2017 年度博士後期課程（ソフトウェア情報学）論文

ヘルスケア予測システムにおけるソーシャルネットワーク解析とセマンティック Web オントロジーアラインメント

Social Network Analysis and Semantic Web Ontology Alignment for Healthcare Prediction System

岩手県立大学学院

ソフトウェア情報学研究科

２３６２０１５００３

Gajo Petrović

研究指導教員　　藤田　ハミド

佐々木　淳

布川　博士

要旨

本論文では高齢者のホームケアのためのシステムについて、オントロジーアライメント、ソーシャルネットワーク分析、GUI シナリオプログラミングの 3 つの研究分野に基づいて提案している。

ヘルスケアにおいて、2 つの大きな問題がある。

- 先進国ではホームケアの必要な高齢者がますます増えている。
- さまざまな互換性のない医療基準とシステムがあり、手作業による統合はコストがかかり、エラーを起こしやすい。

この 2 つの問題を解決するために、以下の解決策が示されている。

- ヘルスケアオントロジーとオントロジーアライメントによる自動での基準やシステムの統合。
- 個人差のあるホームケアを提供するためのセンチメント分析とユーザプロファイルの抽出
- シミュレートされたホームケア状況の迅速な開発のための GUI エディタ

これらの分野ごとに、いくつかの研究貢献がなされている。要約すると、

- 背景知識を使用しない、単語と文の Embedding によるオントロジーアライメントアプローチ。
- より良い結果を得るための、構造化された情報と半構造化された情報を使用したセンチメント分析。
- 不都合なく統合されたメタプログラミングをサポートする拡張可能な GUI エディタ

提案システムでは、ヘルスケアオントロジー、ソーシャルネットワークから抽出されたユーザプロファイル、およびシミュレートされたコマーシャルオフシェルフデバイス（COTS）に基づいた GUI エディタを使用して、ホームケア状況の作成を行う。

Abstract

A system for elderly homecare is proposed in this thesis, along with research contributions in three relevant areas: Ontology alignment, Social Network Analysis and GUI scenario programming.

There are two main issues in healthcare that have been addressed:

- Developed countries have increasingly high numbers of elderly people that require constant supervision and monitoring.
- There are many different, incompatible healthcare standards and systems, and manual integration is costly and error-prone.

To solve these two problems, the following solutions are presented:

- Automated integration of healthcare ontologies with Ontology Alignment
- Sentiment analysis and user profile extraction in order to provide personalized homecare
- GUI editor for rapid development of simulated homecare situations

For each of these areas, several research contributions have been made. To summarize, these are:

- Ontology Alignment approach with word and sentence embeddings, without the use of background knowledge.
- Sentiment Analysis using structured and semi-structured information to obtain better results.
- Extensible GUI editor with seamlessly integrated meta-programming support.

In the presented system, creation of homecare situations is done using the GUI editor, based on healthcare ontologies, user profiles extracted from Social Networks and simulated Commercial Off the Shelf devices (COTS).

Contents

Chapter 1. Introduction

Care for the elderly has always been an important topic in healthcare, and recently it has been gaining more attention due to large demographic shifts in developed countries. There is an increasing large elderly population in developed countries. In Japan, it is expected to exceed 43.3% by 2060 [1], with only 49.7% working age population. It is therefore important to provide a cost-effective healthcare solution with constant supervision and monitoring.

At the same time, there are many different healthcare data standards[11]. These standards are used to define, categorize and relate healthcare data for use in healthcare information systems (IS). Manual integration between healthcare IS is costly[1] and error-prone. This often leads to vendor lock-in, which provides less flexibility and can also lead to cost increase of its own, due to fewer choices. Automated integration is not a solved problem.

Figure 1. Elderly population increase

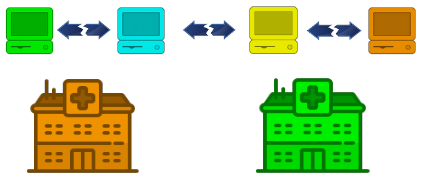


Figure 2. Different healthcare standards

In order to address these issues, a homecare system has been proposed in this paper. The goal of such a system is to reduce healthcare cost and provide care from home. It is based on structured (ontology) healthcare data with COTS (Commercial off the shelf) devices. Additionally, using user likes and dislikes it is possible to personalize it. An automated integration of systems (ontologies) has also been proposed, with the goal of reducing integration costs and providing more flexibility with regards to vendor choice. Such automated integration provides a way to automatically link equivalent concepts between two different systems.

To create such a homecare system, research has been done to develop and improve certain systems and methods.

In particular, the approach to automated integration was achieved with Ontology alignment. This was tested on Ontology alignment between biomedical ontologies (e.g. SNOMED CT, FMA, NCI), as well as Social Network ontologies: (FOAF (Friend of a Friend) ontology and custom sentiment ontologies).

1 Integration cost in the US: http://www.calgaryscientific.com/blog/bid/284224/Interoperability-CouldReduce-U-S-Healthcare-Costs-by-Thirty-Billion

Work in Social Network Analysis has been done in order to provide personalization, with research in sentiment analysis of unstructured (and semi-structured) data source Social Networks (e.g.Twitter data) as well as analysis and ranking of structured data sources (e.g. FOAF ontology network data).

Scenario creation was used in order to rapidly prototype and create homecare situations. Models were created using a GUI editor with simulation. Such simulation models can be described as triggers, consisting of events, conditions and actions. The editor is highly extensible with meta-programming support that is seamlessly integrated.

1.1. Homecare System

The concept of a homecare system is one of the more important elements of this thesis. The goal is to provide healthcare support and monitoring for elderly patients at their home. To provide this, it is necessary to rely on modern healthcare devices and intelligent approaches. In addition, it is necessary to provide personalized homecare and interaction with healthcare devices, to promote activity and participation willingness. Elderly users may also have issues interfacing with such devices so support for accessibility may be necessary. Such accessibility support systems, as well as additional intelligent systems can be developed to further enhance the homecare system.

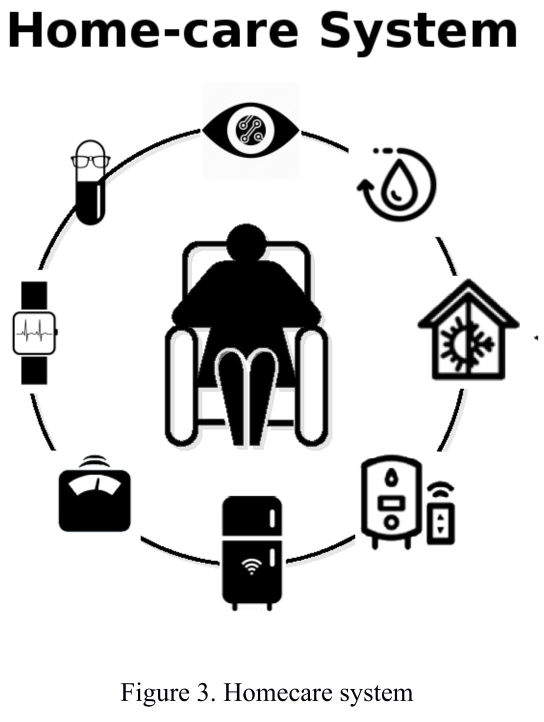


Figure 3. Homecare system

One such example of enhanced accessibility, in terms of color vision deficiency support using data driven intelligent methods is shown in the Deep Correct paper [13]. In case of a home-care system, it is expected that users would interface with it not just through voice commands and gestures, but also using screen displays (e.g. PC, TV, smartphone, etc.). There are multiple reasons for this, including cost, existing presence of smartphone/PC devices in many homes, user familiarity with such devices, and also the ability to display information that would otherwise be difficult to present using just voice.

Some users may have Color Vision Deficiency or other vision impairments, which should be handled using color correction methods. Most classical color correction methods are based on the physical model of the human brain, which is imperfect. In the Deep Correct paper, a method for providing CVD correction using a data driven approach is presented. In addition to being data-driven, the approach is also inherently useful for the specific solution (data problem set) so it can be customized for the screen display that is presented to the user.

It consists of two subsystems, as shown in figure 4:

- **Corrector.** Corrector performs the color correction, and is a Neural Network system that has the goal of producing enhanced images, as qualified by the referee. It is trained by the referee.

- **Referee:** evaluates the correction and improves the Corrector, by training it to produce enhanced images, which produce the best result when trying to solve a supervised task (often image classification, object detection, etc.). It is also often a Neural Network, that is trained separately (before), the referee.

The Corrector-Referee architecture is a general purpose architecture, and also potentially useful for other tasks (outside of CVD), as long as the task has some labeled data. As it enhanced the input data (images), it can be used for denoising purposes. The architecture is similar to GANs [32] (Generative Adversarial Networks), but the training of the Corrector and the Referee are done separately, providing a stable termination criteria and making it possible to train the corrector using a pretrained referee, which is getting exceedingly common especially in image processing.

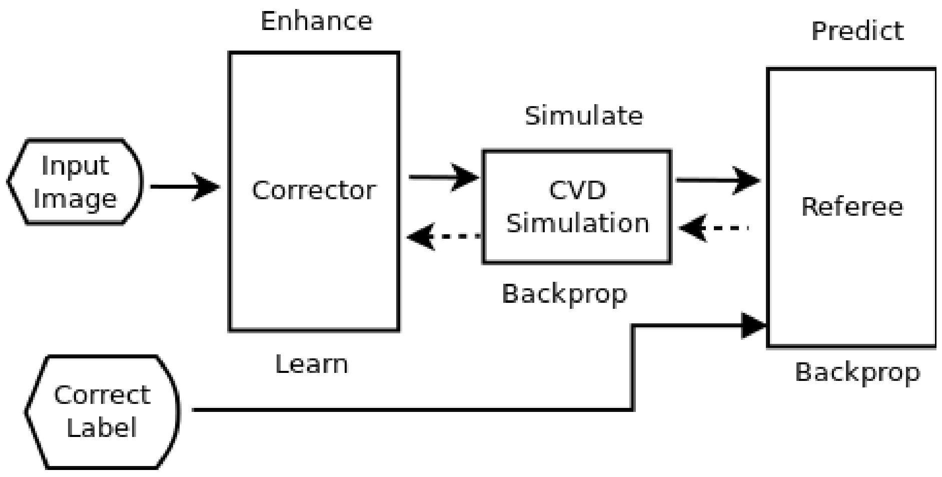


Figure 4. Corrector-Referee architecture.

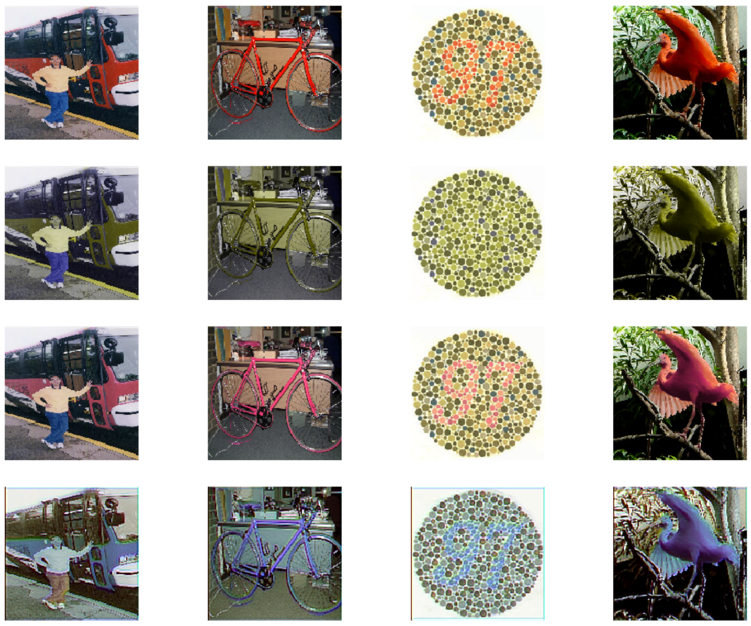


Figure 5. CVD correction results. From top to bottom: original image, CVD simulation, basic CVD color correction, Corrector-Referree CVD correction

Demonstration of using the proposed Deep-Learning based color correction is shown in figure 5. While not necessarily outperforming the model-based method, in the sense that produced images are often unnatural looking, it is possible to achieve results which are obviously usable for discerning objects.

CVD correction is just one of the possibilities of intelligent methods that are possible once a home system exists that can collect relevant user data. Depending on the user needs, and engineering effort, additional intelligent homecare systems can be developed.

1.2. Commercial off-the-shelf hardware

As described in paper [10]: in recent years we have seen rise of IoT (Internet of Things), IoP (Internet of People) and IioT (Internet of Industrial Things), which have led to many advances in the hardware area. To provide homecare support, it is necessary to rely on modern COTS (Commercial off-the-shelf) hardware, which is an ever increasing industry, especially in healthcare and fitness. Many new devices are developed by both big companies and small, sometimes Kickstartered startups. They provide support for cheap, reliable and modular healthcare systems, suited to the need of each patient.

The main goal of using COTS devices [10] instead of making proprietary health devices, is that COTS devices are already present in a large number of households, and the expertise of vendors for certain areas can and should be utilized. Further, competition in areas such are smart watches, fitness trackers, and smart home appliances, can lead to faster development, reduced cost and increased quality. In the COTS component of the proposed architecture, we have identified three types of devices that could serve as a source of health data: (i) wearable health devices, (ii) stationary health devices, and (iii) environment devices. Wearable health devices provide direct and continuous measuring of health indicators as they are worn by people for longer periods of time. They can represent a major source of data as they are in contact with a user more than any other type of device in this architecture component. Products from this category can be further roughly categorized into the following main subcategories: Smart wristbands and watches are usually used for activity tracking and heart rate monitoring (e.g. Fitbit Blaze[2]), but can be also used for blood pressure measurement (e.g. Omron Project Zero2[3]). Smart patches and skin attached sensors are used for more precise health monitoring as they need to be attached and sometimes even embedded in the user's skin. Main representatives of this subcategory are blood glucose measurement devices (e.g. The FreeStyle Libre Sensor[4]) and electrocardiogram bands and patches (e.g. Quardiocore[5]).

Smart clothing and accessories comprises smart apparel (e.g. Hexoskin [6]) and footwear (e.g. Zhor DigitSole [7]) that are either equipped with external sensors or have the sensors woven into the their fabric. Smart hearing meters and enhancers (e.g. ReSound [8]) and smart glasses and contact lenses measuring sight related characteristics (e.g. Right-Eye[9]) can be also classified into this subcategory.

Stationary health devices provide health measurements but in a discrete way as they are usually used daily or less often. Stationary devices like smart blood pressure monitors (Withings Wireless Blood Pressure Monitor[10]) are larger than their wearable alternatives, but they provide more accurate measurements and are usually more trusted by medical doctors. Sometimes, devices from this category provide data that cannot be measured with a wearable device. An example of such a measure is person's weight (e.g. Garmin IndexTM Smart Scale[11]). Environment devices provide measurements about the environment in which a user lives. Such information may be useful to medical staff in order to determine a cause of a problem or to detect patterns in person's behavior. If a person breaks a pattern marked critical by medical staff it may be considered an emergency and appropriate actions could be taken. Data collected from smart home appliances such as fridges and stoves may be used to determine person's diet habits and patterns. Smart air conditioning and heating systems could provide information about the temperature and humidity which may be of importance for the analysis of some health issues and

2 https://www.fitbit.com/
3 http://www.omron-healthcare.com/
4 http://www.freestylelibre.nl/
5 https://www.getqardio.com/
6 http://www.hexoskin.com/
7 http://zhor-tech.com/
8 http://www.resound.com/
9 http://www.righteye.com/
10 https://www.withings.com
11 http://www.garmin.com/en-GB

diseases. In order for the health monitoring system to be functional, the following issues must be resolved: sensor accurateness, device up-time, and integration between devices and the cloud. The first two issues are closely related to the hardware aspect of the COTS component. Depending on a chosen device, measuring quality may vary. In addition to the obvious solution to buy a more accurate and usually a more expensive device, this issue may be mitigated by deliberately introducing redundancy. Multiple multifunctional devices may be used as a source of the same health indicator. For example, heart rate could be monitored by both wearable device (chest patch, smart apparel, or smart wristband) and stationary blood pressure monitor. The integration issue is the toughest one to solve for the COTS component. There are no communication standards that all vendors can implement in their health devices. Currently, every vendor has its own web service or mobile application that is paired with the device and the two communicate by a protocol chosen by a vendor. The service or application is able to store, analyze, and present the collected data to the user often using proprietary protocols and security measures that prevent third party software from accessing it. One possible solution for the integration problem is the creation of adapters at the interactivity layer in the proposed architecture.

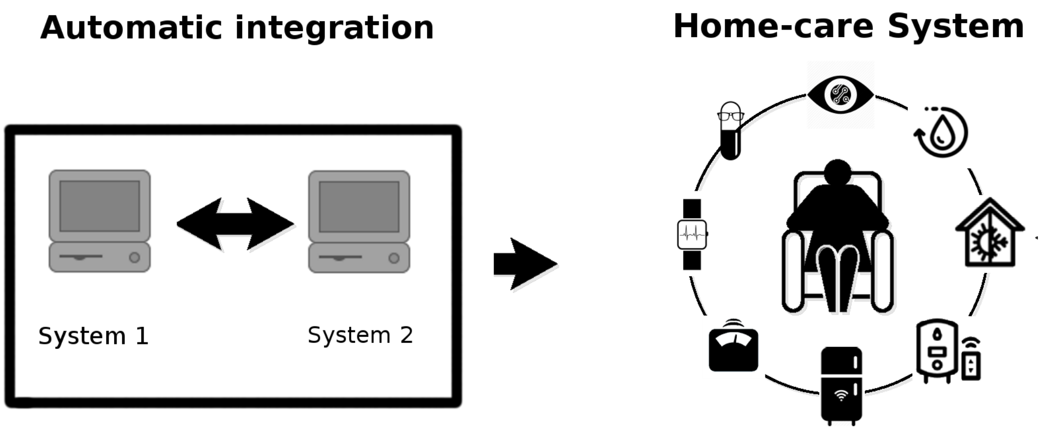


Figure 6. Automated integration and home-care system

The prerequisite for creating the homecare system is to solve the issue of integration between different healthcare systems, which would allow interoperability between the different COTS devices mentioned before. The problem of automated integration is approached by the means of ontology alignment. Systems are specified in different standards and the goal is to provide an alignment between those standards. This is described in more detail in Chapter 2.

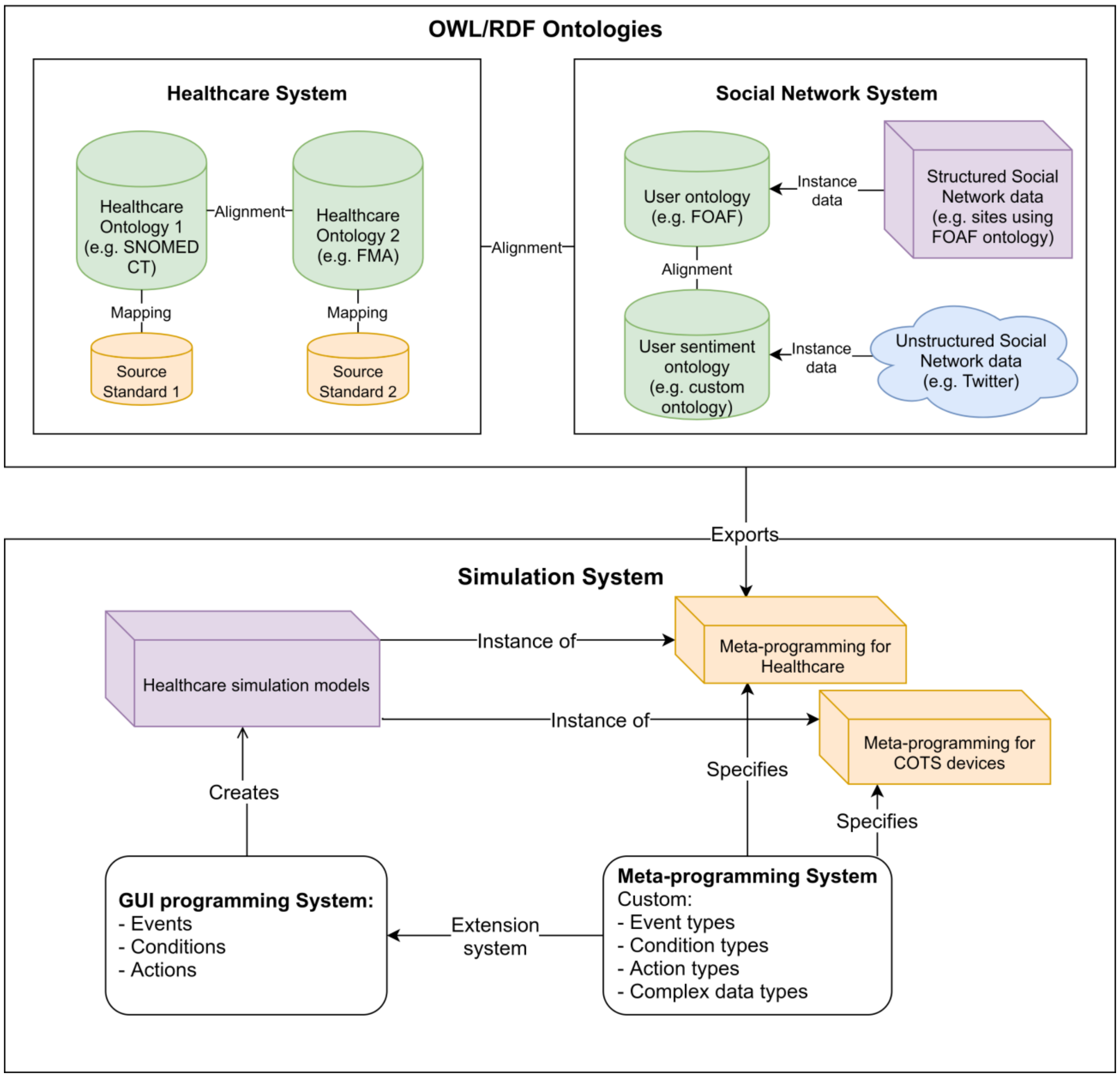


Figure 7. Homecare system model

The detailed overview of the entire proposed system is given in Figure 7. It consist of three main elements:

1) Ontology Alignment between healthcare systems, which is presented in detail in Chapter 2.

2) Social Network Analysis with sentiment analysis and structured graph analysis, presented in Chapter 3.

3) Simulation and scenario creation, using the GUI editor with Meta-programming support, defined in Chapter 4.

The goal of this system is to provide integration between different healthcare standards, and a platform for rapid homecare situation creation based on those healthcare models, with support for personalization using user profile extracted from social networks.

Chapter 2. Healthcare system integration: Ontology alignment

Healthcare systems are software systems, often used by a health providing institution, such as a hospital, in order to model and store real world healthcare data and provide an interface which uses such data for decision making and data storage. They are implemented with a model of the healthcare knowledge which models the healthcare domain and with software that provides an interface on that data. Knowledge in healthcare systems is stored using healthcare data formats, represented using healthcare standards, There are many different healthcare data formats and standards, and while there are some popular ones, it seems unlikely that there will only be single, universally used data standard anytime in the near future. In case of incompatibility, it is necessary to manually provide data integration, which can be costly, error prone and time consuming. The high cost[12], in turn leads to vendor lock-in, which reduces flexibility and decreases incentive to use the best tool for each purpose, but rather those tools which are developed by established healthcare marker share leaders.

The goal is therefore to increase interoperability between healthcare systems. At the core of interoperability is compatibility between knowledge standards, which is split into two types of issues:

- Syntax compatibility. Syntax compatibility refers to compatibility of data formats, due to the file format in which they are stored. There are many different types of data formats, such as XSD[13], RDF/XML[14], RDF/N3[15], HL7 2.x[16], HL7 CDA[17], , etc.

- Semantic compatibility. Semantic compatibility refers to compatibility between the healthcare knowledge models themselves, on the level of their abstract models. There are many different knowledge models, usually one per healthcare standard, and in a way, different version of the same standard which model entities differently can also considered semantically incompatible.

In many cases, systems are neither syntactically nor semantically compatible. In this case, the first step is to convert them to the same syntax, and then perform the semantic mapping, which tends to be harder. Knowledge formats which exhibit syntax compatibility but not semantic compatibility also occur often, as they are the first step towards data format compatibility. In case when the formats are semantically

12 http://www.calgaryscientific.com/blog/bid/284224/Interoperability-CouldReduce-U-S-Healthcare-Costs-by-Thirty-Billion
13 https://www.w3.org/XML/Schema
14 https://www.w3.org/TR/rdf-syntax-grammar/
15 https://www.w3.org/TeamSubmission/n3/
16 https://www.hl7.org/implement/standards/product_brief.cfm?product_id=214
17 http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7

compatible, but not syntactically, it usually refers to different file formats of the same system. Knowledge formats which are both syntactically and semantically compatible represent equivalent systems.

Generally, it is easier to automate the process of converting from one syntax to another, provided the data format is clearly described. This is usually the least error-prone part of the system, but it can lead to data-loss if one of the formats is less expressive. Because of its simplicity, RDF is often used as the format of choice when representing systems. That said, even RDF can be represented in different file format, such as RDF/XML, RDF/N3, RDF/Turtle[18], etc.

Healthcare knowledge formats represented using RDF and other elements of the Semantic Web are called healthcare ontologies. Ontologies are standardized representation formats, which provide a general solution to the syntax compatibility issue. Often as the first step of ontology integration, systems will be converted to the RDF format. Semantic Web also provides a standardized API for data querying, using the SPARQL API standard[19], which further enhances interoperability when using such formats.

2.1. Semantic Web

Classic Web has been in widespread use for a considerable while, with many different services available on it. It is also called Web of Documents as it defines how documents are presented to the user, in a standardized way. That said, the Classic Web doesn’t hold any information about the semantics of the content that is displayed. This has lead to the development of its extension, the Semantic Web or Web of Data, which would allow storing data in a Web-like manner, under well-defined standards. The goal of this is to automate various interactions on the network, and to provide better interoperability and standardization of various domains.

One of the basic concepts of the Semantic Web is the ontology. Ontologies are formal representations of domain knowledge, as a set of concepts and relationships between those concepts. Ontology is defined as an explicit specification of a conceptualization.

Semantic Web can also be viewed as an approach with a relevant set of technologies which were created in order to add semantics to the Classic Web, in order to extend the web with ontologies which add meaning to the resources and their links. Common technologies include: Hypertext Web technologies (URL/URI, Unicode, XML), data representation (RDF), knowledge representation and formalization (RDFS, OWL) and query technologies (SPARQL).

18 https://www.w3.org/TR/turtle/
19 https://www.w3.org/TR/sparql11-query/

2.1.1 Resource Description Framework

Resource Description Framework (RDF) is a group of W3C specifications for data modeling with the goal of describing metadata, and is a general purpose technology for describing Web resources and their links. The basic element in RDF is the RDF triplet, which is a basic statement that contains three elements: subject, verb and object and is written in SVO order (subject-verb-object).

There are three basic concepts that Semantic Web relies on: 1) basic resource identifier with URIs, 2) triplet data model as defined in RDF, and 3) ontology as base for knowledge representation and reasoning.

Fundamental concepts of RDF are resources, properties and statements:

- Resources define models of real life entities and are uniquely identified with a URI.
- Properties are relationship models between resources and are also identified with URIs. Properties can only define binary relationships (relationships between exactly two resources).
- Statements represent resource property assignment. Such assigned values can be either atomic types or other resources.

RDF has the open world assumption, which means that while we known the truthfulness of the expressed statements, there might still be statements which are true, but we lack such information. This can lead to intractability of certain reasoning. Additionally, RDF doesn't have an ability to express negative expressions, that is, there is no way to express untruthful expressions.

RDF alone is not enough to represent ontologies, as it is inherently domain independent. It is therefore necessary to use other technologies such as RDFS or OWL.

2.1.2 RDF Schema

RDF Schema (RDFS)[20] can be used to define domain models on RDF. It defines classes of entities and properties on those entities. Individual entities (objects) are treated as class instances.

It is possible to define restrictions on class properties: by defining limits which kind of values can be assigned to properties, and therefore limiting the domain of the properties. In this way it is possible to define relationships between classes, similar to how it is done in UML (Unified Modeling Language).

RDFS also allows the ability to inherit properties, which can be used both on classes and on properties. Individuals of the child class can be used in any place where the parent class can exist. For properties, the domain of the child class is a subdomain of the parent class.

20 https://www.w3.org/TR/rdf-schema/

2.1.3 Ontology Web Language

Ontology Web Language (OWL)[21] is a group of W3C standards which defines multiple dialects of the Semantic Web, which further solves the limitations of RDFS. RDFS is not capable of defining: restrictions on values of a specific class, disjoint classes, cardinality bounds, etc. While OWL is more expressive than RDFS, not all dialects of OWL necessarily include all of RDFS.

There are three main OWL dialects, which represent different expressibility of OWL, and are aptly named OWL Lite, OWL DL and OWL Full. More expressive dialects inherently include less expressive ones (OWL DL includes OWL Lite, and OWL Full includes OWL DL).

The main reason for the existence of dialects such as OWL Lite and OWL DL is that they include a smaller subset of OWL, which makes them faster and simpler, and queries on such OWLs are less likely to be intractable.

OWL can be represented using various syntax formats, such as XML, N3, N4, and so on. As OWL often relies on RDF technologies, it is common to refer to such ontologies as OWL/RDF, as the syntax itself is less important.

2.2. Healthcare ontologies

As described in [11]: healthcare ontologies are electronic representation of knowledge, that provide a standard for sharing, categorizing and relating knowledge, represented in a formal way. They may also additional provide support for logical reasoning depending on the way they are defined.

Many standards have been created to allow interchange of data and integration of healthcare ISs, usually focusing on low level protocols and predefined message formats. In the era of Internet, high connectivity and openness introduced an opportunity for a different kind of integration approach. Such an approach may utilize semantic-based technologies to represent and communicate knowledge between the healthcare ISs. Ontologies are often used as a way of representing such knowledge. Two main reasons for using them are their ability to capture healthcare knowledge in a formal way and an easy application of reasoning processes that is performed by a medical decision support system. Resource Description Framework (RDF)[22] in conjunction with Web Ontology Language (OWL)[23] can be considered as a de facto standard when it comes to semantic web and linked data technologies, and represents a foundation for defining healthcare ontologies. Despite the popularity of OWL/RDF format, systems are often centered around traditional eXtensible Markup Language (XML) technology and relational databases, partly because of good validation tools and support from major manufacturers. Majority of traditionally

21 https://www.w3.org/TR/owl-features/
22 https://www.w3.org/RDF/
23 https://www.w3.org/2001/sw/wiki/OWL

used data representation languages offer some sort of input validation like parsing grammars and metamodels for domain specific languages, XML Schema (XSD) for XML, and Data Definition Language for Structured Query Language (SQL). These properties of traditional systems can be preserved while using the OWL/RDF technology as an additional layer of integration[33], in order to obtain a semantically rich representation of underlying knowledge.

Despite the benefit of a single standard, there are domains that have multiple competing ontologies. Some of them were developed at different times, by different vendors or interest groups (sometimes due to specific requirements), or were not public in the early versions. Examples of these situations can be seen in healthcare (FMA[24], SNOMED CT[25], NCI[26], CEN/ISO EN13606[27], openEHR[28]), agriculture (NALT[29], AGROVOC[30]), and many more.

Health Level Seven (HL7). HL7 is a set of healthcare standards for the exchange, integration, sharing, and retrieval of electronic healthcare data. It is developed by Health Level Seven International (HL7), which is a non-profit organization with a goal to develop healthcare standards for interoperability of healthcare information systems. The latest version of the standard at the time of this writing is the HL7 Version 3, centered around HL7 Reference Information Model (RIM)[31], which is an object model representing HL7 clinical data. HL7 also specifies a data standard, called HL7 V3 Data Types (DT), which is using in extensively throughout the suite.

openEHR is an open-standard set of specifications in healthcare informatics with the aim to standardize management, storage, retrieval, and exchange of health data in EHRs. It follows the dual-model approach [34] that differentiates between two levels: (i) information level represented by a reference model specifying statements that are applied to all entities of the same class and (ii) knowledge level which is represented through archetypes that are statements about specific entities. The openEHR standard provides both functional and semantic interoperability, allowing for it to be read and processed by both humans and machines respectively.

The Electronic Health Record Communication (EN13606) is a European norm from the European Committee for Standardization (CEN) that specifies the normative for exchanging patient records between EHR systems. Although a stand-alone standard, it can be viewed as a subset of openEHR. Both EN13606 and openEHR follow the dual-model approach, however slightly different archetypes were defined. Transformations between archetypes were developed in [22] allowing future ontology-based

24 Foundational Model of Anatomy: https://bioportal.bioontology.org/ontologies/FMA

25Systematized Nomenclature of Pathology Clinical Terms: https://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

26National Cancer Institute Thesaurus: https://bioportal.bioontology.org/ontologies/NCIT

27 http://www.en13606.org/

28 http://www.openehr.org/

29National Agricultural Library's Agricultural Thesaurus: https://agclass.nal.usda.gov/

30 AGROVOC Multilingual Agricultural Thesaurus: http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural-thesaurus

31http://www.hl7.org/implement/standards/rim.cfm integration approaches to consider the two standards in the following ways: (i) separately, where the ontology of each of the solutions should be developed or used if already exists, and (ii) together, using one ontology and transforming the EHRs described in one standard to the other using the predefined archetype transformations. openEHR OWL ontology covering reference model, data types, and data structures of the openEHR was developed by Roman[32]. Also, authors of [35] have developed an OWL ontology of the archetype library by following the guidelines from official openEHR specification. Therefore, first two ontologies can be viewed as a single, complete, ontology that can be used in the integration approaches.

An ontology for both EN13606 and openEHR were developed in [36] as a part of an attempt to provide semantic interoperability between the standards. However, more complete ontology for EN13606 was developed by the authors of [37] while trying to develop an architecture comprising of different EHR systems capable of inter-operating so as to offer an integrated service using interoperability patterns based on EN13606 and the semantic technologies such as OWL. A partial OWL ontology for the definition and validation of archetypes was also developed in [38].

The Clinical Element Model (CEM)

The goal behind the development of the Clinical Element Model (CEM) [39] was to provide a single, referent, architecture for representing information in EHRs. CEM comprises two models: Abstract instance model for representing individual instances of collected data, and Abstract constraint model for representing constraints on the data instances. These two models are abstract specifications and can be implemented using different programming languages. The main purpose of such an abstract implementation is to provide a way to normalize different data from EHRs.

Originally, CEM was implemented using the Clinical Element Modeling Language (CEML) [39] which has a XML-like syntax. Additional implementation was in Constraint Definition Language (CDL) [40] that extends CEML to allow specification of new constraints to the modeling language. In order for CEM to be integrated using OWL/RDF technologies, the CEM-OWL ontology is developed by the authors of [41]. They have also developed an automatic transformation from CEM-XML specification to the corresponding CEM-OWL specification. This will allow future researchers to focus more on the process of integration than on data acquisition.

In addition to the previously described ontologies, that are based on widely used standards, several other healthcare ontologies were developed. These ontologies usually focus on some specific parts of EHR, but it could be beneficial, in the context of globally transferable healthcare data, to integrate systems that are using these ontologies.

Open Biomedical Ontologies (OBO) [42] is a set of ontologies developed and maintained by the scientific community with a goal to allow easier representation and integration of biomedical data. To clarify the terminology, the biomedical domain is broader than just the healthcare domain as it comprises

32 http://trajano.us.es/~isabel/EHR/6

other knowledge not only specific to patient medical care and EHRs. Disease Ontology (DO) [43] is a part of OBO repository and can be used to describe patient disease history in EHRs. A benefit of using this ontology is a fact that it is heavily referencing SNOMED and other medical thesauri. Another ontology form the OBO repository is the Gene Ontology (GO) [44] that provides structured, controlled vocabularies and classifications used in the annotation of genes, gene products, and sequences.

The Foundational Model of Anatomy Ontology (FMA) is the representation of classes or types and relationships necessary for the symbolic representation of the phenotypic structure of the human body. FMA is a domain ontology that represents a coherent body of explicit declarative knowledge about human anatomy but can be also applied and extended to all other species.

Due to the lack of a common vocabulary, healthcare ontologies often reference terms from various existing vocabularies. Vocabulary standards are used to describe clinical problems, terms, categories, procedures, medications, and allergies. Various medical vocabulary standards exist and in order to implement a usable healthcare standard interoperability, these vocabularies must be also taken into the consideration. Medical Subject Headings (MeSH) [45] consists of sets of terms naming descriptors in a hierarchical structure that permits searching at various levels of specificity. It is not an ontology per se, but it is referenced from [33] a vast majority of ontologies as to provide classification and categorization of the biomedical terms. Therefore, it should be considered in all integration approaches as it provides structure and hierarchical information about the medical categories. MeSH is often used in conjunction with RxNorm [46], a pharmaceutical vocabulary used for e-prescribing, medication history, government reporting, and drug compendium mapping, and Logical Observation Identifiers Names and Codes (LOINC) [47], a database and universal standard for identifying medical laboratory observations. NCIt[48] is a widely recognized standard for biomedical coding and reference, used by a broad variety of public and private partners both in the U.S. and internationally.

The Unified Medical Language System (UMLS) [49] aims to alleviate the problem that exists when using multiple vocabularies in a healthcare informatics system. UMLS comprises several controlled vocabularies in the biomedical sciences including SNOMED-CT, ICD, RxNorm, etc. It provides a mapping structure among these vocabularies and it may also be viewed as a comprehensive thesaurus and ontology of biomedical concepts. Therefore, it is often referred as the UMLS meta-thesaurus. Although it provides a mapping structure, it does not make semantically integrated terminology interoperable. However, it provides enough information about term relations to be used in an integration process.

Additionally, UMLS provides facilities for natural language processing.

Attempts have been made to solve this problem, by creating a new or extending an existing ontology, that would then be all encompassing and provide a shared conceptualization satisfactory to all interest groups. However, this has not shown to be a complete solution, and would often result in there now being one extra ontology system, at least for a certain amount of time. Even if the new ontology is objectively

33 http://sig.biostr.washington.edu/projects/fm/AboutFM.html

better, it takes time (and effort) to convert the existing data to a new standard, as well as to upgrade the IT system.

2.3. Healthcare ontology alignment

The main issues with manual integration are the high cost and error-proneness. In addition, it is necessary to do the same process essentially from scratch for every pair of healthcare ontologies. Also, currently accurate ontology alignment has not yet been achieved. In this thesis an automated approach to ontology alignment has been proposed. It uses word and sentence embedding to achieve highly accurate and fast ontology alignment.

As described previously, to achieve healthcare compatibility, it is necessary to achieve two things: syntax compatibility and semantic compatibility. There are many different types of file formats, so it would be rather cumbersome to support any combination of them, which would result in n * (n-1) types of format combinations. It is therefore ideal to first transfer all data formats to only one file format, and then use that format for semantic mapping. Due to its simplicity and flexibility, OWL/RDF, or just RDF is often chosen for this.

OWL/RDF based integration approaches are not a novel idea proposed in thesis, and are evidenced both in many research papers published in previous years and many projects and movements such as the Yosemite manifesto[34], SemanticHealthNet[35], and Clinical Information Modeling Initiative[36].

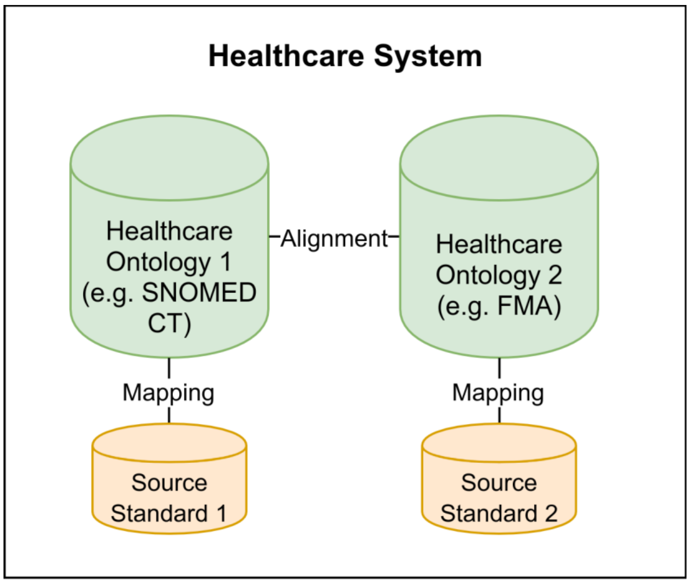


Figure 8. Healthcare system integration using ontologies as an intermediate format

34 http://yosemitemanifesto.org/
35 http://www.semantichealthnet.eu/
36 https://www.amia.org/the-standards-standard/2012-volume3-edition1/clinical-information-modeling-initiative

The Yosemite initiative suggests a two-step approach to healthcare ontology integration: (1) transforming any ontology format to OWL/RDF and (2) creating an integration algorithm for two OWL/RDF ontologies. Once the ontologies are transformed to OWL/RDF representation, in order to implement the second step, one must create an ontology alignment algorithm. This entire process is shown in figure 8. Although this is the hardest and the most work-intensive part, it is always easier to perform integration and create more general solutions for a single representation technology than to create transformations on by-representation basis. Therefore the first step of the approach is the prerequisite to have such an universal technology, and due to that fact it is a subject of numerous discussions and criticism. The main issue concerns the choice of OWL/RDF for the universal representation technology for both ontologies and alignments/mapping between ontology concepts. There are several reasons of why the OWL/RDF is a suitable technology [50]:

– It is possible to map any other representation to RDF. RDF is made up of atomic statements (triplets of subject, object, and predicate). As triplets are atomic pieces of information, all other more complex information can be implemented by a set of triplets. Sometimes, this may lead to more verbose representations.

– RDF captures information, not syntax. Therefore, many different syntaxes (XML, Json, etc.) may be used to provide serialization for triplets. Therefore, usual storage mechanisms may be used for RDF-based solutions.

– RDF is self describing as it uses Uniform Resource Identifiers (URIs) as main identifiers. This reduces ambiguity and allows the creation of term definitions to be referenced by any other documents. This reduces ambiguity and allows single points of knowledge.

– OWL/RDF enables inference that derives new assertions from existing ones. This can lead to more automation of data translation processes.

2.4. Ontology alignment methods

There are a number of existing systems and methodologies for ontology alignment:

– AML[15]

– FCA-Map [2]

– Lily [3]

– LogMap [4]

– CroMatcher [5], [6]

The main techniques that are common in most top systems include basic string and word matching (Lexical Matcher), usually by hashing words and matching them exactly. Matching candidates are found by searching for entities that have the same words or strings. There are multiple variants of the basic lexical matcher, based on how strictly they match words:

String matcher. String matcher matches only identical words. There are various implementations, but a common idea is to calculate the ratio of the number of words matched, and a number of words not matched, as expressed in this formula:

$$\frac{\left(len\left(w_i, w_i \in A \land w_i \in B\right)\right)}{\left(len\left(u_i, u_i \in A, u_i \notin B\right) + len\left(u_i, u_i \notin A, u_i \in B\right)\right)}$$

Edit distance. Edit distance (or Levenshtein distance), is the approach where string similarity can be partial, depending on how many characters are matched. Edit distance calculates the number of necessary basic operations (insertion, deletion, replacement) performed on characters, in order to transform one string to another. For example, the edit distance between *cat* and *bat* is 1, and the distance between *dog* and *frog* is 2. If we want to transform distance into similarity in the [0, 1] range, we can divide this number by the total number of characters of the two words, as shown below:

$$Similarity = 1 - \frac{editDistance}{totalWords}$$

The formula can then be modified to include a sum of all similarity measures, as seen below:

$$\sum \left(Similarity\left(A_i, B_i\right), for\ A_i \in A\right), where\ B_i = argmin_{Bi}\left(Similarity\left(Ai, Bi\right)\right)$$

It’s important to note the difference between String matching and Edit-distance matching. While generally simple, and sometimes capable of giving better results, Edit-distance matching can be rather inefficient when matching similar elements between documents. Because string matching is based on identical matching, it is possible to create an associative array (hash-map) for each of the matching ontologies. When searching for potential matches, it is possible to lookup entities which contain the same strings, so it is not necessary to check every entity, largely speeding up the matching process. While this speedup might seem like a minor implementation detail, due to the speed complexity of comparing all entities $O(n^2)$, it is far more efficient to use a hash-based approach to obtain the initial list of candidates. Checking all entity combinations can be so inefficient for large ontologies, that it becomes impractical. Most state of the art approaches use a similar fast, hash-based method to create the candidate list.

All of the previously mentioned methods have variations that determine with what importance words are matched. Varying these weights can often lead to better performance on some datasets, although there is no universal solution that is guaranteed to produce better results.

A common approach is weight words differently based on how important they are. For this, it can be sensible to calculate the word TF-IDF (term frequency, inverse document frequency), which is a metric commonly used to obtain document keywords. We can then alter the above formula to compute weighted sum of the word, where each word is weighted by its TF-IDF value. Other variants of this can also be used.

Some words like stop-words can be ignored completely, as they tend to hold no information. Having matching stop-words tends to not increase similarity between concepts. Besides ignoring stopwords, other common NLP preprocessing techniques can also be implied, which includes stemming (reducing the word to its most common form), lemming (reducing the word to its most common, grammatically correct and semantically equivalent form), and similar.

This provides a fast and efficient way to match two large ontologies. Such a method tends to have high precision, but low recall as they cannot produce results in non-exact matches.

In addition, many of these methods use some kind of background knowledge, such as medical knowledge bases. Selection (via ranking) of possible matches is a helpful addition that can be used to filter only the most likely matches. Some systems additionally use simple techniques for structural matching, although they tend to be too time inefficient to use when matching large ontologies.

2.5. Proposed methodology

The most impactful module in the previous approaches is the Lexical Matcher, which provides the majority of the alignments while being highly efficient. That said, the issue with this approach is that it only works with direct string matching. Even if we used string matching to generate initial candidates, and more sophisticated methods to filter those candidates to a smaller list, they would completely miss words and concepts which are semantically similar but written differently.

To solve the first problem, we can do word comparisons based not just on their characters, but also on their meaning. Techniques such as Word2Vec[51] make it possible to transform words into latent variable vectors, where semantically similar words are positioned nearby in the latent space. Once transferred into the vector space, it is possible to calculate concept similarity using common vector distance measures, such as Euclidean distance or cosine similarity.

The main idea behind Word2Vec is that words which appear together share similar meaning, and words which can be used instead of other words, are often semantically related. These ideas are older than the Word2Vec paper itself, and were first introduced in linguistic.

Word2Vec calculates word vectors based on their distributional properties, using a large text corpus, and words that are located in nearby proximity (as defined by distance measures such as Euclidean or cosine distance) in the vector space are said to be semantically similar.

As Word2Vec only has vectors for words that exist in the dictionary, it can sometimes be necessary to first extend the dictionary, learning new domain-specific words, especially if Word2Vec was initially trained on a general-purpose corpus such as Wikipedia or News corpuses.

Glove[53] and fasttext[52] are methods that follow the same idea of Word2Vec and further extend it. Fasttext in particular extends it to support character-based vector generation.

What comes as a natural extension, we can also design systems which generate latent vectors based on the entire sentence. Therefore, in order to improve this particular element, an approach to ontology alignment with sentence embedding is proposed. The system (Ontoline) creates a sentence vector for each sentence, which is then used to efficiently obtain similar sentences.

Sentences are encoded into vectors using modern NLP (Natural Language Processing) techniques, mainly relying on word embedding systems (Word2Vec, Glove, fastText) to first obtain word vectors, and then converting a sequence of word vectors to a sentence vector. The word vector systems use embeddings (such as Word2Vec, Glove or fastText) pretrained on concept-agnostic corpuses such as WikiPedia.

This approach provides an accurate matching (as it uses not just string similarity, but also semantic similarity), while still remaining highly efficient.

The main approaches to creating sentence embeddings based on word embeddings are as follows:

– Simple techniques (average, weighted average, product) sentence embedding

– LSTM/Deep Learning based sentence embeddings (Skip-thought vectors[54], InferSent[55], etc.).

To index these highly dimensional sentence vectors, it is possible to use locality sensitive hashing and provide a fast mapping system. Maintaining efficient matching is possible if we transfer concepts into vectors, and then search for similar concepts in the vector space, using LSH[56] (locality sensitive hashing).

Once we have generated a list of candidates using sentence similarity search based on LSH, we normally do two things:

1. Decide on the best candidate using more accurate, but time-consuming approaches.

2. Rank the elements based on similarity factor and remove duplicate matchings.

Rank based selection is done similarly to previous work. It is important to be careful when trimming duplicates, as some type of duplicates (e.g. one-to-many) can actually occur, although many-to-many seems less frequent. In these cases it might be useful to provide such options to the user, making this a semi-automated approach.

2.6. Dataset

To evaluate the validity of this approach, popular datasets have been used. In particular, Ontology Alignment Evaluation Initiative[37] organized ontology matching competitions yearly from 2004, with the aim to evaluate and improve ontology alignment systems and evaluation techniques.

There are multiple different types of tasks (as of 2018):

The matching tasks relevant to healthcare include:

1) LargeBio matching task, where the goal is to match between three healthcare ontologies: Foundational Model of Anatomy (FMA), SNOMED CT and National Cancer Institute Thesaurus (NCI). It uses UMLS Metathesaurus for reference alignments.

2) Anatomy matching task: a much simpler task where the goal is to match similar elements in the Adult Mouse Anatomy and the NCI Thesaurus (human anatomy)

Task 1. has six subtasks, divided into two categories (subset matching tasks and full ontology matching tasks). In case of subset matching, datasets tend to be much smaller and only contain relevant elements. The larger, full-ontology matching tasks tend to disqualify a lot of algorithms which exceed the challenge-specified computation time, and they demonstrate the importance of efficient lexical matches. Task sizes are given below http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2017/ :

	Small (Shared classes)	Full
NCI-FMA		
NCI	**3696**	**66724**
FMA	**6488**	**78989**
NCI-SNOMED		
NCI	**51128**	**66724**
SNOMED	**23958**	**122464**
FMA-SNOMED		
FMA	**10157**	**78989**
SNOMED	**13412**	**122464**

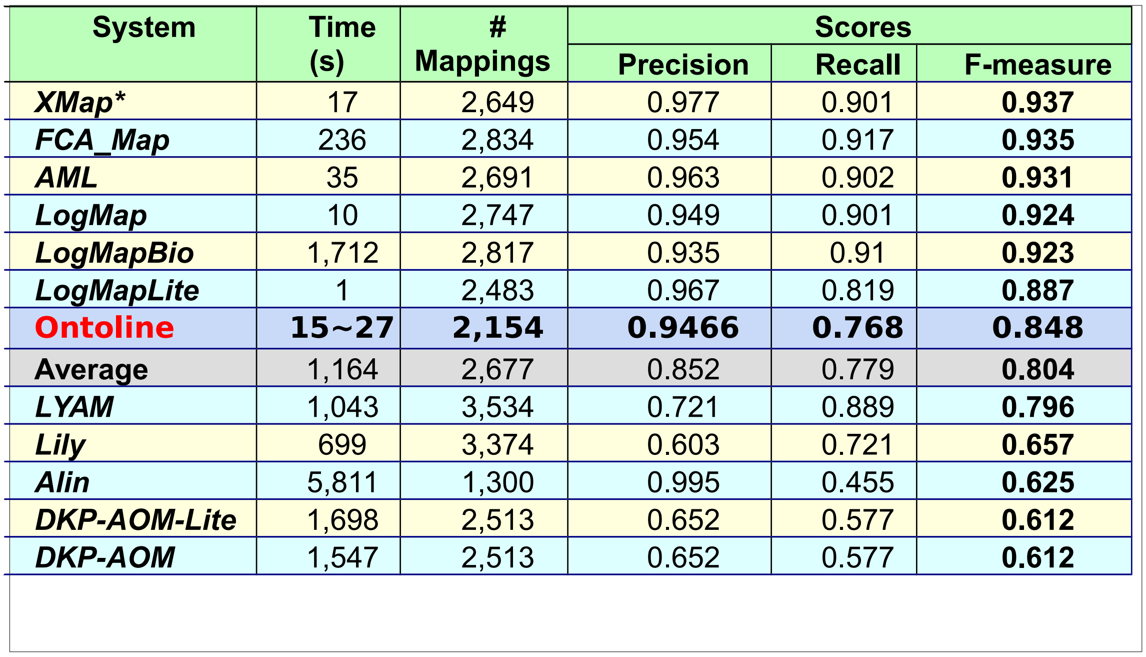
Table 1. LargeBio matching subtask class sizes

The proposed method was mainly tested on the NCI-FMA subtask, although as it doesn't rely on background knowledge, its ability to work with different types of ontologies is theoretically possible, but requires more exact testing.

37http://oaei.ontologymatching.org/

2.7. Results

A basic algorithm has been developed that beats baseline Lexical Matching approaches. Speed is comparable to (and often beats) state of the art systems. Accuracy is better than the average state of the art system, but still currently worse than some top ones.



System	Time (s)	# Mappings	Scores		
			Precision	Recall	F-measure
*XMap**	17	2,649	0.977	0.901	**0.937**
FCA_Map	236	2,834	0.954	0.917	**0.935**
AML	35	2,691	0.963	0.902	**0.931**
LogMap	10	2,747	0.949	0.901	**0.924**
LogMapBio	1,712	2,817	0.935	0.91	**0.923**
LogMapLite	1	2,483	0.967	0.819	**0.887**
Ontoline	**15~27**	**2,154**	**0.9466**	**0.768**	**0.848**
Average	1,164	2,677	0.852	0.779	**0.804**
LYAM	1,043	3,534	0.721	0.889	**0.796**
Lily	699	3,374	0.603	0.721	**0.657**
Alin	5,811	1,300	0.995	0.455	**0.625**
DKP-AOM-Lite	1,698	2,513	0.652	0.577	**0.612**
DKP-AOM	1,547	2,513	0.652	0.577	**0.612**

Table 2. Results (LargeBIO: FMA-NCI small overlapping task). Ontoline (red) is the proposed system.

Despite the lower F-measure, an important benefit of the proposed approach is that it doesn’t rely on domain specific background knowledge.

Additionally, ontology alignment between different domains (e.g. healthcare and social network ontologies) is also something worth considering, despite generally being a task where there are much fewer similar concepts.

Chapter 3. Social Network Analysis

As based on papers [7, 8, 12, 14]: in recent years we have seen a rise in the use of Social Networks such as Facebook, Twitter and many others. More and more users are posting their opinions and feelings on a daily basis about a wide variety of issues, and those posts are propagated throughout the Web. Often such data is publicly available, and it has seen use in a number of areas, such as: product marketing, political analysis, election prediction, and so on. Data analysis can be done on a per-message level, but when it's performed for users as a whole it results in user profile extraction. In either case, the goal of such analysis is to determine what people think about a certain topic, which can be represented as a positive or negative polarity in the most basic model. Often, it is also important to extract the exact topic of the post, but this issue (topic extraction) is usually considered a separate research topic.

In many cases, users post their opinions and sentiments in an unstructured format, without information such as numerical ratings or opinions picked from a predefined set (structured data). Instead, users would post text written in a natural language with some additional information such as URLs, emoticons, hashtags and similar. This makes the sentiment analysis and opinion mining an NLP problem, relying on methods such as POS tagging and chunking, coreference resolution, named entity recognition, sentence breaking, relationship extraction, word and topic segmentation, stemming, and so on. In the end, this results in a process that has low accuracy, especially since the messages written in Social Networks often contain non-standard words, abbreviations, acronyms, incorrect grammar and are extended by new symbols and special spellings. Also, these posts would often rely on information located on other sites (in case where URLs are included) as well as contain irony and sarcasm that even humans sometimes have difficulty identifying properly.

In most cases, the content that needs to be processed is written almost exclusively in a natural language and is often completely unstructured. The algorithms that are capable of analyzing such input belong to the family of NLP (Natural Language Processing) algorithms, and the problems that are tackled with them, while usually solvable by human experts (still not trivial) tend to be very hard for machines. Because of this, the accuracy of such methods tends to either be very low, or the problem is reduced to a slightly simplified version, often by only looking at well written text, with correct grammar and very little or no ambiguity, irony or sarcasm. Thankfully, many of these social networks also contain some semi-structured information present in the format of the messages, which can be of tremendous help for sentiment analysis. Example of such semi-structured data can be seen in emoticons (e.g. "John :feelinghappy:") which are chosen from a limited, predefined set, as well as sentiment hashtags[16] which are popular terms used that contain a certain sentiment towards a specific topic (e.g. “#NoNetNeutrality”). The only drawback of the sentiment hashtags is that these aren’t predefined in the platform, and in order to use them, they must first be identified. The identification is most often done manually, and in [12] paper a method was proposed for a semi-automatic detection of sentiment hashtags, which should help jumpstart sentiment analysis research by automating yet another part of it.

Cold start is an issue that’s also commonly seen in data mining and it’s also present in the case of user profile extraction. The cold start issue arises when attempts are made to calculate user profiles for new users that still don’t have enough information due to their lack of activity. Examples of this in Social Networks would be users that still haven’t posted much or interacted with many people, making it hard to discern their sentiments from their posts alone. This was also a motivation in [8] paper, in which it was accomplished using additional information such as follower relationships, replies and retweets in order to supplement the lack of information from the posts alone. In addition to posting messages in Social Networks, there are certain interactions between users that can be observed. People can associate with one another, mention other users in messages, as well as share and reply to each other. It is often the case that these interactions are performed by people who have similar interests, and this is the assumption we're basing this paper on. In fact, this behavior is called homophily and has been commonly observed in Social Networks. We combined post-level sentiment analysis with user relationships to calculate user profiles with a higher degree of accuracy. We calculated the initial user profiles based on their posts and we then improve that by using the user’s relationships. The assumption of homophily leads to a formula that recalculates user profiles while including not just their posts, but also the user sentiments of other users they interact with. This process is then done repeatedly until a given set of iterations is completed.

3.1. Related Work

Sentiment analysis has been a popular research topic for a while now. It has seen use in the political domain, such as the case of the prediction of German election results and emerging topic tracking[16]

It has also seen use in health care, as a way to track patient feedback[17].

In certain Social Networks, due to either limitations of the system, or popularity, there are trends to use uncommon abbreviations, symbols and grammar that industry standard NLP tools cannot handle well. This has lead to the development of Social Network specific NLP tools, such as the Twitter NLP[18], [19].

As far as topic extraction goes, LDA[20] is a well known model, and there have been some attempts in using it for Twitter such as in [21]. For the purpose of obtaining and searching Twitter data, various software architectures have been developed such as in [22].

The continued research into sentiment analysis has lead to the development of publicly available tools, which include SenticNet[23], ConceptNet[24] and similar. Still, despite the existence of these tools, sentiment analysis presents an open problem. Sarcasm detection proves to be a difficult challenge and an open problem, and there have been many attempts to tackle it like the authors did in [25]. Cold start is also usually considered a separate research domain on its own, and there have been many approaches in order to solve it, most often in the domain of recommender systems [26][27]. Regarding graph information, there have even been a number of attempts using social relationships to calculate user sentiment profiles such as in [28], however, unlike in this paper, user sentiments were calculated by relying on biographical information rather than tweets, and the way social networking relationships are used is also different.

3.2. Methodology

To compute tweet sentiments we have relied on Cambria's SenticNet [23]. SenticNet uses a bag of concepts model[29] instead of the simple word or term frequency (bag of words model). Concepts can include word pairs such as “net neutrality” which in the bag of words model would be split into two words “net” and “neutrality” for which the sentiment calculation would be done separately, and that would results in a loss of its original meaning. Sentiments are represented by the Hourglass model[30], which includes four attributes: sensitivity, aptitude, attention and pleasantness. From that, polarity can be computed by the following equation (quoted from [30]):

$$p=\sum_{i=1}^{N}\frac{Pleasantness(c_i)+|Attention(c_i)|-|Sensitivity(c_i)|+Aptitude(c_i)}{3N}\quad(1)$$

where is an input concept, *N* the total number of concepts, and *3N* the normalization factor (as the Hourglass dimensions are defined as float in [-1,+1]). We have used the tools available on the SenticNet 2 website in order to extract concepts and retrieve their sentiments using the ontology provided. In addition to the bag of concepts model described above, we have used sentiment hashtags. When sentiment hashtags were detected in a tweet, we would add a 0.5 polarity in the case of positive sentiments, -0.5 in the case of negative, and 0 if both a positive and negative sentiment hashtags were contained (mixed sentiments). These numbers were chosen arbitrary, and it is likely that better values could be obtained using a supervised learning algorithm on a classified dataset, but we decided it was not in the scope of this paper.

Next, we use the sentiments extracted from each tweet to calculate the aggregated user profile. For each topic that the user has expressed sentiments towards in a tweet, the aggregated user sentiment is calculated as an average, which can be seen in formula

$$AggregatedSentiment(u,p)=\sum_{i=1}^{N}\frac{Sentiment(t_i)}{N}(2)$$

SenticNet website download section: http://sentic.net/downloads/ where *u* is the user, *p* is the topic, t_i is a tweet made by user *u* on topic *p*, and *N* is the total number of tweets by user *u* on topic *p*. Since we also want to have different values for user sentiments depending on the amount of tweets they were generated from, the resulting user sentiment profile for each topic is a tuple, where the weight is represented by the number of tweets (*N*).

We also calculate the connectivity users have with each other. The assumption is that two users are connected if they have a relationship between them, or if one of them has mentioned the other in a post. The most common relationship is the follower relationship, which is non-symmetric and represents that one user follows another. The user that's doing the following is called a follower, and the user being followed is called a friend. Other things that imply there's a connection between users are retweets and replies. We calculate the strength of the connection between two users as defined in formula 2, where connectivity from user *i* to user *j* is calculated as the weighted sum of four relationships: follower, friendship, retweet and reply. The parameters *α, β, γ* and *δ* have in this case been manually chosen as 0.25, 0.25, 0.25 and 0.25.

$$Connectivity(i,j)=\frac{\alpha\cdot fo(i,j)+\delta\cdot fr(i,j)+\beta\cdot rt(i,j)+\gamma\cdot\Re(i,j)}{\alpha+\beta+\gamma+\delta}(3)$$

These parameters could be automatically calculated if there was a manually classified dataset. One such dataset could also be created by the users themselves, where they would rank other users by what they conceive is the strength of their relationship.

The novel approach of this paper is in using the sentiments of other, related users to recalculate user sentiments. We do this by including the sentiments of connected users, and combining it with the user's sentiment profile. The resulting sentiment profile is based on the collective sentiment of user tweets, as well as the weighted average of each of the connected users. It is calculated as shown in formulas 4 and 5.

$$US_{related}(u_1,p)=\frac{\sum_{u2\in Users}N_{u2,p}\cdot Connectivity(u_1,u_2)\cdot US(t_i,p)}{\sum_{u2\in Users}N_{u2,p}\cdot Connectivity(u_1,u_2)}(4)$$

$$US_{new}(u_1,p)=\frac{N_{u2,p}\cdot X\cdot US_{old}(u_1,p)+US_{related}(u_1,p)\cdot\sum_{u2\in Users}N_{u2,p}\cdot Y}{(X+Y)}(5)$$

where *US* stands for user sentiment, defines how many tweets user *u* posted on topic *p* and *X* and *Y* are constants that define how much of an impact should the related users have in comparison to the old sentiment. *X* and *Y* are chosen to be 0.75 and 0.25 respectively. This motivation behind this approach is based on the idea of homophily[31], that suggests that users on social networks have a tendency to form groups with users of similar interest, that is, that users of similar interests tend to associate with each other.

From this, we assume that the correct user sentiment profile depends on other connected users proportionally to the strength of their connection. Thus, in order to calculate the user profile and user connectivity, we apply iterative the approach given as follows (shown in figure 9):

1. Calculate initial user sentiment profiles based on (1) and (2)
2. Repeat for a number of iterations: Calculate new user sentiment profiles based on (4) and (5)

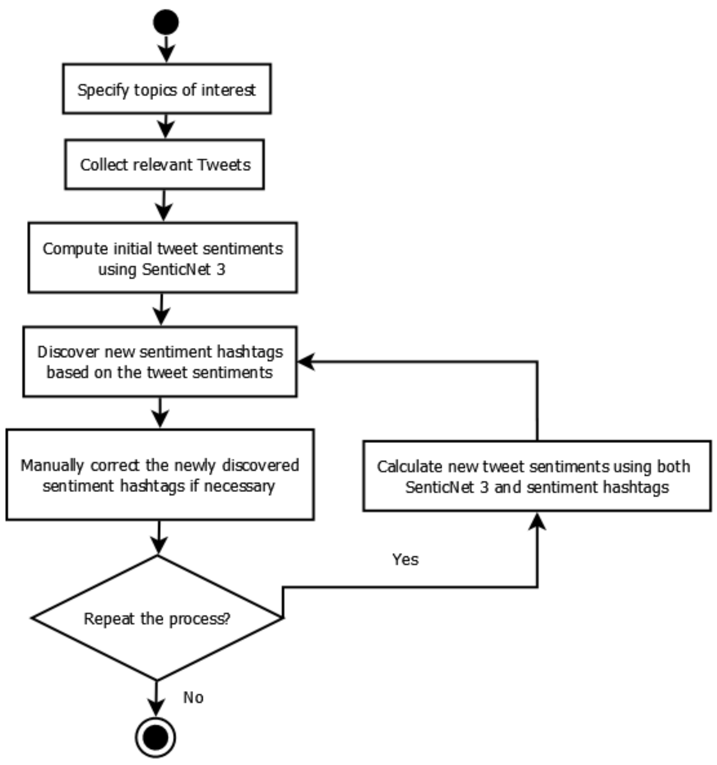

Specify topics of interest
Collect relevant Tweets
Compute initial tweet sentiments using SenticNet 3
Discover new sentiment hashtags based on the tweet sentiments
Manually correct the newly discovered sentiment hashtags if necessary
Calculate new tweet sentiments using both SenticNet 3 and sentiment hashtags
Yes
Repeat the process?
No

Figure 9. Iterative discovery of sentiment hashtags and their use in tweet sentiment calculation

3.3. Social Network module

Personalized homecare is important in order to achieve better user satisfaction and to motivation interaction with a homecare system. Responding to user likes and dislikes is likely to engage them better. It is therefore necessary to extract user profiles, which can be done with Social Network Analysis. In particular there are two main works done in this thesis in order to achieve this kind of user profile extraction (as shown in figure 10):

- Sentiment analysis using unstructured data (e.g. Twitter)[8], [12]
- Graph analysis – synonym detection on structured data[7] (e.g. FOAF - Friend of a Friend ontology)

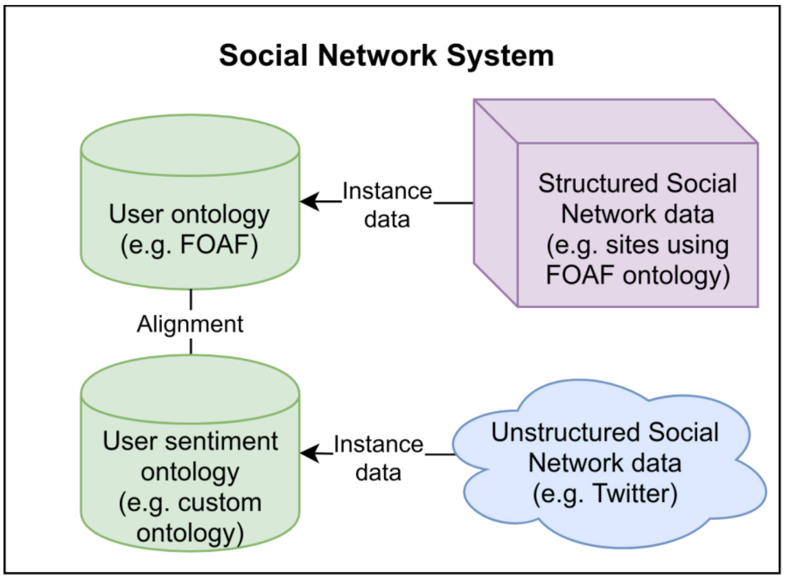

Social Network System
User ontology (e.g. FOAF)
Instance data
Structured Social Network data (e.g. sites using FOAF ontology)
Alignment
User sentiment ontology (e.g. custom ontology)
Instance data
Unstructured Social Network data (e.g. Twitter)

Figure 10. Social Network Analysis for personalized homecare.

Sentiment analysis usually relies on unstructured Social Network data analysis, where the goal is to extract user sentiments towards specific topics. Data sources contain messages written in a natural language, and therefore rely on Natural Language Processing techniques.

Synonym detection was done [7] as part of structured Social Network data analysis, and it was verified on user ranking with PageRank and HITS algorithms. It was performed on FOAF ontology structured data, where the goal was to detect multiple user online identities. The motivation behind this lies in the assumption that combining identities can help obtain better results in other graph-related tasks, such as the previously mentioned user ranking algorithms. This was performed on a Semantic Web (FOAF) graph with user profiles, where users were nodes and knows relationships were links, as shown in figure 11.

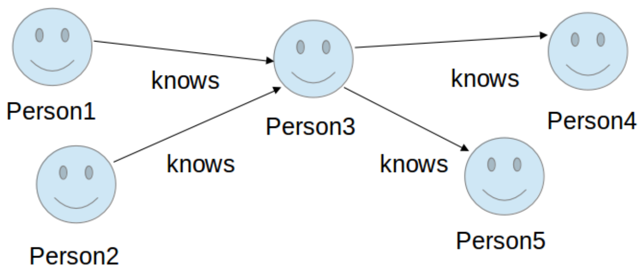


Figure 11. Example FOAF Social Network graph.

FOAF is an OWL/RDF ontology for social networks, which models people and relationships between them. Generally there are two ways synonyms can occur:

- Manual creation of accounts on multiple sites. This can occur when people register on multiple sites, and there is no direct connection between the two profiles. It is the more difficult version we're trying to detect, and this kind of detection is based on attribute similarity (e.g. *foaf:familyName, foaf:mbox_sha1sum, foaf:phone, foaf:birthday foaf:homepage*) of user profiles.

- Local definitions (in FOAF files) of people located in remote documents. This is a property of OWL/RDFS, and trivial to detect by just reading the value of the *rdfs:seeAlso* property.

Once synonyms are detected, they can be merged like shown in figure 12.

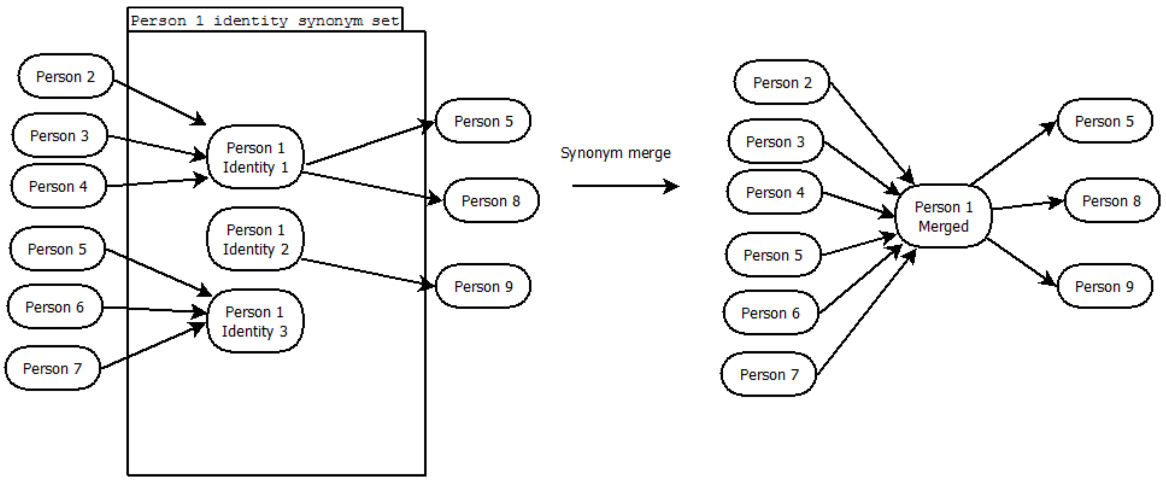
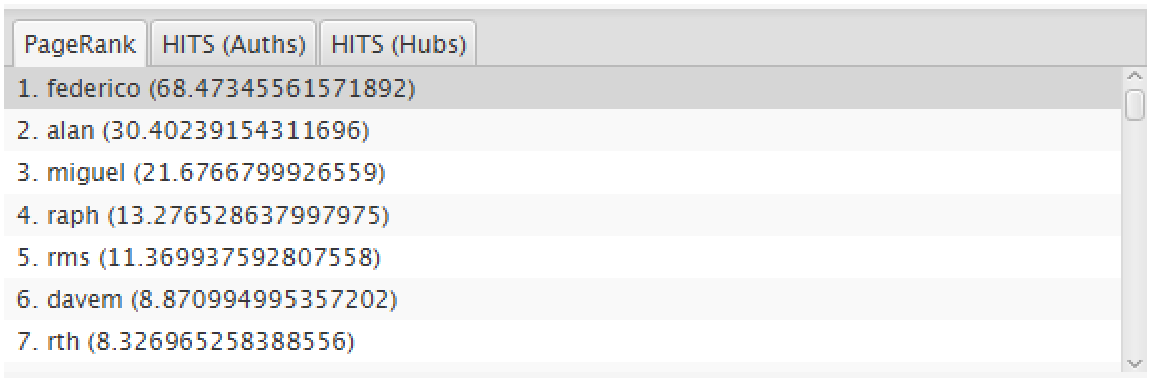


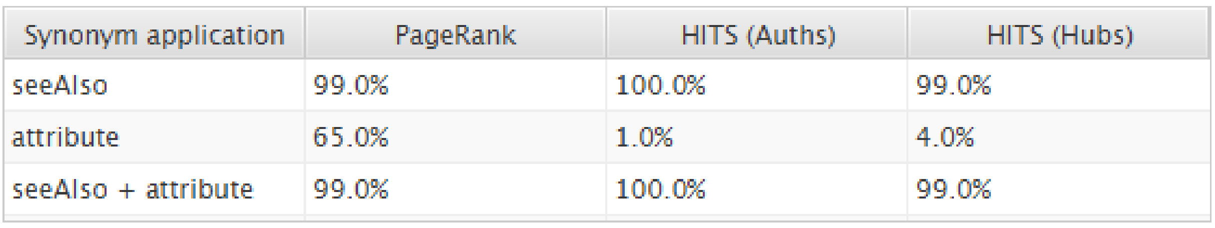
Figure 12. Effects of synonym merge.

Attribute similarity was calculated using attribute difference, as calculated by the Levenshtein distance and stored in a vector. Once it was calculated, SVM (Support Vector Machine) was used to classify potential synonym pairs. This was lastly validated using PageRank and HITS ranking algorithms, as shown in figure 13 and figure 14, to calculate user popularity, and difference in order with and without applying synonym similarity detection.



```
1. federico (68.47345561571892)
2. alan (30.40239154311696)
3. miguel (21.6766799926559)
4. raph (13.276528637997975)
5. rms (11.369937592807558)
6. davem (8.870994995357202)
7. rth (8.326965258388556)
```

Figure 13. Popularity ranking of FOAF users with PageRank.



Synonym application	PageRank	HITS (Auths)	HITS (Hubs)
seeAlso	99.0%	100.0%	99.0%
attribute	65.0%	1.0%	4.0%
seeAlso + attribute	99.0%	100.0%	99.0%

Figure 14. Difference in the resulting popularity order of the first 1000 people

To further modernize Social Network Analysis, in paper [14] Deep Learning techniques were used to detect similar software repositories, although a similar approach could be applied for various social networks and graphs. The approach used unsupervised (or self-supervised) data, which meant there was little manual work required in labeling, and thus it could easily scale to very large networks. This also eliminates the labeling factor presented earlier with SVM classifier for user attribute similarity in FOAF networks.

Software Repositories contain source code, documentation and meta data, and they are used to organize software development. Many open source projects are hosted online using version control systems such as: Git, Mercurial, Apache Subversion (SVN), Concurrent Versions System (CVS) and similar. Popular hosting platforms include: Github[38], Bitbucket[39], SourceForge[40], Assembla[41], and more.

Software repository representation as graphs is shown in figure 15, and the synonym merge is shown in figure 16, akin to the previous user profile synonym detection and merge application.

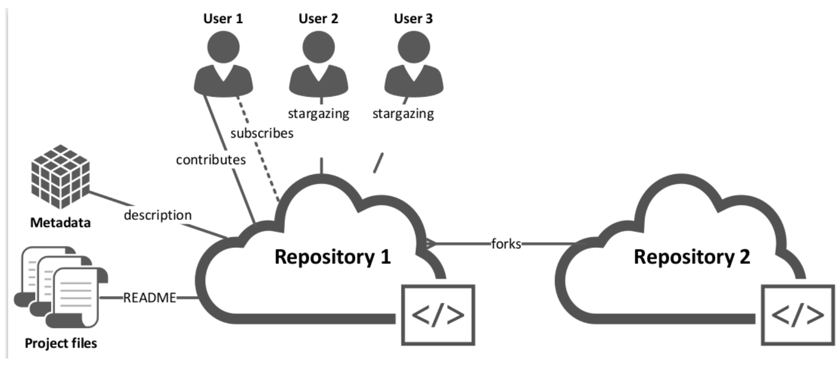
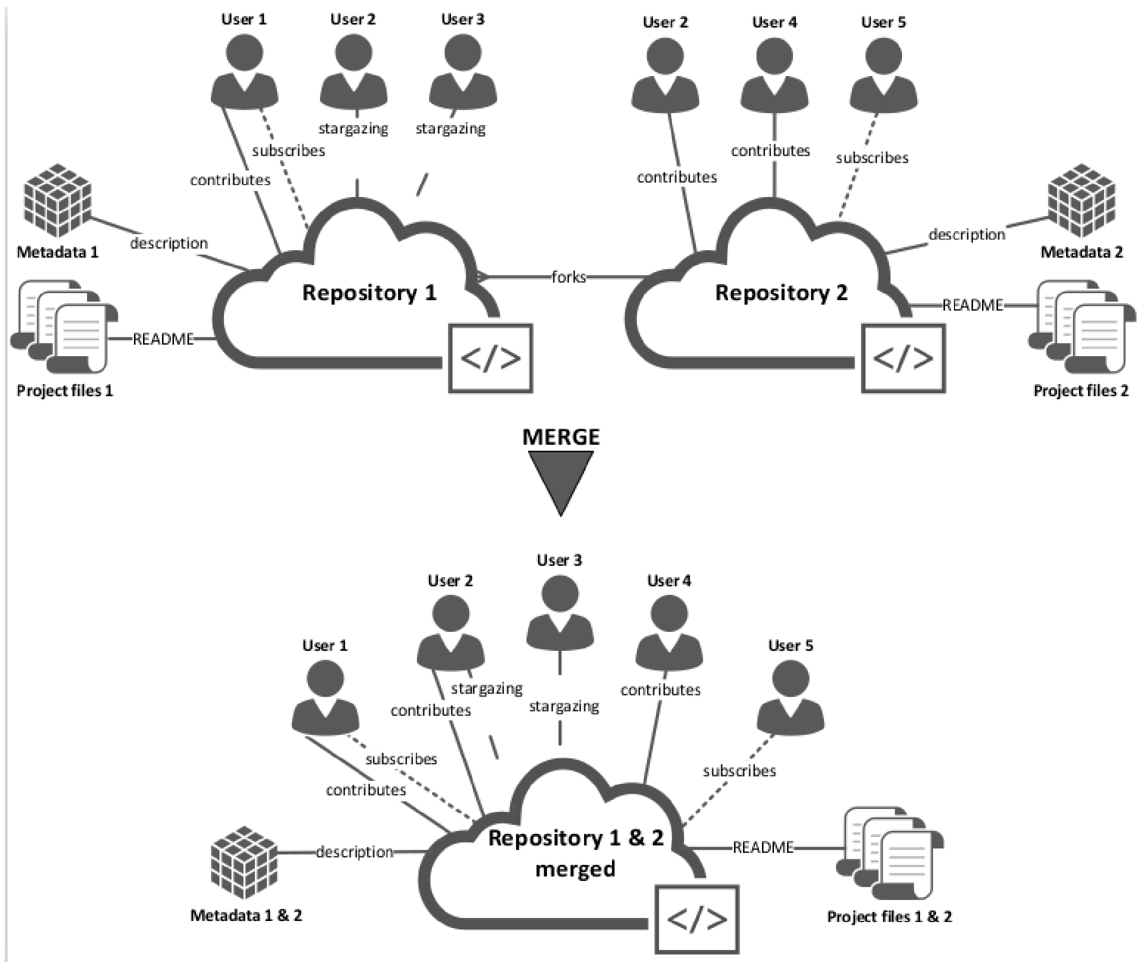


Figure 15. Github Social Network Graph, with user and repository interaction.

38 https://github.com/
39 https://bitbucket.org/
40 https://sourceforge.net/
41 https://www.assembla.com/home



User 1
User 2
User 3
User 2
User 4
User 5
subscribes
stargazing
stargazing
contributes
contributes
contributes
subscribes
Metadata 1
description
Repository 1
forks
Repository 2
description
Metadata 2
README
README
Project files 1
Project files 2
MERGE
User 3
User 2
User 4
User 1
User 5
stargazing
contributes
stargazing
contributes
subscribes
contributes
subscribes
description
Repository 1 & 2 merged
README
Metadata 1 & 2
Project files 1 & 2

Figure 16. Github repository merge.

While the synonym merge looks similar to what was done before with user synonym detection and merge, the introduction of Deep Learning allow synonym detection to become a largely automated process, based purely on the unsupervised social network data.

To use unsupervised data in such a fashion, Autoencoders [57] were used. Autoencoders are Neural Networks that consist of two elements, both of which are neural networks:

- **Encoder.** Encodes the input data to a compressed representation, often called latent space.

- **Decoder.** Decodes the compressed representation to create the reconstructed input.

The process of autoencoding is inherently lossy, and it has a couple of key features: 1) it's self supervised, which means we don't need to provide any additional labels in order to train the network, and 2) the compressed representation can be used to look for similar objects. Searching in this latent space is akin to searching in the latent space of word vectors, where methods like euclidean distance or cosine similarity can be used to find similar objects. The Autoencoder architecture is shown in figure 17.

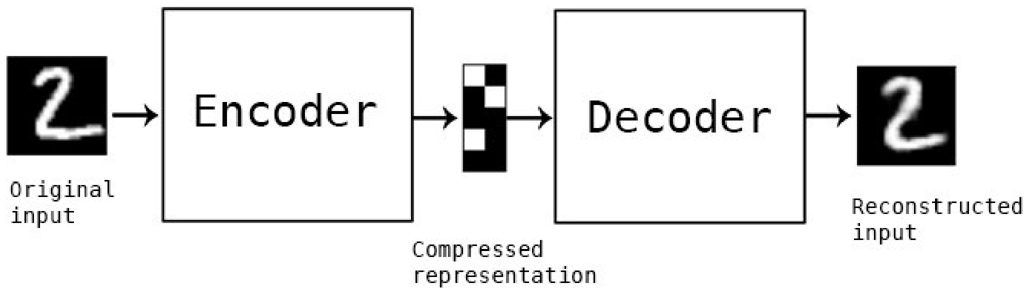


Figure 17. Image autoencoder, source: https://blog.keras.io/building-autoencoders-in-keras.html

To use the graph data, we first need to convert it to a format directly usable by Neural Networks. This was c

value,

(desc

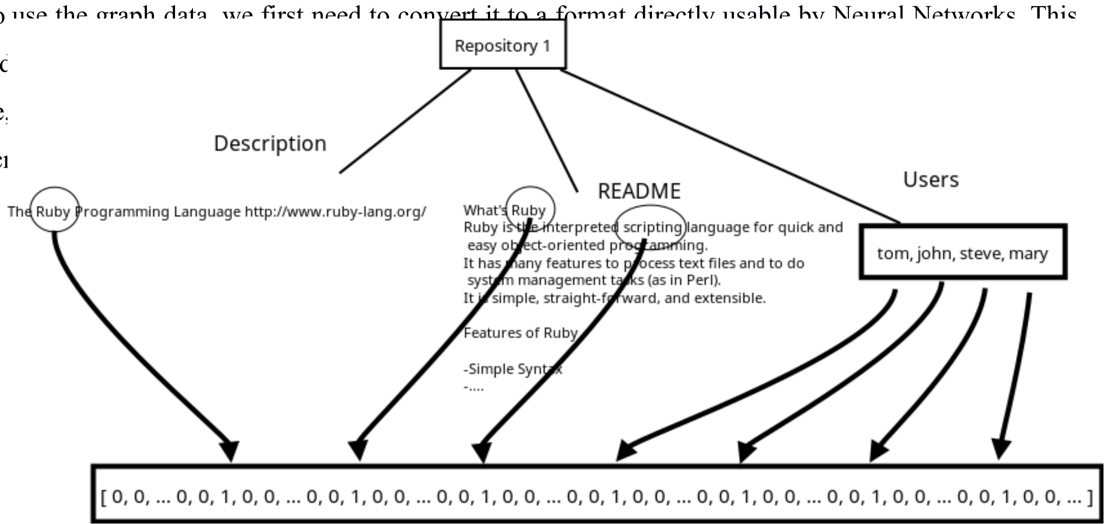

Repository 1
Description
README
Users
The Ruby Programming Language http://www.ruby-lang.org/
What's Ruby
Ruby is the interpreted scripting language for quick and easy object-oriented programming.
It has many features to process text files and to do system management tasks (as in Perl).
It is simple, straight-forward, and extensible.
Features of Ruby
-Simple Syntax
-....
tom, john, steve, mary
[0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ... 0, 0, 1, 0, 0, ...]

Figure 18. One-hot encoding of graph data.

Chapter 4. Scenario creation

Implementing the entire system in the real world is expensive and time consuming. Creating a simulation of such a homecare system first would allow for rapid prototyping and extensive testing. The simulation should be done using a model of the real world. To rapidly create such test cases, a flexible GUI editor is helpful. This would be useful even after such a system has been developed, and it can be used by less experienced users to define their needs.

SpringBoard[9] is a GUI editor with extensible meta-programming support that was developed for this particular use case. It can be used to model and simulate healthcare data exported from ontologies and COTS devices, and it is a GUI editor for rapid prototyping. In it, home-care simulation is treated as a game, where specific healthcare scenarios are created as game scenarios (or levels). It presents a game-agnostic GUI Scenario editor, that provides GUI editing using the trigger concept which consists of:

– Events (things that happen in the world, such as an object entering an area)

– Conditions (certain rules that need to be satisfied to trigger actions)

– Actions (functions that modify the real world)

This is extended with Meta-programming, which allows the ability to define and seamlessly integrate custom event, condition and action types, as well as custom (complex) data types

4.1. GUI programming

Scenario editors are GUI (graphical user interface) programs that are used to create scenarios. That said, sometimes the distinction between a scenario editor and a general purpose engine editor is not clear, and scenario editor functionalities may be contained within the engine editor. There are some reasons why it makes sense to keep the two separate, or at least create a specific scenario export of the engine editor that limits user actions to those that create a valid scenario for the game. Creating such specialized tools, we can further improve user workflow and efficiency. In [9] paper we use the term scenario instead of level or mission, since it is more general and does not assume progress within the game or existence of a specific goal the user needs to achieve. That said, these three terms tend to mean the same thing and can be considered synonymous. The process of creating scenarios usually includes a number of different tasks:

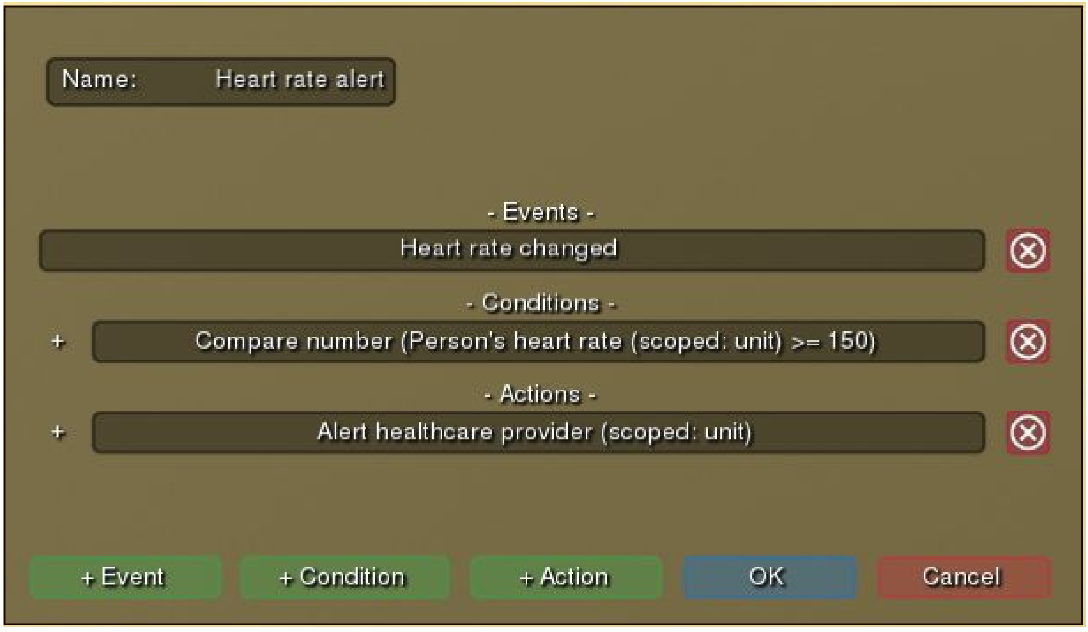
1. Creating the terrain by defining its heightmap and visual properties (diffuse, specular, normal, etc.).
2. Creating and placing predefined objects in the game world.

3. Defining the scene environment, lighting and similar effects.

4. Defining custom game mechanics for the particular scenario.

While all four steps benefit greatly from using an editor instead of having to describe each element by code (and SpringBoard supports all four elements), the focus will particularly be on the fourth step and how we approach the problem of describing custom game mechanics.

In SpringBoard, a simple trigger model has been developed for describing scenario mechanics. Each trigger ti consists of a list of events, conditions and actions, ti(Ei,Ci,Ai), where Ei stands for a list of events: Ei = (ei1, ei2, ...ein), Ci represents a list of conditions: Ci = (ci1, ci2, ...cin), and Ai denotes a list of actions: Ai = (ai1, ai2, ...ain). Events are caused by changes in the game world, which are sometimes due to player actions, but can also simply be a result of game and scenario mechanics. Conditions define checks that have to be satisfied in order for the trigger to fire, and all conditions in the list have to evaluate to true. If all conditions are true, then the list of actions will be executed in the order they are defined. It is not necessary for a trigger to have any events, conditions or actions, although a trigger without actions is unusual, as it has no purpose. A trigger with no formally defined events can still be useful as it can be directly invoked from another trigger. While input elements to conditions and actions can be values, they can also have a complex, tree structure by including other functions. A SpringBoard function is a function that produces one output and has zero or more inputs. While conditions always produce a boolean output type, functions can be specified to produce any defined type. Essentially, conditions are a special case of functions with boolean output type.


Name: Heart rate alert

- Events -

Heart rate changed

- Conditions -

Compare number (Person's heart rate (scoped: unit) >= 150)

- Actions -

Alert healthcare provider (scoped: unit)

Figure 19. Example SpringBoard trigger with one event, condition and action.

The tree structure obtained by combining functions can be arbitrarily deep, as long as all input and output types are present and equivalent.. As triggers themselves do not hold any state information, and the game world itself might not be enough, it is necessary to introduce variables. Variables have user-defined values assigned to them at game start, and they can change by trigger actions as the game progresses. Variables can be queried in conditions and can therefore influence whether or not triggers execute. Once game testing is over, their values are restarted to the originally defined user-defined values.

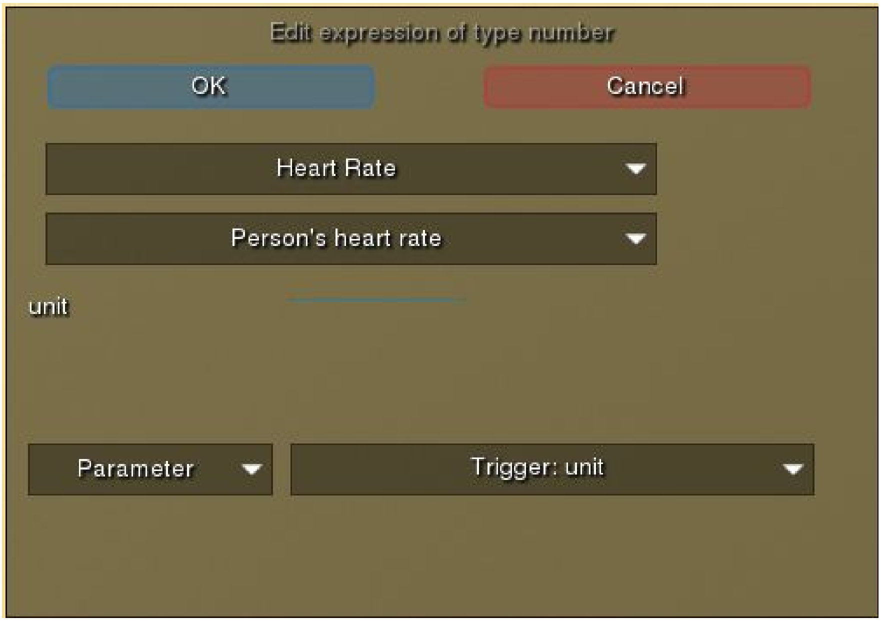


Figure 20. Example of a custom SpringBoard expression that returns a person’s heart rate value.

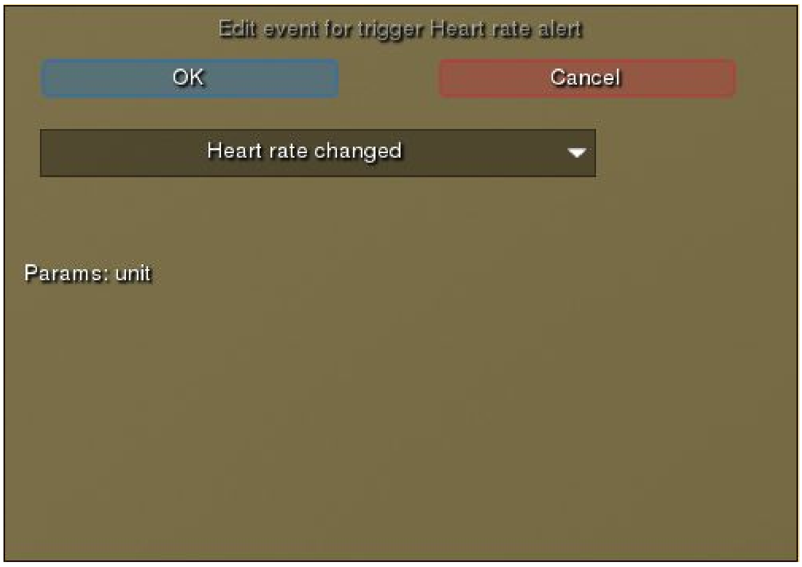


Figure 21. Example of a custom event, triggered on Heart rate change.

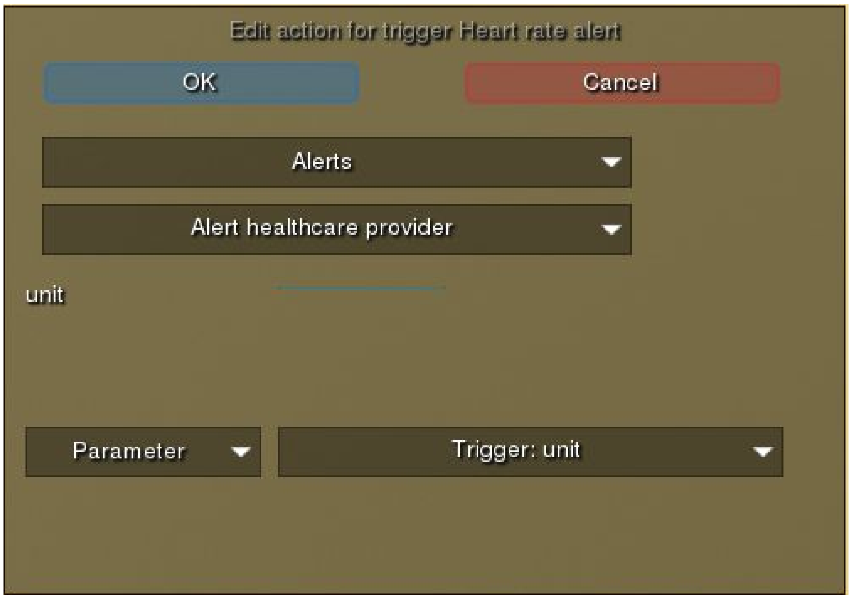

Edit action for trigger Heart rate alert
OK
Cancel
Alerts
Alert healthcare provider
unit
Parameter
Trigger: unit

Figure 22. Example of a custom SpringBoard action, which alerts the healthcare institution.

Due to its WYSIWYG editing nature, scenario editor is particularly useful for describing mechanics that depend on the game world. This includes the simple process of placing objects as well as defining triggers that reference a predefined object or space in the world. However, there are situations where using the scenario editor might not be the best idea. For certain types of problems, such as writing complex intelligent computer opponent behavior (AI bots), it is much better to write them as separate scripts or programs than to try and model such a behavior through That said, simple AI bots can still be created with the trigger system, and it is possible to use the trigger system to control complex AIs in order to achieve a specific effect. The output of the entire scenario trigger model is given as data (a Lua table), which has a couple of benefits. All references in the produced file are done with string keys, which makes it easy to modify the trigger model programatically. As the trigger model is interpreted, it allows us to provide optimizations based on the data model. In case there are errors in certain meta-programs, any bug fixes can be applied without having to modify or recompile each scenario separately. Both the model (and meta-model) are saved in plain text .lua files, which makes it relatively simple to understand changes and have a reasonable idea of historical progression when using a VCS (version control system) software. Additionally as art elements get replacedwith newer versions, the user only needs to overwrite the old files and the scenario will work without requiring any manual changes. This makes it possible to create scenarios using only placeholder objects, and therefore not having to depend on the art production pipeline. Figures 19-22 demonstrate the SpringBoard GUI editor, and how it can be used within a healthcare domain.

4.2. Meta-programming

Meta-programming is the process of specifying trigger types that can be used in the programming process. It involves specifying event, function (condition) and action types as Lua code described by metadata. As we followed the “eat your own dog food” line of thinking, all event, condition and action types are specified this way, including the ones supplied with the editor (core trigger types). This also allows us to provide the ability to write powerful extensions using the same expressive language with which the core of SpringBoard was designed. Before going into details with meta-programming, it is necessary to first describe the type system. The Lua programming language recognizes a few different types, from which we utilize number, string and bool. In addition,we utilize additional types from the Spring system, unitType, feature, featureType, order (for describing objects and their interactions), team (for describing players), and trigger, position and area (as SpringBoard core types). Multiple elements of the same type can be represented as an array, which is defined as a new type, e.g. U1 = (unit1, unit2, ...unitn), with U1 being of type unit array. Meta programming is defined using Lua script files which contain a list of meta event, function (condition) and action definitions.

Definitions comprise the following attributes that determine how they get used:

–humanName (mandatory). A human readable namethat will be displayed in the editor. It is possible for this field to be duplicate, but it is not recommended, because of the potential confusion users may experience.

– name (mandatory). A machine name and also the unique identifier (and key) used when referencing. It should still ideally be kept human understandable so models are verbose.

– input (optional). Zero or more input definitions, that can be defined as a single string (which determines the input type), list of strings (one for each input type), or list of tables. In case of input is specified as tables, each each table defines the input properties: name, humanName, type, raw (whether variables should be sent by value or reference, false by default), allowNil (possible for the value to be nil, false by default).

– output (optional). Zero (in case of actions) or one (in case of functions) string defining the output type.

- execute (mandatory). Lua function that is executed each time the definition is invoked. Inputs are passed in a table, and in case of functions the output should be returned as the function result. Actions should change the game state in execute blocks, while functions should only return the appropriate output, without modifying the game world in any way.

– param (optional). While events do not have input, output and execute fields, they use the param field to specify data sources that become available to the entire trigger. This kind of data becomes resolved at runtime, when the event happens.

– tags (optional). List of tags that are used for grouping definitions. Should be a human-readable name.

The separation of functions and actions in two distinct categories was done by design. This is to guarantee higher program correctness and provide possible optimization techniques. As functions do not modify the world state, they produce the same results for each invocation, for the same world state. They are not necessarily pure functions in the sense of functional programming, as their output depends on the world state (in pure functions it is always the same for the same inputs). Still, the separation gives a higher guarantee of producing clearer code and still has room for potential optimizations, such as lazy evaluation (evaluating the functions as needed). As we mentioned previously, SpringBoard supports basic Lua data types and Spring concepts. We have created a GUI for using these data types in the level editor, which ensures that each specified value is valid (type correct and, in case of Spring concepts, refers to an existing object). While this interface is intuitive, as game-specific needs increase, higher levels of abstractions are needed to represent game concepts. It is therefore necessary to be able to define custom data types. Using them allows developers to specify game concepts in the meta-model.

We have created support for defining custom data types as composites of basic data types. These data types are defined and used in the meta-model, which allows them to be used in the GUI editor seamlessly. In the model, custom data types are treated as an expression which produces an associative array of its inputs. When comparing to GPLs (general-purpose languages) they are similar to C's structs, Java's POJOs (plain old Java objects) and Lua's associative tables. In the meta-model, custom data types are specified under the dataTypes field. A custom data type consists of the humanName, name and input attributes, each of which is defined in the same way as in the meta-model expression attributes. The comprising basic data types are listed in the input attribute.

Custom data types can be used in meta-model expressions in the same way as basic data types. The comprising basic data types are accessed by their name key, much like it is done with expression parameters. As a sidenote, it is possible to define composites of custom data types (composites of composites), much like it is done in GPLs. The only requirement is that the child data types are defined beforehand. Example: a game might define a hand to consist of five fingers, each of which is defined as a composite of some basic types, like health(number) and size(number). It is necessary to define the finger before the hand as it consists of it. Both the hand and the finger can be used as types in meta-models. The limitation of this approach is that custom data types can only be defined as composites of other data types. It is therefore not possible to create new data types by putting restrictions on existing data types (e.g. capitalizedString, integerNumber, etc.), or to define a custom GUI for editing the data type. These are features that we intend to implement in future versions.

4.3 Validation

SpringBoard was additionally validated in a small user study, with 7 participants. In the user study, users were given 5 tasks, to be completed with GUI editing and Lua programming in 20 minutes.

The user study contained both quantitative measurements, where it was measured how many tasks they managed to complete in the alloted time, and how correct their solution was, and also in the qualitative measure, where they expressed feedback regarding usability of the GUI program in comparison to programming.

All users were complete novices to the SpringRTS Lua programming ecosystem and SpringBoard, as well as any other scenario creation tool. They also lacked any gamedev experience, but they did have previous programming experience, as they have all completed at least four years of study at a university. In both GUI editing and Lua programming, they were given a short (5min) introduction on how to complete basic tasks, and they were given translation assistance during task execution, so as not to make the distinction between programming and GUI editing be due to natural language knowledge.

The results were as follows:

- **Quantitative measures** (average tasks completion):

GUI editing.	Mean: **3.12**	Median: **3.0**
Lua programming.	Mean: **1.43**	Median: **1.0**

- **Qualitative measures** (feedback)[42], as shown in table below:

(GUI vs Programming)	User satisfcation	Expressibility	Aesthetics	Robustness
Mean	4.57	4.43	4.14	4.43
Median	5	5	4	5

From these results we can conclude that SpringBoard GUI editing outperforms Lua Programming both in quantitative terms (where users managed to complete about 2 more tasks in average), as well as their satisfaction.

42 Rated from 1 to 5 while comparing Programming to GUI. Smaller numbers meant that Programming was better than GUI, large numbers meant that GUI was better than Programming

Conclusion

In this thesis two key healthcare issues have been addressed: 1) integration of healthcare systems and 2) homecare system for the elderly.

In particular, an ontology alignment method has been proposed for automatic integration using word and sentence embeddings. This approach allows to obtain fast and accurate alignments without the use of structured background knowledge, such as a medical database.

The proposed homecare system consists of two modules: 1) sentiment analysis for personalized healthcare, and a 2) GUI editor for rapid system modeling. Contributions in the field of sentiment analysis include synonym detection and an improvement on existing techniques by using structured and semi-structured graph information. A highly extensible GUI editor with meta-programming has been created, that allows for rapid prototyping and creation of scenarios for homecare situation modeling.

The future work required to complete the thesis is:

- Further improvement of ontology alignment and submission as a journal paper. While the current results beat average state of the art systems, as well as baseline string comparison methods, there is still room for improvement and the goal should be to compete with top systems.

- Validation of the GUI editor's effectiveness through a use-case. It is necessary to test how effective it is in comparison to classical programming techniques.

- Creation of ontology alignment between different-domain ontologies (e.g. healthcare and social network ontologies).

References

[1] National Institute of Population and Social Security and Research, "Population Projections for Japan (January 2012): 2011 to 2060,", 2012.

[2] M. Zhao and S. Zhang, "FCA-Map Results for OAEI 2016," Ontol. Matching, pp. 172, 2016.

[3] W. Wang and P. Wang, "Lily results for OAEI 2015," Ontol. Matching, pp. 162, 2015.

[4] E. Jiménez-Ruiz, B. C. Grau, and V. Cross, "LogMap family participation in the OAEI 2016," Ontol. Matching, p. 185, 2016.

[5] M. Gulić, B. Vrdoljak, and M. Banek, "CroMatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment," Web Semant. Sci. Serv. Agents World Wide Web, vol. 41, pp. 50–71, Dec. 2016.

[6] Gulić, Marko, Vrdoljak, Boris, and Banek, Marko, "CroMatcher - Results for OAEI 2016," 2016.

[7] G. Petrović and H. Fujita, "SoNeR: Social Network Ranker," Neurocomputing, vol. 202, pp. 104–107, Aug. 2016.

[8] G. Petrovic and H. Fujita, "Effect of relationships in Social Networks on calculating user sentiment profiles," Front. Artif. Intell. Appl., pp. 17–24, 2015.

[9] G. Petrovic and H. Fujita, "SpringBoard: game-agnostic tool for scenario editing with meta-programming support," Appl. Intell., Oct. 2017.

[10] G. Petrović, V. Dimitrieski, and H. Fujita, "Cloud-based health monitoring system based on Commercial Off-The-Shelf hardware," in Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on, pp. 3713–3718 , 2016.

[11] V. Dimitrieski, G. Petrović, A. Kovačević, I. Luković, and H. Fujita, "A Survey on Ontologies and Ontology Alignment Approaches in Healthcare," in Trends in Applied Knowledge-Based Systems and Data Science, vol. 9799, H. Fujita, M. Ali, A. Selamat, J. Sasaki, and M. Kurematsu, Eds. Cham: Springer International Publishing, pp. 373–385, 2016.

[12] G. Petrovic and H. Fujita, "Semi-automatic Detection of Sentiment Hashtags in Social Networks," in Intelligent Software Methodologies, Tools and Techniques, vol. 532, H. Fujita and G. Guizzi, Eds. Cham: Springer International Publishing, pp. 216–224, 2015.

[13] G. Petrovic and H. Fujita, "Deep Correct: Deep Learning Color Correction for Color Blindness," Front. Artif. Intell. Appl., pp. 824–834, 2017.

[14] G. Petrović, V. Dimitrieski, and H. Fujita, "A Deep Learning Approach for Searching Cloud-Hosted Software Projects," Front. Artif. Intell. Appl., pp. 358–368, 2016.

[15] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto, "The AgreementMakerLight Ontology Matching System," in On the Move to Meaningful Internet Systems: OTM 2013 Conferences, vol. 8185, R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. De Leenheer, and D. Dou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 527–541, 2013.

[16] S. Rill, D. Reinel, J. Scheidt, and R. V. Zicari, "PoliTwi: Early detection of emerging political topics on twitter and the impact on concept-level sentiment analysis," Knowl.-Based Syst., vol. 69, pp. 24–33, 2014.

[17] E. Cambria, A. Hussain, and C. Eckl, "Bridging the Gap between Structured and Unstructured HealthCare Data through Semantics and Sentics", 2011.

[18] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-speech tagging for twitter: Annotation, features, and experiments," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pp. 42–47, 2011.

[19] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith, "Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters.," in HLT-NAACL, pp. 380–390, 2013.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.

[21] L. Hong and B. D. Davison, "Empirical study of topic modeling in twitter," in Proceedings of the First Workshop on Social Media Analytics, pp. 80–88, 2010.

[22] M. Oussalah, F. Bhat, K. Challis, and T. Schnier, "A software architecture for Twitter collection, search and geolocation services," Knowl.-Based Syst., vol. 37, pp. 105–120, Jan. 2013.

[23] E. Cambria, D. Olsher, and D. Rajagopal, "SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis," in Twenty-eighth AAAI conference on artificial intelligence, 2014.

[24] C. Havasi, R. Speer, and J. Alonso, "ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge," in Recent advances in natural language processing, pp. 27–29, 2007.

[25] R. Justo, T. Corcoran, S. M. Lukin, M. Walker, and M. I. Torres, "Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web," Knowl.-Based Syst., vol. 69, pp. 124–133, 2014.

[26] X. Zhang, J. Cheng, S. Qiu, G. Zhu, and H. Lu, "DualDS: A dual discriminative rating elicitation framework for cold start recommendation," Knowl.-Based Syst., vol. 73, pp. 161–172, 2015.

[27] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems,", pp. 208, 2008.

[28] C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, and P. Li, “User-level sentiment analysis incorporating social networks,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1397–1405, 2011.

[29] E. Cambria and B. White, “Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article],” IEEE Comput. Intell. Mag., vol. 9, no. 2, pp. 48–57, 2014.

[30] E. Cambria, A. Livingstone, and A. Hussain, “The hourglass of emotions,” in Cognitive behavioural systems, Springer, 2012, pp. 144–157.

[31] M. McPherson, L. Smith-Lovin, and J. M. Cook, “Birds of a Feather: Homophily in Social Networks,” Annu. Rev. Sociol., vol. 27, no. 1, pp. 415–444, 2001.

[32] Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua, “Generative Adversarial Networks”, arXiv:1406.2661, 2014

[33] A. Hoffmann, Byeong-ho Kang, D. Richards, S. Tsumoto, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. P. Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum, “Advances in Knowledge Acquisition and Management”, volume 4303 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[34] T. Beale. “Archetypes: Constraint-based domain models for future-proof information systems”, In OOPSLA 2002 workshop on behavioural semantics, volume 105, Seattle, USA, November 2002.

[35] C. Martínez-Costa, M. Menárguez-Tortosa, and J. Tomás Fernández-Breis. “An approach for the semantic interoperability of ISO EN 13606 and OpenEHR archetypes". Journal of Biomedical Informatics, 43(5):736–746, October 2010.

[36] J. Tomás Fernández-Breis, P. José Vivancos-Vicente, M. Menárguez-Tortosa, D. Moner, J. Alberto Maldonado, R. Valencia-García, and T. Gonzalo Miranda-Mena. “Using semantic technologies to promote interoperability between electronic healthcare records’ information models”. In Proceedings of the 28th IEEE EMBS Annual International Conference, pages 2614–2617, New York City, USA, 2006.

[37] M. R. Santos, M. P. Bax, and D. Kalra. “Building a logical EHR architecture based on ISO 13606 standard and semantic web technologies”. Studies in Health Technology and Informatics, 160(1):161–165, 2010.

[38] M. Menárguez-Tortosa and J. T. Fernández-Breis. “OWL-based reasoning methods for validating archetypes”. Journal of Biomedical Informatics, 46(2):304–317, April 2013.

[39] J. Coyle, Y. Heras, T. Oniki, and S. Huff. “Clinical element model.” Technical report, University of Utah, 2008.

[40] A. James. Qualibria “Constraint Definition Language (CDL)”. Technical report, 2010.

[41] C. Tao, G. Jiang, T. A. Oniki, R. R. Freimuth, Q. Zhu, D. Sharma, J. Pathak, S. M. Huff, and C. G. Chute. “A semantic-web oriented representation of the clinical element model for secondary use of

electronic health records data.” Journal of the American Medical Informatics Association, 20(3):554–562, May 2013.

[42] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S. A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis. “The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration.” Nature Biotechnology, 25(11):1251–1255, November 2007.

[43] L. Marie Schriml, C. Arze, S. Nadendla, Y. W. W. Chang, M. Mazaitis, V. Felix, G. Feng, and W. Alden Kibbe. “Disease Ontology: a backbone for disease semantic integration.” Nucleic acids research, 40(1):940–946, 2012.

[44] Gene Ontology Consortium. “The Gene Ontology (GO) database and informatics resource.” Nucleic Acids Research, 32(1):258–261, January 2004.

[45] C. E. Lipscomb. “Medical Subject Headings (MeSH).” Bulletin of the Medical Library Association, 88(3):265–266, July 2000.

[46] S. Liu, W. Ma, R. Moore, V. Ganesan, and S. Nelson. “RxNorm: prescription for electronic drug information exchange.” IT professional, 7(5):17–23, 2005.

[47] C. J. McDonald, S. M. Huff, J. G. Suico, G. Hill, D. Leavelle, R. Aller, A. Forrey, K. Mercer, G. DeMoor, J. Hook, and others. “LOINC, a universal standard for identifying laboratory observations: a 5-year update.” Clinical chemistry, 49(4):624–633, 2003.

[48] C. A. Puri, K. Gomadam, P. Jain, P. Z. Yeh, and K. Verma. “Multiple Ontologies in Healthcare Information Technology: Motivations and Recommendation for Ontology Mapping and Alignment.” In International Conference on Biomedical Ontologies (ICBO), 2011.

[49] O. Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research, 32(1):267–270, 2004.

[50] D. Booth, C., Huff, S.M., Fry, E., Dowling, C.,Mandel, J.C.: “RDF as a Universal Healthcare Exchange Language” 2013

[51] T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, CoRR, 2013

[52] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, “Enriching Word Vectors with Subword Information “, arXiv:1607.04606, 2016

[53] J. Pennington, R. Socher, and C. D. Manning. “GloVe: Global Vectors for Word Representation.” 2014.

[54] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. "Skip-Thought Vectors." arXiv preprint arXiv:1506.06726, 2015.

[55] C. Alexis, K. Douwe, S. Holger, B. Loic, B. Antoine, “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data“, arXiv preprint arXiv:1705.02364, 2017

[56] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions" Communications of the ACM, vol. 51, no. 1, pp. 117-122. 2008

[57] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. “Stacked convolutional auto- encoders for hierarchical feature extraction. In International Conference on Artificial Neural Networks”, pages 52–59. Springer, 2011.