw text is preprocessed and vectorized into features with the built-in Apache Spark MLlib and the external NLP library for Apache Spark, namely Spark-NLP from the Johnsnow labs. Data modeling takes into the extracted features from the NLP pipeline to train the latent Dirichlet allocation (LDA) for topic extraction and the Naive Bayes model for document classification.

### 5.4.3 Evaluation Metrics

The performance metrics applied in the evaluation include execution time, estimated cost, and three common prediction measures: accuracy, precision, and recall. Execution time refers to the amount of time it takes to complete the execution of a Spark application in the cloud. Long execution time would significantly slow the development of text mining solutions. In most cases, data scientists obtain their best results through numerous rounds of execution. Thus, less execution time means more efficiency and synergies in text mining. Since cloud computing shares computational resources through the Internet, execution time can be reduced by expanding the number of cloud servers. However, the bigger the cloud is, the expensive it becomes. Taking affordability into consideration, we calculated the cloud computational cost for

each experiment. Since all cloud infrastructure providers support the pay-as-you-go model, costs are estimated on the basis of execution time.

Besides execution time and estimated cost, accuracy, precision, and recall are employed to measure the prediction performance of the document classification model trained on the second dataset. From the classification point of view, four possible outcomes are defined: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Based on the four outcomes, accuracy, precision and recall are calculated.

- Accuracy: it is calculated as the ratio of correctly classified document to the total number of documents. Accuracy = (TP + TN) /(TP + TN + FP + FN)
- Precision: it measures the exactness of a prediction. It refers the percent of positive predictions that are accurate. Precision = TP/(TP + FP)
- Recall: it measures the sensitivity of a classifier. It is calculated as Recall = TP/(TP + FN) To compare the performance of our approach,

we used the performance of a big data framework in (Ye et al., 2016) as the benchmark to conduct comparison.

## 5.5 Results

In this section, we present the results of two experiments and discuss the implications and limitation of the evaluation.

### 5.5.1 Topic Modelling on the PMC OA Subset

We demonstrated the scalability of SELM with a big biomedical literature corpus which contains over one million biomedical full-texts articles. To our best knowledge, there is no research on biomedical literature mining that has analyzed data of this size, even in those where big data frameworks were developed for biomedical literature mining (Ye et al., 2016; Tafti et al., 2017; Xing et al., 2018). Moreover, as a generative probabilistic model, LDA is computationally intensive. The result of this experiment also applies to text mining tasks using other common classification models, including Support Vector Machine (SVM), Logistic Regression, and Random Forest.

Performance comparison of our approach and other systems for large-scale biomedical literature mining is shown in Table 5.3 The analysis of over one million full-text articles took around 25 minutes with a cost of around 1.5 US dollars. Our implementation only took nine minutes to analyze 267,471 full-text articles. It took six minutes to build a classification model with 29,437 full-text articles on SparkText. The execution time of running the same analysis on the same dataset in the second experiment is only one minute. Therefore, in terms of time efficiency, our framework is around six times faster than SparkText. Furthermore, when it comes to cost, our framework's is much lower than that of the other three systems. It offers an affordable means for biomedical scientists to fully utilize the available biomedical literature.

| | **Size** | **Model** | **Time (minutes)** | **Cost** ($) |
|---|---|---|---|---|
| SELM | 1,010,787 | LDA | 25 | 1.5 |
| | 428,152 | LDA | 15 | 0.75 |
| | 267,471 | LDA | 9 | 0.45 |
| | 29,437 | Naive Bayes | 1 | 0.02 |
| bigNN | 14,017 | SVM | 88.9 | 597* |
| | 21,843 | Naive Bayes | 135.3 | |
| SparkText | 29,437 | Naive Bayes | 6 | 1100* |
| ParaBTM | 61,078 | Conditional Random Fields | 240 | High** |

* Monthly cost estimated on server specifications provided on the papers and the prices of Google Cloud.

** The authors stated that the cost of a full run of over 1 million full-text articles from PMC is beyond their funding budget.

**Table 5.3:** Performance Comparison with Large Corpus

### 5.5.2 Cancer Articles Classification

Table 5.4 shows the comparison of execution time, estimated cost and the prediction metrics between our solution and SparkText. Like SparkText, we trained a Naive Bayes classification model on a dataset of 29,437 full-text articles. First of all, SELM is six times faster than SparkText. Secondly, the cost of execution with our framework is much less and more flexible. In comparison with SparkText our approach do not require a server running at 24/7. Cloud infrastructure only starts on demand and costs are calculated based on the amount of actual running time. Last but not least, prediction results are improved with our approach. Therefore, it is safe to claim that our framework is a better choice in handling large-scale biomedical literature mining.

| | **Accuracy** | **Precision** | **Recall** | **Time (minutes)** | **Cost ($)** |
|---|---|---|---|---|---|
| SELM | 90.28% | 90.36% | 90.28% | 1 | 0.02 |
| SparkText | 86.44% | 87.61% | 89.12% | 6 | 1100 |

**Table 5.4:** Performance on Cancer Articles Classification

## 5.6 Discussions

The execution time and estimated costs from both experiments indicate that the proposed framework is more efficient and affordable in large-scale biomedical literature mining. The fact that the framework supported the analysis of over one million full-text articles exhibits its good scalability in handling big data. No existing biomedical literature mining system has demonstrated similar capacity. Besides, the prediction performance in the second experiment suggests that SELM is capable of being scaled down to medium-sized datasets. As mentioned previously, the framework is designed as a flexible platform that supports a variety of biomedical literature mining tasks. During the evaluation, we performed two literature mining tasks on two different datasets, which ensures that our framework can be easily expanded to other biomedical literature mining tasks. Therefore, we argue that the purpose of designing a scalable, flexible, efficient and affordable framework for biomedical literature mining is fulfilled.

The target users of the platform built on the framework include any biomedical researcher who works at discovering knowledge from a huge amount of biomedical literature. It relieves biomedical researchers from the burden of technical details on configuring computational infrastructure. With the platform, biomedical researchers focus on building text mining Spark applications and distributing them as Docker images. Spark applications in the format of Docker images are reusable modules that facilitate reproducible research.

There are some noted limitations in our study. First, not all 2.4 million full-text articles in the PMC OA Subset were used in our evaluation. Although there is no doubt that the framework can scale up to analyze all 2.4 million full-text articles, it is of great interest to see all articles being analyzed. Secondly, common biomedical literature mining tasks, such as entity recognition and relation extraction, are not validated in the evaluation. However, these tasks are crucial in knowledge discovery from biomedical literature. For instance, relation extraction helps advance precision medicine by discovering the association between gene and disease (Pletscher-Frankild et al., 2015), gene and drug (Cañada et al., 2017), and other associations. Future studies should include the implementation of these tasks. At last, the framework was introduced as cloud infrastructure independent, meaning it can seamlessly integrate with all major cloud service providers, including Amazon, Google, and Microsoft. However, only Google Cloud was tested in the evaluation.

## 5.7 Conclusion

In this study, we presented a big data framework for large-scale biomedical literature mining in the cloud. The scalability and cost-effectiveness are guaranteed by the cloud infrastructure upon which the framework was built. Besides, the storage and management components supported by Kubernetes and Docker make developing, managing and reproducing Spark based biomedical literature mining as easy as possible. Experimental results validate that our platform offers a scalable, flexible, efficient and affordable solution for large-scale biomedical literature mining.

In future work, we plan to further evaluate the platform by inviting more biomedical researchers to evaluate it and provide feedback. Then we will improve its usability based the feedback, We also would like to investigate the performance of our platform in analyzing literature or textual data from other domains.

# Chapter 6

# A Systematic Review of Open Source Clinical Software

The plethora of open source clinical software offers great reuse opportunities for developers to build clinical tools at lower cost and faster pace. However, the lack of research on open source clinical software poses a challenge for code reuse in clinical software development. This paper aims to help clinical developers better understand open source clinical software by conducting a thorough investigation of open source clinical software hosted on GitHub. We first developed a data pipeline that automatically collected and preprocessed GitHub data. Then, an in-depth analysis using several methods, such as statistical analysis, hypothesis testing, and topic modeling, was conducted to reveal the overall status and various characteristics of open source clinical software. There were 14,971 clinical-related GitHub repositories created during the last 10 years, with an average annual growth rate of 55%. Among them, 12,919 are open source clinical software. Our analysis unveiled a number of interesting findings: Popular open source clinical software in terms of the number of stars, most productive countries that contribute to the community, important factors that make open source clinical software popular, and 10 main groups of open source clinical software. The results can assist both researchers and practitioners, especially newcomers, in understanding open source clinical software.

This work was originally published as: Shen Z, Spruit M. A Systematic Review of Open Source Clinical Software on GitHub for Improving Software Reuse in Smart Healthcare. *Applied Sciences*. 2019; 9(1):150. https://doi.org/10.3390/app9010150

## 6.1 Introduction

There have been numerous open source projects created across a variety of domains after decades of open source software advocacy (Anthes, 2016). With their accelerating popularity, more and more will likely be added. The plethora of open source projects offers great reuse opportunities for developers to build tools at lower cost and at a faster pace. Therefore, software reuse, also called code reuse, has become an essential topic in software development (Frakes and Fox, 1996; McIlroy et al., 1968; Zaimi et al., 2015). This is especially true in the clinical domain, where the lack of a well-trained IT workforce and an uneven geographic distribution of clinical informaticians between developed and developing countries restrain the development of clinical IT (Luna et al., 2014; Russo et al., 2016). Given that successful reuse of existing open source software enables the rapid and cost-effective development of clinical applications, it is of great importance to promote reuse awareness in the clinical community and facilitate software reuse practices among clinical developers and applied data scientists (Zhang and Ho, 2017; Schots, 2014; Badgeley et al., 2016; Spruit and Lytras, 2018). Zhang and Ho (2017) have recognized this importance and have called for more reuse of open source projects in clinical settings.

As a challenging topic, software reuse still faces many difficulties, particularly in clinical software development (de Oliveira, 2015). The lack of resources to support the selection of suitable reusable source codes or software from tens of thousands of open source projects is one of them. Although systematic literature studies on clinical software have helped developers obtain a deeper understanding of the existing tools and their performances (Rehim et al., 2017; Guaitoli et al., 2014; Marien et al., 2017), they are not particularly useful in clinical software development. First, systematic literature studies cover only a small proportion of the vast amount of clinical software. Moreover, the rapid updates of clinical software are not reflected due to the long publication process. Most importantly, these systematic literature studies do not examine source codes, software issues, and code usage, which are crucial in reuse practices.

Recently, researchers in life sciences have combined literature studies and large-scale analysis on open source repositories to achieve better reusability (Wang et al., 2017; Russell et al., 2018). A source code repository is a file archive and web hosting facility where a large amount of source code is kept, either publicly or privately (Wikipedia, 2018). Well-organized repositories contain both brief descriptions and detailed README files that explain what the projects are and how to use them. Quantitative metrics such as stars and forks indicate how popular a repository is.

Wang et al. built an online software discovery platform with biomedical software-related data collected from PubMed literature and GitHub. It empowers biomedical researchers to easily find the (open source) tools they need (Wang et al., 2017). A large-scale analysis of bioinformatics open source projects on GitHub was conducted to uncover the unknown state of bioinformatics software. The analysis covers a list of 1743 GitHub repositories manually selected from bioinformatics articles and online forums (Russell et al., 2018). By focusing on the analysis of data extracted from open source repositories, these studies offered more information for evaluating software in reusability. However, both studies only included a limited amount of biomedical software that were mentioned in the literature. To our knowledge, there is no such research that has investigated open source clinical software.

Therefore, this work intends to contribute to clinical software reuse research by conducting a large-scale analysis of open source repositories in the clinical domain. Instead of analyzing a shortlist of open source repositories extracted from literature, this study includes all clinical-related open source repositories on GitHub. As the largest code host in the world, GitHub reached 24 million developers working across 67 million repositories in 2017 (Octoverse, 2018). Numerous open source projects in various domains could be found and reused. A systematic study on GitHub repositories could provide a good representation of available open source tools in clinical domains. Besides, GitHub offers an easy application programming interface (API) for external users to retrieve data from repositories.

Natural language processing (NLP) quantitative research via topic modeling was conducted to uncover the distribution of open source repositories over different topics. Additionally, quantitative analysis of numerical data revealed the status of repositories in terms of their reuse capability. The analyses provide sufficient information to support clinical developers' decisions in the software development process. Moreover, this study developed a reproducible and scalable method for systematic studies on GitHub repositories. Reproducing the research can be completed by running the data collection and analysis scripts we host on GitHub. To scale up the study to other domains, one merely needs to customize the search terms.

In particular, the purpose of the paper is to shed light on the following questions, so that clinical developers can make more informed decisions:

- What is the current status of open source clinical software?
- What are the impacts of various factors, such as the number of contributors and the number of commits, on the popularity of an open source clinical software?

- What are the main focus areas of all the collected open source clinical software?

The paper is structured as follows. section 6.2 illustrates a number of methods used in our study. Then the results are presented in section 6.3 . Discussions in section 6.4 summarize the main findings of this research and list some limitations. section 6.5 concludes the study.

## 6.2 Materials and Methods

This study underwent two main steps: (1) Data collection that extracted data from GitHub repositories via the GitHub REST API v3 (GitHub, 2018), and (2) data analysis in which both numerical and textual data were analyzed to reveal insights on open source repositories in the clinical domain. The data collection produced the core material of this study, GitHub data. To uncover the current status, we applied descriptive analysis to obtain a summary of open source clinical software from many aspects. Second, since generalized additive models (GAMs) have an interpretability advantage, the study employed a GAM to explore the relationships between various factors and the number of stars of an open source clinical software. Lastly, the description of an open source clinical software offers a quick way of understanding what it does and what it is for. Therefore, the main focus areas were derived from thousands of descriptions with topic modeling.

### 6.2.1 Data Collection and Processing

#### Search Terms

This study refers to clinical software as software that is implemented in healthcare to help doctors improve patient care. Software that is developed for biological research, such as genome sequencing (Pabinger et al., 2014) and cell screening (Omta et al., 2016), were excluded. Therefore, "(clinical OR medical) OR (patient OR doctor)" was selected as the search term. English terms were used given that the majority of repositories have their names and descriptions in English. Figure 6.1 shows the numbers of repositories returned using the search term "clinical" in different languages. Second, the search term covered the main entities in the clinical domain.

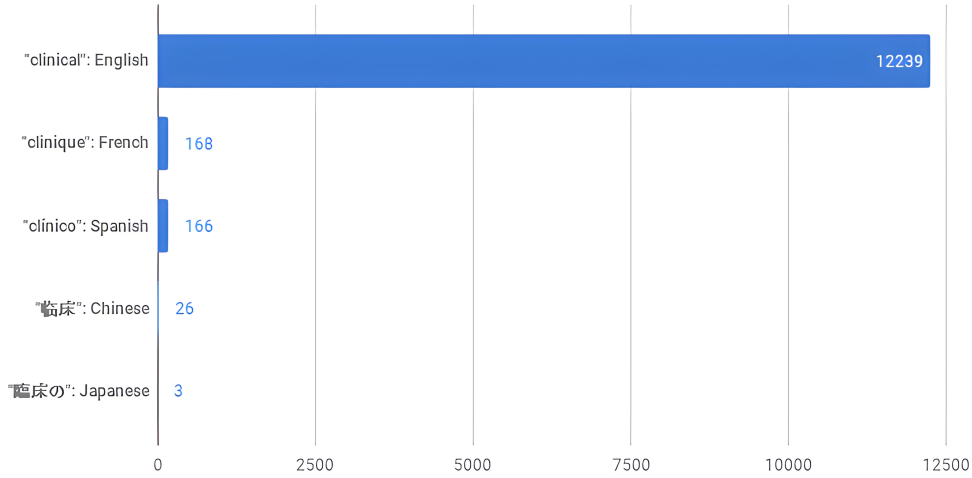



**Figure 6.1:** The number of repositories returned with the search term "clinical" in different languages.

### Data Extraction

GitHub REST API v3 exposes GitHub repository data to external users. Its search API offers an optimized solution for users to locate the specific items that interest them most, such as repositories, users, and issues. GitHub repositories in this study were obtained by using the search repository with the above term, and data for each repository including name, description, README files, stars, forks, the number of contributors, and the number of commits were extracted. Table 6.1 displays the features extracted and the purposes of extracting them.

### Automated Data Pipeline

The data extraction pipeline was written in Python using a third-party library, PyGitHub. The pipeline took the chosen search terms as input and received repository data in JSON. The JSON responses were first filtered and then converted to database records and pushed to tables in a MySQL database. Repositories with no description or no programming language specified were excluded from further analysis for the reason that clinical software was the focus of our study. The whole process is reproducible by running the Python scripts at (Open Source Clinical Software, 2018). Moreover, replacing the search term with others scales the pipeline to other domains.

| Data Type | Data Extracted | Description |
|---|---|---|
| **Textual** | Description | A brief explanation of what a repository does, what it is for, and other items. |
| | README files | A README file summarizes the most important aspects of your project. It generally includes project name, description, table of contents, installation, usage, contributors, credits, and license. |
| | Languages | Programming languages of the source codes. |
| | Owner type | Two types of owners exist: Organization and individual developers. |
| | Owner location | Location of an owner. |
| **Numerical** | Stargazers | Stargazers refers to the number of times a repository is bookmarked. It reflects an approximate level of interest in the repository. |
| | Forks | A fork is a copy of a repository. Forking is necessary for developers to contribute a project. Forks refers to the number of forks. |
| | Contributors | The number of contributors who have worked for a repository. |
| | Commits | The total number of commits. |
| | Created date | The date that a repository was created. |
| | Updated date | The latest date that a repository was updated. |
| | Issues | Issues is the number of open issues in a repository. |
| | Size of source codes | The size is valued as the size of the whole repository (including all of its history), in KB. |
| | Size of README file | The size of the README file of a repository, in B. |

**Table 6.1:** Data extracted from GitHub REST API v3.

#### Text Processing

Although GitHub has allowed users to attach topics as labels of the repositories since 2017, only a small fraction of them have topics. Specifically, only 6.6% (982/14,971) of the collected repositories have topic labels. Therefore, we utilized keywords ex-

tracted from a repository description as labels. Keywords extracted from the description presented an insightful indication into a repository. An external API, namely IBM Watson Natural Language Understanding (IBM Watson NLU, 2018), was requested to process the descriptions, which yielded a list of informative keywords for each repository.

Nevertheless, the extracted keywords could not be directly added as labels because a large number of keywords had semantically similar forms. Examples included "patients' data" versus "patients' information", "medical appointments" versus "doctor appointments", and "diabetes patients" versus "diabetic patients". To address this semantic issue, we employed a normalization process in which semantically similar keywords were combined. The process consisted of three steps: (1) Calculating the semantic similarity between keyword pairs, (2) labeling keyword pairs based on their similarity, and (3) replacing keywords with their semantically similar keywords.

To calculate the semantic similarity, we utilized Word2Vec (Mikolov et al., 2013; Goldberg and Levy, 2014) to represent keywords. Word2Vec is an unsupervised algorithm developed by Google that tries to learn meaningful vector representations of words from a dataset of text (Goldberg and Levy, 2014). It does so based on the distributional hypothesis, which states that words that appear in the same context probably have a similar meaning. With Word2Vec, each keyword becomes a word vector that is compared to other keyword vectors to generate a similarity score. Based on their similarity, we divided keyword pairs into two groups, equal and tag. The threshold of the similarity score was 0.85. Keywords with a similarity score greater than or equal to 0.85 were considered as an equal keyword pair. Otherwise, they were a taggable pair. For equal keyword pairs, we replaced one of the keywords, the less frequent one, to replace the other. The "tag" meant that the more frequent keyword was added as a tag to the other. Table 6.2 shows some prominent examples of these keyword pairs.

### 6.2.2 Descriptive Analysis

Descriptive analysis is applied to acquire the characteristics of a collection of data (Mann, 2010). Together with simple graphics, it presents knowledge hidden in large datasets in a manageable way. To identify interesting characteristics that reflected the current status of the GitHub repositories, we conducted a descriptive analysis. Specifically, we identified popular repositories in terms of stars, most rapidly grow-

| Original Keyword | Replacement | Tag |
|---|---|---|
| Patients' information | Patients' data | Patients |
| Electronic medical record | Medical records | |
| Clinical data | Patients' data | Data |
| Medical information | Medical data | Medical records |
| Medications | Medicines | |
| Breast cancer patients | Cancer patients | |
| Healthcare | Medical care | Health |
| Pharmacists | Physicians | |
| Clinical studies | Clinical research | Clinical data |
| Simple web application | Web application | Application |

**Table 6.2:** Examples of 10 keyword pairs based on their semantic similarity.

ing repositories according to stars per day, most diverse repositories in terms of the number of contributors, and most active repositories in terms of the number of commits per day.

Graphics are of great importance to descriptive analysis (Mann, 2010). A graphic was created with geographic visualization to show the geographic distribution of open source clinical software. Geographic visualization refers to a set of techniques for analyzing spatial data (Dodge et al., 2011). The graphic reflected the current development of clinical software across countries, which provided some insights into the uneven geographic distribution of clinical informaticians between developed and developing countries.

### 6.2.3 Generalized Additive Models

As an extension of generalized linear models (GLMs), a GAM is an additive modeling technique that captures the impact of the predictive variables through smooth functions (Wood, 2017). It is best known for its interpretability, which can determine the contribution of each independent variable to the prediction. Therefore, we chose a GAM to help us identify the relationships between several repository features and its popularity (the number of stars). To begin with, we built a linear regression model

that predicted the number of stars of a repository based on a number of predictors: Forks, the number of issues, the repository size, owner type, the size of README files, the number of contributors, the total number of commits, and creation year. Then, partial dependency plots were produced to explicitly show the relationships between the predictors and the number of stars on GitHub repositories. We chose pyGAM, a Python package for GAM, as the package for our implementation (GAM in Python, 2018).

### 6.2.4 Topic Modeling

For broad search queries, thousands of unique keywords may remain, even after the previously described similarity-based processing step. Therefore, labeling the repositories with such keywords could not provide a good overview. Our aim was to identify a few numbers of broader topics that could be used to label repositories. This paper applied an unsupervised machine learning approach, latent Dirichlet allocation (LDA), to infer topics from the extracted keywords. We set 10 to 20 as the range in creating optimal LDA models.

LDA, first introduced by Blei, Ng, and Jordan in 2003, is a generative probabilistic topic model that aims to discover the underlying topics from a corpus (Blei et al., 2003; Jelodar et al., 2019). LDA models document probabilistic distributions over K latent topics, where each latent topic is represented by a probabilistic distribution over words from a fixed vocabulary. The words with the highest probabilities in each topic usually give a good indication of what the topic is. As a robust open source topic modeling toolkit, Gensim offers an easy way of implementing topic modeling (Rehurek and Sojka, 2010). We implemented our LDA with Gensim in Python (Genism, 2018).

We first trained an LDA model on the extracted keywords from all the repositories, and then five more LDA models were built on repositories from different periods: Prior to 2015, 2015–2016, 2016–2017, 2017–2018, and post-2018. Each of these periods contained a roughly similar number of repositories, around 3000. Table 6.3 lists all the models and the parameters that generated the optimal result in terms of the cohesion and explainability of topics. Alpha dictates how many topics a document potentially has. The lower the alpha, the lower the number of topics per document is. Passes is the number of times you want to go through the entire corpus.

| LDA Model | Number of Repositories (Repos) | Number of Topics | Alpha | Passes |
|---|---|---|---|---|
| All repositories | 14,971 | 10 | 0.001 | 10 |
| Prior to 2015 | 2,784 | 8 | 0.002 | 6 |
| 2015–2016 | 1,899 | 6 | 0.01 | 8 |
| 2016–2017 | 2,872 | 8 | 0.01 | 6 |
| 2017–2018 | 4,330 | 8 | 0.02 | 8 |
| Post-2018 | 3,086 | 8 | 0.01 | 10 |

**Table 6.3:** Latent Dirichlet allocation (LDA) models: Five models that built on yearly repository data.

### 6.2.5 Interactive Data Visualization

Data visualization empowers researchers to present their results in a graphical manner (Jiang et al., 2005). In this work, we visualized our findings with interactive data visualizations, which are accessible online. Interactive data visualization engages more with users by allowing them to select specific data points to visualize the data in the way they choose (Dunkerley, 2013). In particular, radar charts and time-series line plots were created with Plotly.py (plotly visualization, 2018), an interactive, browser-based graphing library for Python. In addition, we built an interactive top modeling visualization with pyLDAvis to improve the interpretation of the generated topics (Sievert and Shirley, 2014; pyLDAvis, 2018).

## 6.3 Results

This section presents the main findings of our analysis, which answers the questions described in section 6.1.

### 6.3.1 Current Status

Table 6.4 provides a quantitative summary of all collected GitHub repositories. It highlights some important characteristics of the dataset and illustrates the current status of open source clinical software from different aspects. First of all, there have been 14,971 clinical-related GitHub repositories created since 2008. Among them, 12,919 are open source clinical software for the reason that they own publically accessible source codes. Moreover, there has been an upward trend in the number of repositories created on a yearly basis. Until 2011, the number of repositories was

around 100 per year. The number rapidly increased to around 1000 from 2012 to 2014. Since the average yearly growth rate over the last five years has been 55%, the number of repositories created in a year will reach around 9300 in 2020.

Table 6.4 lists repositories according to a variety of metrics. From "popular repos", "most rapidly growing repos", "10 frequency topics extracted from repos", and "top 10 libraries", we found that clinical software on machine learning, medical images, and electronic medical records attract more attention from developers. The most rapidly growing repositories list indicated that deep learning-related repositories are becoming dominant, with five out of seven of the most rapidly growing ones addressing deep learning issues in the clinical field. Nevertheless, the most diverse repositories and most active repositories did not show a clear pattern.

### 6.3.2 Geographic Distribution

Table 6.4 shows the top 10 countries that contribute most to the open source community based on the number of GitHub repositories. Figure 6.2 is the geographic visualization of a country's repositories (accessible at Reference (GeoVis, 2018)). As shown in the figure, the U.S. is unsurprisingly the biggest player, taking up to 38.8% of the total GitHub repositories. Among developing countries, India contributes the most, with around 10%.

### 6.3.3 Key Success Factors of Open Source Clinical Software

Many factors affect the popularity of GitHub repositories. However, given the data we could collect, we mainly examined the impact of eight factors: The number of forks, the number of open issues, repository size, owner type, README file size, total contributors, total commits, and year of creation. The partial dependence plots, as shown in Figure 6.3, reveal the impacts of different factors on the number of stars. First, forks and the number of contributors exhibited a clear positive effect on the number of stars. A higher number of forks or contributors led to more stars. Meanwhile, the README file size had litter influence. Whether a repository is owned by an organization or individual developers was also trivial. With regard to the year of creation, we noticed that repositories that had existed significantly longer tended to earn more stars, but the impact diminished after 2010.

When the number of open issues was between 60 and 120, the number of stars was at a lower point. When the value was larger than 120, there was a positive correlation. A possible explanation is that the number of issues rises as developers of a

| Characteristics | Statistics |
|---|---|
| # of repos that were not empty | 114,971 |
| Open source clinical software (# of repos with codes) | 12,919 |
| Top 10 countries (# of repos) | U.S., India, United Kingdom, Canada, China, Germany, France, Bangladesh, Australia, Brazil |
| Popular repos (# of stars) | ResearchKit, openemr, ClearCanvas, dwv, cornerstone, Deep-Learning-for-Medical-Applications, CTK, NiftyNet, 3DUnetCNN, OpenClinica |
| Most rapidly growing repos (# of stars per day) | ResearchKit, NiftyNet, DLTK, Attention-Gated-Networks, Deep-Learning-for-Medical-Applications, chart-doctor, 3DUnetCNN |
| Most diverse repos (# of contributors) | ResearchKit, openmrs-module-bahmniapps, medical-appointment-scheduling, sweetdoc, SwasthIndia, CTK, sofa-framework, MITK, dicom_tools |
| Most active repos (# of commits per day) | MITKats, openeyes, cds-stack, sofa-framework, myclinic-spring, renalware-core, BluePearlViewer, SwasthIndia, clinical-meteor-tool |
| Top 10 frequency topics extracted from repos | machine learning, healthcare, medical image, deep learning, clinical trial, patients, medical information, natural language processing |
| Top 10 libraries/framework | React, Jquery, Angular, Tensorflow, Ajax, Redux, Keras, OpenCV, Pytorch, Scikit-learn |

**Table 6.4:** Overview of open source clinical software.

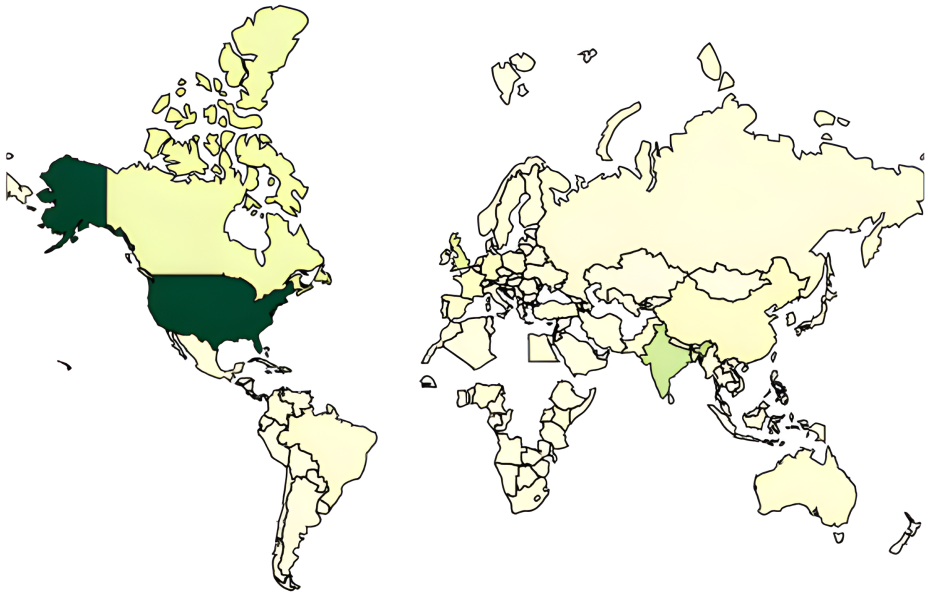


**Figure 6.2:** Geographic visualization of GitHub repositories by country.

repository stop further development, which then drives potential users away. Lastly, the impacts of repository size and the number of commits were unclear.

### 6.3.4 Main Focus Areas

As shown in Figure 6.4, ten topics discovered in all repositories were as follows: Medical images analysis, applications for medical practice, medical data reuse, clinical text processing, applications for doctors, medical research, medical education, healthcare management systems, clinical trials, and medical data analysis. Table 6.5 offers a brief description of each of these topics and their top keywords.

Analysis of topics generated from the yearly data revealed that new technologies emerged over time. For instance, the open source community has devoted a lot to medical images analysis since the very beginning. It has also been a topic throughout all the years. Top keywords in the topic showed that deep learning and neural networks emerged as popular techniques in 2015. Moreover, we noticed that recently developers have been starting to evaluate blockchain in medical record sharing.

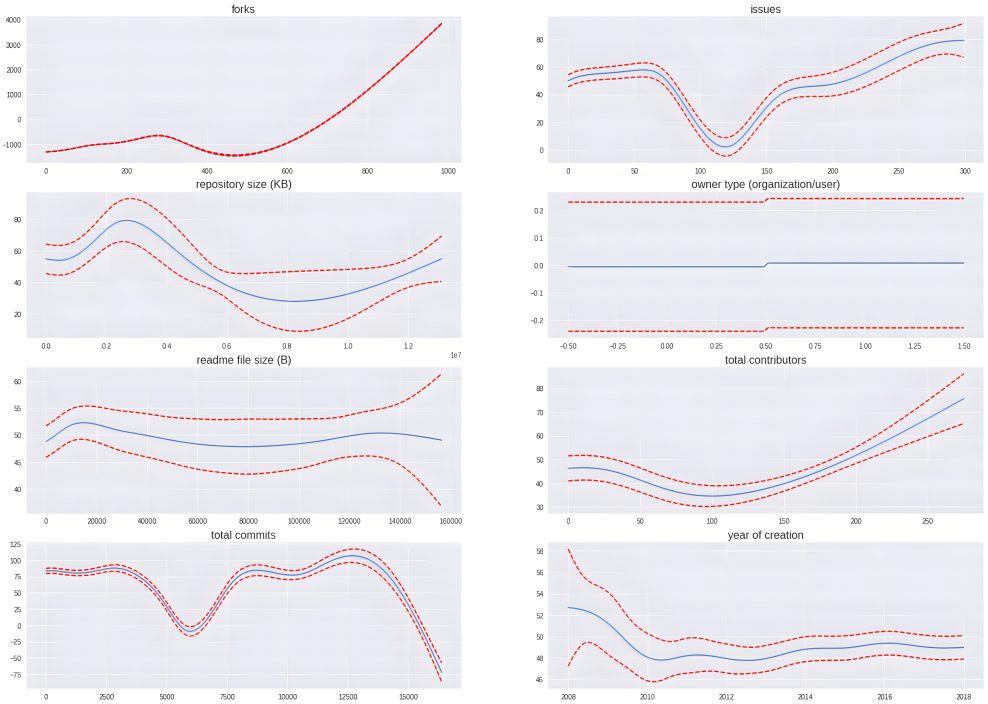



**Figure 6.3:** Partial dependency plots showing the relationships between the predictors (forks, open issues, repository size, owner type, README file size, total contributors, total commits, year of creation) and the number of stars on GitHub repositories.

| # in Figure 6.4 | Topic Label | Top Keywords |
|---|---|---|
| 1 | Medical images analysis | Medical images, medical image processing, deep learning, medical image analysis, medical image segmentation |
| 2 | Applications for medical practice | Android application, medical applications, prescriptions, medical practice, appointments, medical care |
| 3 | Medical data reuse | Medical data, website, API, medical services, database, MySQL |
| 4 | Clinical text processing | Medical documents, symptoms, medical history, clinical notes, natural language processing (NLP) |
| 5 | Application for doctors | Doctors, applications, doctor appointments, medical professionals, doctor office, nearby doctor, medical staff |
| 6 | Medical research | Medical research, clinical research, clinical data, patient registration, projects, design, clinical decision support, medical informatics |
| 7 | Medical education | Medical students, final project, medical center, project Gutenberg book, experiments, paperwork |
| 8 | Healthcare management systems | Patient management, medical appointments, medical store management, patient monitoring, patient management systems, platform, software |
| 9 | Clinical trials | Clinical trials, patients, data analysis, new patients, survival analysis |
| 10 | Medical data analysis | Patient data, medical imaging, machine learning, model, data science, gene expression, predictive model |

**Table 6.5:** Ten topics of all clinical GitHub repositories. API: Application programming interface.

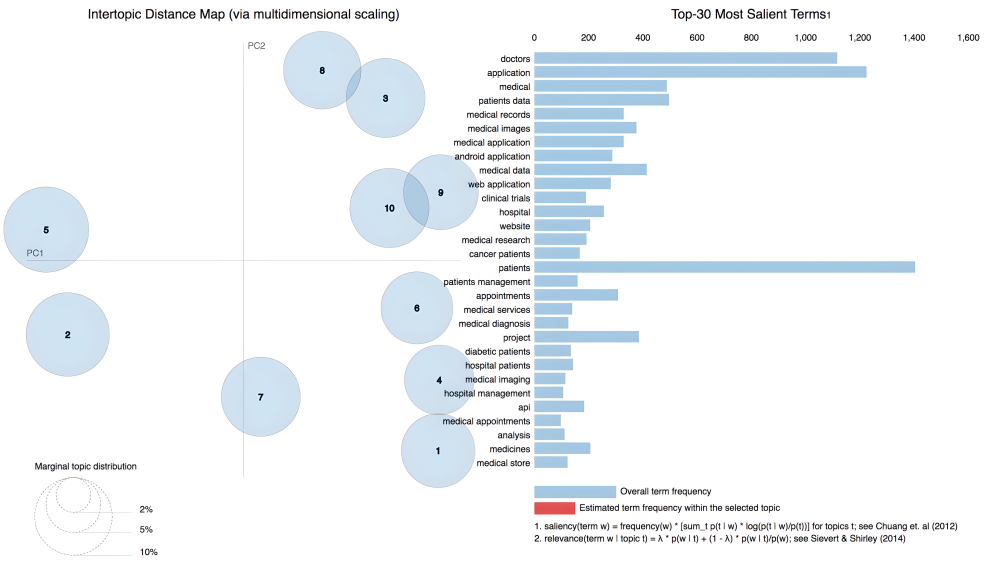



**Figure 6.4:** Interactive visualization of clinical topic clusters.

## 6.4 Discussion

With the purpose of shedding light onto the vast amount of clinical-related open source software, our study collected data through the GitHub API and then conducted various analyses to uncover both overall and specific characteristics among GitHub repositories. The analysis methods included a descriptive analysis, trends analysis, and topic clustering. The main findings are summarized as follows:

- The number of repositories and the growth rate indicated an increasing interest, which suggests the potential of utilizing open source software in both academic and practical clinical settings;
- Clinical software on machine learning, medical images, and electronic medical records attract more attention from developers, and deep learning-related repositories are becoming dominant;
- The U.S. is the biggest player in the world, taking up to 38.8%. Among developing countries, India contributes the most, with around 10%;
- Factors that affect the popularity of a GitHub repository include forks and the number of contributors. The README file size and owner type had litter influ-

ence. Other factors, such as the number of open issues, the repository size, and the number of commits, are unclear.

- The LDA model clustered keywords extracted from repository descriptions into 10 groups, ranging from clinical text processing to medical education. Moreover, analysis of the topics generated by yearly data revealed that topics evolve over time.

Besides the above findings, our study contributes to the scientific community by providing a reproducible method of studying GitHub repositories. Over the years, researchers have been struggling with reproducing research results such as literature studies. As mentioned above, both data collection and analysis could be reproduced with a rerun of the Python scripts. Scaling up the approach to other domains requires some additional inputs, such as search terms, and the number of topics for topic modeling.

Nevertheless, some limitations to this study should be noted. Although GitHub is a major platform where developers work and share their codes, there are other platforms that are also of great importance to the open source community, such as SourceForge and GitLab. Collecting more data from such platforms may therefore produce more complete and deeper insights.

Furthermore, README files were also employed to extract keywords because some repositories only describe coarsely what the code does in the project description. Keywords extracted from README files could provide more information so that the following topic modeling would be more accurate. For example, borochris/CLinUiP has a very short and uninformative description, "Clinical UI Portal", but its README file contains a clear explanation, "Clinical UI Portal is an Open Source demonstration medical portal".

Last but not least, the topic labeling was limited. First of all, some of the topics contained overlapping keywords, given that they were related. For instance, medical image analysis has "medical images" as a top keyword, and "medical imaging" belongs to medical data analysis. In addition, the labels were manually assigned by authors only, instead of a group of experts.

## 6.5 Conclusions

This paper provided an overview of the open source GitHub projects in clinical domains with the purpose of understanding the status of open source clinical software.

It, on the one hand, helps clinical developers be aware of the status quo so that they can make an informed decision while they design their systems and tools. Instead of creating all code from scratch by investing a lot of resources and time, making good use of existing tools and code can be a more efficient strategy. On the other hand, researchers might get possible research direction from this study.

# Chapter 7

# LOCATE: A web application to link open-source clinical software with literature

Nowadays, the effective utilization of open-source software could significantly boost both clinical research and practices, especially in resource-poor countries. However, the plethora of open-source clinical software has left many people unable to quickly locate the appropriate one for their needs. Commonly available software quality metrics and software documentation, such as downloads, forks, stars, and readme files, are useful selection criteria, but they only indicate the software quality from the perspective of IT experts. This paper proposes a method that offers additional insights on the performance and effectiveness of clinical software. It links open-source clinical software with relevant scientific literature, such as papers that use case studies of clinical software to reveal the strength and weakness of a given software from the clinical perspective. To interactively present the open-source clinical software and their related literature, we have developed the LOCATE web application that enables users to explore related literature for a given open-source clinical software. Moreover, the peer-review cycle of the application allows users to improve the application by confirming, adding or removing related literature. An evaluation experiment of the five most popular open-source clinical tools demonstrates the potential usefulness of LOCATE.

This work was originally published as: Shen, Zhengru, and Marco Spruit "LOCATE: A web application to link open-source clinical software with literature." *In Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2019)*-Volume 5. SciTePress, 2019. https://doi.org/10.5220/0007378702940301

## 7.1 Introduction

Open-source software plays a more important role in today's research, especially in the clinical field where using IT to leverage clinical practices has become ubiquitous (McDonald et al., 2003; Karopka et al., 2014; Reynolds and Wyatt, 2011). Furthermore, both the huge volume and the various types of clinical data accumulated on a daily basis require researchers to develop more advanced data analysis methods including both machine learning and deep learning algorithms (Raghupathi and Raghupathi, 2014). By using openly accessible tools or algorithms, both clinical researchers and practitioners can improve their work in many ways. To begin with, it significantly reduces the IT development cost, which, in return, allows us to focus (usually limited) resources on clinical issues (McDonald et al., 2003). Secondly, open-source clinical software, especially the well-supported in the open-source community, ensures the public accessibility of software platforms and tools in research and thus empower the scientific community to verify its reproducibility (McCormick et al., 2014). Lastly, your involvement contributes to the open-source community by verifying open-source clinical software in real-world settings (Kiah et al., 2014; Zettinig et al., 2015; Akowuah et al., 2015).

There are numerous open-source tools created across a variety of domains after decades of the open-source software advocacy (Anthes, 2016). With its accelerating popularity of open science, more and more will be added. Open-source clinical software covers its various research topics and clinical practices: medical images analyses (McCormick et al., 2014), medical text processing (Cunningham et al., 2013), clinical trials management systems (Haak et al., 2016), electronic health records systems (de Abajo and Ballestero, 2012) and so on. The plethora offers great opportunities for both clinical researchers and practitioners to accelerate their work with available open-source tools or algorithms. However, it also leaves many people unable to quickly locate the tools most suitable to their clinical research or practices. To make things worse, the lack of adequate usage examples in software documentation is a common issue for open-source software (McColl et al., 2014).

To select appropriate open-source clinical software, researchers or practitioners need to investigate a long list of available open-source software and go through a large volume of relevant literature. The search functions of most popular open-source software hosting platforms, like GitHub, enable users to easily retrieve clinical software to their specific needs. The commonly used databases, such as Google Scholar, PubMed, and Scopus, could help us obtain relevant literature for any open-

source software. But there is no unified platform that links open-source software directly to literature so that users can directly obtain related literature of a given open-source clinical software, instead of collecting and processing information from different sources. Recently, researchers in the life sciences have started to work on combining literature and open-source software (Wang et al., 2017). Wang et al. (2017) built an online biomedical software discovery platform based on data collected from PubMed literature and GitHub. It empowers biomedical researchers to easily find the suitable (open-source) tools they need. However, the study only included biomedical software that was reported in the biomedical literature from PubMed. To date, to the best of our knowledge, no single study has directly focused on building an online platform where users can easily locate and confidently select appropriate open-source clinical software.

Thus, the objective of this work is to bridge the research gap by developing a tool that links open- source clinical software to their literature. Currently, links between clinical software and their literature are described either in the readme files, like the Attention-Gated-Networks repository from GitHub, or in a separate GitHub repository where papers for some clinical software, such as deep learning methods for medical image processing, or blockchain for medical platforms and healthcare, are summarized. Both readme files and literature summaries have related papers hidden in unstructured text so that it is troublesome to extract such information and present it in a structured way. In this work, we first collected a large number of available open-source clinical software and then retrieved literature related to each of them through Google Scholar. Based on the collected data we built a web application with the following main functionalities: 1) searching open-source clinical software with any given topic; 2) showing relevant literature for a selected software, if there is any; 3) updating the existing clinical software and related literature; 4) adding new open-source software and related literature.

The remainder of the paper is organized as follows: section 7.2 explains the research approach, including data collection and processing methods, and artifact design strategies. In section 7.3, we present the system and two common use cases. Evaluation is discussed in section 7.4. section 7.5 and section 7.6 conclude the paper and outline future work.

## 7.2 Methods

### 7.2.1 Design Science

Our research followed the design science research as we built the application, because of its strength and popularity in solving a real-world problem by designing and building an innovative IT artefact (Hevner et al., 2008). In our case, the artefact is a system that links open-source clinical software and literature so that both clinical researchers and practitioners are able to efficiently locate their supporting resources, including both open-source software and papers. Specifically, we follow the design science research methodology (DSRM) proposed by Peffers et al. (2007), which consists of six steps: problem identification and motivation, a definition of the objectives for a solution, design and development, demonstration, evaluation, and communication.

The DSRM was initiated by the (I) problem identification and motivation, which we addressed by literature study and by reviewing other relevant online resources. As stated before, so far little research has been performed on linking open-source clinical tools to literature. But literature is a reliable resource that could provide additional information about clinical software such that we can make an informed decision on choosing the most suitable software. Such additional information could be the technical details of a clinical software, the strength and weakness of a given clinical software when comparing with other similar ones, or the software implementation advices learned from case studies. Based on the identified problem, we inferred (II) the objectives for a solution: creating a tool that links clinical open-source clinical software to their literature. In the (III) design and development, we built a web application. At first, we started by building a data pipeline in which open-source clinical software and literature were collected, processed and stored in our database. Then the artefact, namely the web application, was developed with Node.js and the React framework. To (IV) demonstrate the use of the system, two common use cases were presented. An (V) evaluation experiment measures the system reliability by comparing the automatically generated results from the system with the manually evaluated results. The final step of the DSRM is the (VI) communication. This paper serves as the start of our communication on this topic.

### 7.2.2 Data Pipeline

This section describes the data pipeline which collects open-source clinical software data and literature

#### Data Sources

Nowadays the open-source community has adopted a transparent version control system to manage source codes and other related files for long-term reproducibility and usability (Russell et al., 2018). Git dominates the open-source community with 87.2% of developers using it according to the 2018 Stack Overflow Developer Survey (Stack Overflow Survey, 2018). Open-source software hosted on Git platforms often refers to as a Git repository in which files along with all tracked changes are stored under version control. There are a number of online hosting platforms for Git repositories, including GitHub, SourceForge, Bitbucket and GitLab.

As the largest code host in the world, GitHub has reached 24 million developers working across 67 million repositories in 2017 (Octoverse, 2018). It hosts source codes of numerous open-source software in various domains. Figure 7.1 demonstrates GitHub's rising popularity in the clinical field by the proportion of PubMed articles mentioning GitHub in the title or abstract. In comparison with other platforms, GitHub has become the most used one and grows at the highest rate. Therefore, Given that GitHub repositories provide a good representation of available open-source software in the clinical domain, this study obtained open-source clinical software solely from GitHub. Besides, GitHub offers an easy API for external users to retrieve data from repositories (Russell et al., 2018).

#### Data Collection

This study refers to clinical software as software that is developed for either clinical practices or clinical research. Biomedical software, such as genome sequencing (Pabinger et al., 2014) or cell screening (Omta et al., 2016), are excluded. Therefore, we selected "(clinical OR medical) OR (patient OR doctor)" as the search term while using GitHub API to retrieve Git repositories. Furthermore, English terms were chosen because more than 97% of GitHub repositories have their names and descriptions in English. Figure 7.2 shows the numbers of Git repositories returned using the search term 'clinical' in different languages.

We chose Google Scholar to conduct our literature search. The literature search for each Git repository contains two main steps: 1) determining search terms for the

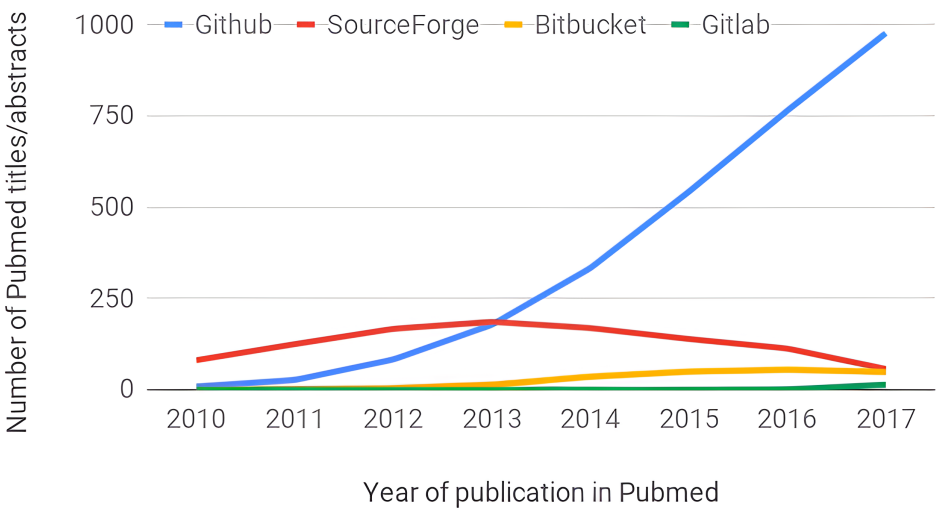



**Figure 7.1:** Trends of Git platforms mentioned in Pubmed titles or abstracts.

repository; 2) searching literature via Google Scholar with the search terms. Each Git repository derives two types of search terms. The first term is the full name of a Git repository, including both the owner name and the repository name, which retrieves papers that specifically mention the Git repository. Another search term contains both the repository name and keywords extracted from the repository description. This term identifies relevant literature that covers similar topics as a given Git repository. For each term, the top ten papers in terms of relevance are collected while there are more than ten papers discovered.

#### Data Extraction

GitHub REST API v3 exposes GitHub repository data to external users. Its search API offers an optimized solution for users to locate the specific items that interest them most, such as Git repositories, users, and issues. GitHub repositories in this study were first obtained by using the search repository API with the above terms. Then we extracted relevant data for each Git repository including name, description, readme files, stars, forks, and programming languages. Afterwards, a filter process excluded repositories based on their popularity and whether they contain source codes or not. We argue that Git repositories that receive no forks or stars three months after their creation are not reliable software or tools. Therefore, we filtered out such Git reposi-

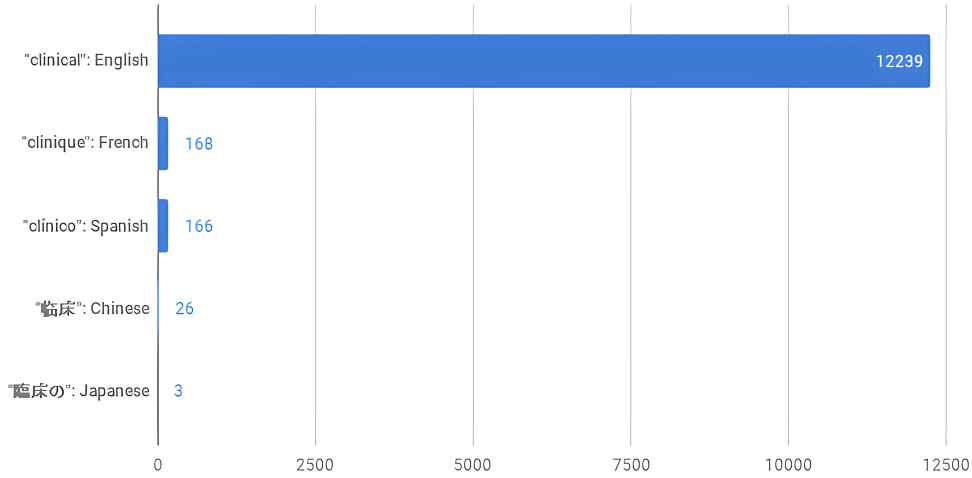



**Figure 7.2:** The number of repositories returned with the search term 'clinical' in different languages.

tories.

### Data Processing

Keywords extracted from a GitHub repository description present an insightful indication of the repository. An external API, namely IBM Watson, is called to process the descriptions, which yields a list of informative keywords for each Git repository.

The extracted keywords cannot be directly employed as search terms for the literature retrieval as the list of keywords will contain many semantically similar keywords. A number of examples are: "patients data" vs "patients information", "medical appointments" vs "doctor appointments", "diabetes patients" vs "diabetic patients". To address this issue, we developed a normalization process in which semantically similar keywords were combined. The process consists of three steps: 1) calculating the semantic similarity between keywords pairs; 2) labelling keyword pairs based on their similarity; 3) replacing keywords with their semantically similar keywords.

Not all papers collected with the abovementioned method are in the clinical scope, especially those retrieved based on the second search term. For instance, ClearCanvas, a medical imaging tool, obtained a few papers about video games, titled as "Implementing Common Components of Video Games", "Build Your Own 2D Game Engine and Create Great Web Games". Therefore, we filtered out literature according

to their relevance to the clinical domain. Specifically, the abstract of each paper was examined to see if it mentions clinically relevant terms like 'clinical', 'medical', 'patient' or 'doctor'. If not, we exclude the paper. Prior to the paper filtering, our GitHub filtering helped us obtain a subset of all collected GitHub repositories. As mentioned earlier, the criteria include whether it has source code, the number of forks larger than 0, the number of stars larger than 0, and the readme file is not empty. In the end, 5119 GitHub repositories and 8820 related papers were collected. Figure 7.3 gives a detailed view of the above-mentioned data pipeline.

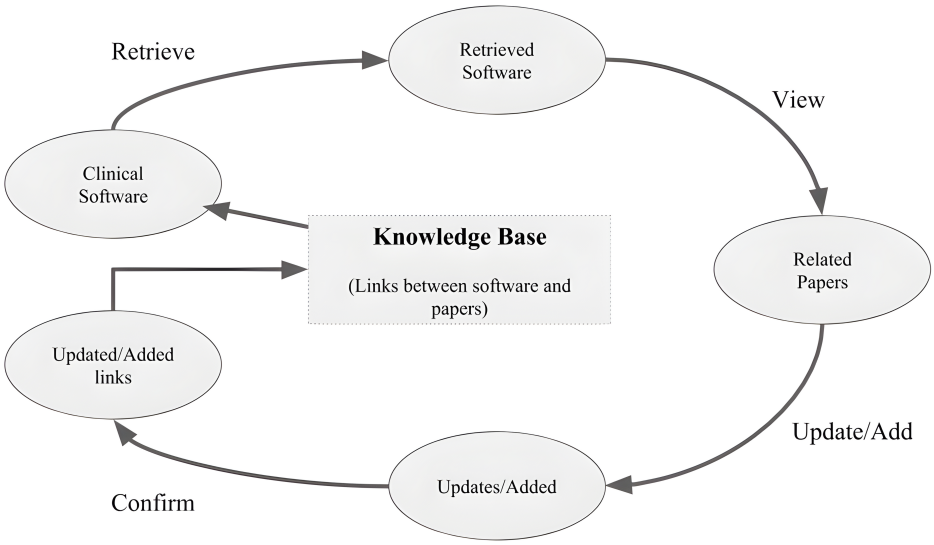



**Figure 7.3:** The peer review cycle.

## 7.3 LOCATE

### 7.3.1 Overview

LOCATE is a web application developed purely in JavaScript with support of popular open-source web development tools: Node.js and React. The web application, running online, is composed of interactive user interfaces and a REST (Representational State Transfer) API as the backend.

### 7.3.2 Key Design Principles

Continuously improving software is crucial in today's ever-changing environment. It is especially true to open-source software developed in academia where reproducibility is essential (Boettiger, 2015). Therefore, we adopted continuous integration which is a software development practice where software is continuously improved (Dingsøyr and Lassenius, 2016). Specifically, we implemented two methods to improve the core of our system, i.e. the data source. Firstly, since both open-source clinical software and literature are constantly added, we accordingly update our database on a regular basis in an incremental manner. Secondly, peer review was introduced to assess and modify the automatically extracted knowledge, particularly the links between open-source clinical software and papers. Figure 4 outlines the peer review cycle. Firstly, experts can retrieve clinical software of their interest and obtain a list of retrieved software. Experts give feedback on current links between the retrieved clinical software and literature, such as confirming, adding or removing links, then the system administrators make the final decisions upon the aggregated feedback. While there is no related paper found, experts can add relevant papers.

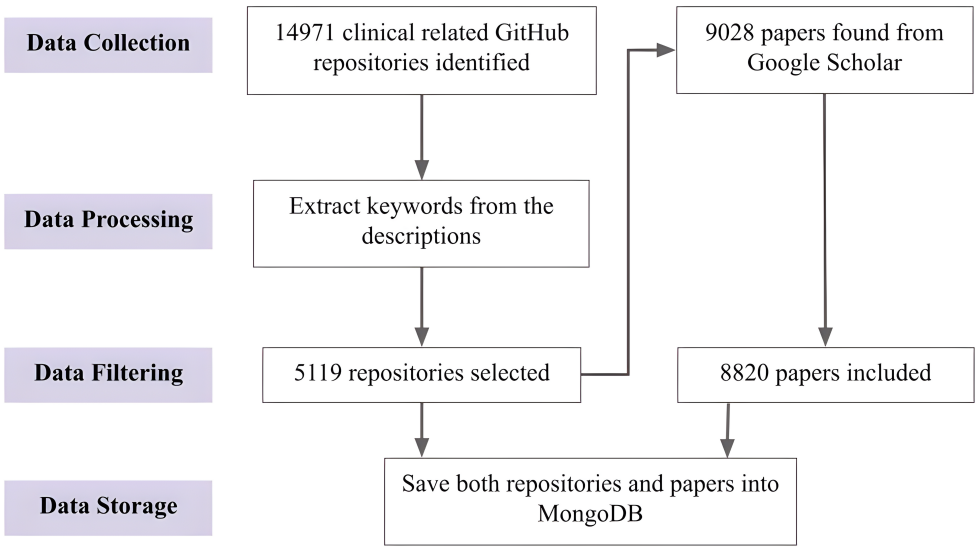



**Figure 7.4:** The data pipeline.

In the spirit of open-source, we made a choice to open source the application. Stewart (2016) grouped open-source software success into two broad types: development success which measures the success of attracting contributors from the open-

source community, and usage success that refers to the user interest/adoption. To ensure development success, source codes are modularized with the Model View Controller (MVC) architecture (Pop and Altar, 2014). The code and architecture simplicity clears barriers faced by newcomers to open-source software projects (Steinmacher et al., 2015). Moreover, we built a REST API that allows you to complete the CRUD (Create, Read, Update, Delete) operations. The REST API handles the server-side requests of the web application while providing a great deal of flexibility in expanding to other applications or the additional requirements of new use cases.

### 7.3.3 Technical Details

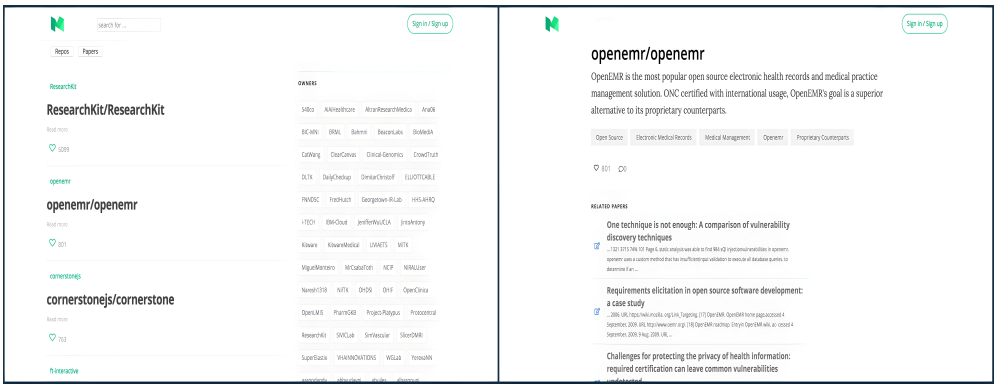
Since we intend to continuously improve the application, its data schema evolves accordingly. To support the dynamic schema of our data and the need for continuously redefining data structures, a non-relational (NoSQL) database, specifically MongoDB, was implemented. As one of the most popular document-based NoSQL databases, MongoDB allows us 1) to update schemas without modifying the existing data, 2) to easily manage the database without complicated database administrator skills, 3) to have good performance and availability (Bradshaw et al., 2019).

Full-text search is utilized to enable users to retrieve clinical software with specified search terms. Since both open-source clinical software and papers in our database contain large chunks of free text that describes them in detail, full-text search is a reasonable option. MongoDB offers a built-in full- text search feature that supports case-insensitive searches on text content.

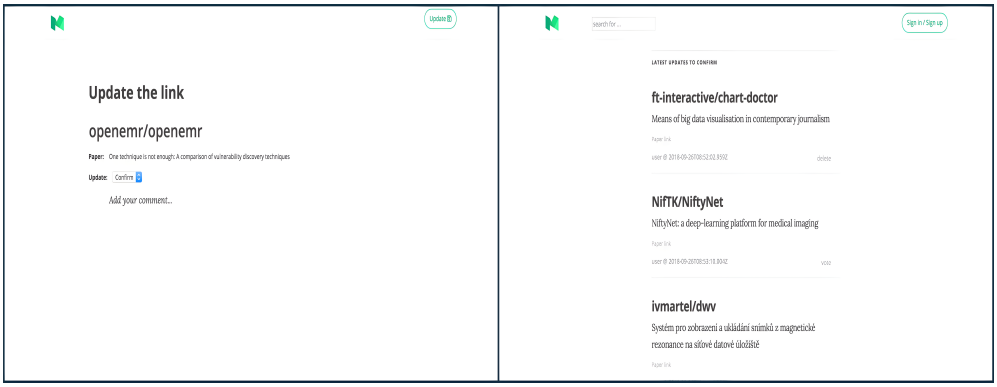
The following screenshots capture several key user interfaces of the application. Figure 7.5a shows the process of searching for open-source clinical software with or without a search term and assessing them based on related papers. The left screenshot shows a list of software returned by a request, while the right one displays a software called 'openemr' and its related papers. The peer review cycle that helps improve the application continuously is demonstrated in Figure 7.5b. The left interface demonstrates how a user update a link between software and literature. To better understand how the application works, we invite you to examine the web application at https://locate-repo.herokuapp.com.

### 7.3.4 Use Cases

As indicated above, LOCATE is capable of assisting both clinical researchers and practitioners. This section describes two use cases in detail to show how the applica-




**(a)** the search and view of open-source clinical software and related papers




**(b)** interfaces of the peer-review cycle

**Figure 7.5:** Screenshots of the application LOCATE

tion can be useful in clinical settings.

The first use case is that of suggesting open- source tools based on literature: given a clinical research topic, e.g. medical image segmentation, researchers who are conducting research on this topic need to investigate relevant literature and explore the potential appropriate tools to support them, preferably open-source ones. In this case, our application is able to recommend useful open-source software based on related literature. Without our application, researchers might need to search literature and software separately, which is a tedious process.

The second use case discussed in detail is that of choosing open-source tools. The lack of sufficient documentation in much open-source clinical software makes it difficult to assess their usability and reliability. In this context, the system matches open-

source software to their relevant literature which elucidates open-source clinical software in various aspects: ranging from detailing the technical features, comparing similar tools, to applications of tools in solving practical issues. With such additional information, researchers can make a more informed decision with regard to which tools to use.

## 7.4 Evaluation

This section presents the results of our evaluation experiment on whether the system offers additional knowledge about open-source clinical software. We selected five popular open-source clinical software projects to review the value of linked papers. Specifically, we examine the associated papers of each software and compare them with their GitHub repository documentation.

| Software | Description | Docs | Papers from LOCATE |
|---|---|---|---|
| OpenEMR | Electronic health records and medical practice management solution | Features, manuals, and forum where people share their knowledge | Comparing OpenEMR with other open-source EMR systems; Case studies on specific issues of OpenEMR, such as performance and security. |
| DICOM Viewer | Medical image viewer | Setup manual | A survey of DICOM viewer software; Web case studies |
| DLTK | Deep Learning Toolkit for Medical Image Analysis | Setup manual, tutorials, sample applications | Paper explaining the tool; Case studies |
| Open Clinica | Open-source clinical trial software for Electronic Data Capture (EDC) Clinical Data Management (CDM) | Features, setup manual, forum | Case studies |
| NiftyNet | An open-source convolutional neural networks platform for research in medical image analysis and image-guided therapy | Features, manuals, other resources including StackOverflow questions | Paper explaining the platform; Survey papers; Paper on similar tools |

**Table 7.1:** Comparison between software documentation and linked papers from LOCATE.

As shown in Table 7.1, related papers from LOCATE provide additional knowledge about open-source clinical software from several aspects: 1) a more detailed description of the development of the software; 2) studies that compare a number of similar open-source software projects; 3) case studies that apply the software to solve a specific clinical issue. Nevertheless, such knowledge cannot be obtained from software documentation which commonly exists as readme files, setup manuals, tutori-

als and so on. Therefore, the results confirm our assumption that enriching software documentation with its linked papers enables more informed decision making.

## 7.5 Discussion

This study developed a web application that links open-source clinical software with their related literature. To our best knowledge, no such studies have been conducted and our study is the first attempt to combine the two valuable components of today's clinical research. The tool which is available as an online web application offers an easy and openly accessible representation of our study. Moreover, an evaluation experiment which compared software documentation with the linked papers from LOCATE, outlines how the application enables more informed decision making.

The application has the potential of being beneficial for both practitioners and researchers in the clinical community. Practitioners obtain knowledge about the clinical tools of their interest from related literature so that they can better assess which one to choose. On the other hand, clinical researchers are provided with a list of potential useful open-source tools based on their research topics so that it might save a substantial amount of time by directly using available software or customizing them to fit their own research goals.

Furthermore, source codes of our study are open- sourced under the MIT license and hosted at https://github.com/ianshan0915/locate. People in other domains are able to utilize the source codes and conduct similar research in other domains.

Nevertheless, some limitations regarding this study should be noted. Firstly, GitHub is the only source for open-source clinical software in the study. Although GitHub is a major platform where developers work and share their codes, there are other platforms which are of great importance to the open-source community, such as SourceForge and GitLab. Collecting more data from other platforms might yield a more complete list of open-source clinical software.

Secondly, a more comprehensive evaluation is necessary, for example through a system usability measurement with a customized System Usability Scale (SUS) and case studies with clinical researchers or participating practitioners. Then, the usability study will quantitatively assess our application in terms of functionality and user-friendliness. Furthermore, the case study could provide a qualitative evaluation of the application from the perspective of potential users. Specific suggestions for further improvement are the expected results.

Last but not the least, the development of LOCATE is ongoing. A revised version

is being developed. New features will also be added. For instance, natural language processing techniques will be used to process the textual data so that the application can support keyword-based retrieval.

## 7.6 Conclusions

The primary goal of the paper has been to develop an application that supports the clinical community to easily and confidently locate open-source software. The web application we built offers user-friendly interfaces and are publicly accessible online. Furthermore, its iterative development process ensures the continuous improvement of the application and that the rapid updates of open-source clinical software are incorporated. As the first attempt to link open-source clinical software to literature, we have laid down the groundwork for more research on this topic so that open-source clinical software can be better utilized to contribute to both research and practice.

# Chapter 8

# Conclusions

The purpose of this research is to enhance the development and deployment of healthcare information systems in practice. The interdisciplinarity of HIS engineering research provides us research opportunities from both IT and healthcare perspectives. This dissertation focused on: 1) how to solve specific real-world healthcare problems with HIS; 2) how to employ state-of-the-art of information technology to accelerate HIS engineering. The empirical work of this dissertation demonstrates the usefulness and effectiveness of HIS in practice.

In this concluding chapter, we answer the Main Research Question (MRQ) in section 8.1 by discussing answers to all Research Questions (RQs) as presented in section 1.4. We end by providing a reflection on this dissertation in section 8.2, which includes a discussion of its limitations (subsection 8.2.1), an outlook on future work (subsection 8.2.2), and a personal reflection (**??**).

## 8.1 Answers to Research Questions

**MRQ**

The aim of this research is formalised in the following research question:

> *How can we employ artificial intelligence technologies – such as machine learning algorithms, knowledge systems and natural language processing techniques – based on open source principles to accelerate healthcare information system engineering in solving real-world clinical problems?*

Turning to our main research question, we have investigated the application of various AI technologies in healthcare. The dissertation started with the clinical descision support system we developed in chapter 2, where we focused on designing a federated information architecture to integrate various healthcare data sources from different countries. The system supported a large multinational randomized clinical trial over 2 years without any technical issue. However, we also identified a number of improvements to the system, such as enhancing the data entry process with NLP-enabled data extraction. To address this we experimented with an NLP prototype in chapter 3. The prototype built on a proposed lightweight NLP framework which extracted medical entities such as diseases and medications and then transformed these into standardized medical codes. The evaluation of the prototype shows promising results, so there is a potential of using the system to replace data entry in healthcare information systems. In chapter 4, we further explored the use of NLP data extraction for adverse drug reactions from drug product labels. An NLP pipeline was developed to collect drug product labels from Electronic Medicines Compendium (EMC) and then extract adverse drug reactions from them. The evaluation conducted on 647 common medicines shows that the pipeline is capable of extracting adverse drug reactions with high precision and recall. It has the potential of being used to solve ADR related clinical problems, such as creating adverse drug reaction databases (Kuhn et al., 2016; Garcia et al., 2023).

We demonstrated the potential of NLP techniques in extracting information from clinical texts. But all the studies were conducted on small datasets. Given the accumulation of huge amounts of clinical data over last decades, it is worth exploring the use of NLP techniques in extracting information from large-scale clinical data. In chapter 5, we proposed a big data framework for scalable and efficient information extraction from large-scale biomedical and clinical data. This framework enabled us to perform topic modeling on **over 1 million full-text articles** from the PMC Open Access Subset in about 25 minutes at a cost of $1.5 USD on Google Cloud.

With the development of AI, there are so many AI technologies including NLP techniques, algorithms, packages. In chapter 6, a systematic literature review is conducted to identify the state-of-the-art of open source clinical software. In chapter 7, we have developed a web-based application to support clinical developers' decisions in the software development process with a reproducible and scalable method for systematic studies on open source clinical software.

This dissertation has demonstrated how to help accelerate healthcare information system engineering with AI technologies in real-world clinical settings. Both

scientific and technical contributions are of great importance in our studies. Besides designing federated information architecture, big data framework and systematic literature review, we also developed NLP pipelines and web-based applications with open source principles. We hope that clinical developers and researchers will reuse our public accessible source codes and data for their research and clinical applications (Tanaka et al., 2024; Garcia et al., 2023).

**RQ1 (chapter 2)**

> *How can we design a GDPR-compliant clinical decision support system that supports a multi-center multi-lingual clinical trial?*

To answer this question, this chapter designed a federated information architecture that is capable of integrating data from different countries in a unified schema. A clinical decision support system named STRIPA was developed based on the above architecture to support a large multinational randomized clinical trial which requires multilingual support, data security and privacy, data accessibility and consistency. Evaluation of the system was conducted in both development and deployment. Throughout development random manual comparisons of data between databases and data sources were performed to test on data completeness, data consistency and data integrity. During the clinical trial information of 2008 older adults from 110 clusters of inpatient wards within university based hospitals in four European countries were stored and processed. STRIPA supported the clinical trial for 2 years without any technical issue (Blum et al., 2021a). Sensitive patient data in the system were kept secure and private.

From a broader perspective, the success of this clinical decision support system leads us to believe that the federated information architecture and data integration methods have a good potential to be reused in similar scenarios, especially, in large-scale clinical trials.

**RQ2 (chapter 3)**

> *How can we improve rapid and cost-effective development of clinical NLP systems with external NLP APIs?*

There are various methods and frameworks for developing clinical NLP systems. However, most of them require a tremendous amount of resources to ensure successful implementation (Chiang et al., 2010). In this chapter we presented a lightweight

framework which enables a rapid development of clinical NLP systems with external NLP APIs. In addition, a web-based open source clinical NLP application was built to validate the framework and they together answer RQ2.

The lightweight framework consists of four main components: external APIs, infrastructure, NLP pipelines, and Apps. External API refers to cloud-based NLP services which are available for everyone via APIs. The pay-as-you-go billing model offers end-users flexibility and freedom. The infrastructure layer ensures data privacy and security by preparing clinical data with deidentification and authentications before sending them to external APIs. NLP pipelines parse unstructured medical text and map them into structured and standardized medical codes with the external APIs component. By assembling suitable cloud-based NLP services on demand in NLP pipelines, we are able to process unstructured medical text with the least effort. In the application layer, clinical NLP-enabled applications for various needs can be created.

A clinical concept extraction app is created based on the proposed framework in the chaper. It extracts clinical concepts from clinical free texts. Evaluation of the application on four test datasets shows satisfactory overall results, which proves the effectiveness and capabilities of the proposed framework. Moreover less time and resources are required to create and maintain NLP-enabled clinical tools given that all NLP tasks are outsourced.

**RQ3 (chapter 4)**

> *How can we utilize NLP techniques to automatically extract adverse drug reactions from the summary of product characteristics in European drug product labels?*

In this chapter, we designed an NLP pipeline that automatically scrapes the summary of product characteristics online and then extracts adverse drug reactions from them. It starts with scraping side effects data of 647 commonly used medicines from the Electronic Medicines Compendium (EMC) (Electronic Medicines Compendium (EMC), 2020). The second step of the pipeline is to extract adverse drug reactions from different formats of side effects data. For side effects that are recorded in free-text, named entity recognition (NER) is used. Side effects written in structured text can be easily extracted by syntax parsing.

The manual experts review results demonstrate that our proposed method is effective and has the potential of being used to solve ADR related clinical problems.

Specifically, our method achieves an overall recall of 0.990 and a precision of 0.932.

**RQ4 (chapter 5)**

> *How can we design an open source big data framework to facilitate cost-effective, large-scale, biomedical literature mining?*

To address this research question, we proposed a big data framework specifically designed for large-scale biomedical literature mining. Our framework consists of three key components: storage layer, management layer and spark apps layer. These components work together on top of a cloud infrastructure layer (Infrastructure-as-a-Service) to provide a scalable, flexible, efficient, and affordable solution for large-scale biomedical literature mining.

To validate the effectiveness of our framework, we conducted two case studies to evaluate the performance of our framework. To begin with, we performed topic modeling on over 1 million full-text articles from the PMC Open Access Subset, demonstrating the framework's scalability by analyzing a much larger dataset than previous studies. It processed over 1 million articles in about 25 minutes at a cost of $1.5 USD, proving significantly faster and more cost-effective than other systems. The second study conducted cancer article classification using 29,437 labeled full-text articles, comparing performance with SparkText. Our framework was 6 times faster, with lower costs and improved prediction results. These case studies effectively showcased the framework's scalability, efficiency, and cost-effectiveness for large-scale biomedical literature mining tasks, outperforming existing solutions in execution time, cost, and prediction performance.

**RQ5 (chapter 6)**

> *How can we support clinical developers' decisions in the software development process with a reproducible and scalable method for systematic studies on open source clinical software?*

To conduct the systematic studies on open source clinical software, we developed a reproducible and scalable data pipeline to collect and analyze GitHub repository data, providing insights into the current status, popularity factors, and main focus areas of open source clinical software. Our approach offers several key contributions:

- A quantitative overview of the open source clinical software landscape, including growth trends, popular repositories, and geographic distribution.

- Identification of factors impacting repository popularity, such as forks and number of contributors.
- Discovery of 10 main focus areas through topic modeling, revealing the evolving trends in clinical software development.
- A reproducible methodology that can be easily adapted to study open source software in other domains.

By providing these insights and a reusable approach, this study empowers clinical developers to make more informed decisions in their software development process. They can better understand the available open source resources, identify popular and actively maintained projects, and recognize emerging trends in clinical software development. The reproducible methodology in this study also encourages collaboration and continuous improvement in the field, as other researchers can build upon and extend this work to provide even more comprehensive insights in the future.

While limitations exist, such as the focus on GitHub and manual topic labeling, this research lays the groundwork for future studies on open source clinical software. The reproducible nature of our method encourages further refinement and application to support ongoing improvements in clinical software development and reuse.

**RQ6 (chapter 7)**

> *How can we design an online platform where clinical developers can easily locate and confidently select appropriate open-source clinical software based on associated scientific literature?*

This chapter presents LOCATE, a web application designed to address the challenge of efficiently linking open-source clinical software with relevant scientific literature. The system answers the main research question by:

1. Developing a data pipeline that collects and processes information about open-source clinical software from GitHub and associated literature from Google Scholar.
2. Creating an interactive web application that allows users to search for clinical software and view related papers, providing additional insights beyond typical software documentation.

3. Implementing a peer review cycle that enables continuous improvement of the platform through expert feedback.

4. Offering an open-source, modular architecture that facilitates further development and expansion.

The evaluation demonstrates that LOCATE provides valuable additional knowledge about clinical software through linked papers, enabling more informed decision-making for both researchers and practitioners. While limitations exist, such as relying solely on GitHub for software data, LOCATE represents a significant step towards bridging the gap between open-source clinical software and scientific literature. This work lays the foundation for future research and improvements in this area, ultimately contributing to more effective utilization of open-source tools in clinical research and practice.

## 8.2 Reflection

This section of the dissertation discusses the Limitations of this work, proposes some future research opportunities, and ends with a personal reflection.

### 8.2.1 Limitations

We first reflect on the AI technologies used in this dissertation. We then discuss the datasets we used to evaluate the performance of the proposed methods. We continue with discussing the privacy and security issues in HIS. We conclude with elaborating on the impact of LLMs.

#### AI technologies

AI technologies cover a variety of topics including machine learning, deep learning, NLP and LLM. AI technologies used in this dissertation include traditional NLP techniques like topic modeling and text classification. In chapter 4, chapter 5 and chapter 6, we employed AI technologies to process big clinical data. While these methods demonstrated good performance for their respective tasks, they have some inherent limitations. The NLP techniques used in this dissertation are relatively basic compared to current state-of-the-art approaches, such as deep learning or transformer-based models. For example, chapter 4 uses rule-based parsing techniques specifically designed for SmPC (Summary of Product Characteristics) documents, rather

than more generalizable approaches. chapter 5 and chapter 6 rely on traditional topic modeling and text classification methods instead of exploring recent advances in neural architectures that could potentially yield better results. Given the wide adoption of deep learning methods in NLP tasks, some comparison studies would be useful.

#### Data bias

One key threat to the validity of this dissertation is the data bias. First, most of data used to evaluate the proposed methods and prototypes in this study are in English. For instance, the ADR extraction pipeline in chapter 4 was tailored for English drug products from SmPC. The decision to overlook this issue was not intentional but rather a consequence of broader limitations in clinical data resource availability. Specifically, public medical and clinical datasets in general remain scare, and datasets languages other than English are exceptionally scarce. However, these methods can be adapted for application in non-English clinical data. The big data framework from chapter 5 is applicable to large clinical datasets in any language. Secondly, the open source research in chapter 6 and chapter 7 are purely based on GitHub. Although GitHub serves as the main platforms for open source projects, there are also other platform such as GitLab and Bitbucket.

#### Privacy and security

Data privacy and security is considered as one of the most important aspects in research using clinical data. Although systems from chapter 3 and chapter 4 included the privacy and security component, this study lacks explicit research focusing on privacy-preserving methodologies and security measures for clinical data storage and transmission. Given the sensitive nature of health information, ensuring patient confidentiality and safeguarding data against breaches are critical in real-world applications. Therefore, privacy and security related rules and guidelines should be carefully examined, and it lays the foundation for the utilization of AI technologies in clinical settings.

The advocacy of open source clinical software in this study has its security limitations. While open source software offers benefits such as cost-effectiveness, transparency, and community-driven innovation, its application in healthcare has some security concerns. First of all, open source software may contain undiscovered vulnerabilities or dependencies on inadequately maintained libraries, which could expose sensitive patient data to breaches if not rigorously audited or updated. Unlike

commercial software which has built-in safeguards for compliance with healthcare-specific regulations (e.g., HIPAA, GDPR), open source software needs extra work to meet the compliance requirements. Lastly, security patches and updates depend on community contributions, which may lag behind emerging threats, leaving systems unprotected during critical windows.

### Impact of LLMs

This dissertation was completed before the release of ChatGPT. The rapid adoption and advancement of LLMs after ChatGPT have fundamentally transformed the application of clinical natural language processing, making many of the techniques presented in this work obsolete. The powerful capabilities of LLMs in general tasks surpass the rule-based parsing and traditional NLP methods employed in this research. While the approaches introduced in this research were state-of-the-art at the time of their creation, LLMs have introduced a paradigm shift in processing clinical text.

The API-based framework proposed in chapter 3 aligns remarkably well with today's LLM ecosystem, as both rely on external service integration through APIs. This architectural similarity makes the framework highly adaptable to LLM integration. Furthermore, LLMs significantly simplify the framework by consolidating multiple NLP tasks (entity extraction, negation detection, etc.) into a single API call, reducing the need for complex pipeline orchestration and local NLP implementations. The framework's modular design allows for seamless replacement of traditional NLP APIs with LLM APIs while maintaining its core benefits of flexibility and configurability.

chapter 4 employed various methods, such as rule-based parsing, named entity recognition, and HTML structure analysis, to extract ADR from SmPC documents. Although being an effective method with high precision and recall, it is quite limited in comparison with today's LLMs.

1. **Generalization:** The rule-based approach is tailored to SmPC documents and requires manual rule creation. LLMs can generalize accross document formats without explicit rules. By simply adjusting prompts, LLMs would process different product labels with ease.

2. **Context Understanding:** Traditional NER struggles with contextual understanding, while LLMs excel at understanding context and can better handle ambiguous cases.

3. **Error Handling:** To handle specific error cases (e.g., handling "or", "and" in ADR lists), manual checks were introduced. LLMs can better handle such variations naturally.

Traditional NLP techniques, such as topic extraction and text classification employed in other chapters can be easily replaced by LLMs. It simplifies the clinical text processing pipelines in this dissertation and improves the generalization of our approaches. However, due to the limitations of today's LLM models in explainability, transparency and reliability, extensive tests and evalation are necessary before deployment in clinial setting which requires trust, transparency and consistency. The architectural complexity and hundreds of billions parameters make it difficult to trace how LLMs generate outputs. Moreover, AI companies which built the best LLMs in the world are more reluctant to share how they train their models and what data is used. As a well-known feature of LLMs from the beginning, hallucination produces false information and incorrect responses which is a huge challenge in high-stakes domains like healthcare.

### 8.2.2 Future Work

This dissertation has contributed to overcoming challenges in healthcare information system engineering, especially in clinical settings. Besides developing frameworks and tools to solve specific HIS problems, we also introduced open source methodology as best practice for HIS engineering. Nevertheless, there are still many improvements and unexplored territories in this field. A few future research areas can be imagined, and we will describe three of them below.

1. **Transitioning from prototype to practice –** Due to limited resources and time, we only managed to validate the architecture proposed in chapter 2 via a multinational randomized clinical trial. Further research or experiments on applying the proposed tools and frameworks from Chapter 3-5 in practice are necessary to prove their true value to healthcare. The lightweight API-based NLP framework from chapter 3 could be integrated into hospital information systems (HIS) to automate clinical concept extraction from discharge summaries or medical records, thereby reducing manual effort and improving decision-making. Similarly, The big data framework for biomedical literature mining designed in chapter 5 could be deployed in clinical research institutions to analyze large-scale biomedical/clinical literature of any given topic. Moreover, chapter 4 constructed an adverse drug reaction (ADR) extraction pipeline which is

intended for the development of ADR knowldege base. Then the knowledge base can be used for automatic ADR detection from electronic patient records.

However, practical implementation requires addressing several challenges, including privacy and security concerns, computational efficiency, and seamless integration with existing clinical workflows. Therefore, before deploying anything in pratice we should start with a small proof of concept in a well-defined context.

2. **Leveraging LLMs in clinical text processing –** As discussed in subsection 8.2.1, LLMs are superior to traditional NLP techinques in many tasks. Future research should focus on integration of LLMs with frameworks or approaches introduced in this dissertation. Given the generalization of LLMs, it is worth investigating the replacement of various NLP APIs with LLMs, and get a more lightweight framework with better performance. As a follow-up research on ADR extraction pipeline from chapter 4, we could explore using LLMs to simplify the extraction pipeline and add multilingual support. chapter 5 only used two NLP tasks to validate the efficiency and effectiveness of the framework, more NLP tasks need to be tested. LLMs could make this very easy by just adjusting the prompts. Therefore, future research on integrating LLMs will improve the framework.

3. **Expanding knowledge sources –** As the largest code host platform in the world, Github hosted hundreds of millions projects with millions of active developers. However, besides GitHub, there are other platforms which also host open source projects, such as GitLab, Bitbucket and SourceForge. Therefore including open source projects from other platforms will provide a more hollistic view of open source clinical software. Likewise, literature search and collection in chapter 6 was limited to Google Scholar. Since Google Scholar often fails to index the latest literature in time, especially in fields with rapid development, like AI, we could fill the gap by including literature from arXiv. Another future research direction lies with developing a retrieval augmented generation (RAG) based chatbot on the expanded knowledge sources. The chatbot will offer an easier and interactive way for retrieving related literature for a given opensource clinical software.

# Bibliography


A Seoane, J., Aguiar-Pulido, V., R Munteanu, C., Rivero, D., R Rabunal, J., Dorado, J., and Pazos, A. (2013). Biomedical data integration in computational drug design and bioinformatics. *Current computer-aided drug design*, 9(1):108–117.

Abdallah, Z. S., Carman, M., and Haffari, G. (2017). Multi-domain evaluation framework for named entity recognition tools. *Computer Speech & Language*, 43:34–55.

Akowuah, F., Lake, J., Yuan, X., Nuakoh, E., and Yu, H. (2015). Testing the security vulnerabilities of openemr 4.1. 1: a case study. *Journal of Computing Sciences in Colleges*, 30(3):26–35.

Albahri, A. S., Hamid, R. A., Alwan, J. K., Al-Qays, Z., Zaidan, A., Zaidan, B., Albahri, A., AlAmoodi, A. H., Khlaf, J. M., Almahdi, E., et al. (2020). Role of biological data mining and machine learning techniques in detecting and diagnosing the novel coronavirus (covid-19): a systematic review. *Journal of medical systems*, 44:1–11.

Alharthi, A., Krotov, V., and Bowman, M. (2017). Addressing barriers to big data. *Business Horizons*.

Ansper, A., Buldas, A., Freudenthal, M., and Willemson, J. (2013). Protecting a federated database infrastructure against denial-of-service attacks. In *Critical Information Infrastructures Security: 8th International Workshop, CRITIS 2013, Amsterdam, The Netherlands, September 16-18, 2013, Revised Selected Papers 8*, pages 26–37. Springer.

Anthes, G. (2016). Open source software no longer optional. *Communications of the ACM*, 59(8):15–17.

APIs, N. (2017). Api-based cms buyer's guide. Accessed: 2024-08-22.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79-80:3 – 15. Special Issue on Scalable Systems for Big Data Management and Analytics.

Athey, B. D., Braxenthaler, M., Haas, M., and Guo, Y. (2013). transmart: an open source and community-driven informatics and data sharing platform for clinical and translational research. *AMIA Summits on Translational Science Proceedings*, 2013:6.

Badgeley, M. A., Shameer, K., Glicksberg, B. S., Tomlinson, M. S., Levin, M. A., McCormick, P. J., Kasarskis, A., Reich, D. L., and Dudley, J. T. (2016). Ehdviz: clinical dashboard development using open-source technologies. *BMJ open*, 6(3):e010579.

Bahrami, M. and Singhal, M. (2015). *The Role of Cloud Computing Architecture in Big Data*, pages 275–295. Springer International Publishing, Cham.

Banda, J. M., Evans, L., Vanguri, R. S., Tatonetti, N. P., Ryan, P. B., and Shah, N. H. (2016). A curated and standardized adverse drug event resource to accelerate drug safety research. *Scientific data*, 3(1):1–11.

Bankhead, P., Loughrey, M. B., Fernández, J. A., Dombrowski, Y., McArt, D. G., Dunne, P. D., McQuaid, S., Gray, R. T., Murray, L. J., Coleman, H. G., et al. (2017). Qupath: Open source software for digital pathology image analysis. *Scientific reports*, 7(1):1–7.

Batini, C., Lenzerini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM computing surveys (CSUR)*, 18(4):323–364.

Beynon-Davies, P., Tudhope, D., and Mackay, H. (1999). Information systems prototyping in practice. *Journal of information technology*, 14(1):107–120.

Björkman, I. K., Fastbom, J., Schmidt, I. K., Bernsten, C. B., and of the Elderly in Europe Research (PEER) Group, P. C. (2002). Drug—drug interactions in the elderly. *Annals of Pharmacotherapy*, 36(11):1675–1681.

Blaya, J. A., Fraser, H. S., and Holt, B. (2010). E-health technologies show promise in developing countries. *Health Affairs*, 29(2):244–251.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Blum, M. R., Sallevelt, B. T. G. M., Spinewine, A., O'Mahony, D., Moutzouri, E., Feller, M., Baumgartner, C., Roumet, M., Jungo, K. T., Schwab, N., Bretagne, L., Beglinger, S., Aubert, C. E., Wilting, I., Thevelin, S., Murphy, K., Huibers, C. J. A., Drenth-van Maanen, A. C., Boland, B., Crowley, E., Eichenberger, A., Meulendijk, M., Jennings, E., Adam, L., Roos, M. J., Gleeson, L., Shen, Z., Marien, S., Meinders, A.-J., Baretella, O., Netzer, S., de Montmollin, M., Fournier, A., Mouzon, A., O'Mahony, C., Aujesky, D., Mavridis, D., Byrne, S., Jansen, P. A. F., Schwenkglenks, M., Spruit, M., Dalleur, O., Knol, W., Trelle, S., and Rodondi, N. (2021a). Optimizing therapy to prevent avoidable hospital admissions in multimorbid older adults (operam): cluster randomised controlled trial. *BMJ*, 374. `https://www.bmj.com/content/374/bmj.n1585`.

Blum, M. R., Sallevelt, B. T. G. M., Spinewine, A., O'Mahony, D., Moutzouri, E., Feller, M., Baumgartner, C., Roumet, M., Jungo, K. T., Schwab, N., Bretagne, L., Beglinger,

S., Aubert, C. E., Wilting, I., Thevelin, S., Murphy, K., Huibers, C. J. A., Drenth-van Maanen, A. C., Boland, B., Crowley, E., Eichenberger, A., Meulendijk, M., Jennings, E., Adam, L., Roos, M. J., Gleeson, L., **Shen, Z.**, Marien, S., Meinders, A.-J., Baretella, O., Netzer, S., de Montmollin, M., Fournier, A., Mouzon, A., O'Mahony, C., Aujesky, D., Mavridis, D., Byrne, S., Jansen, P. A. F., Schwenkglenks, M., **Spruit, M.**, Dalleur, O., Knol, W., Trelle, S., and Rodondi, N. (2021b). Optimizing therapy to prevent avoidable hospital admissions in multimorbid older adults (operam): cluster randomised controlled trial. *BMJ*, 374.

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Boettiger, C. (2014). An introduction to docker for reproducible research, with examples from the r environment. *ACM SIGOPS Oper. Syst. Rev.*, 49.

Boettiger, C. (2015). An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90.

Bradshaw, S., Brazil, E., and Chodorow, K. (2019). *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media.

Branson, A., Hauer, T., McClatchey, R., Rogulin, D., and Shamdasani, J. (2008). A data model for integrating heterogeneous medical data in the health-e-child project. *arXiv preprint arXiv:0812.2874*.

Brazhnik, O. and Jones, J. F. (2007). Anatomy of data integration. *Journal of biomedical informatics*, 40(3):252–269.

Bretthauer, D. (2002). Open source software: A history. *UConn Libraries Published Works*, 21.

Cai, T., Giannopoulos, A. A., Yu, S., Kelil, T., Ripley, B., Kumamaru, K. K., Rybicki, F. J., and Mitsouras, D. (2016). Natural language processing technologies in radiology research and clinical applications. *Radiographics*, 36(1):176–191.

Calì, A., Calvanese, D., De Giacomo, G., and Lenzerini, M. (2002). Data integration under integrity constraints. In *Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Toronto, Canada, May 27–31, 2002 Proceedings 14*, pages 262–279. Springer.

Canonico, M. and De Russis, L. (2018). A comparison and critique of natural language understanding tools. In *Cloud Computing 2018*, pages 110–115. CLOUD COMPUTING 2018: The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization ; Conference date: 01-01-2018.

Carchiolo, V., Longheu, A., Reitano, G., and Zagarella, L. (2019). Medical prescription classification: a nlp-based approach. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 605–609. IEEE.

Cardinal, R. N. (2017). Clinical records anonymisation and text extraction (crate): an open-source software system. *BMC medical informatics and decision making*, 17:1–12.

Carrell, D. (2011). A strategy for deploying secure cloud-based natural language processing systems for applied research involving clinical text. In *2011 44th Hawaii International Conference on System Sciences*, pages 1–11. IEEE.

Cavelaars, M., Rousseau, J., Parlayan, C., de Ridder, S., Verburg, A., Ross, R., Visser, G. R., Rotte, A., Azevedo, R., Boiten, J.-W., et al. (2015). Openclinica. In *Journal of clinical bioinformatics*, volume 5-1, pages 1–2. BioMed Central.

Cañada, A., Capella-Gutierrez, S., Oyarzabal, J., Rabal, O., Valencia, A., and Krallinger, M. (2017). LimTox: a web tool for applied text mining of adverse event and toxicity associations of compounds, drugs and genes. *Nucleic Acids Research*, 45(W1):W484–W489.

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., and Buchanan, B. G. (2001). Evaluation of negation phrases in narrative clinical reports. In *Proceedings of the AMIA Symposium*, page 105. American Medical Informatics Association.

Chapman, W. W., Cooper, G. F., Hanbury, P., Chapman, B. E., Harrison, L. H., and Wagner, M. M. (2003). Creating a text classifier to detect radiology reports describing mediastinal findings associated with inhalational anthrax and other disorders. *Journal of the American Medical Informatics Association*, 10(5):494–503.

Chard, K., Russell, M., Lussier, Y. A., Mendonça, E. A., and Silverstein, J. C. (2011). A cloud-based approach to medical nlp. In *AMIA Annual Symposium Proceedings*, volume 2011, page 207. American Medical Informatics Association.

Che, D., Safran, M., and Peng, Z. (2013). From big data to big data mining: challenges, issues, and opportunities. In *Database Systems for Advanced Applications: 18th International Conference, DASFAA 2013, International Workshops: BDMA, SNSM, SeCoP, Wuhan, China, April 22-25, 2013. Proceedings 18*, pages 1–15. Springer.

Chiang, J.-H., Lin, J.-W., and Yang, C.-W. (2010). Automated evaluation of electronic discharge notes to assess quality of care for cardiovascular diseases using medical language extraction and encoding system (medlee). *Journal of the American Medical Informatics Association*, 17(3):245–252.

Chyu, M.-C., Austin, T., Calisir, F., Chanjaplammootil, S., Davis, M. J., Favela, J., Gan, H., Gefen, A., Haddas, R., Hahn-Goldberg, S., et al. (2015). Healthcare engineering defined: a white paper. *Journal of healthcare engineering*, 6(4):635–648.

Claxton, A. J., Cramer, J., and Pierce, C. (2001). A systematic review of the associations between dose regimens and medication compliance. *Clinical therapeutics*, 23(8):1296–1310.

Clifton, C., Housman, E., and Rosenthal, A. (1998). Experience with a combined approach to attribute-matching across heterogeneous databases. In *Data Mining and Reverse Engineering: Searching for semantics. IFIP TC2 WG2. 6 IFIP Seventh Conference on Database Semantics (DS-7) 7–10 October 1997, Leysin, Switzerland*, pages 428–451. Springer.

Creswell, J. W., Fetters, M. D., and Ivankova, N. V. (2004). Designing a mixed methods study in primary care. *The Annals of Family Medicine*, 2(1):7–12.

Crowley, E. K., Sallevelt, B. T., Huibers, C. J., Murphy, K. D., **Spruit, M.**, **Shen, Z.**, Boland, B., Spinewine, A., Dalleur, O., Moutzouri, E., et al. (2020). Intervention protocol: Optimising therapy to prevent avoidable hospital admission in the multi-morbid elderly (operam): a structured medication review with support of a computerised decision support system. *BMC health services research*, 20(1):1–12. https://doi.org/10.1186/s12913-020-5056-3.

Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS computational biology*, 9(2):e1002854.

Currie, W. L., Desai, B., and Khan, N. (2004). Customer evaluation of application services provisioning in five vertical sectors. *Journal of Information Technology*, 19(1):39–58.

Dale, R. (2015). Nlp meets the cloud. *Natural language engineering*, 21(4):653–659.

Davis, L. (2012). *Basic methods in molecular biology*. Elsevier.

de Abajo, B. S. and Ballestero, A. L. (2012). Overview of the most important open source software: analysis of the benefits of openmrs, openemr, and vista. In *Telemedicine and e-health services, policies, and applications: Advancements and developments*, pages 315–346. IGI Global.

De Castilho, R. E. and Gurevych, I. (2014). A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11.

de Oliveira, M. S. (2015). *On the use of visualization for supporting software reuse*. PhD thesis, Universidade Federal do Rio de Janeiro Rio de Janeiro, Brazil.

Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA.

Demner-Fushman, D., Shooshan, S. E., Rodriguez, L., Aronson, A. R., Lang, F., Rogers, W., Roberts, K., and Tonning, J. (2018). A dataset of 200 structured product labels annotated for adverse drug reactions. *Scientific data*, 5(1):1–8.

Dingsøyr, T. and Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and software technology*, 77:56–60.

Dodge, M., McDerby, M., and Turner, M. (2011). *Geographic visualization: concepts, tools and applications*. John Wiley & Sons.

Dunkerley, M. (2013). Information visualization: perception for design. *Waltham, MA: Morgan Kaufmann Publishers*.

Electronic Medicines Compendium (EMC) (2020). Home—electronic medicines compendium (emc). Accessed: 2020-10-14.

Evans, R. S. (2016). Electronic health records: then, now, and in the future. *Yearbook of medical informatics*, 25(S 01):S48–S61.

Ferrucci, D. and Lally, A. (2004). Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Fichman, R. G., Kohli, R., and Krishnan, R. (2011). Editorial overview—the role of information systems in healthcare: current research and future trends. *Information systems research*, 22(3):419–428.

Frakes, W. B. and Fox, C. J. (1996). Quality improvement using a software reuse failure modes model. *IEEE Transactions on Software Engineering*, 22(4):274–279.

Fung, K. W., Jao, C. S., and Demner-Fushman, D. (2013). Extracting drug indication information from structured product labels using natural language processing. *Journal of the American Medical Informatics Association*, 20(3):482–488.

GAM in Python (2018). Generalized additive models in python.

Garcia, R., Treude, C., and La, W. (2023). Towards understanding the open source interest in gender-related github projects. In *2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 68–79. IEEE.

Genism (2018). Genism: Topic modeling for humans.

GeoVis (2018). Geographic visualization of the number of github repositories by country.

Gerner, M., Sarafraz, F., M Bergman, C., and Nenadic, G. (2012). Biocontext: An integrated text mining system for large-scale extraction and contextualization of biomolecular events. *Bioinformatics (Oxford, England)*, 28:2154–61.

Gibson, E., Li, W., Sudre, C., Fidon, L., Shakir, D. I., Wang, G., Eaton-Rosen, Z., Gray, R., Doel, T., Hu, Y., et al. (2018). Niftynet: a deep-learning platform for medical imaging. *Computer methods and programs in biomedicine*, 158:113–122.

GitHub (2018). Github api v3: Github developer guide.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

GoogleCloudPlatform (2019). Kubernetes operator for managing the lifecycle of apache spark applications on kubernetes.

Gregor, S. and Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS quarterly*, pages 337–355.

Guaitoli, P. R., Jansma, E. P., de Vet, H. C., et al. (2014). Nutrition screening tools: does one size fit all? a systematic review of screening tools for the hospital setting. *Clinical nutrition*, 33(1):39–58.

Haak, D., Page, C.-E., and Deserno, T. M. (2016). Electronic data capture and dicom data management in multi-center clinical trials. In *Medical Imaging 2016: PACS and Imaging Informatics: Next Generation and Innovations*, volume 9789, pages 118–123. SPIE.

Haffari, G., Tran, T., and Carman, M. (2017). Efficient benchmarking of nlp apis using multi-armed bandits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 408–416.

Hansen, M. S. and Sørensen, T. S. (2013). Gadgetron: an open source framework for medical image reconstruction. *Magnetic resonance in medicine*, 69(6):1768–1776.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating nlp using linked data. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, pages 98–113. Springer.

Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. (2016). Serverless computation with openlambda. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, Denver, CO. USENIX Association.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2008). Design science in information systems research. *Management Information Systems Quarterly*, 28(1):6.

IBM (2021a). Watson natural language understanding. Accessed: 2021-02-27.

IBM (2021b). Watson natural language understanding api demo. Accessed: 2021-02-27.

IBM Watson NLU (2018). Ibm watson natural language understanding.

Ide, N., Suderman, K., and Kim, J.-D. (2018). Mining biomedical publications with the lapps grid. In *LREC*.

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., and Zhao, L. (2019). Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia tools and applications*, 78:15169–15211.

Jiang, M., Machiraju, R., and Thompson, D. (2005). Detection and visualization of vortices. the visualization handbook.

Kalliamvakou, E., Damian, D., Blincoe, K., Singer, L., and German, D. M. (2015). Open source-style collaborative development practices in commercial projects using github. In *2015 IEEE/ACM 37th IEEE international conference on software engineering*, volume 1, pages 574–585. IEEE.

Karasavvas, K., Baldock, R., and Burger, A. (2004). Bioinformatics integration and agent technology. *Journal of biomedical informatics*, 37(3):205–219.

Karopka, T., Schmuhl, H., and Demski, H. (2014). Free/libre open source software in health care: a review. *Healthcare informatics research*, 20(1):11–22.

Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., and Chouvarda, I. (2017). Machine learning and data mining methods in diabetes research. *Computational and structural biotechnology journal*, 15:104–116.

Kiah, M. L. M., Haiqi, A., Zaidan, B., and Zaidan, A. (2014). Open source emr software: Profiling, insights and hands-on analysis. *Computer methods and programs in biomedicine*, 117(2):360–382.

Koopman, B., Zuccon, G., Nguyen, A., Bergheim, A., and Grayson, N. (2015). Automatic icd-10 classification of cancers from free-text death certificates. *International journal of medical informatics*, 84(11):956–965.

Korolev, V. and Joshi, A. (2014). PROB: A tool for Tracking Provenance and Reproducibility of Big Data Experiments. In *Reproduce '14. HPCA 2014*.

Kreimeyer, K., Foster, M., Pandey, A., Arya, N., Halford, G., Jones, S. F., Forshee, R., Walderhaug, M., and Botsis, T. (2017). Natural language processing systems for capturing and standardizing unstructured clinical information: a systematic review. *Journal of biomedical informatics*, 73:14–29.

Kreuzthaler, M. and Schulz, S. (2015). Detection of sentence boundaries and abbreviations in clinical narratives. In *BMC medical informatics and decision making*, volume 15, pages 1–13. Springer.

Kuhn, M., Letunic, I., Jensen, L. J., and Bork, P. (2016). The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079.

Labropoulou, P., Galanis, D., Lempesis, A., Greenwood, M., Knoth, P., Eckart de Castilho, R., Sachtouris, S., Georgantopoulos, B., Anastasiou, L., Martziou, S., Katerina, G., Manola, N., and Piperidis, S. (2018). Openminted: A platform facilitating text mining of scholarly content. In *WOSP 2018 Workshop Proceedings*, Luxemburg.

European Language Resources Association (ELRA). Held as a workshop at the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Lamurias, A. and Couto, F. (2019). *Text Mining for Bioinformatics Using Biomedical Literature*, page 602–611.

Li, X., Ng, S.-K., and Wang, J. T. (2013). *Biological data mining and its applications in healthcare*, volume 8. World scientific.

Liao, X., Xiao, L., Yang, C., and Lu, Y. (2014). Milkyway-2 supercomputer: system and application. *Frontiers of Computer Science*, 8:345–356.

Ljubicic, V., Ketikidis, P. H., and Lazuras, L. (2020). Drivers of intentions to use healthcare information systems among health and care professionals. *Health Informatics Journal*, 26(1):56–71.

Louie, B., Mork, P., Martin-Sanchez, F., Halevy, A., and Tarczy-Hornoch, P. (2007). Data integration and genomic medicine. *Journal of biomedical informatics*, 40(1):5–16.

Luna, D., Almerares, A., Mayan, J. C., de Quirós, F. G. B., and Otero, C. (2014). Health informatics in developing countries: going beyond pilot practices to sustainable implementations: a review of the current challenges. *Healthcare informatics research*, 20(1):3–10.

Luo, J., Wu, M., Gopukumar, D., and Zhao, Y. (2016). Big data application in biomedical research and health care: A literature review. *Biomedical Informatics Insights*, 8:1.

Ly, T., Pamer, C., Dang, O., Brajovic, S., Haider, S., Botsis, T., Milward, D., Winter, A., Lu, S., and Ball, R. (2018). Evaluation of natural language processing (nlp) systems to annotate drug product labeling with meddra terminology. *Journal of biomedical informatics*, 83:73–86.

Mann, P. S. (2010). *Introductory statistics*. John Wiley & Sons.

Marafino, B. J., Boscardin, W. J., and Dudley, R. A. (2015). Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to icu risk stratification from nursing notes. *Journal of biomedical informatics*, 54:114–120.

Marien, S., Krug, B., and Spinewine, A. (2017). Electronic tools to support medication reconciliation: a systematic review. *Journal of the American Medical Informatics Association*, 24(1):227–240.

Martínez, P., Martínez, J. L., Segura-Bedmar, I., Moreno-Schneider, J., Luna, A., and Revert, R. (2016). Turning user generated health-related content into actionable knowledge through text analytics services. *Computers in Industry*, 78:43–56.

McColl, R. C., Ediger, D., Poovey, J., Campbell, D., and Bader, D. A. (2014). A performance evaluation of open source graph databases. In *Proceedings of the first workshop on Parallel programming for analytics applications*, pages 11–18.

McCormick, M., Liu, X., Jomier, J., Marion, C., and Ibanez, L. (2014). Itk: enabling reproducible research and open science. *Frontiers in neuroinformatics*, 8:13.

McDonald, C. J., Schadow, G., Barnes, M., Dexter, P., Overhage, J. M., Mamlin, B., and McCoy, J. M. (2003). Open source software in medical informatics—why, how and what. *International journal of medical informatics*, 69(2-3):175–184.

McIlroy, M. D., Buxton, J., Naur, P., and Randell, B. (1968). Mass-produced software components. In *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany*, pages 88–98.

McKiernan, E. C., Bourne, P. E., Brown, C. T., Buck, S., Kenall, A., Lin, J., McDougall, D., Nosek, B. A., Ram, K., Soderberg, C. K., et al. (2016). Point of view: How open science helps researchers succeed. *Elife*, 5:e16800.

MedDRA (2020). Meddrahierarchy. Accessed: 2020-10-16.

Mehrabi, S., Krishnan, A., Sohn, S., Roch, A. M., Schmidt, H., Kesterson, J., Beesley, C., Dexter, P., Schmidt, C. M., Liu, H., et al. (2015). Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.

Menger, V., Scheepers, F., and Spruit, M. (2018a). Comparing deep learning and classical machine learning approaches for predicting inpatient violence incidents from clinical text. *Applied Sciences*, 8(6):981.

Menger, V., Scheepers, F., van Wijk, L. M., and Spruit, M. (2018b). Deduce: A pattern matching method for automatic de-identification of dutch medical text. *Telematics and Informatics*, 35(4):727–736.

Menger, V., Spruit, M., Van Est, R., Nap, E., and Scheepers, F. (2019). Machine learning approach to inpatient violence risk assessment using routinely collected clinical notes in electronic health records. *JAMA network open*, 2(7):e196709–e196709.

Meulendijk, M. C., Spruit, M. R., Drenth-van Maanen, A. C., Numans, M. E., Brinkkemper, S., Jansen, P. A., and Knol, W. (2015a). Computerized decision support improves medication review effectiveness: an experiment evaluating the strip assistant's usability. *Drugs & aging*, 32:495–503.

Meulendijk, M. C., Spruit, M. R., Jansen, P. A., Numans, M. E., and Brinkkemper, S. (2015b). Stripa: A rule-based decision support system for medication reviews in primary care. In *ECIS*.

Meulendijk, M. C., Spruit, M. R., Willeboordse, F., Numans, M. E., Brinkkemper, S., Knol, W., Jansen, P. A., and Askari, M. (2016). Efficiency of clinical decision support systems improves with experience. *Journal of medical systems*, 40:1–7.

Meystre, S. and Haug, P. J. (2006). Natural language processing to extract medical problems from electronic clinical documents: performance evaluation. *Journal of biomedical informatics*, 39(6):589–599.

Midha, V. and Palvia, P. (2012). Factors affecting the success of open source software. *Journal of Systems and Software*, 85(4):895–905.

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

Mitchell, K. J., Becich, M. J., Berman, J. J., Chapman, W. W., Gilbertson, J., Gupta, D., Harrison, J., Legowski, E., and Crowley, R. S. (2004). Implementation and evaluation of a negation tagger in a pipeline-based system for information extraction from pathology reports. In *MEDINFO 2004*, pages 663–667. IOS Press.

Mozzicato, P. (2007). Standardised meddra queries: their role in signal detection. *Drug safety*, 30:617–619.

Muilu, J., Peltonen, L., and Litton, J.-E. (2007). The federated database–a basis for biobank-based post-genome studies, integrating phenome and genome data from 600 000 twin pairs in europe. *European Journal of Human Genetics*, 15(7):718–723.

Müller, H.-M., Auken, K. V., Li, Y., and Sternberg, P. W. (2018). Textpresso central: a customizable platform for searching, text mining, viewing, and curating biomedical literature. In *BMC Bioinformatics*.

Munger, M. A. (2010). Polypharmacy and combination therapy in the management of hypertension in elderly patients with co-morbid diabetes mellitus. *Drugs & aging*, 27:871–883.

Ngafeeson, M. N. (2015). Healthcare information systems opportunities and challenges. *Encyclopedia of Information Science and Technology, Third Edition*, pages 3387–3395.

Ni, Y., Kennebeck, S., Dexheimer, J. W., McAneney, C. M., Tang, H., Lingren, T., Li, Q., Zhai, H., and Solti, I. (2015). Automated clinical trial eligibility prescreening: increasing the efficiency of patient identification for clinical trials in the emergency department. *Journal of the American Medical Informatics Association*, 22(1):166–178.

Nolden, M., Zelzer, S., Seitel, A., Wald, D., Müller, M., Franz, A. M., Maleike, D., Fangerau, M., Baumhauer, M., Maier-Hein, L., et al. (2013). The medical imaging interaction toolkit: challenges and advances: 10 years of open-source development. *International journal of computer assisted radiology and surgery*, 8:607–620.

Octoverse (2018). The state of octoverse.

O'Leary, D. (1988). Expert system prototyping as a research tool. In *Applied expert systems*, pages 17–32. North-Holland Amsterdam.

Omta, W. A., van Heesbeen, R. G., Pagliero, R. J., van der Velden, L. M., Lelieveld, D., Nellen, M., Kramer, M., Yeong, M., Saeidi, A. M., Medema, R. H., et al. (2016). Hc stratominer: A web-based tool for the rapid analysis of high-content datasets. *Assay and drug development technologies*, 14(8):439–452.

Omta, W. A., van Heesbeen, R. G., **Shen, Z.**, Feelders, A. J., Brinkhuis, M., Egan, D. A., and **Spruit, M.** (2020a). Purifyr: An r package for highly automated, reproducible variable extraction and standardization. *Systems Medicine*, 3(1):1–7. `https://api.semanticscholar.org/CorpusID:215898718`.

Omta, W. A., van Heesbeen, R. G., **Z.Shen**, de Nobel, J., Robers, D., van der Velden, L. M., Medema, R. H., Siebes, A. P., Feelders, A. J., **S. Brinkkemper**, Klumperman, J. S., **M. Spruit**, Brinkhuis, M. J., and Egan, D. A. (2020b). Combining supervised and unsupervised machine learning methods for phenotypic functional genomics screening. *SLAS Discovery*, 25(6):655–664. `https://doi.org/10.1177/2472555220919345`.

Open Source Clinical Software (2018). Source codes of open source clinical software.

Osheroff, J. A., Teich, J. M., Levick, D., Saldana, L., Velasco, F., Sittig, D. F., Rogers, K. M., and Jenders, R. A. (2012). *Improving outcomes with clinical decision support: an implementer's guide*. CRC Press.

Pabinger, S., Dander, A., Fischer, M., Snajder, R., Sperk, M., Efremova, M., Krabichler, B., Speicher, M. R., Zschocke, J., and Trajanoski, Z. (2014). A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in bioinformatics*, 15(2):256–278.

Pachidi, S., Spruit, M., and Van De Weerd, I. (2014). Understanding users' behavior with software operation data mining. *Computers in Human Behavior*, 30:583–594.

Pagliari, C., Sloan, D., Gregor, P., Sullivan, F., Detmer, D., Kahan, J. P., Oortwijn, W., MacGillivray, S., et al. (2005). What is ehealth (4): a scoping exercise to map the field. *Journal of medical Internet research*, 7(1):e391.

Pandey, A., Kreimeyer, K., Foster, M., Dang, O., Ly, T., Wang, W., Forshee, R., and Botsis, T. (2019). Adverse event extraction from structured product labels using the event-based text-mining of health electronic records (ether) system. *Health informatics journal*, 25(4):1232–1243.

Passi1, K., Lane, L., Madria, S., Sakamuri, B. C., Mohania, M., and Bhowmick, S. (2002). A model for xml schema integration. In *E-Commerce and Web Technologies: Third International Conference, EC-Web 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings 3*, pages 193–202. Springer.

Patrick, J. and Li, M. (2010). High accuracy information extraction of medication information from clinical notes: 2009 i2b2 medication extraction challenge. *Journal of the American Medical Informatics Association*, 17(5):524–527.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Pestian, J. P., Matykiewicz, P., Linn-Gust, M., South, B., Uzuner, O., Wiebe, J., Cohen, K. B., Hurdle, J., and Brew, C. (2012). Sentiment analysis of suicide notes: A shared task. *Biomedical informatics insights*, 5:BII–S9042.

Pletscher-Frankild, S., Pallejà, A., Tsafou, K., Binder, J. X., and Jensen, L. J. (2015). Diseases: Text mining and data integration of disease–gene associations. *Methods*, 74:83 – 89. Text mining of biomedical literature.

plotly visualization (2018). Modern analytics apps for the enterprise.

Pop, D.-P. and Altar, A. (2014). Designing an mvc model for rapid web application development. *Procedia Engineering*, 69:1172–1179.

Przybyła, P., Shardlow, M., Aubin, S., Bossy, R., Eckart de Castilho, R., Piperidis, S., McNaught, J., and Ananiadou, S. (2016). Text mining resources for the life sciences. *Database*, 2016:baw145.

Pubmed Central (2019). Pubmed central (pmc).

Pubmed Open Access (2019). Pubmed open access subset.

Purkayastha, S., Allam, R., Maity, P., and Gichoya, J. W. (2019). Comparison of open-source electronic health record systems based on functional and user performance criteria. *Healthcare informatics research*, 25(2):89–98.

pyLDAvis (2018). pyldavis: Python library for interactive topic model visualization.

Raghupathi, W. and Raghupathi, V. (2014). Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2:1–10.

Rago, A., Ramos, F. M., Velez, J. I., Díaz Pace, J. A., and Marcos, C. (2016). Textract: a web-based tool for building nlp-enabled applications. In *Simposio Argentino de Ingeniería de Software (ASSE 2016)-JAIIO 45 (Tres de Febrero, 2016).*

Rajdho, A. and Biba, M. (2013). *Plugging Text Processing and Mining in a Cloud Computing Framework*, pages 369–390. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ramesh, V. and Ram, S. (1995). A methodology for interschema relationship identification in heterogeneous databases. In *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, volume 3, pages 263–272. IEEE.

Rehim, S. A., DeMoor, S., Olmsted, R., Dent, D. L., and Parker-Raley, J. (2017). Tools for assessment of communication skills of hospital action teams: a systematic review. *Journal of surgical education*, 74(2):341–351.

Rehurek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. lrec 2010 workshop on new challenges for nlp frameworks. *University of Malta, Msida, Malta*, pages 45–50.

Reynolds, C. J. and Wyatt, J. C. (2011). Open source, open standards, and health care information systems. *Journal of medical Internet research*, 13(1):e1521.

Rizzo, G. and Troncy, R. (2012). Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76.

Roberts, K., Demner-Fushman, D., and Tonning, J. M. (2017). Overview of the tac 2017 adverse reaction extraction from drug labels track. In *TAC*.

Russell, P. H., Johnson, R. L., Ananthan, S., Harnke, B., and Carlson, N. E. (2018). A large-scale analysis of bioinformatics code on github. *PloS one*, 13(10):e0205898.

Russo, E., Sittig, D. F., Murphy, D. R., and Singh, H. (2016). Challenges in patient safety improvement research in the era of electronic health records. In *Healthcare*, volume 4-4, pages 285–290. Elsevier.

Sallevelt, B., Huibers, L., Op Heij, J., Egberts, T., Puijenbroek, E., **Shen, Z.**, **Spruit, M.**, Jungo, K., Rodondi, N., Dalleur, O., Spinewine, A., Jennings, E., O'Mahony, D., Wilting, I., and Knol, W. (2021). Frequency and acceptance of clinical decision support system-generated stopp/start signals for hospitalised older patients with polypharmacy and multimorbidity. *Drugs & Aging*, 39. https://doi.org/10.1007/s40266-021-00904-z.

Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., and Chute, C. G. (2010). Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Schots, M. (2014). On the use of visualization for supporting software reuse. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 694–697.

Shekhani, R., Steinacher, L., Swen, J. J., and Ingelman-Sundberg, M. (2020). Evaluation of current regulation and guidelines of pharmacogenomic drug labels: opportunities for improvements. *Clinical Pharmacology & Therapeutics*, 107(5):1240–1255.

Shen, Z., Meulendijk, M., Knol, W., Huibers, L., Wilting, I., Jansen, P., and Spruit, M. (2016a). Stripa investigational medical device dossier (imdd). Technical report, UU/UMCU. https://marcospruit.nl/pub/2016%20-%20Shen%20et%20al%20-%20IMDD.pdf.

Shen, Z., Meulendijk, M., and Spruit, M. (2016b). A federated information architecture for multinational clinical trials: Stripa revisited. *24th European Conference on Information Systems (ECIS)*. https://aisel.aisnet.org/ecis2016_prototypes/2.

Shen., Z. and Spruit., M. (2019). Locate: A web application to link open-source clinical software with literature. In *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2019) - HEALTHINF*, pages 294–301. INSTICC, SciTePress. https://doi.org/10.5220/0007378702940301.

Shen, Z. and Spruit, M. (2019). A systematic review of open source clinical software on github for improving software reuse in smart healthcare. *Applied Sciences*, 9(1). https://www.mdpi.com/2076-3417/9/1/150.

Shen, Z. and Spruit, M. (2021). Automatic extraction of adverse drug reactions from summary of product characteristics. *Applied Sciences*, 11(6). https://www.mdpi.com/2076-3417/11/6/2663.

Shen, Z., van Krimpen, H., and Spruit, M. (2019a). A lightweight api-based approach for building flexible clinical nlp systems. *Journal of Healthcare Engineering*, 2019(1):3435609. https://onlinelibrary.wiley.com/doi/abs/10.1155/2019/3435609.

Shen, Z. and Wang, X. (2019). Source code.

Shen, Z., Wang, X., and Spruit, M. (2019b). Big data framework for scalable and efficient biomedical literature mining in the cloud. In *Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval*, NLPIR '19, page 80–86, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3342827.3342843.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee.

Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70.

Spruit, M. and Lytras, M. (2018). Applied data science in patient-centric healthcare: Adaptive analytic systems for empowering physicians and patients.

Stack Overflow Survey (2018). 2018 stack overflow developers survey.

Steinmacher, I., Silva, M. A. G., Gerosa, M. A., and Redmiles, D. F. (2015). A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59:67–85.

Steinman, M. A., Seth Landefeld, C., Rosenthal, G. E., Berthenthal, D., Sen, S., and Kaboli, P. J. (2006). Polypharmacy and prescribing quality in older people. *Journal of the American Geriatrics Society*, 54(10):1516–1523.

Stewart, K. (2016). Studies of success in open source software projects 1. In *Successful OSS Project Design and Implementation*, pages 131–148. Routledge.

Sutton, R. T., Pincock, D., Baumgart, D. C., Sadowski, D. C., Fedorak, R. N., and Kroeker, K. I. (2020). An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ digital medicine*, 3(1):17.

Tafti, A. P., Badger, J. C., LaRose, E. R., Shirzadi, E., Mahnke, A. N., Mayer, J., Ye, Z., Page, D., and Peissig, P. L. (2017). Adverse drug event discovery using biomedical literature: A big data neural network adventure. In *JMIR medical informatics*.

Tanaka, Y., Chen, H. Y., Belloni, P., Gisladottir, U., Kefeli, J., Patterson, J., Srinivasan, A., Zeitz, M., Sirdeshmukh, G., Berkowitz, J., et al. (2024). Onsides (on-label side effects resource) database: Extracting adverse drug events from drug labels using natural language processing models. *medRxiv*, pages 2024–03.

Towns, J., Cockerill, T., Dahan, M., Foster, I. T., Gaither, K. P., Grimshaw, A. S., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., Roskies, R., Scott, J. R., and Wilkins-Diehr, N. (2014). Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16:62–74.

Uzuner, Ö. (2009). Recognizing obesity and comorbidities in sparse data. *Journal of the American Medical Informatics Association*, 16(4):561–570.

Uzuner, Ö., Solti, I., and Cadag, E. (2010). Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518.

van de Weerd, I. and Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications*, pages 35–54. IGI Global.

van Tuijl, G., Leenen, W., **Z. Shen**, van de Weerd, I., and **S. Brinkkemper** (2011). Prioritizing requirements: An experiment to test the perceived reliability, usability and time consumption of bubblesort and the analytical hierarchy process. In *Proceedings of the International Requirements Engineering Efficiency Workshop (REEW 2011)*, pages 37–48. `https://api.semanticscholar.org/CorpusID:53900285`.

Wager, K. A., Lee, F. W., and Glaser, J. P. (2021). *Health care information systems: a practical approach for health care management*. John Wiley & Sons.

Wang, W., Bleakley, B., Ju, C., Kyi, V., Tan, P., Choi, H., Huang, X., Zhou, Y., Wood, J., Wang, D., et al. (2017). Aztec: A platform to render biomedical software findable, accessible, interoperable, and reusable. *arXiv preprint arXiv:1706.06087*.

Wikipedia (2018). Comparison of source code hosting facilities.

Wood, S. N. (2017). *Generalized additive models: an introduction with R*. chapman and hall/CRC.

Wright, R. M., Sloane, R., Pieper, C. F., Ruby-Scelsi, C., Twersky, J., Schmader, K. E., and Hanlon, J. T. (2009). Underuse of indicated medications among physically frail older us veterans at the time of hospital discharge: results of a cross-sectional analysis of data from the geriatric evaluation and management drug study. *The American journal of geriatric pharmacotherapy*, 7(5):271–280.

Wu, L., Ingle, T., Liu, Z., Zhao-Wong, A., Harris, S., Thakkar, S., Zhou, G., Yang, J., Xu, J., Mehta, D., et al. (2019). Study of serious adverse drug reactions using fda-approved drug labeling and meddra. *BMC bioinformatics*, 20:129–139.

Xing, Y., Wu, C., Yang, X., Wang, W., Zhu, E., and Yin, J. (2018). Parabtm: A parallel processing framework for biomedical text mining on supercomputers. In *Molecules*.

Yang, C., Yu, M., Hu, F., Jiang, Y., and Li, Y. (2017a). Utilizing cloud computing to address big geospatial data challenges. *Computers, Environment and Urban Systems*, 61:120 – 128. Geospatial Cloud Computing and Big Data.

Yang, C. P., Huang, Q., Li, Z., Liu, K., and Hu, F. (2017b). Big data and cloud computing: innovation opportunities and challenges. *Int. J. Digital Earth*, 10:13–53.

Yang, Y. C., Islam, S. U., Noor, A., Khan, S., Afsar, W., and Nazir, S. (2021). Influential usage of big data and artificial intelligence in healthcare. *Computational and Mathematical Methods in Medicine*, 2021.

Ye, Z., Tafti, A. P., He, K. Y., Wang, K., and He, M. M. (2016). Sparktext: Biomedical text mining on big data framework. *PloS one*, 11(9):e0162721.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65.

Zaimi, A., Ampatzoglou, A., Triantafyllidou, N., Chatzigeorgiou, A., Mavridis, A., Chaikalis, T., Deligiannis, I., Sfetsos, P., and Stamelos, I. (2015). An empirical study on the reuse of third-party libraries in open-source software development. In *Proceedings of the 7th Balkan Conference on Informatics Conference*, pages 1–8.

Zelkowitz, M. V. and Wallace, D. R. (1998). Experimental models for validating technology. *Computer*, 31(5):23–31.

Zettinig, O., Shah, A., Hennersperger, C., Eiber, M., Kroll, C., Kübler, H., Maurer, T., Milletarì, F., Rackerseder, J., Schulte zu Berge, C., et al. (2015). Multimodal image-guided prostate fusion biopsy based on automatic deformable registration. *International journal of computer assisted radiology and surgery*, 10:1997–2007.

Zhang, M. W. and Ho, R. (2017). Personalized reminiscence therapy m-health application for patients living with dementia: Innovating using open source code repository. *Technology and Health Care*, 25(1):153–156.

Zhang, Y., Qiu, M., Tsai, C., Hassan, M. M., and Alamri, A. (2017). Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95.

Zhao, H. and Ram, S. (2007). Combining schema and instance information for integrating heterogeneous data sources. *Data & Knowledge Engineering*, 61(2):281–303.

Ziegler, P. and Dittrich, K. R. (2004). Three decades of data intecration—all problems solved? In *Building the Information Society: IFIP 18th World Computer Congress Topical Sessions 22–27 August 2004 Toulouse, France*, pages 3–12. Springer.

# Summary

Healthcare Information Systems (HIS) play a pivotal role in modern today's healthcare, managing patient data, supporting clinical decisions, and enabling large-scale clincal trails. However, developing effective HIS faces numerous challenges, including structured and unstructed data in various format, massive data volume, regulatory compliance, and the increasing need of using AI for better decision making. Simultaneously, a large number of open source software or projects are created to address healthcare problems, with some well-known and successful ones like openEHR (open Electronic Health Records) and cTAKES (clinical Text Analysis and Knowledge Extraction System). These openly accessible OSS (Open Source Software) can benefit the HIS development, especially for circumstances where IT resources are restrained.

This dissertation investigates how artificial intelligence technologies and open source principles can accelerate HIS development to solve real-world clinical problems. The main research question is:

> **How can we employ artificial intelligence technologies – such as machine learning algorithms, knowledge systems and natural language processing techniques – based on open source principles to accelerate healthcare information system engineering in solving real-world clinical problems?**

We approached this research question through a practical, two-step methodology. The first part of the dissertation (chapter 2 and chapter 3) built a few healthcare information systems and prototypes to address specific clinical challenges, showing the real-world user cases of AI technologies in healthcare settings. Then, building on these practical experiences, we investigated how open source software and methodologies could accelerate HIS development. This progression from concrete system development to broader engineering methodology reflects our belief that effective HIS

engineering must be grounded in practical implementation while embracing open collaboration.

The research begins with a concrete clinical need – supporting polypharmacy reviews through rule-based clinical decision support system called the STRIP Assistant (STRIPA). To supoort its use in a multinational clinical trial, we design GDPR-compliant systems while maintaining data security and accessibility in chapter 2. STRIPA has a manual data entry process which is time consuming and error-prone. To address such issue, chapter 3 proposes a lightweight, API-based architecture for clinical NLP systems, which orchestrates various external NLP services to create flexible, cost-effective solutions that overcome the limitations of monolithic clinical NLP systems. We develop a prototype which demonstrates the potential of reducing manual data entry for HIS like STRIPA. The dissertation then continues on designing an automated NLP pipeline for extracting structured adverse drug reactions from European product labels in chapter 4. It shows how specialized NLP techniques can unlock structured knowledge from regulatory documents with high precision. chapter 5 presents a cloud-based big data framework for large-scale biomedical literature mining, providing a scalable solution for processing millions of research articles while maintaining cost-effectiveness. The need for such framework becomes apparent as the size of biomedical data grows rapidly.

The second part of the research focuses on exploring how open source software can enhance HIS engineering. A reproducible methodology is developed for systematic studies of open source clinical software in chapter 6, which provides insights into the landscape of available tools and their characteristics. Furthermore, we develop a web platform, named LOCATE, that bridges the gap between open source clinical software and scientific literature and enables developers to make more informed choices about software reuse. Except this first clinical decision support system, all the source codes of this research are publically accessible on GitHub with the aim of supporting open source software.

# Samenvatting

Healthcare Information Systems (HIS) spelen een cruciale rol in de moderne gezondheidszorg. Deze systemen beheren patiëntgegevens, ondersteunen klinische besluitvorming en faciliteren grootschalige klinische trials. De ontwikkeling van effectieve HIS wordt echter geconfronteerd met talrijke uitdagingen, waaronder de verwerking van gestructureerde en ongestructureerde data in verschillende formaten, enorme datavolumes, naleving van wet- en regelgeving, en de toenemende behoefte aan het gebruik van kunstmatige intelligentie voor betere besluitvorming. Tegelijkertijd wordt er een groot aantal open source projecten gecreëerd om gezondheidszorgproblemen aan te pakken. Prominente en succesvolle voorbeelden zijn openEHR (open Electronic Health Records) en cTAKES (clinical Text Analysis and Knowledge Extraction System). Deze publiek toegankelijke open source software (OSS) kan de ontwikkeling van HIS bevorderen, met name in omstandigheden waar IT-middelen beperkt zijn.

Dit proefschrift onderzoekt hoe kunstmatige intelligentie technologieën en open source principes de ontwikkeling van HIS kunnen versnellen om klinische problemen uit de praktijk op te lossen. De hoofdonderzoeksvraag luidt:

> **Hoe kunnen we kunstmatige intelligentie technologieën – zoals machine learning algoritmen, kennissystemen en natuurlijke taalverwerking technieken – op basis van open source principes inzetten om de ontwikkeling van healthcare information systems te versnellen bij het oplossen van klinische problemen in de dagelijkse praktijk?**

Deze onderzoeksvraag hebben we benaderd door middel van een praktische, tweestaps-methodologie. Het eerste deel van dit proefschrift (Hoofdstuk 2 en Hoofdstuk 3) ontwikkelde verschillende HIS en prototypes om specifieke klinische uitdagingen aan te pakken, waarmee praktische toepassingen van AI-technologieën in de

gezondheidszorg worden gedemonstreerd. Voortbouwend op deze praktische ervaringen hebben we onderzocht hoe open source software en methodologieën de ontwikkeling van HIS kunnen versnellen. Deze progressie van concrete systeemontwikkeling naar een bredere ontwikkelingsmethodologie weerspiegelt onze overtuiging dat effectieve HIS-ontwikkeling op praktische implementatie moet zijn gebaseerd en tegelijkertijd open samenwerking moet omarmen.

Ons onderzoek begint met een concrete klinische behoefte – het ondersteunen van polyfarmacie-reviews door middel van een regel-gebaseerd klinisch beslissingsondersteunend systeem genaamd de STRIP Assistant (STRIPA). Om het gebruik ervan in een multinationale klinische trial te ondersteunen, ontwerpen we in Hoofdstuk 2 GDPR-conforme systemen die tegelijkertijd databeveiliging en toegankelijkheid waarborgen. In eerste instantie had STRIPA een handmatig data-invoerproces dat tijdrovend en foutgevoelig is. Om dit probleem aan te pakken, introduceert Hoofdstuk 3 een lichtgewicht, API-gebaseerde architectuur voor klinische NLP-systemen. Dit is een architectuur die verschillende externe NLP-diensten orkestreert om flexibele, kosteneffectieve oplossingen te creëren die de beperkingen van monolithische klinische NLP-systemen overwinnen. We ontwikkelen een prototype dat het potentieel aantoont om handmatige data-invoer voor HIS zoals STRIPA te verminderen. Het proefschrift vervolgt met het ontwerpen van een geautomatiseerde NLP-pipeline voor het extraheren van gestructureerde bijwerkingen van geneesmiddelen uit Europese productlabels in Hoofdstuk 4. Dit laat zien hoe gespecialiseerde NLP-technieken gestructureerde kennis kunnen ontsluiten uit regelgevingsdocumenten met hoge precisie. Hoofdstuk 5 presenteert een cloud-gebaseerd big data framework voor grootschalige biomedische literatuur mining, dat een schaalbare oplossing biedt voor het verwerken van miljoenen onderzoeksartikelen met behoud van kosteneffectiviteit. De behoefte aan een dergelijk framework wordt duidelijk aangezien de omvang van biomedische publicaties nog altijd gestaag sneller groeit.

Het tweede deel van het onderzoek richt zich op het verkennen van hoe open source software de ontwikkeling van HIS kan verbeteren. In Hoofdstuk 6 wordt een reproduceerbare methodologie ontwikkeld voor systematische studies van open source klinische software, die inzicht biedt in het landschap van beschikbare tools en hun kenmerken. Verder ontwikkelen we een webplatform, genaamd LOCATE, dat de kloof overbrugt tussen open source klinische software en wetenschappelijke literatuur en ontwikkelaars in staat stelt beter geïnformeerde keuzes te maken over het hergebruik van software. Met uitzondering van het eerste klinische beslissingsondersteunende systeem, zijn alle broncodes van dit onderzoek publiekelijk toegankelijk

op GitHub met als doel aan open source software bij te dragen.

# Curriculum Vitae

Zhengru Shen was born in Anhui, China on the 23rd of July 1987. He completed his BSc in Information System Management at Anhui University in China (2006-2010). He moved to the Netherlands in 2010 to pursue his MSc in Business Informatics at Utrecht University, which he completed in February 2013. After working as an engineer in Shanghai, China for one year, he returned to the Netherlands in 2015 to start his PhD under the supervision of promotores prof.dr. Marco Spruit and prof.dr. Sjaak Brinkkemper at Utrecht University. His research focused on employing artificial intelligence technologies – including machine learning algorithms, knowledge systems, and natural language processing techniques – based on open source principles to accelerate healthcare information system engineering in solving real-world clinical problems. Upon completing his PhD research, he has been working as a data scientist at Odido since 2019.

# Acknowledgements

Ten years ago, when I decided to embark on this PhD journey, I knew it would be a long road ahead. But I never imagined in my wildest dreams that it would take this long—a full decade of my life. Throughout this journey, there have been countless ups and downs, moments of excitement and frustration, progress and setbacks that have taught me resilience and patience. Most importantly, however, I have learned to appreciate and be grateful for the people in my life. Without their support and patience, I would never have reached this point.

Marco, I am deeply grateful for your unwavering belief in me and for never giving up on me throughout this lengthy journey. Your persistent encouragement and support have been the driving force that has enabled me to reach this milestone. Thank you for pushing me forward when I needed it most, for your patience. Without your dedication and guidance, I would not be writing these words today. Sjaak, thank you for showing us how to conduct research and how to be a proper researcher. Your guidance at a strategic level has been invaluable in shaping the direction of this work.

I am grateful to my colleagues at UMCU for their collaboration and support. Bastiaan and Lianne, thank you for working together on our clinical trials. You helped me understand much more about the clinical domain and made our research collaboration both productive and enjoyable. I would also like to thank Wilma and Paul for their contributions and support during my time at UMCU.

I am also indebted to my colleagues at UU. Michiel, without you, my PhD project would not exist. Your research set the foundation for our EU project, and I am deeply grateful for your help, especially during the beginning of my PhD. Wienand, you were my daily supervisor during my master thesis, and I have continued to learn from you ever since. Thanks for your support and the interesting discussions we had. I am also grateful to my fellow colleagues at the ICS department Vincent, Noha, Shaheen, Siamak, Ingy for the encouraging chats, valuable discussions, and the pleasant

time we shared.

I would also like to thank my current colleagues at Odido. Even though you did not directly contribute to my PhD research, and I tried my best to avoid talking about my PhD at work, I really enjoy working with you and being surrounded by such a group of bright people. Thank you for being part of my long PhD journey and for celebrating this completion together.

To my dear friends who have supported me through this journey: Armel and Guru, our Bunnik days turned out to be a highlight of our PhDs, creating memories I will cherish for years to come. Honghong, Bilge and Baris, thank you for the wonderful chats and drinks, especially during COVID. Your friendship has meant so much to me. Last but certainly not least, I'd like to thank my friends Wang Xi and Xu Yudi for sharing dinners, drinks, talking about work and life, and helping with house moves. You have made the Netherlands feel more like home.

Ju, my dearest, thank you for your infinite patience and unwavering support throughout these ten years. You have been my rock through all the ups and downs, and your belief in me has made all the difference. To Grandma, Mom, Dad, and my sister—I would not be here without your unwavering support and love. Although we are separated by distance and cannot see each other as often as I would like, I have always felt your presence and encouragement. Your love has been a constant source of strength throughout this journey, and I am deeply grateful to have you in my life.