UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES ÉCONOMIQUES

Ontology Engineering using Formal Concept Analysis from Unstructured Textual Data

PhD thesis submitted to the Faculty of Economics and Business

Institute of Management of Information Systems

University of Neuchâtel

For the PhD degree in Computer Science

by

Simin JABBARI

Approved by the dissertation committee:

Prof. Kilian Stoffel, University of Neuchâtel, thesis director

Prof. Adrian Holzer, University of Neuchâtel, head of jury

Prof. Paul Cotofrei, University of Neuchâtel, expert

Prof. Jacques Savoy, University of Neuchâtel, external expert

Dr. Giovanni D'Ario, F. Hoffmann-La Roche, external expert

Defended on 29.10.2019

Faculté des sciences économiques
Avenue du 1er-Mars 26
CH-2000 Neuchâtel
www.unine.ch/seco

LETTRES ET SCIENCES HUMAINES
SCIENCES
DROIT
SCIENCES ECONOMIQUES

UNIVERSITÉ DE NEUCHÂTEL

IMPRIMATUR POUR LA THÈSE

Ontology Engineering using Formal Concept Analysis
from Unstructured Textual Data

Simin JABBARI

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES ÉCONOMIQUES

La Faculté des sciences économiques,
sur le rapport des membres du jury

Prof. Kilian Stoffel (directeur de thèse, Université de Neuchâtel)
Prof. Adrian Holzer (président du jury, Université de Neuchâtel)
Prof. Paul Cotofrei (Maître d'enseignement et de recherche, Université de Neuchâtel)
Prof. Jacques Savoy (Faculté des sciences, Université de Neuchâtel)
Dr. Giovanni d'Ario (senior data scientist, Roche)

Autorise l'impression de la présente thèse.

Neuchâtel, le 2 décembre 2019

Annik Dubied

La doyenne
Annik Dubied

■ Téléphone : +41 32 718 15 00 ■ E-mail : secretariat.seco@unine.ch ■ www.unine.ch

Abstract

Knowledge extraction especially from unstructured data such as texts has been always considered as one of the highly demanded requests with lots of applications in almost all industries. Design and building of solutions that are capable of extracting knowledge, in an almost automated way, is not an easy task at all. Many researchers have proposed variety of methodologies and algorithms to describe how one can give some structure to textual data with the ultimate goal of knowledge extraction since decades ago. One of the key elements of those solutions is to utilize *ontology* as a graph-like structure for representing knowledge. Building ontologies especially from textual data, however, is not quite straightforward. To the best of our knowledge, there is no yet a comprehensive methodology to describe how one can form an ontology from processing textual data in a given domain of interest to be later used for explicit as well as implicit (or semantic) knowledge extraction.

In this thesis, we propose a pipeline to describe how we can start from analyzing texts to end up with an ontology, which is equipped with the most informative statements of that text corpus about a given context, in order to be used for knowledge extraction. The proposed pipeline is based on utilization of three different yet complementary data analysis methods including (i) natural language processing, (ii) formal concept analysis, and (iii) ontology learning. In a nutshell, the pipeline will start by mining the input text corpus (in a given domain of interest) using state-of-the-art natural language processing techniques. The formal concept analysis will then be used to form the concepts and build the hierarchies among them (i.e., a concept lattice) as the cornerstone of the desired ontology. Finally, the most informative statements extracted from text corpus will be embedded into the ontology, that has been derived from a set of proposed algorithms applied on the aforementioned concept lattice.

To validate the accuracy of the proposed pipeline we tested it on a few toy examples as well as a real use case in the context of pharmaceuticals. We could demonstrate that such an engineered ontology can be used for querying valuable knowledge and insights from unstructured textual data, and to be employed as the core component of *smart search engines* with applications in semantic analysis. One of the advantages of our proposed solution is that it does not require so much of human intervention, as opposed to many existing solutions whose performance highly depends on the presence of a subject matter expert along the ontology engineering process. It does not, however, mean that our proposed pipeline cannot benefit from existence of such additional information resources to be further empowered by human expertise in shaping ontologies.

Key words: Ontology Learning, Knowledge Representation, Formal Concept Analysis, OWL, Unstructured Data

Résumé

L'extraction de connaissances, en particulier à partir de données non structurées tels que les textes, a longtemps était considérée comme une des demandes les plus souhaitées, avec un grand nombre d'applications dans presque toutes les industries. La conception, ainsi que la mise en place de solutions capables d'extraire des connaissances, de façon presque automatique, est loin d'être une tâche facile. Depuis déjà plusieurs décennies, différents chercheurs ont proposé une variété de méthodologies et d'algorithmes afin de décrire comment donner une certaine structure à des données textuelles, avec pour but ultime l'extraction de connaissances. Un des éléments clés de ces solutions est d'utiliser une ontologie reposant sur une structure de graphe, et rendant possible une représentation de connaissances. Cependant, la construction d'ontologies, en particulier à partir de textes, n'est pas aisée. A notre connaissance, il n'y a pas encore de méthodologie complète décrivant la construction d'une ontologie à partir du traitement de données textuelles, dans un domaine d'intérêt donné, pour être par la suite utilisée pour l'extraction de connaissances explicites ainsi que implicites (sémantiques).

L'objectif de cette thèse est de proposer un pipeline décrivant comment partir de l'analyse de textes pour finalement arriver à une ontologie comprenant les propositions les plus informatives de ce corpus de textes sur un contexte donné, et dans le but d'être utilisé pour l'extraction de connaissances. Ce pipeline repose sur l'utilisation de trois méthodes d'analyse de données, tout aussi différentes que complémentaires, incluant (i) le traitement du langage naturel, (ii) l'analyse formelle de concepts, et (iii) l'apprentissage d'ontologies. En résumé, le pipeline débutera par une exploration du corpus de textes en entrée (pour un domaine d'intérêt bien défini), faisant usage des techniques de traitement du langage naturel les plus avancées. L'analyse formelle de concepts sera par la suite utilisée pour former les concepts et construire leurs hiérarchies (i.e., un treillis de concepts), constituant le fondement de l'ontologie désirée. Enfin, les propositions les plus informatives du corpus de textes seront intégrées au sein de l'ontologie, dérivée au préalable d'un ensemble d'algorithmes proposés dans cette thèse et appliqués au treillis de concepts susmentionné.

Afin de valider la précision de notre pipeline, nous l'avons testé avec quelques exemples synthétiques ainsi qu'avec de vrais cas d'utilisation dans le contexte de l'industrie pharmaceutique. Nous avons pu démontrer qu'une telle ontologie obtenue peut être utilisée pour interroger d'importantes connaissances ainsi que des informations extraites de données textuelles non structurées, et peut être employée comme élément central au sein d'un moteur de recherche intelligent, avec des applications en analyse sémantique. Un des avantages de

notre solution est de minimiser l'intervention humaine, contrairement à beaucoup d'autres solutions déjà existantes et aux performances fortement dépendantes de la présence ou non d'un expert en la matière tout au long du processus de construction de l'ontologie. Lors du processus de validation, nous impliquons toutefois toujours l'expertise humaine, afin de garantir la constante amélioration de notre ontologie.

Mots clefs : Apprentissage de l'Ontologie, Représentation des Connaissances, Formal Concept Analysis, OWL, Données Non Structurées

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Dr. Kilian Stoffel for his continues support throughout my Ph.D. study. His immense domain knowledge, patience and motivation have always given me the more power for conducting my doctoral research. I am very grateful to him for giving me the chance of becoming part of his research team and for forging my scientific personality. He has provided me an opportunity to grow and to enhance my academic and industry skills.

I would also like to thank my thesis committee members : Prof. Adrian Holzer, Prof. Paul Cotofrei, Dr. Giovanni D'Ario, and Prof. Jacques Savoy, for accepting to assess my thesis and for their insightful comments, which incensed me to widen my research from various perspectives.

I am also pleased to F. Hoffmann-La Roche, for allowing me the opportunity to expand my academic skills through the internship. I know the skills I have learned as a result will help catapult my success. I was so grateful of being able to take part in the Diagnostics Data Science Lab (DDSL) where I could apply my theoretical skills into real case studies. I will not forget to express the gratitude to all DDSL members, for giving the encouragement and sharing insightful suggestions and endless guidelines.

I am also very thankful to all my colleagues in Information Management Institute (IMI), past and present, for their friendship, and for all the fun we have had in the last four years. They have all played a major role in completing my doctoral study. I would always remember their sympathy and kindness in tough moments.

My foremost gratitude and respect go to my parents, for their unconditional love and dedication. I would like to deeply appreciate them for all the sacrifices they made for me, and for providing me an everlasting support for making progress. They receive my deepest gratitude and love for their dedication. I have no doubt that I am truly indebted to them for all the success I have achieved in my life.

Especially, I express my deepest appreciation to my beloved husband Mohammad, for standing beside me in every single step towards this achievement. I cannot express my feelings about having such a wonderful partner in my life. He has always been my inspiration and motivation for making progress. I truly appreciate the love of my life, for believing in me, and for the encouragement he has always given to me for a brighter future.

Neuchâtel, 29 October 2019 Jabbari Simin.

Wings are a constraint that makes
it possible to fly.
— Robert Bringhurst

This thesis is dedicated to my flying wings, my *mom* and my *husband*. For their continuous support and impatient love throughout my life. For giving me the strength to reach for the stars and chase my dreams and for having always supported me and encouraged me to work hard for the things that I aspired to achieve. I am truly grateful for having you both in my life...

Contents

List of Figures

List of Tables

1 Introduction

"It is vital to remember that information - in the sense of raw data - is not knowledge, that knowledge is not wisdom, and that wisdom is not foresight. But information is the first essential step to all of these."

Arthur C. Clarke

1.1 Motivations

In today's world, data is considered as a valuable asset. The volume of data being generated in the past decades has dramatically increased compared to the past. Besides the availability of huge amount of data, an incredible computational power as well as cheap data storage, together with recent advances in the field of data science motivate all enterprises to invest on generating actionable insights from their data assets. Many data assets are currently unstructured, i.e., there is no well-defined tabular structure to store such data. But textual data assets are among the most valuable unstructured data assets we have today. However, extracting knowledge from textual data is not as easy as knowledge extraction from structured data.

Many organizations, including pharmaceutical companies, are producing vast amounts of data in the form of unstructured texts. These include manuals for products, articles about launching new products, medical electronic reports, news articles and web-pages containing market information, etc. In today's competitive world, analyzing such data provides actionable insights and competitive intelligence for large companies about their major competitors. The primary application of this dissertation is Market and Competitive Intelligence (MCI), as one of the major topics in pharmaceutical organizations. The ambition of MCI is to understand the business environment around the company by analyzing and retrieving information from its competitors data. This supports future-oriented decision making of the company based on high quality and up-to-date insights. Nevertheless, the application of this study could be

expanded into other domains such as the medical area in which information extraction is required from medical historical records.

Knowledge extraction from huge collection of text documents, however, requires an enormous human effort in order to meticulously go through the content of each document. Therefore, small and big companies are currently investing in automating this process by benefiting from various advanced analytic techniques in Artificial Intelligence (AI).

One of the key challenges in dealing with textual data is the lack of proper knowledge representation models. Providing an abstract view of important concepts and their relationships would be a great starting point for giving structure to unstructured textual data. In other words, there should be a more convenient way (such as an intelligent AI-based solution) than visually screening all those documents by a human in order to extract knowledge and realize the value hidden in the content of those text documents. Therefore, automating the knowledge extraction process becomes a crucial task in the digital universe of unstructured textual data.

Ontologies have been proposed as a promising solution for representing knowledge. In this research work, we are proposing an *ontology learning* technique for meaningful knowledge extraction from textual documents by automatically acquiring the main concepts, properties and their corresponding relationships for a domain of interest. Like any other information extraction technique, learning ontologies requires powerful data analysis methods and an extensive background knowledge, which could be gathered from both unstructured sources such as text documents and structured sources in the form of tabular data bases.

As described in WordNet [1], the background knowledge [1] refers to the *"information that is essential to understanding a situation or problem"*. The majority of the ontology learning approaches highly rely on pre-built ontologies to construct the core of their ontology. This dependency, however, introduces the challenge of re-adjustment of the core units whenever a change is happening into the source ontology. The significance of the work in this dissertation is the propose of a pipeline for learning an ontology from scratch which solely depends on the provided text documents as its background knowledge and other official data sources for enhancing the constructed ontology.

1.2 Problem Description

Ontology learning from text corpora is one of the promising research areas with the aim of extracting explicit and implicit information from the body of text by transforming hidden facts and patterns into shareable constructs. The process of ontology learning from text is to extract it's fundamental components such as domain concepts, relations and potential axioms.

There is no doubt that Natural language processing (NLP) is a powerful toolset for analyzing textual data. However, existing NLP methods also suffer from some shortcomings. One of them

[1] http://wordnetweb.princeton.edu/perl/webwn?s=background%20knowledge

is that existing classic modules in NLP had been trained on casual texts (usually from wikipedia or web contents) and are barely trained on documents that contain a domain specific context such as the medical domain. This may cause some issues when we use casual NLP tools for the medical domain. This further strengthens the importance of training domain-specific NLP tools for different contexts. Therefore, more powerful linguistic and statistical data analysis and AI techniques are needed to enable valuable information extraction.

Another important concern when building ontologies from a corpus is that, the textual documents within the corpus may not be complete in the sense that relevant information about that context may not be available in that corpus. Therefore, we may face a situation where the learned ontology misses important data that could potentially change the engineered ontology. To address such challenge, we suggest to further augment the corpus by other domain-relevant textual data repositories to ensure the engineered ontology becomes as precise as possible, although it is possible that such a comprehensive and precise ontology may never be engineered. Nevertheless, the point is that if we can leverage other existing resources besides the initial corpus for engineering the ontology for that domain, there are no arguments for not doing so. There are many researches being conducted for addressing the issue of incompleteness of background knowledge, where each of them is focused on a specific requirement. For instance in the following studies, Wikipedia has been widely used for the purpose of relation acquisition and word sense disambiguation [2], [3], [4] and [5]. Moreover, many authors used web search engines with the goal of text corpus construction [6], [7], similarity measurements [8] and word collection [9], [10]. However, a fundamental research that focuses on choosing the proper supplementary data source (besides an existing text corpus) is long overdue and it highly depends on the application domain.

We are faced with the following challenges throughout the ontology construction process:

1. How to figure out if we have sufficient background knowledge to perform all ontology learning tasks?
2. Which knowledge base could cover inadequate pieces of the original data?
3. And most importantly which statistical and linguistic techniques shall be used in each step to guarantee the completeness of the constructed ontology based on the provided data?

1.3 Literature Review

There were many studies being conducted in the scope of ontology learning from text corpora. Before starting our discussion on the current prominent techniques and recent tools, let us recap the key findings of a few performed surveys. The OntoWeb Consortium has conducted a survey in 2003 on ontology learning techniques from text [11]. The findings highlighted in this report are summarized as follows:

1. There is no fully automated system for ontology learning from unstructured data.
2. There is no systematic approach for ontology learning from text and it is highly dependent on the domain of interest and discretion of the ontology designer.
3. There is no general way for evaluating the correctness and completeness of the constructed ontology.

Another survey is presented in [12] with the goal of introducing a general framework for classifying and comparing the state of the art Ontology Learning (OL) systems. This framework composed of three dimensions concerning the following aspects of the OL systems: what to learn, from where to learn and how to learn. In this study, the authors studied around 50 different OL approaches and only retained 7 prominent ones for detailed analysis according to the framework. The summary of their finding over the the most recent techniques is the following:

1. Most of the studies were focused on extracting taxonomic relation among classes and there was less attention paid to non-taxonomic relation extraction.
2. The main concern of the majority of proven techniques is to generate domain related ontologies. This decreases the possibility of generalizing the proposed approach across different domains.
3. Despite an enormous effort of many researchers in various domain, fully automated tools and techniques are still unexplored. Most systems and techniques require either human involvement or support of a pre-built ontology.
4. Evaluation of ontologies requires more research and exploration and a formal, standard method for assessing the constructed ontology is highly required.
5. Learning ontological axioms is missing in most of the studies.

Moreover, by placing the selected systems into the proposed framework, they could describe the strengths and weaknesses of each dimension. Consequently, a general guideline is presented for researchers to choose the appropriate features for creating an OL system from scratch or re-using existing techniques for their own domain of interest.

The third survey was conducted by Ding and Foo [13] on 12 ontology learning projects. The authors claimed that most of the projects cover ontology learning from structured data. They found out that discovering relations is more difficult than discovering concepts. In [14] ontology learning is considered as composed of multiple subtasks. The authors mainly compared the various approaches based on their effort on automating the different steps. Their study has similar findings as in the aforementioned surveys such as:

1. One of the open problems remains evaluating the ontology. No standard and formal method has yet been discovered for proving the accuracy, efficiency and completeness of the built ontology.

2. Not all of the ontology learning steps are automated. Full ontology development process is still an open issue.

3. Ontology learning will remain as an active field of research due to its complexity and importance in various research areas.

One of the latest survey in this domain was published in 2018 [15]. One of their finding is that a combination of linguistic and statistical approaches provides better results and that data preprocessing techniques play a crucial role in the performance of final ontology. Moreover, none of the projects could fully replace human-based evaluation of ontologies with a new approach. The authors provide comprehensive information about the major challenges in ontology learning and also propose some solutions for further improvement of each phase. In order to validate some of those conclusions, we will have a closer look at recent advances in ontology learning systems and techniques:

OntoLearn [16], [17], [18], [19] is one of the prominent ontology learning systems, which is based on both linguistic and statistical approaches for discovering domain terms, concepts and relations. Term extraction has been achieved through part of speech tagging and sentence parsing in order to produce syntactic structure of noun phrases and propositional phrases. The next step which is called semantic interpretation, performs concept and glossary extraction by benefiting from WordNet. Generated terms are associated to existing concepts and their corresponding definitions are extracted from WordNet. Finally, taxonomic relations among concepts are established by discovering their respective hypernyms in WordNet. The evaluation process of the constructed ontology is done through the performance measure called F-measures. F-measure (also known as F_1 score or F-score) measures the accuracy of a classifier by considering the harmonic average of precision and recall for computing the final score (see [20] for more information). In their study, they estimated precision and recall of the constructed ontology by using manually created tourism corpora.

Text-to-Onto discussed in [21], [22], [23], [24] follows a semi automated approach for constructing ontologies. The authors claimed that their proposed approach is applied to tourism and insurance case study data. Likewise *OntoLearn*, *Text-to-Onto* also employs both statistical and linguistic approaches for term extraction, concept formation and hierarchy construction. It however considers both taxonomical and non-taxonomical relation extraction between generated concepts. For the purpose of term extraction, HTML and pdf documents are converted into plain text and part of speech tagging is used for syntactic parsing of sentences. In order to identify important sentences, they have used a syntactic structure analysis. However, this tool employs a natural language processing system called Saarbruecken Message Extraction System (SMES) [25]. Concept forming is achieved through the same process as explained in

OntoLearn with a major difference in its background knowledge. For doing so, they have used a domain-specific knowledge base which contains around 120,000 terms. Taxonomic relation extraction is also very similar to *OntoLearn* with an extra step of using lexico-syntactic patterns for identifying hypernyms relation in the text. Association rule mining is applied in order to generate non-taxonomical relations by identifying potential connection among concept with user-defined support and confidence. In [26], authors used *Text2Onto* for the main ontology learning steps (concepts, taxonomies, relations and axioms extraction) from legal documents.

A semi-automated ontology learning system is also introduced by the Laboratoire d'automatique Documentaire et Linguistiques de L'université de Paris named **ASIUM** [27], [28], [29]. It works very similarly to previously described systems by employing both statistical and linguistic techniques across all ontology learning phases. Their proposed method has been tested on texts being extracted from the French newspaper *Le Monde*. The evaluation process of the constructed ontology is then performed by two domain experts where they assessed the extracted information by their correctness.

Other similar ontology learning systems are **SYNDIKATE** [30], [31] and **TextStrom/Clouds** [32], [33]. The former applies a purely linguistic based approach and the later employs both logic and linguistics techniques for discovering various components of an ontology from text.

As mentioned in the conducted ontology learning surveys [13], [12], [11], [34] and [15], the main focus of the ontology learning community lies on 1) improving relation acquisition, 2) labelling domain concepts and relations in an automatic or a semi-automatic way, and 3) benefiting from available structured and unstructured data on the Web for relation acquisition. Researchers have consistently studied the relation acquisition process. In [35], the authors proposed a relation extraction approach which aims to identify the semantic relations between extracted concepts from input text. In this study, the authors applied NLP techniques such as lemmatization, syntactic parsing, part-of-speech tagging, named entity recognition and word sense disambiguation patterns for analysing the input text. Generated named entities are semantically attached with their corresponding properties. Relation extraction in [36] is performed through syntactic dependencies. Dependency paths among terms are captured as by using bi-grams and their corresponding scores are calculated via statistical measures of correlation. Their proposed method is applied to the GENIA corpus [37]. They also applied an in-depth analysis of processes for obtaining abstract concepts by the means of *Selection Restriction Learning* [38]. Besides that, in [39] and [40], extraction of taxonomic relations is achieved by means of different types of lexico-syntactic patterns. Moreover in [41], [42], the authors presented a supervised approach which automatically identifies unknown hypernyms based on a group of known hypernyms being collected in sets of text. Their proposed method employs extracted dependency paths from parse trees.

[43] is also another recent study where the authors applied Google's word2vec to emulate an ontology learning system. Their approach, however, doesn't recognize non-taxonomical relations and only supports term/concept extraction and taxonomy generation. In the context

of automatic concept and relation labeling, [44] proposed a method for non-taxonomical relation extraction with the support of background knowledge and semantically tagged corpus. In this study, the authors argued that extracting verbs from a part-of-speech tagging process provides qualified information for relation labeling. Their technique has been implemented as an extension to the aforementioned ontology learning tool called *Text-to-Onto*. On the other hand, in [45], the author proposed a semi-automated technique for identifying concepts. Once domain concepts are extracted, they have been labelled with the support of previously provided set of seed terms. Another similar study has been conducted for generating domain concepts and relations by employing statistical semantic parsing [46]. In this study, several machine learning algorithms are applied for role extraction between the noun compounds being generated from the input text. The authors also tackled the problem of attaching semantic relations to the noun compounds. For this aim, primarily, multiple semantic relations are identified for a collection of named compounds, then a classification algorithm is described for automatic categorization of the relations. The proposed method, however is applied only on small amounts of data in the domain of bioscience text. In [47], the authors proposed the use of seed words as a guidance for ontology learning processes. One of the recent studies in this domain is conducted in [48], where Fraga and Vegetti generated a text file with a set of seed words to simplify the concept extraction process. Ontologies Translational Medicine Ontology ([49]), is a good example of applications of ontology learning based on web data in eHealth domain. [50], is another recent study in this domain where authors designed a framework which enables concept formation and hypernym/hyponym extraction.

On the side of benefiting from Web data in the process of relation acquisition, [4] proposed a method for the automatic acquisition of hyponyms and hypernyms for any term using search engine and natural language processing techniques. Their approach follows 3 main steps: The first step relates to the construction of a set of queries based on the patterns such as *term1 is a term2* being generated from term pairs. In the second step, the corpus is constructed based on the web-pages extracted by the search engine. Finally by employing sentence parsing and syntactic structure analysis, their method achieved to extract taxonomic relation between the terms. [51] also proposed a method for non-taxonomic relation extraction using Web data. Based on their method, domain patterns can be learned through relevant verb phrases being extracted from web-pages. The list of domain patterns is then used for labelling non-taxonomic relations. In another study [52], the authors profited from the structured Web data for constructing ontologies. Their proposed approach employs Wikipedia as a background knowledge to generate relations between domain concepts. A similar method has been proposed in [2] where Wikipedia plays a vital role in triplet extraction. Their technique, which is called *Catriple*, automatically extracts *category-property* and *category-value* pair type expressions (e.g. *"Category:Books by writer"* and *"Category:William Shakespeare"* where *"Books"* is a property and *"William Shakespeare"* is a value). Similar to other approaches, explicit and implicit properties are extracted through sentence parsing and syntactic rule analysis. In [53], the authors proposed an ontology learning method called **ISOLDE** (stands for Information System for Ontology Learning and Domain Exploration) which operates on sets of

manually created text corpora, structured data from the Web (such as Wikipedia, Wiktionary and a known German online dictionary called DWDS), and a named entity tagger.

1.4 Overview of the Thesis

The purpose of this thesis is to effectively decrease human involvement in the ontology learning process to the greatest possible extent. We propose a pipeline which automate the discovery of domain concepts, their corresponding individuals, as well as coarse-grained relations (both taxonomic and non-taxonomic). Besides performing standard NLP techniques, we applied an extremely powerful mathematical technique named *Formal Concept Analysis (FCA)* for constructing the core components of the ontology. Moreover, the constructed ontology is enhanced by means of an absolute knowledge base which has been proposed by our domain experts. The effectiveness of our proposed method is assessed by its ability to satisfy the following requirements:

1. Constructing a domain specific ontology with the minimum human effort.
2. Avoiding using any pre-built ontology and relying entirely on constructing an ontology from scratch by benefiting from the strength of FCA generated structures.
3. The proposed mapping rules should have the potential for converting any FCA structure into an ontological format.

In this thesis, we have addressed the following problems by taking advantage of FCA as the core component of the ontology learning process as well as using a domain specific reliable data source called *OpenFDA*:

1. Inefficiency of current NLP techniques for extracting domain specific named entities.
2. Difficulty in cleaning noisy text due to lack of integrated techniques and comprehensive knowledge about domain of interest.

The ultimate goal of this thesis is to automate all phases of the ontology learning process by means of linguistic (i.e. NLP) as well as statistical (FCA) techniques. The overall overview of our proposed ontology learning process is shown in Figure 1.1 and a detailed explanation of each phase is provided in each chapter of this thesis.

In chapter 2, we provide a comprehensive overview about applied NLP techniques and their role in constructing coarse-grained ontological concepts, individuals as well as non-taxonomic relations. Our ontology learning process initiates with analysing text corpora for the purpose of extracting the most important contents by producing its abstraction view. It comprises four main stages such as *format conversion, text cleaning, text processing* and generating abstraction

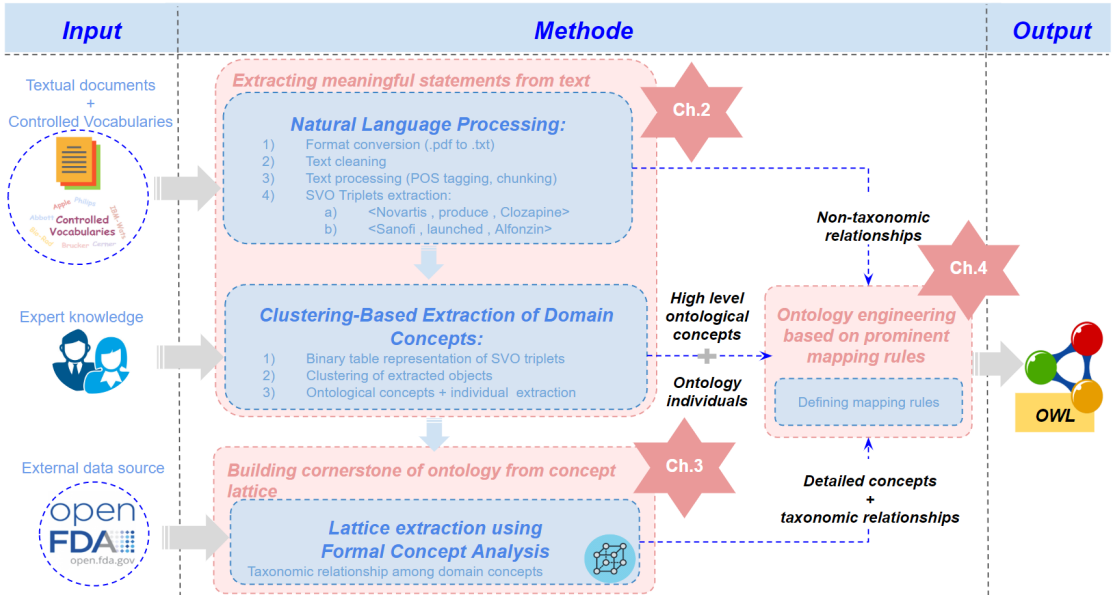


Figure 1.1 – *An overview of different phases in the proposed approach for ontology learning from textual data and how the details of each phase are outlined in each chapter of this dissertation.*

view in the form of *XML* document. The first box called *"Natural Language Processing"* is fed by textual data along with set of seed terms being called controlled vocabularies, which led us to extract the valuable content. As the output, it generates triplets of the form <*subject, verb, object*> from those sentences containing at least one of the seed terms, which are then used to assert non-taxonomic relations among ontological concepts. Furthermore, this chapter also provides a complete explanation of the process of pulling out domain concepts as their respective individuals. The corresponding box called *"Clustering-Based Extraction of Domain Concepts"* performs combinations of standard data analysis and machine learning techniques to generate the desired output. Correctness of the generated outputs is verified by the support of domain experts.

Once high level ontology components are extracted, chapter 3 illustrates FCA's vital role in completing its missing pieces, such as taxonomic relations among sub-classes and asserting their property axioms. A review of FCA's prominent techniques for fetching domain concepts and relations is conducted. Despite the fact that utilizing other statistical techniques such as co-occurrence analysis, latent semantic analysis, hierarchical clustering, association rule mining, term subsumption, etc. is prominent for the sake of ontology learning from unstructured data, there is a little work on employing FCA for ontology construction.

Lastly, chapter 4 presents our defined mapping rules for collecting pieces of ontologies which were generated during the previous steps. A bottom-up approach was used for designing the domain ontologies, therefore, the most specific concepts and their properties generated by FCA's special type lattice are firstly asserted into the ontology. To the best of our knowledge, this is the first effort for clearly defining sets of mapping rules for converting FCA lattice into

ontological components in the context of textual data. Ontology high level classes and their corresponding individuals are then gathered based on the approach described in chapter 2 and non-taxonomic relationships among them are asserted by scanning the triplets of the XML file being generated from NLP analysis.

After the core content, we also elaborate in chapter 4 on the implementation details of proposed approach by a validation of the engineered ontology in a case study in the pharmaceutical domain by analysing a query-answering approach. Finally in chapter 5, we discuss our conclusions and summarize the potential ideas for future work.

The bibliographical details of published papers of this dissertation is outlined below:

1. Jabbari, S., & Stoffel, K. (2018, December). A Methodology for Extracting Knowledge about Controlled Vocabularies from Textual Data using FCA-Based Ontology Engineering. In 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 1657-1661). IEEE. [54]

2. Jabbari, S., & Stoffel, K. (2018, December). FCA-Based Ontology Learning from Unstructured Textual Data. In International Conference on Mining Intelligence and Knowledge Exploration (pp. 1-10). Springer, Cham. [55]

3. Jabbari, S., & Stoffel, K. (2017, October). Ontology extraction from MongoDB using formal concept analysis. In 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA) (pp. 178-182). IEEE. [56]

4. Jabbari, S., & Stoffel, K. A Hybrid Algorithm for Generating Formal Concepts and Building Concept Lattice Using NextClosure and Nourine Algorithms.In international Workshop on Algorithms for FCA and Data Mining (AFCADM 2016), Moscow, Russia. [57]

5. Jabbari, S., & Stoffel, K. Parallel Execution of Binary-Based NextClosure Algorithm. In international Workshop on Algorithms for FCA and Data Mining (AFCADM 2016), Moscow, Russia. [58]

2 Extracting Meaningful Statements from Text

"You can dig out facts, but if you don't go deep enough, the facts won't tell you much."

- Jeff Catlin

2.1 Overview

As demonstrated in the ontology learning pipeline (in Fig. 1.1), the completeness and integrity of an acquired ontology is highly dependent on the the quality of the output generated by the text mining box. Therefore, applying significant linguistic techniques is a prerequisite for generating ontological concepts and relationships. In this section we are going to discuss about various natural language processing techniques which have been used in this research.

Figure 1.1 summarizes different steps required to produce an *.xml* file containing the most demanding and informative sentences from unstructured text documents. The unstructured text is parsed into *.txt* format using *Apache Tika* and the text cleaning process is done using well known Python libraries for natural language processing. Then, sentence tokenization and word tokenization are applied to all sentences as preliminary steps for part-of-speech tagging in order to extract grammatical information from sentences. The information is then used for chunking noun phrases (NP) and verb phrases (VP) in the form of *NP-VP-NP*. Finally, the set of chunked phrases is converted into *.xml* format to be imported into the ontology learning box. In this chapter, we discuss about the main linguistic techniques which have been used in our ontology learning process.

2.2 Natural Language Processing (NLP)

2.2.1 NLP: Background and Context

Natural Language Processing (NLP) has originated approximately seven decades ago as a branch of computer science and artificial intelligence that enables computers to process and analyze large amounts of natural language data. Moreover, NLP techniques play a key role in semantic knowledge extraction and reasoning from a domain of interest by deriving meaning from human languages.

Nowadays, there is a rapid growth in demand for knowledge extraction from unstructured data. Therefore, proper approaches and techniques are required to perform in-depth research for extracting valuable information. Various methods are employed on different types of documents in order to extract some knowledge from free texts. In [59], [60], [61] and [62], the authors provide a complete overview of information retrieval tools and techniques being used in the biomedical domain. Among all, famous biomedical IR/IE systems are listed as MEANS [63],*iHOP* [64], *EBIMed* [65], *GoPubMed* [66], *PubFinder* [67] and *Textpresso* [68].

In the domain of information extraction from clinical documents, NLP techniques facilitate automated terminology management, prediction analysis of the progress of various chronic diseases and side effect analysis of disease treatment. For instance, in [69] the authors designed an open-source NLP system called cTAKES (stands for "clinical Text Analysis and Knowledge Extraction System") for IE from electronic medical records which is developed based on existing information management architecture frameworks and the *OpenNLP* toolkit to create linguistic and semantic annotations. Another study ([70]) has been conducted to develop an information extraction system with the ability of identifying events (e.g. medical concepts), temporal expressions (e.g. dates associated with events), and their corresponding temporal relations in clinical text. The authors claimed that their proposed hybrid approach as a combination of rule-based and machine learning techniques could achieve encouraging results for information extraction from clinical texts. Moreover, other studies such as [71], [72], [73], [71], [74] are among the recent attempts of benefiting from NLP in medical data analysis.

Information extraction from unstructured text by means of NLP techniques is not just applicable to clinical and biomedical domains but is also intensively used for any online applications. [75] covers a comprehensive study of information retrieval technologies, information extraction and text categorization and highlights both basic and mathematical background in NLP. However, in online applications such as processing the content of web pages, standard NLP techniques are not sufficient to generate accurate results. In [76], the authors proposed an approach to transform text into a formal representation equivalent to relational database entries. Their approach is based on a preprocessor called *Webfoot* and an existing NLP system (*CRYSTAL*) to parse web-pages into logically coherent segments, and learn text extraction rules from examples. Similar studies in this domain are [77] and [78].

As mentioned in [79] and [80], the common approach being used in any typical information

extraction system based on NLP techniques consists of lower level and higher level analysis. Lower level analysis consists of tokenization, part of speech tagging, chunking (shallow parser) and named entity recognition. Higher level analysis such as morphological analysis, template extraction, template combination etc. require careful modeling based on the application domain.

The study of NLP began in 1950's with machine translation systems where Donald Booth and William Locke discussed around feasibility of translating the essential content of documents in languages which are foreign to the reader using techniques developed in World War II to break enemy codes [81]. In the 1950's, NLP intensely evolved from an automatic to a statistical machine translation system which is highly dependent on statistical models with parameters being extracted from the analysis of bilingual text corpora [82], [83], [84], [83].

During the 1960's, there was a huge interest in developing NLP systems such as *SHRDLU*, with the capability of answering questions, executing commands and accepting information with restricted vocabularies [85]. Another system, called *ELIZA* is developed as a computer program for the study of natural language communication between man and machine by means of pattern matching techniques [86]. Since then, there have been many studies in this domain such as [87], [88], [89], [90] and [91].

NLP systems have been extensively used in filling the gap among real word unstructured information and computer understandable data. Some notably successful progress occurred in 1970's in designing conceptual ontologies such as TaleSpin [92], QUALM [93], PAM [94] and Plot Units [95]. At that time, many programming languages were also invented for NLP applications. *Prolog* was one of those languages with rich set of syntaxes for writing grammars and parsing input data in a top-down approach [96].

In 1980's, new machine learning algorithms from the world of computer science brought a revolution in natural language processing. During this decade, some basic machine learning algorithms such as decision trees could replace complex sets of hand-written rules [85], [97] by similar hard *"if-then"* rules. On the other hand, statistical models improved the performance of existing NLP systems while dealing with unfamiliar and imprecise real world data. IBM research widely contributed to developing complicated statistical models with the goal of introducing more adaptable, robust and user friendly NLP systems. Later on, focus shifted towards inventing new machine learning algorithms which support unsupervised or semi-supervised learning approach and could handle non-annotated real word data such as content of World Wide Web.

Nowadays, deep neural network based approaches [98] play a key role in many natural language processing tasks such as language modelling [99] and parsing [100], [101]. Additionally, neural network architectures simplify the process of extracting semantically similar words and their roles in the sentence from the unstructured text corpora.

Currently NLP is heavily interconnected with information retrieval and there is has been a

huge interest in this field from many researchers and developers from diverse domains during last 50 years. "Ask Me Anything" is one of the recent studies being conducted in [102] in which a type of neural network architecture, called dynamic memory network (DMN), is used to process input sequences and questions, and produces relevant answers. Another relevant study is [103], where the authors proposed a method for solving the problem of language model smoothing in the context of information retrieval. Their proposed method primarily estimates a language model for each document. Subsequently, document ranking is obtained based on the likelihood of the query to the respective language model.

2.2.2 NLP: Tricks and Techniques

NLP techniques are applicable to almost all types of unstructured text documents. In this section, we briefly introduce primary NLP techniques such as *tokenization*, *noisy word removal*, *parsing* and *part-of-speech tagging*, *lemmatization* and *chunking*. It should be emphasized that a tailored pipeline of NLP techniques could be applied depending on the application domain. For instance, in keyword-based document classification, the text cleaning process (such as stop words and punctuation removal) might be essential to achieve a more accurate result. Conversely, retaining punctuation marks such as :) is required for the purpose of sentiment analysis as they carry valuable emotional information.

- **Tokenization**: this is the process of breaking a text into sentences, words, punctuation marks and other meaningful elements, which is prerequisite for the next processing steps.

  Although, it seems an easy task to split the text using white spaces, tabs, newlines or any similar characters, it is of great importance to consider that the syntax is not strictly defined in natural languages and the same character might be used for various purposes. This gets more challenging when we are analyzing biomedical text in which specific expressions and drug names contain hyphens, forward slashes and any other type of token boundaries (i.e. *"15 mg/day"*, *"HER2./neu"*). In this situations, the tokenization process becomes a challenging task and using any type of end line symbols, like full stops, does not provide highly accurate results.

  In [104], the authors discriminated among tokenization approaches for *space-delimited* languages and *unsegmented* languages. Based on their study, in *space-delimited* languages, such as most European languages, words boundaries are mainly indicated by white-spaces. Hence, any word being followed by space, could be considered as an individual token. This approach successfully splits words constructed as sequences of alphabetic characters. However, for *unsegmented* languages, since words boundaries are not space-delimited, approaches other than simple lexical analysis are required. Therefore, word tokenization requires more lexical and morphological information. To the best of our knowledge, no general solution has been developed with high accurate precision. However, some algorithms have been developed to obtain an overall

approximation for different languages.

The following box, demonstrates an example of a sentence and its corresponding tokenized form:

Trastuzumab; is treatment for all <HER2 positive+ > & breast cancer.
DisplayName is: Herceptine
https://en.wikipedia.org/wiki/Trastuzumab →

```
['Trastuzumab', ';', 'is', 'treatment', 'for', 'all', '<', 'HER2',
'positive+', '>', '&', 'breast', 'cancer', '.', 'DisplayName', 'is',
':', 'Herceptine', 'https://en.wikipedia.org/wiki/Trastuzumab']
```

- **Text Cleaning and Text Simplification**: The quality of the input textual data might be extremely poor, therefore several preprocessing steps may be required to prepare more appropriate text corpora. In the following, we briefly introduce some of potential techniques being used for treating noisy sentences while keeping their core concepts and meanings:

  1. *Stopwords removal:* Stopwords are words that are frequently used in any text document. Some examples of stopwords are *"of", "are", "the", " it", "is", etc.* Depending on the application domain, more words can be added to or removed from the list of stopwords. For instance, document classification or document indexing is highly dependent on extracting keywords from the provided documents. Hence, removing common stopwords, modal verbs such as *"would", "could", "should"*, and human expressions such as *"lol , "wow"* has a huge effect on document simplification for the successive NLP steps. On the other hand, in *sentiment analysis*, which has application in many domains such as marketing or customer relationship management, NLP techniques must be applied on comments of reviewers or surveys of responses with the objective of extracting their expressed opinion in a domain of interest. Hence, some human expressions such as *"hahaha", "lol", "brb", etc.* are informative and should not be removed from the original text.

     There are two common approaches for stopwords removal from text corpora. The first approach is relaying on the frequency of a word within a document. Hence, all words occurrences must be counted in the given corpus. Words appearing in excess of a given threshold are then removed. The other approach is to use a predefined list of stopwords defined in language specific libraries. *Stopwords Corpus* is available in the *Natural Language Toolkit (NLTK)* [105] which contains 2400 stopwords for 11 languages. Figure 2.1 demonstrates list of *NLTK's* stopwords for English:

     Removing stopwords from the previous sentence results in:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Figure 2.1 – *English stopwords in NLTK.*

```
['Trastuzumab', ';', 'treatment', '<', 'HER2', 'positive+', '>',
'&', 'breast', 'cancer', '.', 'DisplayName', ':', 'Herceptine',
'https://en.wikipedia.org/wiki/Trastuzumab']
```

2. *Punctuation removal:* In linguistics, punctuation marks are a set of marks such as comma, full stop sign, question mark, apostrophes, brackets and any other type of non-lexical orthography that are used in writing to clarify the meaning. In general, as discussed in [106], there are three types of punctuation marks: delimiting (comma, dash, parenthesis), separating and disambiguating. Some punctuations such as comma, could fall into multiple categories based on their role in the sentence.

   How to deal with punctuation marks depends on the application domain. As discussed in [107], punctuation marks play a crucial role in syntactic processing and not all of them must be removed from the input text. For instance, consider the application of sentiment analysis to marketing based on the classification of customers reviews about a specific product. To do so, considering both human generated text as well as emotion icons such as *:D, :), :(* is crucial to correctly differentiate between very happy, happy and sad. Therefore, removing such punctuation marks would heavily impact the accuracy of sentiment analysis. On the other hand, in text summarization, where the ultimate goal is to retain the core content and to create a coherent summary of the main points, removing the aforementioned punctuation marks is crucial.

   Most of natural language processing packages (like *NLTK*), are trained to distinguish among different roles of a punctuation mark within a sentence. Example of that is the dot sign that might have different roles; the first role is indicating the end of a sentence and the second role is to mark an abbreviation, such as *"E.g."* and the main challenge is to recognize that the dot sign in *"E.g."* is not a sentence separation mark (*"E."* and *"g"*) but it indicates an abbreviation. *Punkt tokenizer* is a useful *NLTK* tokenizer which has been trained to identify abbreviations [108]. In the following box, punctuation marks such as <, +, &, *, ; are removed:

```
['Trastuzumab', 'treatment', 'HER2', 'positive+',
'breast', 'cancer', 'DisplayName', 'Herceptine',
'https://en.wikipedia.org/wiki/Trastuzumab']
```

3. *Removal of URLs and email addresses:* Many URLs, hyperlinks and email addresses exists in text documents extracted from World Wide Web. Depending on the required criteria for the type of information to be extracted in a specific domain, one might require to remove urls and hyperlinks. In the following, this process is applied by means of a Python's regular expression pattern defined as:
`"r = re.compile(r"(https?:(?:[-\ w.]|(?:%[\ da-fA-F]2))+)")".`

```
['Trastuzumab','treatment', 'HER2', 'positive+', 'breast',
'cancer', 'DisplayName', 'Herceptine']
```

4. *Other data cleaning processes:* There exist many other types of data cleaning steps such as *"splitting attached words" , "slang words removal", "words standardization", etc.* which might be required depending on how noisy the input data is. For instance, in the application of social media analysis, natural language processing must deal with completely informal text data being generated by humans in social forums such as Twitter. In social media like Twitter, attaching a hash tag to any word, is a common expression that people use to categorize some Tweets. Words such as *"#HappyLife"* and *"#RainyDay"* can be splitted into their normal forms using simple rules and regular expressions. In Python, a short line of code`" ".join(re.findall('[A-Z][^A-Z]*', text))` could extract all attached words in *"text"* and join them by a space sign.

   Additionally, preprocessing is also needed to normalize words and convert them into their proper form. Words for expressing human feelings mainly contain longer characters such as *"Soooooooo Saaaaad"* and defining regular expressions in the form of`''.join(''.join(s)[:1] for _, s in itertools.groupby(text))` converts it to its normal form *"So Sad"*.

   Moreover, depending on the application domain, formatting characters must also be treated differently. Typically, retaining formatting characters such as HTML tags (e.g. *"&lt, &gt, &amp"*) does not provide valuable information for understanding and analyzing the text. Therefore, removing these components could noticeably simplify the sentences and increase the quality of input text.

- **Parsing** and **Part-of-Speech tagging**: Text parsing enables syntactic analysis that explores various dependencies among words in a sentence and represents them in the form of a data structure called *parsing tree*. The most famous tools for parsing sentences are Natural Language Toolkit (NLTK), Stanford Parser [109], General Architecture for Text Engineering (GATE) [110], Principar [111]), Minipar [112] and Link Grammar Parser [113].

  Part-of-speech (POS) tagging on the other hand, is the process of labeling words with

their corresponding part of speech which indicates their syntactic role, for example, noun, verb, adverb, etc. Many studies have been conducted to increase the accuracy of *POS-tagging* process using both rule-based and statistical-based approach. In [114] authors applied inductive learning of statistical decision trees to part-of-speech tagging in natural language processing. They claimed that their system outperforms the previous models, especially when dealing with unknown data. In [115], a statistically based approach is introduced for designing a new tagger, called *Trigrams'n'Tags (TnT)*, based on Markov models. [116] also proposed a statistical model (based on Maximum Entropy) which simultaneously uses many contextual features to predict the POS-tag of words. Conversely, the authors of [117] applied a rule-based approach which automatically acquires the rules and tags the words with comparable accuracy to stochastic taggers. Brill Tagger [117] and TreeTagger [114] are also among famous part of speech taggers which have been used in various domains.

Machine learning algorithms have been immensely used in old and still state of the art POS tagging process. In [118], the authors applied a specific type of recurrent neural network called bi-directional LSTM (Long Short-Term Memory) ([119]) for POS tagging. Their proposed method is evaluated across 22 languages and its performance is measured at different levels of granularity. Compared to its counterparts such as CRF and HMM taggers [120], their model is less robust to label noise. Another study is conducted in [121] where the authors applied weightless artificial neural networks ([122]) for accelerating the POS tagging. Despite the accuracy of their proposed approach, which matches or outperforms the-state-of the art methods, it doesn't successfully tag the languages such as Arabic, Hebrew and Syriac. Besides aforementioned techniques, rule-based POS tagging is used in [123], where the focus is to apply a set of predefined rules for tagging the words.

For instance, the following output is generated by importing the sentence *"Trastuzumab is treatment for HER2 positive breast cancer."* and running the *NLTK's* parts-of-speech tagging module. Please note that it is not necessarily a subsequent step after text cleaning and, depending on the use case, one may need to design a sequence of appropriate text mining operators to extract required information. Part of speech tagging tremendously helps to identify the role of each word (i.e., whether it is a name or verb) in a given sentence.

```
[('Trastuzumab', 'NNP'), ('is', 'VBZ'), ('treatment', 'NN'), ('for',
'IN') , ('HER2', 'NNP'), ('positive', 'JJ'), ('breast', 'NN'),
('cancer', 'NN'), ('.', '.')]
```

As shown above, the tag-set of the input sentence mostly contains nouns indicated by *NN* and *NNP*. Let's use another example with more diverse types of part of speech tags as well as some homonyms:

Word	POS-Tag	Stands for...
After	IN	preposition/subordinating conjunction
the	DT	determiner
heart	NN	noun, singular
attack	NN	noun, singular
he	PRP	personal pronoun (I, he, she)
decided	VBD	verb, past tense
to	TO	to
attack	VB	verb, base form
his	PRP$	possessive pronoun (my, his, hers)
cancer	NN	noun, singular
with	IN	preposition/subordinating conjunction
medication	NN	noun, singular

Table 2.1 – *Words and their corresponding pos-tag.*

After the heart attack, he decided to attack his cancer with medication

```
[('After', 'IN'), ('the', 'DT'), ('heart', 'NN'), ('attack',
'NN'), (',', ','), ('he', 'PRP'), ('decided', 'VBD'), ('to', 'TO'),
('attack', 'VB'), ('his', 'PRP$'), ('cancer', 'NN'), ('with', 'IN'),
('medication', 'NN')]
```

Parts of speech tagging can be important for syntactic and semantic analysis. Therefore, for something like the sentence above, the word *attack* has different roles and meanings. The first *attack* in the above sentence represents a disease symptom (in the context of heart attack), while the second *attack* indicates the verb meaning resisting against cancer. In other words, the same word appears in two different roles, i.e., noun and verb. There are plenty of words being used under different meanings and different roles in a sentence that need to be distinguished during POS tagging. Handling such homonyms while pre-processing a text is crucial for the subsequent text analysis steps.

Table 2.1 summarizes the list of words and their corresponding part of speech tags in the above example. The following command lines provide a long list of available tag sets in *NLTK*:

```
>>> nltk.download('tagsets')
>>> nltk.help.upenn_tagset()
```

- **Lemmatization** and **Stemming**: In natural language processing, converting the tokens into their root forms (i.e., stems) reduces the complexity of text by handling various morphological variants of one term. Even though the basic functionality of lemmatization and stemming methods, which do such conversion, is the same, they have slightly different approaches to do so. In the *stemming* process, the *stem* form of the

word is mostly acquired without having any knowledge about the context of the word occurrence, such as it's part of speech tag. To do so, some rules are defined to strip off anything that looks like a suffix from the end of the word. The Table 2.2 demonstrates some rules and examples from this approach.

Rules	**Examples**
SSES –>SS	Caresses –>Caress
IES –>I	Ponies –>Poni
SS –>SS	Caress –>Caress
S –>	Cats –>Cat
EED –>EE	Agreed –>Agree
ED –>	Plastered –>Plaster
ING –>	Monitoring –>Monitor
Y –>I	Happy –>Happi

Table 2.2 – *A few stemming rules and examples for conversion of words into their stems.*

Contrary to stemming, lemmatization gets the root form of the word considering it's part of speech tag within a sentence. For instance, the lemma of the word *"talking"* is *"talk"* which is matched in both stemming and lemmatization. However, the word *"meeting"* can be either the base form of a noun or a verb (*"to meet"*); unlike stemming, lemmatization will correctly select the appropriate lemma depending the word's part of speech tag in the sentence:

"in our last meeting": meeting → meeting
"We are meeting again tomorrow": meeting → meet

Like other natural language processing techniques, lemmatization and stemming are application dependent processes. In applications such as search engines and document clustering, which highly rely on terms and keywords, we should apply lemmatization or stemming. However, in other type of applications, where acquiring semantic information is also valuable, these steps must be dropped. As an example, the attached affix *"er"* and *"est"* to the adjectives makes them have different semantic meanings (i.e. *"slow , slower, slowest"* have semantically different meaning). Considering the fact that the stemmed form of a verb might not be an actual word, in this study we considered the lemmatized form of the verb for the task of non-taxonomical relation acquisition in ontology learning process.

To better understand the differences between a stemmer and a lemmatizer, the outcome of stemming and lemmatizing for the following text is provided using the *WordNet lemmatizer* in Python's *NLTK* library. The *WordNet lemmatizer* provides a list of valid lemmas for all words within a text.

"Roche is a Swiss multinational healthcare company that operates worldwide under two divisions: Pharmaceuticals and Diagnostics. It controls the American biotechnology company Genentech, which is a wholly owned affiliate."

Lemmatized form:

```
['Roche', 'is', 'a', 'Swiss', 'multinational', 'healthcare',
'company', 'that', 'operates', 'worldwide', 'under', 'two',
'division', ':', 'Pharmaceuticals', 'and', 'Diagnostics', '.',
'It', 'control', 'the', 'American', 'biotechnology', 'company',
'Genentech', ',', 'which', 'is', 'a', 'wholly', 'owned',
'affiliate', '.']
```

Stemmed form:

```
['roch', 'is', 'a', 'swiss', 'multin', 'healthcar', 'compani',
'that', 'oper', 'worldwid', 'under', 'two', 'divis', ':',
'pharmaceut', 'and', 'diagnost', '.', 'It', 'control', 'the',
'american', 'biotechnolog', 'compani', 'genentech', ',', 'which',
'is', 'a', 'wholli', 'own', 'affili', '.']
```

- **Chunking**: Another step in natural language processing is called chunking (or shallow parsing), which is the process of splitting a sentence into syntactically correlated segments called chunks, or base phrases. Chunking uses a sentence with its part of speech tags as input and provides an output in the form of chunks. Each chunk consists of a few tokens in a row that constructs a meaningful phrase. Like part of speech tagging rules, there is a standard set of chunk tags such as *Noun Phrase (NP), Verb Phrase (VP),*

  As illustrated in Figure 2.2, chunking divides a sentence into segments, where each segment consists of a series of tokens and their corresponding POS tags. The segments being tagged as *NP*, contain a few tokens in a row and are called *Noun Phrase Chunk (NP)*. They have been extracted based on a defined rule. The other segments, containing a single token, are tagged with their part of speech only, and do not belong to any chunk box.

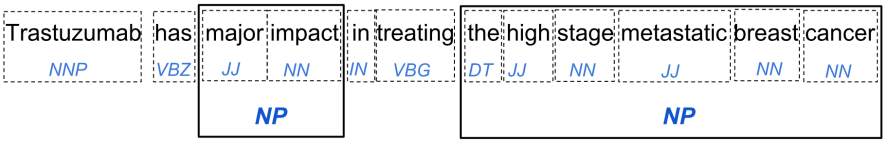


Figure 2.2 – *Noun Phrase (NP) chunking with NLTK*

NP-chunking deals with extracting chunks corresponding to a single noun phrase. To create an *NP-chunker* over one or multiple sentences, a *grammar* must be defined as shown in the following box:

```
grammar = "NP: {<DT>?<JJ><NN><JJ><NN>}"
```

For the sentence provided in Fig. 2.2, a chunking grammar is defined using a simple regular expression. The rule describes the NP chunk to be formed whenever the chunker finds an optional determiner (DT) followed by any number of adjectives (JJ), any number of nouns (NN), any number of adjectives (JJ) and then any number of nouns again (NN). Once a chunk parser is created with a given grammar on the sentence, the result which is in a tree form can be graphically displayed just like Fig. 2.3.

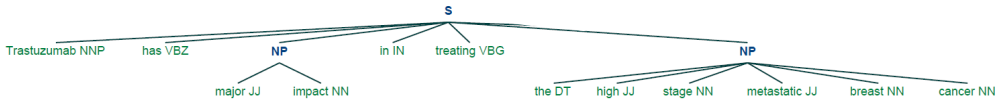


Figure 2.3 – *Noun Phrase (NP) chunking tree.*

It is also possible to define multiple chunking rules (also called "tag patterns") within a single regular expression. The following tag pattern (in the box below) consists of four rules. The first rule is defined for extracting a noun phrase *NP*. The second chunk contains prepositions followed by NP. The third rule is slightly different in that it contains a recursive rule based on other tag patterns which creates an in depth structure for the chunking tree. Finally, the last rule is the combination of predefined chunks to extract the part of the text being tagged as NP and continued as a VP. The tree in Fig. 2.4 shows the chunked form of the sentence *"Herceptin reduces platelet counts and causes liver problems"* based on the defined grammar.

```
grammar = r"""
NP: <DT|JJ|NN.*>+
PP: <IN><NP>
VP: <VB.*><NP|PP|CLAUSE>.*
CLAUSE: <NP><VP>
"""
```

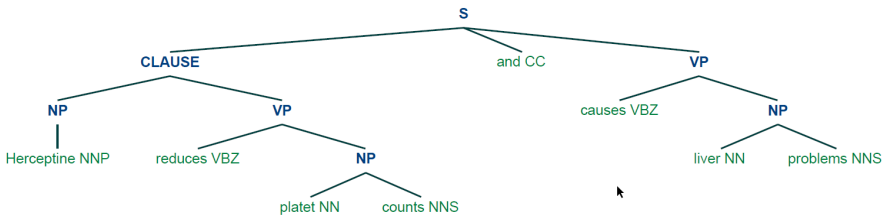


Figure 2.4 – *Recursive chunking tree.*

- **Named Entity Recognition *NER***: This is a process with the task of identifying the main categories and atomic elements of a sentence. These categories refer to specific types of entities such as persons, dates, locations, diseases, medications. NERs provide critical

Named Entity	Examples
Date	July, 1998-07-26
Facility	Brunswick
GPE	Asia, South East
Location	Rhine river
Oraganization	Roche
Money	500 million francs
Percent	Seventy pct, 67.45%
Person	President Obama
Time	15:30 p.m. , Midnight

Table 2.3 – *Commonly used named entity types.*

information for various NLP applications. For instance, assume that a text corpus has been provided as input for a Question Answering (QA) system, containing information regarding different treatments for breast cancer being launched by various companies. Now the question is *"Which company launched Herceptin first?"* and one of the documents in the corpus contain the right answer in the following sentence:

```
Herceptin side effects are reported as fever, infection, cough,
headache and trouble sleeping.  These side effects are reported
based on clinical trials being done in Roche that launched the
product in June 2011.
```

The answer for this question has the form of *"Company X launched Herceptin"* where *X* is a noun phrase which refers to a named entity of type *Company*. Knowing this fact extremely improves the precision of information retrieval system by narrowing down the search space to only those sentences with correct type of named entity, i.e., *Company*. This allows the system to ignore the first sentence in the retrieved document, since it only contains one named entity *Herceptin* with the type *Drug*.

Table 2.3 shows commonly used named entity types and examples. It is however possible to create human-made artifacts based on the application domain such as *GPE*: geo-political entities such as city, state/province, or country based on some geographical dictionary like *"gazetteer"* [124].

The main challenge in NER is that many terms are either ambiguous or not known in available dictionaries. For instance, the name *May* is likely to be part of a named entity of type *Date*, but could also belong to type *Person*. The word *Alexion*, on the other hand, is more likely to be of type *Person*, but needs to be marked as the type *Company* in *Alexion Pharmaceuticals* phrase.

Various powerful and rich NER libraries have been developed, among which the *spacy NER Model* is a Python library which provides an exceptionally efficient system for identifying pre-defined names entities such as *Company-name, Location, Organization,*

Product-name and the possibility of adding arbitrary artifacts to the entity recognition model. Likewise, the *Stanford Named Entity Recognizer* is a Java library which is effectively trained to distinguish different names entity types such as Organization, Person and Location in different languages.

As an example, the following text is imported into the spacy NER library. As illustrated in Figure 2.5, 6 types of names entity types are recognized by NER system and 10 words are tagged with their corresponding category. For instance *"Roche", "Genentech", "SIX Swiss Exchange"* are correctly tagged as *"Organization (ORG)"*. On the other, trained system could properly attach *"Person"* as names entity of *"Fritz Hoffmann La Roche"*.

```
Roche AG is a Swiss multinational healthcare company that operates
worldwide under two divisions:  Pharmaceuticals and Diagnostics.
Its holding company, Roche Holding AG, has bearer shares listed on
the SIX Swiss Exchange.  The company headquarters are located in
Basel.  Roche was founded in 1896 by Fritz Hoffmann La Roche and
controls the American biotechnology company Genentech.[125]
```

Figure 2.5 – *Extracted names entities using spacy.*

Training NER systems requires appropriate data in order to reach extremely high accurate results. Therefore, the crucial step is training the NER engine with appropriate data sources containing well defined named entity annotations. In [126], the authors summarized named entity recognition and classification algorithms ranging from rule-based systems, to statistical models and applied machine learning techniques such as Neural Nets. A comparison of NER tools is also studied in [127] in which the authors compared the existing NER tools in order to identify those suitable for the biographical texts processing domain.

As explained earlier, a pipeline of NLP techniques must be tailored and customized based on domain requirements. In our proposed method, textual documents play a significant role in the process of ontology learning, therefore choosing a proper set of NLP

techniques is of great importance. Considering that the output from the text processing step must be in the form of *subject, verb, object* triplets in order to be integrated into an ontology learning pipeline, the following combination of NLP techniques is employed. Firstly, we performed format conversion and text normalization with the purpose of eliminating noisy components from MCI documents retrieved from online resources. Moreover, in order to extract relevant triplets, we implemented POS tagging to capture syntactic relations among words. Subsequently, a specific chunking grammar of the form *NP, VP, NP* is applied on POS-tagged sentences. Lastly, further filtering is applied on the extracted triplets by considering only those triplets where either their subject or object was included in provided controlled vocabularies. It must be emphasized that all *VP*s are converted into their lemmatized form in order to be used as non-taxonomical relationship between generated subjects and their corresponding objects.

2.3 Application of NLP for Ontology Engineering

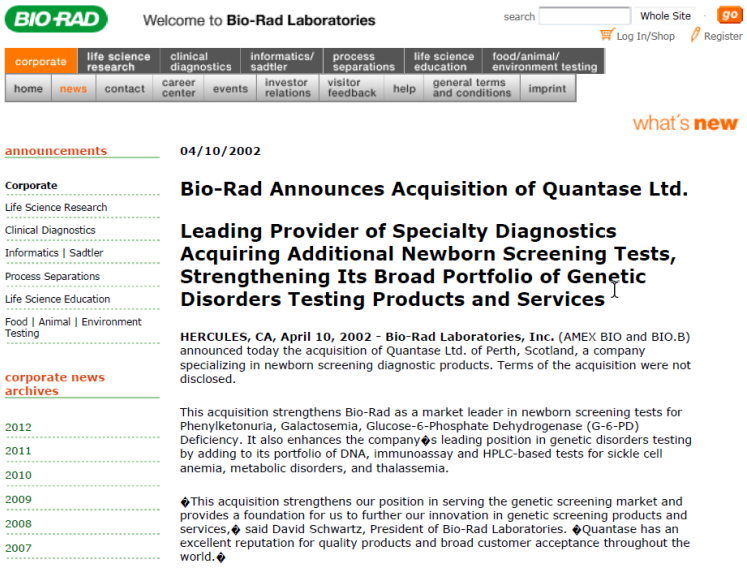
In this section we guide you, step by step, through the applied text processing pipeline which helps us giving some structure to unstructured data embedded in text documents. The output of text the processing step is stored in the form of triplets to enable further processing by other techniques discussed in the next chapters. The proposed approach for ontology engineering from textual data requires some preprocessing steps as discussed below:

2.3.1 Plain Text Extraction

Extracting plain text from input documents is prerequisite for further text analysis steps. For the purpose of corpus construction and prior to any text processing task, plain text extraction is first conducted to exploit relevant meta data by removing any image and extract pure domain text from the input documents ([128]). For doing so, a parser is required to analyze the document contents and convert all *.pdf* files to *.txt* format. *Apache Tika* [1] [129] which is a content analysis toolkit, perfectly fits this purpose by detecting and extracting meta data and text from different file types (such as *.ppt,.xls* and *.pdf*). *ApacheTika* has been also widely used in many other applications such as search engine indexing and translation.

For the sake of implementation, *"tika-python"* has been used as the port of *ApacheTika* in the Python environment. The provided example in Figure 2.6 demonstrates an example of the original *.pdf* file (Figure 2.6a) and its corresponding plain text (Figure 2.6b) as analyzed by *tika-python* library.

[1] https://tika.apache.org/



(a)

search Whole Site
Log In/Shop Register
announcements
Corporate
Life Science Research
Clinical Diagnostics
Informatics Sadtler
Process Separations
Life Science Education
Food Animal Environment
Testing
corporate news
archives
2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001 2000
04/10/2002
Bio Rad Announces Acquisition of Quantase Ltd.
Leading Provider of Specialty Diagnostics
Acquiring Additional Newborn Screening Tests,
Strengthening Its Broad Portfolio of Genetic
Disorders Testing Products and Services
HERCULES, CA, April 10, 2002 Bio Rad Laboratories, Inc. (AMEX BIO and BIO.B)
announced today the acquisition of Quantase Ltd. of Perth, Scotland, a company
specializing in newborn screening diagnostic products. Terms of the acquisition were not
disclosed. This acquisition strengthens Bio Rad as a market leader in newborn screening tests for
Phenylketonuria, Galactosemia, Glucose 6 Phosphate Dehydrogenase (G 6 PD)
Deficiency. It also enhances the company's leading position in genetic disorders testing
by adding to its portfolio of DNA, immunoassay and HPLC based tests for sickle cell
anemia, metabolic disorders, and thalassemia.
"This acquisition strengthens our position in serving the genetic screening market and
provides a foundation for us to further our innovation in genetic screening products and
services," said David Schwartz, President of Bio Rad Laboratories. "Quantase has an
excellent reputation for quality products and broad customer acceptance throughout the
world " The current market for newborn screening tests is estimated to be more than $100
million worldwide and is growing at a rate of approximately 10 percent per year.
Newborn genetic screening is compulsory in the United States and in most developed
nations around the world.
Bio Rad Laboratories, Inc. (is a multinational manufacturer and
distributor of life science research products and clinical diagnostics. It is based in
Hercules, California, and serves more than 70,000 research and industry customers
worldwide through a network of more than 30 wholly owned subsidiary offices.
Various statements made within this press release may constitute "forward looking
statements" for purposes of the Securities and Exchange Commission's "safe harbor"
provisions under the Private Securities Litigation Reform Act of 1995 and Rule 3b 6
under the Securities Exchange Act of 1934. The forward looking statements contained
herein involve risks and uncertainties that could cause results to differ materially from
the Company's expectations.
Ron Hutton, Treasurer
Bio Rad Laboratories, Inc.
Phone: 510 724 7000
E mail: investor_relations@bio rad.com
Trademarks • Site Terms • EU Recycle Program • Privacy • Feedback
Copyright © 2011 Bio Rad Laboratories, Inc. All rights reserved.
Page 1 of 1Bio Rad Laboratories
14.03.2012

(b)

Figure 2.6 – *Plain text extraction using content analysis Toolkit, (a) Original PDF file , (b) Extracted plain text using ApacheTika.*

2.3.2 Text Cleaning

Considering that the original pdf files are extracted from online sources such as news, blogs and published reports of companies, the quality of the generated plain text by *Tika* is somewhat noisy when the content contains external elements (captions, footnotes, etc.). Therefore, text cleaning is an essential prerequisite for constructing a high quality ontology.

Text cleaning is the term we used to describe the overall process of cleaning all irrelevant pieces of data which are captured in the corpus such as web links, email addresses, non-English sentences, etc. We have divided our text cleaning process into two major steps which are mainly accomplished with low level rule-based scripts:

1. *Language detection:* Language identification has been done in order to extract English text only. Text language detection is a key task in our pre-processing pipeline because its failing will nullify subsequent processes.

   Likewise any other NLP technique, in the context of language detection, certain softwares and libraries are well suited to certain languages. For instance, *NLTK* package has been strongly trained for English texts and *FreeLing* ([130]) is well trained for Spanish documents. *"langdetect"* ([131]) is a Python library which supports language detection using defined `"detect()"` function and supports 55 languages out of the box. Another Python package is called *"spaCy-CLD"* ([132]) which adds language detection to *spaCy's* text processing pipeline using defined function `LanguageDetector()`.

   For the purpose of detecting and considering all English documents, we have used a hybrid method by defining a customized function (called `lang_detector()`) and *"langdetect"* package. This method uses the powerful strength of both *langdetect* and *NLTK stop words* packages and accurately distinguish English texts by comparing the result of our customized, *NLTK* based function with output of *langdetect* and considers only those documents being detected as English by both methods.

   Figure 2.7 visualizes the main steps of stopwords based language detection approach. Once the the text is imported for language identification, its content gets tokenized by *NLTK* tokenizer. In the next step, the intersection of tokenized text with stopwords of 21 languages being defined in *NLTK* is calculated and the language with highest common words with input text is selected as the output. Additionally, an in-parallel language investigation process is also done using *langdetect* on the same input data in order to increase the accuracy of identified language for text. Finally the target language is selected based on a simple comparison of generated output from both language detection streams.

2. *Tokenization*: Tokenization, as a prerequisite step for any NLP process, is the task of splitting any document into sentences and then into words. From the view point of implementation, tokenization has robust capabilities compared to native Python splitting function (`".split()"`) such that it tokenizes a sentence into words and punctuations.

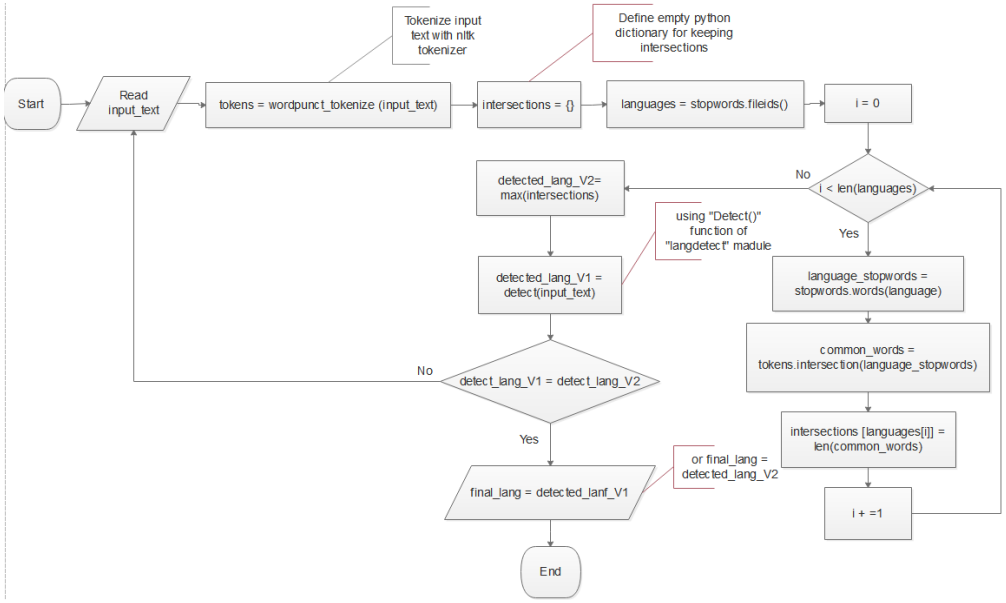


Figure 2.7 – *Customized method for text language detection.*

For instance, consider the following original sentence: *"Abbott Laboratories is an American health care company. Abbot's reported revenues in 2017 was* 27.39$ *billion"*. The following box demonstrates tokenized and splitted version of sentence above. As shown, tokenizer treats ,s , $, and · as separate tokens which is extremely crucial because counteractives have their own semantic meaning and parts of speech tags.

```
>> input_string = "Abbott Laboratories is an American health care
company.Abbott's reported revenues in 2017 was 27.39$ billion"

>> nltk_word_tokenize(input_string)
['Abbott','Laboratories', 'is', 'an', 'American', 'health',
'care', 'company, '.', 'Abbott', ''s', 'reported', 'revenues',
'in', '2017', 'was', '27.39', '$', 'billion']

>> input_string.split()
['Abbott','Laboratories', 'is', 'an', 'American', 'health',
'care', 'company, '.', 'Abbott's', 'reported', 'revenues', 'in',
'2017', 'was', '27.39$', 'billion']
```

Tokenization, as a fundamental step for syntactic analysis of a corpus, is isolated into sentence tokenization and word tokenization. Sentence tokenization is the process of splitting the text into separate sentences based on punctuations and characters. Many applications and natural language processing tools require their input to be divided into sentences for many reasons such as counting average number of words per sentence or part of speech tagging. Considering the fact that parts of speech tagging is in the

NLTK Tokenizer Name	*Python syntax*	*Output*
TreeankWordTokenizer	>>>from nltk.tokenize import TreebankWordTokenizer >>>tokenizer = TreeankWordTokenizer >>>tokenizer.tokenize ("The patient couldn't sleep after his first chemotherapy session")	['The', 'patient', 'could', "n't", 'sleep', 'after', 'his', 'first', 'chemotherapy', 'session']
PunktWordTokenizer	>>>from nltk.tokenize import PunktWordTokenizer >>>tokenizer = PunktWordTokenizer() >>>tokenizer.tokenize ("The patient couldn't sleep after his first chemotherapy session")	['The', 'patient', ' couldn' ', ' 't ', 'sleep', 'after', 'his', 'first', 'chemotherapy', 'session']
WordPunctTokenizer	>>>from nltk.tokenize import WordPunctTokenizer >>>tokenizer = WordPunctTokenizer () >>>tokenizer.tokenize ("The patient couldn't sleep after his first chemotherapy session")	['The', 'patient', 'couldn', "'", 't', 'sleep', 'after', 'his', 'first', 'chemotherapy', 'session']

Table 2.4 – *Syntax and output of Python NLTK Tokenizers on the same text.*

core of *NLP* processes for our proposed method, Python *NLTK's* sentence tokenizer (*"sent_tokenize()"*) is firstly applied on input text. This tokenizer is robustly trained to decide where to cut a sentence based on defined punctuations and characters.

Once the input text is tokenized in the level of sentences, word tokenization is required to break each sentence into words. Python *NLTK* provides multiple word tokenizers as shown in the inheritance tree in Figure 2.8 ([133]). As the name declares, *WhitespaceTokenizer* divides text at whitespaces. *TreeankWordTokenizer*, however, uses conventions being defined in *Penn Treebank* corpus [134] and it works by separating contractions. On the other hand, *PunctWordTokenizer* splits the text based on punctuation while keeping it with the word instead of separate tokens. Lastly, *WordPunctTokenizer* is slightly different such that it splits all punctuations into separate tokens. Table 2.4 clearly presents the difference among outputs of aforementioned tokenizers on the same input string.

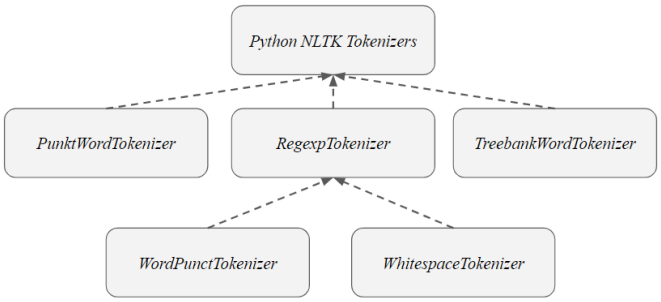


Figure 2.8 – *Python NLTK Tokenizers.*

PunktWordTokenizer is no longer exposed in newest versions of Python. *TreeankWordTokenizer* is selected to be applied on converted version of documents into plain text. It is shown in our exploratory analysis that *TreeankWordTokenizer* was more accurate compare to its counterparts for treating punctuation marks, periods and standard contractions as separate tokens. An example of tokenized plain text is provided in Figure 2.9. As shown in the result *"("*, *":"*), commas and single/double quotes being followed by white spaces are splitted accurately.

3. *Punctuation removal **and** Stop words elimination:* One of the major steps in text cleaning is to eliminate non-informative words and characters. Punctuations such as commas, apostrophes, quotes, question marks, and stop words like *"the"*, *"to"*, *"as"*, *"in"* don't have much contribution in overall meaning of a sentence. Hence, excluding these words could remarkably reduce the search space and considerably influence

search Whole Site
Log In/Shop Register
announcements
Corporate
Life Science Research
Clinical Diagnostics
Informatics Sadtler
Process Separations
Life Science Education
Food Animal Environment
Testing
corporate news
archives
2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001 2000
04/10/2002
Bio Rad Announces Acquisition of Quantase Ltd.
Leading Provider of Specialty Diagnostics
Acquiring Additional Newborn Screening Tests,
Strengthening Its Broad Portfolio of Genetic
Disorders Testing Products and Services
HERCULES, CA, April 10, 2002 Bio Rad Laboratories, Inc. (AMEX BIO and BIO.B) announced today the acquisition of Quantase Ltd. of Perth, Scotland, a company specializing in newborn screening diagnostic products. Terms of the acquisition were not disclosed. This acquisition strengthens Bio Rad as a market leader in newborn screening tests for Phenylketonuria, Galactosemia, Glucose 6 Phosphate Dehydrogenase (G 6 PD) Deficiency. It also enhances the company's leading position in genetic disorders testing by adding to its portfolio of DNA, immunoassay and HPLC based tests for sickle cell anemia, metabolic disorders, and thalassemia.
"This acquisition strengthens our position in serving the genetic screening market and provides a foundation for us to further our innovation in genetic screening products and services," said David Schwartz, President of Bio Rad Laboratories. "Quantase has an excellent reputation for quality products and broad customer acceptance throughout the world." The current market for newborn screening tests is estimated to be more than $100 million worldwide and is growing at a rate of approximately 10 percent per year. Newborn genetic screening is compulsory in the United States and in most developed nations around the world.
Bio Rad Laboratories, Inc. (is a multinational manufacturer and distributor of life science research products and clinical diagnostics. It is based in Hercules, California, and serves more than 70,000 research and industry customers worldwide through a network of more than 30 wholly owned subsidiary offices.
Various statements made within this press release may constitute "forward looking statements" for purposes of the Securities and Exchange Commission's "safe harbor" provisions under the Private Securities Litigation Reform Act of 1995 and Rule 3b 6 under the Securities Exchange Act of 1934. The forward looking statements contained herein involve risks and uncertainties that could cause results to differ materially from the Company's expectations.
Ron Hutton, Treasurer
Bio Rad Laboratories, Inc.
Phone: 510 724 7000
E mail: investor_relations@bio rad.com
Trademarks • Site Terms • EU Recycle Program • Privacy • Feedback

(a)

```
[['search', 'Whole', 'Site', 'Log', 'In/Shop', 'Register', 'announcements', 'Corporate', 'Life', 'Science', 'Research', 'Clinical', 'Diagnostics',
'Informatics', 'Sadtler', 'Process', 'Separations', 'Life', 'Science', 'Education', 'Food', 'Animal', 'Environment', 'Testing', 'corporate', 'news',
'archives', '2012', '2011', '2010', '2009', '2008', '2007', '2006', '2005', '2004', '2003', '2002', '2001', '2000', '04/10/2002', 'Bio', 'Rad', 'Announces',
'Acquisition', 'of', 'Quantase', 'Ltd', '.'],
['Leading', 'Provider', 'of', 'Specialty', 'Diagnostics', 'Acquiring', 'Additional', 'Newborn', 'Screening', 'Tests', ',', 'Strengthening', 'Its', 'Broad',
'Portfolio', 'of', 'Genetic', 'Disorders', 'Testing', 'Products', 'and', 'Services', 'HERCULES', ',', 'CA', ',', 'April', '10', ',', '2002', 'Bio', 'Rad',
'Laboratories', ',', 'Inc.', '(', 'AMEX', 'BIO', 'and', 'BIO.B', ')', 'announced', 'today', 'the', 'acquisition', 'of', 'Quantase', 'Ltd.', 'of', 'Perth',
',', 'Scotland', ',', 'a', 'company', 'specializing', 'in', 'newborn', 'screening', 'diagnostic', 'products', '.'], ['Terms', 'of', 'the', 'acquisition', 'were',
'not', 'disclosed', '.'],
['This', 'acquisition', 'strengthens', 'Bio', 'Rad', 'as', 'a', 'market', 'leader', 'in', 'newborn', 'screening', 'tests', 'for', 'Phenylketonuria',
',', 'Galactosemia', ',', 'Glucose', '6', 'Phosphate', 'Dehydrogenase', '(', 'G', '6', 'PD', ')', 'Deficiency', '.'], ['It', 'also', 'enhances', 'the',
'companyï¿½s', 'leading', 'position', 'in', 'genetic', 'disorders', 'testing', 'by', 'adding', 'to', 'its', 'portfolio', 'of', 'DNA', ',', 'immunoassay',
'and', 'HPLC', 'based', 'tests', 'for', 'sickle', 'cell', 'anemia', ',', 'metabolic', 'disorders', ',', 'and', 'thalassemia', '.'], ['ï¿½This', 'acquisition',
'strengthens', 'our', 'position', 'in', 'serving', 'the', 'genetic', 'screening', 'market', 'and', 'provides', 'a', 'foundation', 'for', 'us', 'to', 'further',
'our', 'innovation', 'in', 'genetic', 'screening', 'products', 'and', 'services', ',', 'ï¿½', 'said', 'David', 'Schwartz', ',', 'President', 'of', 'Bio', 'Rad',
'Laboratories', '.'], ['ï¿½Quantase', 'has', 'an', 'excellent', 'reputation', 'for', 'quality', 'products', 'and', 'broad', 'customer', 'acceptance', 'throughout',
'the', 'world.ï¿½', 'The', 'current', 'market', 'for', 'newborn', 'screening', 'tests', 'is', 'estimated', 'to', 'be', 'more', 'than', '$', '100', 'million',
'worldwide', 'and', 'is', 'growing', 'at', 'a', 'rate', 'of', 'approximately', '10', 'percent', 'per', 'year', '.'], ['Newborn', 'genetic', 'screening', 'is',
'compulsory', 'in', 'the', 'United', 'States', 'and', 'in', 'most', 'developed', 'nations', 'around', 'the', 'world', '.'], ['Bio', 'Rad', 'Laboratories', ',',
'Inc.', '(', 'is', 'a', 'multinational', 'manufacturer', 'and', 'distributor', 'of', 'life', 'science', 'research', 'products', 'and', 'clinical', 'diagnostics', '.'],
['It', 'is', 'based', 'in', 'Hercules', ',', 'California', ',', 'and', 'serves', 'more', 'than', '70,000', 'research', 'and', 'industry', 'customers', 'worldwide',
'through', 'a', 'network', 'of', 'more', 'than', '30', 'wholly', 'owned', 'subsidiary', 'offices', '.'], ['Various', 'statements', 'made', 'within', 'this', 'press',
'release', 'may', 'constitute', '``', 'forward', 'looking', 'statements', "''", 'for', 'purposes', 'of', 'the', 'Securities', 'and', 'Exchange', 'Commission', "'s",
'``', 'safe', 'harbor', "''", 'provisions', 'under', 'the', 'Private', 'Securities', 'Litigation', 'Reform', 'Act', 'of', '1995', 'and', 'Rule', '3b', '6', 'under',
'the', 'Securities', 'Exchange', 'Act', 'of', '1934', '.'], ['The', 'forward', 'looking', 'statements', 'contained', 'herein', 'involve', 'risks', 'and',
'uncertainties', 'that', 'could', 'cause', 'results', 'to', 'differ', 'materially', 'from', 'the', 'Company', "'s", 'expectations', '.'], ['Ron', 'Hutton', ',',
'Treasurer', 'Bio', 'Rad', 'Laboratories', ',', 'Inc', '.'], ['Phone', ':', '510', '724', '7000', 'E', 'mail', ':', 'investor_relations', '@', 'bio', 'rad.com',
'Trademarks', 'â€¢', 'Site', 'Terms', 'â€¢', 'EU', 'Recycle', 'Program', 'â€¢', 'Privacy', 'â€¢', 'Feedback', 'Copyright', 'Â©', '2011', 'Bio', 'Rad', 'Laboratories',
',', 'Inc.', 'All', 'rights', 'reserved', '.'], ['Page', '1', 'of', '1Bio', 'Rad', 'Laboratories', '14.03.2012']]
```

(b)

Figure 2.9 – *Plain text tokenization. (a) Extracted plain text using ApacheTika, (b) Tokenized text using NLTK TreebanckTokenizer.*

the quality of generated output. However, there is no universal answer for keeping or removing punctuations and stop words and it highly depends on the domain of interest. In sentiment analysis, for instance, where the semantics of given text is extremely crucial, removing stopwords and punctuations will produce ambiguous results. Therefore, the approach of punctuation and stopwords removal is not generalizable to all domains and depending on the application, they must be treated in different ways.

In the context of ontology engineering from textual data, our desired outputs are triplets of the form *(subject, predicate, object)*. Therefore, preserving the punctuation marks and stop words considerably adds extra complexity to the process of triplet extraction. For instance in *"'Apple' is [recently] launched the "AppleWatch".! &it has multiple attractive colors."*, we better to omit punctuation marks and simplify the sentence into *"Apple is recently launched the AppleWatch. it has multiple attractive colors."* as they add no remarkable value to the final result.

For this purpose, punctuation elimination and stop words removal are applied on tokenized version of plain text from previous step and the result is shown in 2.10a. Although applying these steps could considerably clean the text, still some noisy data such as combination of ascii characters, short tokens (having $length < 2$) and numbers are present in the output (see Figure 2.10a). In order to accomplish the desired output from text analysis for ontology engineering and to create highly accurate and normalized text, some extra cleaning processes are also applied. As the result, Figure 2.10b demonstrates the final output of text cleaning process on the running example.

2.3.3 Relevant Statements Extraction

In NLP, text simplification is the process of reducing syntactical and lexical complexity of a text while preserving its meaning [135], [136]. In syntactic simplification, the long and structurally complex sentences are converted to shorter and simpler sentences. On the other hand, lexical simplification replaces words with simpler and common terms and expressions.

For the purpose of non-taxonomical relation extraction for the ultimate domain ontology and considering the structure of facts within an ontology, extracting set of triplets of the form <*subject-predicate-object*> is required. In this expression, the *"predicate"* which denotes a relationship between the *"subject"* and the *"object"* can be replaced by part of speech of *"verb"*. Therefore, in this section, syntactic text simplification is performed in order to identify relative clauses in the form of *"Subject-Verb-Object"* through *POS* tagging and *chunking* processes.

In order to simplify a sentence, extracting its structure is highly required to identify the components to be separated out. Obviously *POS* tagging and *chunking* could effectively obtain the desired output.

- *POS-Tagging:* As explained in previous section, *POS* tagging is the process of converting provided sentence to the list of tuples in which each tuple is in the form of *(word , tag)*.

```
[['search', 'Whole', 'Site', 'Log', 'InShop', 'Register', 'announcements', 'Corporate', 'Life', 'Science', 'Research', 'Clinical',
'Diagnostics', 'Informatics', 'Sadtler', 'Process', 'Separations', 'Life', 'Science', 'Education', 'Food', 'Animal', 'Environment',
'Testing', 'corporate', 'news', 'archives', '2012', '2011', '2010', '2009', '2008', '2007', '2006', '2005', '2004', '2003', '2002',
'2001', '2000', '04102002', 'Bio', 'Rad', 'Announces', 'Acquisition', 'Quantase', 'Ltd'],
['Leading', 'Provider', 'Specialty', 'Diagnostics', 'Acquiring', 'Additional', 'Newborn', 'Screening', 'Tests', 'Strengthening', 'Its',
'Broad', 'Portfolio', 'Genetic', 'Disorders', 'Testing', 'Products', 'Services', 'HERCULES', 'CA', 'April', '10', '2002', 'Bio', 'Rad',
'Laboratories', 'Inc', 'AMEX', 'BIO', 'BIOB', 'announced', 'today', 'acquisition', 'Quantase', 'Ltd', 'Perth', 'Scotland', 'company',
'specializing', 'newborn', 'screening', 'diagnostic', 'products'],
['Terms', 'acquisition', 'disclosed'], ['This', 'acquisition', 'strengthens', 'Bio', 'Rad', 'market', 'leader', 'newborn', 'screening',
'tests', 'Phenylketonuria', 'Galactosemia', 'Glucose', '6', 'Phosphate', 'Dehydrogenase', 'G', '6', 'PD', 'Deficiency'], ['It', 'also',
'enhances', companyï¿½s', 'leading', 'position', 'genetic', 'disorders', 'testing', 'adding', 'portfolio', 'DNA', 'immunoassay', 'HPLC',
'based', 'tests', 'sickle', 'cell', 'anemia', 'metabolic', 'disorders', 'thalassemia'], [ï¿½This, 'acquisition', 'strengthens', 'position',
'serving', 'genetic', 'screening', 'market', 'provides', 'foundation', 'us', 'innovation', 'genetic', 'screening', 'products', 'services', 'ï¿½',
'said', 'David', 'Schwartz', 'President', 'Bio', 'Rad', 'Laboratories'], ['ï¿½Quantase', 'excellent', 'reputation', 'quality', 'products', 'broad',
'customer', 'acceptance', 'throughout', 'worldï¿½', 'The', 'current', 'market', 'newborn', 'screening', 'tests', 'estimated', '100', 'million',
'worldwide', 'growing', 'rate', 'approximately', '10', 'percent', 'per', 'year'], ['Newborn', 'genetic', 'screening', 'compulsory', 'United', 'States',
'developed', 'nations', 'around', 'world'], ['Bio', 'Rad', 'Laboratories', 'Inc', 'multinational', 'manufacturer', 'distributor', 'life', 'science',
'research', 'products', 'clinical', 'diagnostics'], ['It', 'based', 'Hercules', 'California', 'serves', '70000', 'research', 'industry', 'customers',
'worldwide', 'network', '30', 'wholly', 'owned', 'subsidiary', 'offices'], ['Various', 'statements', 'made', 'within', 'press', 'release', 'may',
'constitute', 'forward', 'looking', 'statements', 'purposes', 'Securities', 'Exchange', 'Commission', 'safe', 'harbor', 'provisions', 'Private',
'Securities', 'Litigation', 'Reform', 'Act', '1995', 'Rule', '3b', '6', 'Securities', 'Exchange', 'Act', '1934'], ['The', 'forward', 'looking',
'statements', 'contained', 'herein', 'involve', 'risks', 'uncertainties', 'could', 'cause', 'results', 'differ', 'materially', 'Company', 'expectations'],
['Ron', 'Hutton', 'Treasurer', 'Bio', 'Rad', 'Laboratories', 'Inc'], ['Phone', '510', '724', '7000', 'E', 'mail', 'investorrelations', 'bio', 'radcom',
'Trademarks', 'â€¢', 'Site', 'Terms', 'â€¢', 'EU', 'Recycle', 'Program', 'â€¢', 'Privacy', 'â€¢', 'Feedback', 'Copyright', 'Â©', '2011', 'Bio', 'Rad',
'Laboratories', 'Inc', 'All', 'rights', 'reserved'], ['Page', '1', '1Bio', 'Rad', 'Laboratories', '14032012']]
```

(a)

```
[['search', 'Whole', 'Site', 'Log', 'InShop', 'Register', 'announcements', 'Corporate', 'Life', 'Science', 'Research', 'Clinical', 'Diagnostics',
'Informatics', 'Sadtler', 'Process', 'Separations', 'Life', 'Science', 'Education', 'Food', 'Animal', 'Environment', 'Testing', 'corporate', 'news',
'archives', 'Bio', 'Rad', 'Announces', 'Acquisition', 'Quantase', 'Ltd'], ['Leading', 'Provider', 'Specialty', 'Diagnostics', 'Acquiring', 'Additional',
'Newborn', 'Screening', 'Tests', 'Strengthening', 'Its', 'Broad', 'Portfolio', 'Genetic', 'Disorders', 'Testing', 'Products', 'Services', 'HERCULES',
'CA', 'April', 'Bio', 'Rad', 'Laboratories', 'Inc', 'AMEX', 'BIO', 'BIOB', 'announced', 'today', 'acquisition', 'Quantase', 'Ltd', 'Perth', 'Scotland',
'company', 'specializing', 'newborn', 'screening', 'diagnostic', 'products'], ['Terms', 'acquisition', 'disclosed'], ['This', 'acquisition', 'strengthens',
'Bio', 'Rad', 'market', 'leader', 'newborn', 'screening', 'tests', 'Phenylketonuria', 'Galactosemia', 'Glucose', 'Phosphate', 'Dehydrogenase', 'PD', 'Deficiency'],
['It', 'also', 'enhances', 'companys', 'leading', 'position', 'genetic', 'disorders', 'testing', 'adding', 'portfolio', 'DNA', 'immunoassay', 'HPLC',
'based', 'tests', 'sickle', 'cell', 'anemia', 'metabolic', 'disorders', 'thalassemia'], ['This', 'acquisition', 'strengthens', 'position', 'serving',
'genetic', 'screening', 'market', 'provides', 'foundation', 'us', 'innovation', 'genetic', 'screening', 'products', 'services', 'said', 'David', 'Schwartz',
'President', 'Bio', 'Rad', 'Laboratories'], ['Quantase', 'excellent', 'reputation', 'quality', 'products', 'broad', 'customer', 'acceptance', 'throughout',
'world', 'The', 'current', 'market', 'newborn', 'screening', 'tests', 'estimated', 'million', 'worldwide', 'growing', 'rate', 'approximately', 'percent',
'per', 'year'], ['Newborn', 'genetic', 'screening', 'compulsory', 'United', 'States', 'developed', 'nations', 'around', 'world'], ['Bio', 'Rad', 'Laboratories',
'Inc', 'multinational', 'manufacturer', 'distributor', 'life', 'science', 'research', 'products', 'clinical', 'diagnostics'], ['It', 'based', 'Hercules',
'California', 'serves', 'research', 'industry', 'customers', 'worldwide', 'network', 'wholly', 'owned', 'subsidiary', 'offices'], ['Various', 'statements',
'made', 'within', 'press', 'release', 'may', 'constitute', 'forward', 'looking', 'statements', 'purposes', 'Securities', 'Exchange', 'Commission', 'safe',
'harbor', 'provisions', 'Private', 'Securities', 'Litigation', 'Reform', 'Act', 'Rule', '3b', 'Securities', 'Exchange', 'Act'], ['The', 'forward', 'looking',
'statements', 'contained', 'herein', 'involve', 'risks', 'uncertainties', 'could', 'cause', 'results', 'differ', 'materially', 'Company', 'expectations'],
['Ron', 'Hutton', 'Treasurer', 'Bio', 'Rad', 'Laboratories', 'Inc'], ['Phone', 'mail', 'investorrelations', 'bio', 'radcom', 'Trademarks', 'Site', 'Terms',
'EU', 'Recycle', 'Program', 'Privacy', 'Feedback', 'Copyright', 'Bio', 'Rad', 'Laboratories', 'Inc', 'All', 'rights', 'reserved'],
['Page', '1Bio', 'Rad', 'Laboratories']]
```

(b)

Figure 2.10 – *Output text after text cleaning steps are applied. (a) Text after stop words and punctuation removal , (b) Cleaned text.*

The tag is a signifier of word's part of speech and indicates whether the word is a noun, adjective, verb, etc.

In order to assign linguistic information on extracted tokens from cleaned corpus, we have used *pos-tag()* function from *NLTK* library. Figure 2.11 shows pos-tagged tokens from our running example.

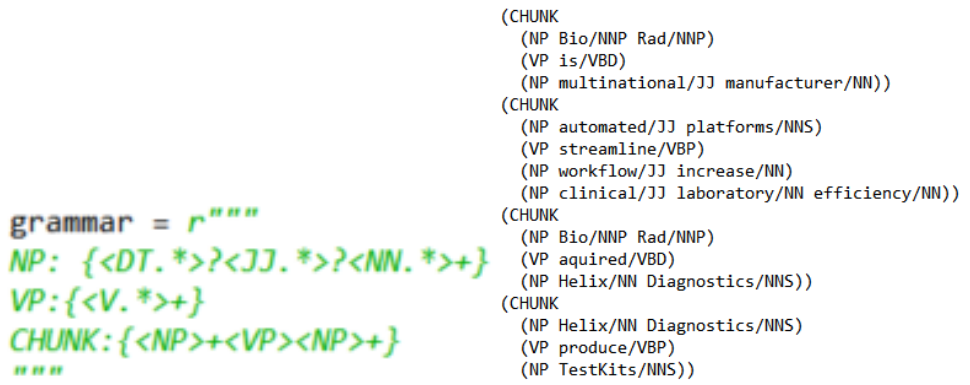
Figure 2.11 – *The processed text after POS Tagging*

As shown in Fig. 2.11, nouns are mainly appearing after determiners (*"DT" pos tag*) and adjectives (*"ADJ" pos tag*) and could be either the subject or the object of a verb. As an example, considering the original sentence as *"Bio-Rad Announces Acquisition of Quantase"*, the pos-tagged on cleaned sentence is *"('Bio', 'NNP'), ('Rad', 'NNP'), ('Announces', 'VBP'), ('Acquisition', 'NNP'), ('Quantase', 'NNP')"*, in which *"BioRad"* has the role of subject of observed verb *"Announce"*.

- *Chunking:* Once pos-tagged tokens are generated, chunking is applied with the goal of segmenting and labelling sequence of tokens which follow required rule. In order to design a chunker, we have defined a chunking grammar which declares the rule of how chunking must be done. Considering the desired form of each chunk (i.e. *"Subject-Verb-Object"*), *"Noun Phrase (NP)"* is defined as optional determinator and any number of adjectives being followed by any type of noun pos-tags. Moreover, *Verb Phrase (VP)* is simply defined as any type of verb pos-tag in *NLTK* (i.e. base form-VB, past tense-VBD, past participle-VBN, etc.). Lastly, a *chunk* is a sequence of *(NP,VP,NP)* which entirely matches the required structure. Figure 2.12 displays a defined grammar as well as a few extracted chunks from the whole plain text.

2.3.4 Chunks Filtering

The goal of our proposed method is to provide a generic information extraction framework which is adoptable to any provided textual data with diverse domains. However, the output of discussed method enables us to produce highly accurate results with minimal human effort. Existing diversity in the content and context of collected documents often brings many challenges for information seeking users. In order to reduce search space for the sake of non-taxonomic relation extraction of ultimate domain ontology, our proposed method employs



(a) (b)

Figure 2.12 – *Extracted chunks based on a predefined grammar. (a) Chunking grammar, (b) Examples of extracted chunks.*

the use of defined controlled vocabularies for extracting the most needed information from input corpus.

Therefore, as the next step, generated chunks are filtered based on the content of their respective subject or object. In this process, those chunks that do not contain at least one of the controlled vocabularies in their subject or object segments are filtered out and removed from the list. Applying this process could remarkably reduce the size of subject-verb-object (SVO) triplets. More importantly it guarantees that only relevant statements are extracted from documents. For instance, let us assume that names of pharmaceutical companies such as *"Bio Rad , Roche, Johnson&Johnson, etc."*) have been provided as the controlled vocabularies. Applying the last step of filtering results in removal of the second chunk among those four being presented in Figure 2.12, as it does not contain any of those controlled vocabularies mentioned earlier.

2.3.5 XML Representation of SVO Triplets

Representing data in an interchangeable and standard format is highly crucial for any information extraction system. In proposed method, the output of text processing section will be delivered as one of the input parameters for ontology engineering from unstructured data. Therefore, once required information is extracted in the form of chunks, it is parsed into *XML* format based on some specified rules.

XML representation of *subject-verb-object* triplets is achieved through retaining each chunk in a separate xml bucket $< SVO > ... < /SVO >$ which composed of three main sub elements:

1. $< Subjectname = "sub" > NP < /Subject >$
2. $< Predicatename = "predicate" > VP < /Predicate >$

3. $< Objectname = "object" > NP < /Object >$

XML representation of a few extracted chunks is shown in Figure 2.13. It must be noticed that, for the sake of simplification, the lemmatized form of all *VP*s are extracted and replaced in *XML* representation.

```
<?xml version='1.0' encoding='utf-8'?>
<root>
 <svo>
    <Subject name="sub">Bio Rad</Subject>
    <Predicate name="predicate">is</Predicate>
    <Object name="obj">multinational manufacturer</Object>
  </svo>
  <svo>
    <Subject name="sub">Bio Rad</Subject>
    <Predicate name="predicate">acquire</Predicate>
    <Object name="obj">Helix diagnostics</Object>
  </svo>
   <svo>
    <Subject name="sub">Helix diagnostics</Subject>
    <Predicate name="predicate">produce</Predicate>
    <Object name="obj">Testkits</Object>
  </svo>
 <svo>
    <Subject name="sub">Mindray</Subject>
    <Predicate name="predicate">collaborate</Predicate>
    <Object name="obj">Novartis</Object>
  </svo>
</root>
```

Figure 2.13 – *XML representation of SVO chunks.*

2.3.6 Clustering-Based Extraction of Domain Concepts

Note that the main objective of the proposed approach is to construct a domain ontology from unstructured textual data. Therefore, discovering proper ontological classes is one of the primary challenges. In other words, we would like to know what the main classes exist that can describe different words within a text corpora such as companies, products, humans, etc. Named entity recognition (NER) as a technique in NLP will be usually used for labeling different words within a corpus with their corresponding entity. However, you may not end up with a trustworthy list of entities mainly due to the fact that most of the NER modules have been trained on domain specific data. Therefore, we decided to take a different approach to verify if we can correctly identify the groups of words which have been talked about in the input documents. In the following we describe our customized approach for pulling domain specific ontological classes.

The advantage of the proposed approach is that it enables us to identify the group of products for which we need to extract some complementary information from other resources (such as *"OpenFDA"*) later on. Moreover, it verifies that one can identify groups of words that in some sense belong to the same family (such as products, or companies, etc). Those families of words will be later established as the main classes for constructing desired ontology. Furthermore, it will be determined which word must belong to which class (based on the group to which it is assigned).

Binary Table Representation of SVO Triplets

Once the triplets containing controlled vocabularies are extracted from text corpus, we have created a binary table by mapping subjects and objects of the triplets to the verbs with which they form a triplet. A binary table is represented in the form of $\mathcal{K} =< X, Y, I >$ where X denotes the set of individuals (rows) that are characterized by a set Y of attributes (columns), where I denotes the binary relation between members of sets X and Y. For the sake of clarity, an example of binary table is presented in Figure 2.14, where the set of all subjects and objects in the list of *SVO* triplets are considered as rows, and the set of all verbs in those triplets are considered as columns, but in two different forms V_{subj} and V_{obj}. In other words, the number of columns in the binary table is twice as big as the number of unique verbs in triplets *SVO*, while the number of rows in the binary table is the number of unique subjects and objects in those triplets. For each triplet *SVO*, the subject S is linked to V_{subj} and the object O is linked to V_{obj}, i.e., $I(S, V_{subj}) = 1$ and $I(O, V_{obj}) = 1$.

The argument behind defining two forms for each verb in the binary table is explained in following example. Imagine we have the triplet *(Google, Hire, John Smith)*. Although *Google* and *John Smith* are both related to verb *Hire*, they are different in the sense that the former can be characterized by its feature that is being able to *hire* someone, while the latter is characterized by its feature that *can be hired* by some organization. Figure 2.14 demonstrates how different extracted triplets will contribute in creating our desired binary table. In this example, a binary table is created using SVO triplets of the form *subject,verb,object*. For each verb V within a triplet *SVO*, two columns V_{subj} and V_{obj} will be created in order for the verb to be linked to the subject S and object O (both as individuals inserted in rows of the binary table). As an example, for the triplet *(Roche , Hire , DataScientist)* two crosses denoted by red boxes are added to the table. The blue and green boxes mark the contribution of *(Apple , Produce , AppleWatch)* and *(Herceptine, Treat , Cancer)* in creating table, respectively.

A	B	C	D	E	F	G
Fromal Context	hire-subj	hire-obj	produce-subj	produce-obj	treat-subj	treat-obj
Roche	X	0	X	0	0	0
Nag	0	X	0	0	0	0
Herceptine	0	0	0	X	X	0
Cancer	0	0	0	0	0	X
DataScientist	0	X	0	0	0	0
Apple	X	0	X	0	0	0
AppleWatch	0	0	0	X	0	0

Figure 2.14 – *Binary table of SVO triplets.*

Clustering of Extracted Objects

Once the binary table is created, we applied K-means clustering algorithm to group similar subjects and objects being located in rows. For this aim, we represented each row by a binary vector w of size $|Y|$, where $w(i) = 1$ if $y_i \in Y$, and $w(i) = 0$ otherwise. Here $|Y|$ denotes the cardinality of the column set Y in the binary table and y_i denotes the i^{th} attribute (i.e., i^{th}

column in the binary table). Clustering could satisfy our initial motivation which was to extract ontological classes by accurately grouping similar domain artifacts together. Figure 2.16 depicts the output of K-means clustering for $K = 3$ and Figure 2.15 shows examples of instances that are categorized into three separate clusters. It indicates that the most of individuals that belong to the same class are really part of the same group (mainly *Companies*, *Products* or *People*). These three groups then form the basis of the ontological classes being used in next phases of ontology learning.

In Figure 2.15, extracted subjects and objects are clustered into three categories based on the similarity of their attributes (i.e., columns). Each circle represents a row and each cluster is identified by a separate colour. For the sake of visualization, the instances of clusters are projected into a two dimensional space, where each dimension represents one of the two principal component of the data. Figure 2.16 depicts examples of individuals that are categorized into three clusters we identified in Figure 2.15. The colour of each column is consistent with the colour of its corresponding cluster in Figure 2.15. This table shows that each cluster identifies a separate group of individuals (here *Persons* in *cluster-1*, *Products* in *cluster-2*, and *Companies* in *cluster-3*). These three clusters form the first three classes in the ontology. Individuals within each cluster will then be considered as instances of its corresponding ontological class.

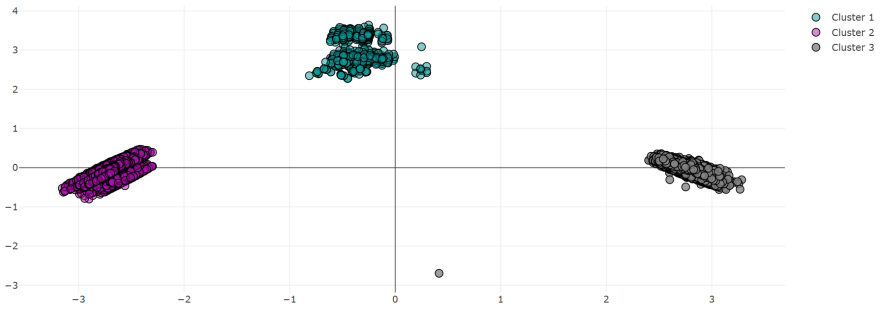


Figure 2.15 – *Subjects and objects of triplets SVOs are clustered into three categories based on the similarity of their attributes.*

Cluster-1	Cluster-2	Cluster-3
Abaxis sales team	Apple Healthkit	Dako
Beijing Leadman	ELISA testing kits	Apple
Grifols product manager	M5 Ultrasound systems	DiaSys
Sarah Li	MIA FORA NGS analysis software	Johnson&Johnson
Dr Michael OReilly	biothreat detection applications	Luminex
Richard Yang	xTAG CYP2D6 Kit	Mindray
Thomas Mac Mahoncurrently	iCa biosensors	Trinity

Figure 2.16 – *Examples of individuals that are categorized into three clusters identified in Fig.* 2.15.

We also suggested to have the possibility of adding / removing or even merging some of

those classes, depending on a domain expert knowledge if it is available. To show such a possibility, we have created a fourth class and named it as *Location*, where its instances were initially grouped in one of those initial three clusters. How to choose an optimal number of clusters is always debatable. For instance by increasing number K of clusters in the beginning, the chance of appearing a new cluster that automatically identifies instances of the class of *location* would increase. However, it may also cause further separation of a bigger class (such as *Products*) into smaller groups (which can be later merged according to a domain knowledge expert).

2.4 Conclusion

In this chapter we described how we can extract relevant statements, containing valuable information about a domain of interest, from a text corpus. The ultimate goal of this work is to engineer an ontology to be used for knowledge extraction from unstructured data. Having a list of triplets of interest (containing some predefined controlled vocabularies) is a prerequisite for the rest of pipeline which will be described in the following chapters.

We first started by cleaning the text corpus so that it becomes ready for some processing. We then applied a sequence of natural language processing techniques to extract the triplets of interest. Those triplets contain information about controlled vocabularies which are of great interest for ontology engineering. We use those triples in later chapters as the input to our engineered ontology in the form of relationships between different ontological classes. In next chapter we will focus on formal concept analysis as a promising data analysis method to build a cornerstone of the desired ontology.

3 Building Cornerstone of Ontology from Concept Lattice

"There is hidden order in Nature, to be found only by patient search."

- Garrett Birkhoff

3.1 Overview

In the previous chapter, we studied how we can generate a list of triplets that represents the most relevant statements about a domain of knowledge within a text corpus. In this chapter, the focus is more on how to build a set of formal concepts and a hierarchy among them that describes a domain of interest. The extracted formal concepts will form the cornerstone of the desired ontology.

In this chapter we demonstrate how formal concept analysis (FCA) can be utilized for the purpose of generating a concept lattice from formal contexts, In the following, we first start with an overview of FCA, and then we move to an in-depth discussion on how we can benefit from FCA for our purpose. We further look into how we can speed up the existing algorithms for formal concept generation and concept lattice construction.

Moreover, we provide some background knowledge about FCA, where it will be used in the next chapter to shape the backbone of the ontology. The ultimate goal is to further augment the ontology using extracted knowledge from a text corpus (see previous chapter).

3.2 Formal Concept Analysis (FCA)

3.2.1 FCA: Background and Context

Formal Concept Analysis (FCA) was introduced and studied by Wille [137],and Ganter [138], [139] as a powerful mathematical theory and a conceptual framework for structuring and

analyzing knowledge extraction and data visualization. FCA extracts the key concepts of the application domain and represents their hierarchical structure according to the concept lattice also known as the Galois Graphs. The strength of FCA in any application relies on using some general rules of mathematical logic to extract domain concepts which can be hierarchically grouped together according to their common attributes.

3.2.2 Formal Context and Formal Concepts

At the core of FCA there is a binary table containing rows and columns in which each row represents a specific domain object and each column corresponds to a domain attribute. Domain specific objects and attributes are connected through Boolean values (i.e., zeros and ones or False and True), so that each object has a set of attributes and might not be related to the rest. In the table, the cross sign *'X'* is used to indicate a relation among a specific object and its respective attributes, so that if an object does not have a value, the intersection of that object and attribute will remain empty.

Formal Context: The following definition is provided by Will and Ganter [138] as short mathematical explanation for *formal context*:

Definition 1 *A formal context is defined as a triplet* $\mathbb{K} : (X, Y, I)$ *which represents a binary relationship* $I \subseteq (X \times Y)$ *between a set of objects* X*, and a set of attributes* Y*.*

Formal context enables basic procedure of FCA by providing simple representation of data in the form of binary table. Basically, a formal context can be constructed based on any set of objects (as table rows) and attributes (as table columns) as well as binary relationship among them. There is no restriction on the nature of objects and attributes. Persons, physical objects, processes, numbers, etc. are just a few examples of objects and attributes of some formal contexts.

Consider a binary table with the size of *(n*m)* in which n is the number of rows and m is the number of columns. The set of objects are defined as $X = \{x_1, ..., x_n\}$, the set of attributes are defined as $Y = \{y_1, ..., y_m\}$, and if the intersection among x_i and y_j is not empty in the formal context, then the relation $I(x_i, y_j) = 1$ is valid which indicates object x_i has attribute y_j.

Notably, FCA generates two types of outputs from a given formal context called *Formal Concepts* and *Concept Lattice*. Generally speaking, each formal concept is a human interpretable unit of knowledge from the application domain. Moreover, the set of formal concepts can be partially ordered in a *subconcept-superconcept hierarchy* known as concept lattice.

Definition 2 *[138] For a formal context* (X, Y, I)*, concept-forming operators are up denoted by 'u':* $2^X \rightarrow 2^Y$ *and down denoted by 'd':* $2^Y \rightarrow 2^X$ *such that for* $A \subseteq X$ *and* $B \subseteq Y$ *they are defined as:*

	Animal	Plant	Lives on Land	Lives in Water
Dog	X		X	
Cat	X		X	
Oak		X	X	
Potato		X	X	
Carp	X			X
Water Illy		X		X
Reed		X	X	X

Table 3.1 – *Formal Context of plants and animals.*[1]

$$A^u := \{y \in Y | \forall x \in A : I(x,y) = 1\}$$
$$B^d := \{x \in X | \forall y \in B : I(x,y) = 1\}$$

Definition 3 *[138] A formal concept in a formal context* (X,Y,I) *is defined as a pair of* (A,B) *, where* $A \subseteq X$,$B \subseteq Y$ *and* $A^u = B$ *and* $B^d = A$.

A **Formal Concept** is a segment of a formal context in which different objects share the same attributes. According to Definition3, each formal concept consists of two parts named *extent* and *intent*. For a formal concept (A,B), the set A is called extent and expresses those objects having attributes in common with B. Alternatively, the set B is called intent and represents the attributes assigned to all objects within A. Mathematically speaking, the set of all extents and intents of a formal context (X,Y,I) are defined as following:

- $Ext(X,Y,I) = \{B^d | B \subseteq Y\}$
- $Int(X,Y,I) = \{A^u | A \subseteq X\}$

In the following, we use the formal context that is represented in Table 3.1 as a running example for clarifying different operations of FCA. It represents a set of entities with some of their characteristics such as where they live and if they are animals or plants, etc. The set of objects and attributes in this example are the following.

$$X = \{Dog, Cat, Oak, Potato, Carp, WaterIlly, Reed\}$$
$$Y = \{Animal, Plant, LivesonLand, LivesinWater\}$$

In Table 3.1, each binary relation (displayed by cross sign) express that object x has the attribute y. An example of formal concept for that context is the pair $\{(Cat, Dog), (Animal, Lives\,on\,Land)\}$ which indicates both *Cat* and *Dog* have the following attributes in common. They are both animal and live on land, i.e., having *Animal* and *Lives on Land* crossed in that table for both *Cat*

[1] Example is captured from: https://www.slideshare.net/ssakpi/formal-concept-analysis

and *Dog*. Moreover, there is no other object than *Cat* and *Dog* that have these two attributes in common (in that context). In other words,

$$(A_1, B_1) = (\{Cat, Dog\}, \{Animal, Lives\,on\,Land\})$$
$$\{Cat, Dog\}^u = \{Animal, Lives\,on\,Land\}$$
$$\{Animal, Lives\,on\,Land\}^d = \{Cat, Dog\}$$

Clearly, many formal concepts might exist within a formal context. In theory, the number of formal concepts grows exponentially with the size of the binary table, i.e., $O(min\{2^n, 2^m\})$. Two more formal concepts of formal context 3.1 are shown in the following tables and highlighted as rectangles. They correspond to

$$(A_2, B_2) = (\{Cat, Dog, Carp\}, \{Animal\})$$
$$(A_3, B_3) = (\{Reed, Water\,Illy\}, \{Plant, Lives\,in\,Water\})$$

.

	Animal	Plant	Lives on Land	Lives in Water
Dog	X		X	
Cat	X		X	
Oak		X	X	
Potato		X	X	
Carp	X			X
Water Illy		X		X
Reed		X	X	X

(a)

	Animal	Plant	Lives on Land	Lives in Water
Dog	X		X	
Cat	X		X	
Oak		X	X	
Potato		X	X	
Carp	X			X
Water Illy		X		X
Reed		X	X	X

(b)

Figure 3.1 – *Two examples of formal concepts from Table 3.1. (a)* (A_2, B_2)*, (b)* (A_3, B_3).

3.2.3 Concept Lattice

As mentioned above, FCA also returns the hierarchical relationship among established concepts in the form of a line diagram called *Concept Lattice*. A conceptual hierarchy, which is in the form of *sub-concept_super-concept* of formal concepts is defined in [138] as follows:

Definition 4 *For formal concepts* (A_1, B_1) *and* (A_2, B_2)*, the sub-concept_super-concept relationship* $\leq$ *is formalized by: (both conditions must be satisfied)*

$$(A_1, B_1) \leq (A_2, B_2) : \Longleftrightarrow (A_1 \subseteq A_2)$$
$$(A_1, B_1) \leq (A_2, B_2) : \Longleftrightarrow (B_1 \supseteq B_2)$$

In Definition 4, the order relation $\leq$ describes the *partial order* between all formal concepts. A partially ordered set is a pair $(P, \leq)$ in which P is a set and $\leq$ is a binary relation on P [140]. As an examples, we can make the subset of $\{0, 1\}$ (which is: $S = \{\emptyset\}, \{0\}, \{1\}, \{0, 1\}$) as a partial order with $\subseteq$. This is because for all A, B, C in S, the following axioms of partial order are satisfied:

- Reflexive: $A \subseteq A$

- Antisymmetry: $A \subseteq B, B \subseteq A \Rightarrow A = B$
- Transitive: $A \subseteq B, B \subseteq C \Rightarrow A \subseteq C$

The order relation (shown in Definition 4) together with the set of all formal concepts of a formal context $\mathbb{K} = (X, Y, I)$ can be transformed into a complete lattice denoted by $\mathscr{B}(\mathbb{K})$ which could be illustrated as a *Hasse Line Diagram* [141].

The corresponding concept lattice of the formal concept in Table 3.1 is shown in Figure 3.2 which consists of 11 formal concepts being represented by 11 circles. A number is assigned to each formal concept, which corresponds to extents and intents that are attached to the rectangle. The nodes are connected through arrows which have been established based on the extracted partial order $\leq$ relation.

The super-concept of any chosen node is reachable by following the ascending path (of generated line diagram) from that node. Note that a concept might have multiple super-concepts. For example, *concept 1, concept 2* and *concept 3* are super-concepts of *concept 6* since there are paths of ascending edges from the node representing *concept 6* to the nodes representing *concept 1, concept 2* and *concept 3*. Similarly, we can find sub-concept(s) of a concept by following all descending paths starting at that concept. Considering *concept 9*, its sub-concepts are *concept 10* and *concept 11*.

In the main theorem on concept lattices introduced by Wille [142], it has been proven that $\mathscr{B}(\mathbb{K})$ is a complete lattice in which for any set of its formal concepts, there is a *join-mutual superconcept (supremum)* and a *meet–mutual subconcept (infimum)* in $\mathscr{B}(\mathbb{K})$. Understanding the following definition and lemma of [138] is prerequisite for discussing the completeness of a concept lattice:

Definition 5 *Let $(P, \leq)$ be a partially ordered set and O be a subset of P. A lower bound of O is an element $s \in P$ such that $s \leq o$ for all $o \in O$. Likewise, the Upper Bound of O is defined as an element $t \in P$ such that $o \leq t$ for all $o \in O$.*

The greatest element in the set of all lower bounds of O is called Infimum (join) of O and the lowest element in the set of all upper bounds of O is called its Supremum (meet)

Lemma 1 *For any given formal concepts (A_1, B_1) and (A_2, B_2):*

- *Infimum (**Greatest Lower Bound**) of (A_1, B_1) and (A_2, B_2) is defined as:*
$$(A_1, B_1) \wedge (A_2, B_2) := ((A_1 \cap A_2), \{(B_1 \cup B_2)^d\}^u) \quad ,$$
- *Supremum (**Lowest Upper Bound**) of (A_1, B_1) and (A_2, B_2) is defined as:*
$$(A_1, B_1) \vee (A_2, B_2) := (\{(A_1 \cup A_2)^u\}^d, (B_1 \cap B_2)) \quad ,$$

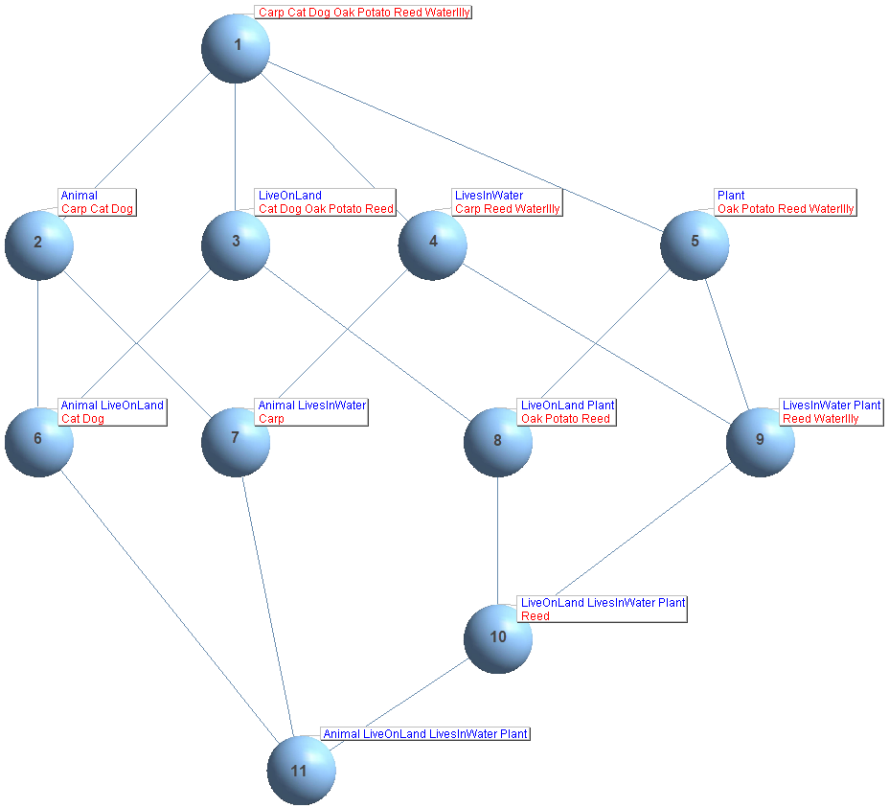


Figure 3.2 – *The Concept Lattice of formal context defined in Table* 3.1

In the running example, we can simply read the infimum and suprimum of any two concepts from the generated lattice diagram. For instance, *concept 7* is the greatest lower bound of *concept 2* and *concept 4* because it is the highest concept which is reachable by descending paths from both *concept 2* and *concept 4* in the diagram. Similarly, *concept 4* is the lowest upper bound of *concept 7* and *concept 9* because it is the lowest node that can be reached from both concepts via ascending paths.

With the support of above explanations, we can use the following definitions for characterizing the hierarchical structure of generated lattice by FCA:

Definition 6 *[138] A partially ordered set* $(P, \leq)$ *could be considered as a* ***lattice*** *if and only if for every pair of elements* $O \in P, S \in P$*, their infimum* $O \wedge S$ *and suprimum* $O \vee S$ *exist in* P*.*

More discussion about the algebraic theory of lattice is out of the scope of this work and is comprehensively studied in [143]. A *complete lattice* is defined in Definition 7. In a complete lattice, there is no restriction on the number of formal concepts which obey the rule of Definition 6. In other words, for any arbitrary chosen set of formal concepts, there must be an infimum and a supremum in lattice.

Definition 7 *[138] A partially ordered set* $(P, \leq)$ *could be considered as a complete lattice if and only if for any arbitrary set of elements* $\{p_1, ..., p_n\} \subset P$*, their infimum* $\bigwedge\{p_1, ..., p_n\}$ *and suprimum* $\bigvee\{p_1, ..., p_n\}$ *exists in* P*.*

This definition must also hold for *the most general* formal concept $\top$ (which is located at top of the line diagram and groups all the objects of a formal context) and *the most specific* formal concept $\bot$ (which is located at the bottom of line diagram and groups all the attributes of formal context).

In order to precisely formulate the algebraic properties of a complete lattice, we must consider definitions of *supremum-irreducible* and *infimum-irreducible*. In a lattice, the supremum-irreducible elements are those from which there exists one and only one edge going downward. Likewise, the infimum-irreducible elements are those from which there is one and only one edge going upward. Therefore, the supremum-irreducible elements cannot be written as the supremum of two other elements, and the infimum-irreducible elements cannot be written as infimum of two other elements.

We now appeal to a general theorem of [138], which states the completeness of FCA lattice based on previously provided terminologies and with the support of Definition 7:

Theorem 2 *Considering* $\mathbb{K} = (X, Y, I)$ *as a given formal context, its corresponding concept lattice* $\mathscr{B}(\mathbb{K})$ *is a complete lattice with the following suprema and infima for any arbitrary set of*

formal concepts $\{(A_i, B_i) | i \in I\} \subseteq \mathscr{B}(\mathbb{K})$*:*

$$\bigvee_{i \in I}(A_i, B_i) = \big(\{(\bigcup_{i \in I} A_i)^u\}^d,\ \bigcap_{i \in I} B_i\big)$$
$$\bigwedge_{i \in I}(A_i, B_i) = \big(\bigcap_{i \in I} A_i,\ \{(\bigcup_{i \in I} B_i)^d\}^u\big)$$

In a complete lattice we call a set of elements ***supremum-dense*** *if every lattice element is a supremum of elements from this set. Likewise, a set is called* ***infimum-dense*** *if every lattice element is an infimum of elements from this set.*

An arbitrary complete lattice $L = (L, \leq)$ *is isomorphic to* $\mathscr{B}(\mathbb{K})$ *if and only if there are mappings* $\gamma : X \to L,\ \mu : Y \to V$ *such that:*

- $\gamma(X)\ is\ infimum-dense\ in\ L$
- $\mu(Y)\ is\ supremum-dense\ in\ L$
- $\gamma(x) \leq \mu(y)\ iff\ I(x, y) = 1$

3.2.4 Reduced Labelling and Galois Sub Hierarchy (GSH)

A concept lattice contains redundant information in the sense that the extent and intent of all concepts overlap with some of their sub-concepts and super-concepts [137]. To make it a bit clearer, let us consider only one of the nodes of a complete lattice we earlier defined in Figure 3.2. For instance, the extent of *concept 7* contains all the objects x whose names are also repeated in its super concepts which are reachable by ascending the path from *concept 7*. Likewise, its intent contains all attributes that are repeated in its sub-concepts and are reachable by descending the path from *concept 7*.

"Reduced Labelling" or *"Galois Inheritance Lattice"* was introduced in [144] as a solution to replace each formal concept (A, B) by its reduced form $(R(A), R(B))$ such that $R(A)$ and $R(B)$ contain the non-redundant elements in A and B, respectively. It has been proven that for an object $x \in A$ in Galois lattice, there exists a smallest concept $\gamma(x) := (\{\{x\}^u\}^d, \{x\}^u)$ containing x in its extent and for each attribute $y \in B$, there exist a largest concept $\mu(y) = (\{y\}^d, \{\{y\}^d\}^u)$ with y in its intent. $\gamma(x)$ denotes the *object concept* associated to x, and $\mu(y)$ denotes the *attribute concept* corresponding to y.

Figure 3.3a represents the reduced form of the full lattice depicted in Figure 3.2. In reduced form, the name of any object x is only attached to the node which illustrates the smallest concept with x in its extent. Likewise, the name of any attribute y is only attached to the node representing the largest concept with y in it's intent.

As illustrated in Figure 3.3a, the reduced form of a lattice contains empty concepts having no

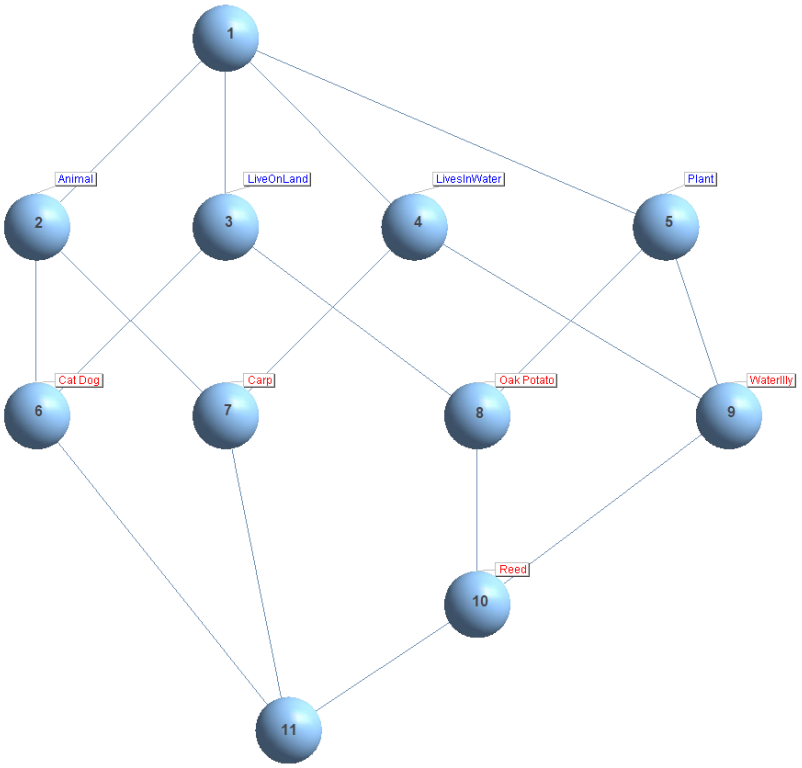
instance in their extent and intent. Depending on the application domain, one might need to eliminate such nodes since they do not add/introduce any additional information. Bypassing these nodes yields an abstract structure which is called a *Galois Sub Hierarchy (GSH)* (*pruned concept hierarchy* or a *AOC-Poset*), and has been widely applied as a data modelling tool in various domains [145],[146],[147] , [148],[149].

As shown in Figure 3.3b, this canonical sub-order representation of the original lattice is deduced from Figure 3.3a in which *concept 1* and *concept 11* are removed due to the empty $R(A)$ and $R(B)$.

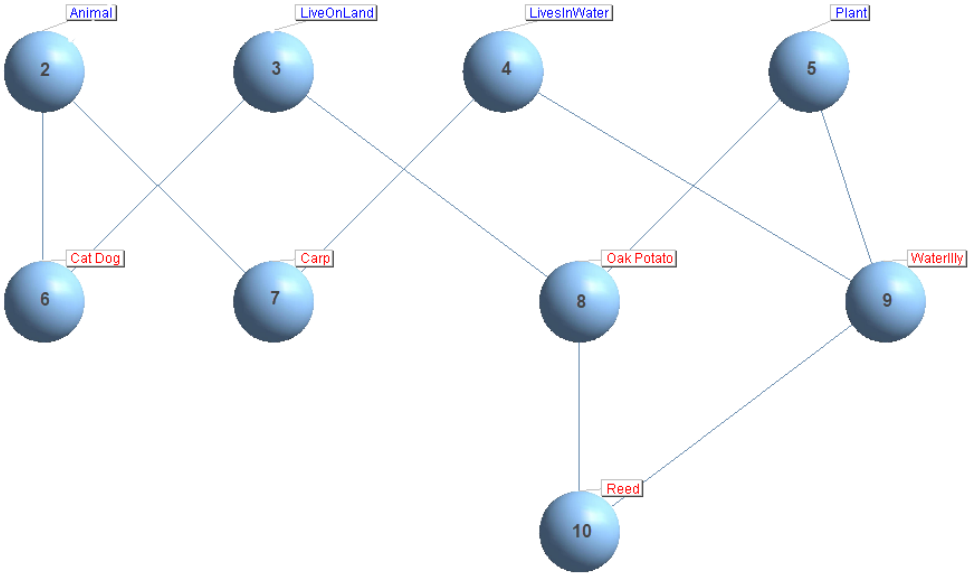
It has been shown that the size of GSH is at most equal to the total number of objects and attributes. Moreover, an introduced object is inherited from bottom to top and similarly an introduced attribute is inherited from top to bottom. Due to the difference in size of *Galois inheritance lattice* (i.e., returned graph after reduced labeling) and *GSH* (i.e., returned lattice after reduced labeling plus removal of concepts with no extent and intent), most of the proposed algorithms directly generate *GSH* instead of deriving that from reduced form of complete Galois lattice.

Many algorithms have been introduced for building GSH, with further comparison discussions on how they differ in terms of running time and computational complexity [150]:

- **Ares** [151] is an incremental Algorithm which accepts as input a Galois sub hierarchy and a new object with set of it's attributes (Y_{new}) and provides the modified version of an existing hierarchy which includes the new object as well. For doing so, the Ares Algorithm traverses the initial GSH on each node (with its corresponding set of intent $Y_{current}$), and checks if $Y_{new} = Y_{current}$, $Y_{new} \subset Y_{current}$, $Y_{new} \supset Y_{current}$ or Y_{new} and $Y_{current}$ are not comparable by set inclusion. If the Algorithm fails to find the new object, it inserts the new object as a new concept to the lattice and binds it to its superclasses and subclasses.
- **Ceres** [148] computes both the elements of GSH and its corresponding Hasse diagram at the same time. First, the Algorithm sorts the columns of a given context based on their number of cross signs. In the next step, attribute concepts are produced based on grouping attributes sharing the same extent. Object concepts are also produced for those intents being covered by intents of the attribute concepts. The Algorithm also produces edges among extracted nodes on the fly.
- **Pluton** [150] is a combination of three main successive operations: *TomThumb* [152] which generate an ordered list of simplified extents and intents which could be mapped to linear extension of GSH; *ToLinext* then merges consecutive pairs consisting of a simplified extent and a simplified intent belonging to the same concept; finally *ToGSH* generates corresponding edges of the Hasse digram of GSH.
- **Hermes** [146] generates GSH of given formal context in 5 main steps: given a domain context $\mathscr{K}$, it first removes redundancy from given context by keeping identical rows



(a)



(b)

Figure 3.3 – *Reduced labeling and Galois Sub Hierarchy of Figure* 3.2. *(a)Lattice with reduced labelling, (b) Galois sub hierarchy.*

and columns in a new context called $\mathcal{K}_i$. Then, the Algorithm computes domination relationship ([153]) among columns by deciding which columns are included in which other columns, and keeps the result in $\mathcal{K}_{Dom}$. In the next step, $\mathcal{K}_i$ and $\mathcal{K}_{Dom}$ are combined and simplified in a sense that no two rows and columns are repeated $\mathcal{K}_{iDom}$. Finally, the main elements of the Hasse diagram are extracted whose intents are rows of $\mathcal{K}_{iDom}$ and whose simplified labels are the labels of these rows in $\mathcal{K}_{iDom}$. The Hasse diagram of generated concepts is then constructed.

3.3 Accelerating *NextClosure* Algorithm

In this section, we discuss our proposed approach for reducing the execution time of one the well-known FCA algorithms called *NextClosure*, for generating formal concepts of a given formal context. Our two-fold approach involves a two-step process. First, we implement a modified form of *NextClosure*, named binary-based, which accelerates the process of its concept forming operators. Second, we take advantage of parallel processing in Python for implementing the proposed binary-based *NextClosure*. Our experiments prove that parallel execution of binary-based *NextClosure* significantly reduces the execution time of the Algorithm for sparse density and high dimensional (in the number of objects) input formal contexts.

Nevertheless, detailed discussion of our contribution in accelerating *NextClosure* Algorithm requires an in-depth knowledge of how it performs for generating canonical concepts of given formal context. The following section provides a general overview of well known FCA algorithms for constructing formal concepts. Furthermore, a detailed description and implementation of *NextClosure* is presented by means of multiple pseudocodes.

3.3.1 Overview of the *NextClosure* Algorithm

Several algorithms have been proposed for computing formal concepts of a given context. In general, FCA algorithms are divided into two categories: incremental algorithms [154], [155], [156], [157] versus batch algorithms such as [158], [159], [160] , [161], [162], [163]. Incremental algorithms produce, at each step *i*, the set of concepts and their corresponding diagram. Batch algorithms, on the other hand, generate concepts and the line diagram for the entire formal context from scratch.

In [164], the authors studied the main characteristics of various algorithms for constructing formal concept sets from a binary table. As shown in Table 3.2, the main features of all Algorithms are gathered in a formal context in which each attribute corresponds to the technique being used to generate concepts by different algorithms. For instance, attributes of columns 2 to 6, are various solutions to avoid generating the same concept multiple times.

In the following we shortly review some of the algorithms used for extracting formal concepts and the hierarchy between them.

	Attr 1	Attr 2	Attr 3	Attr 4	Attr 5	Attr 6	Attr 7	Attr 8	Attr 9	Attr 10
Bordat						X	X			
Ganter		X					X			
Close By One (CBO)		X							X	
Linding					X				X	
Chein		X	X					X		
Nourine	X				X				X	
Norris	X								X	
Godin	X		X	X					X	
Dowling	X	X							X	
Titanic										X

Table 3.2 – *Properties of FCA algorithms for constructing concept lattices. [164]. Att 1: incremental; Att 2: uses canonicity based on the lexical order; Att 3: divides the set of concepts into several parts; Att 4: uses hash function; Att 5: maintains an auxiliary tree structure; Att 6: uses attribute cache; Att 7: computes intents by subsequently computing intersections of object intents; Att 8: computes intersections of already generated intents; Att 9: computes intersections of nonobject intents and object intents; Att 10: uses supports of attribute sets.*

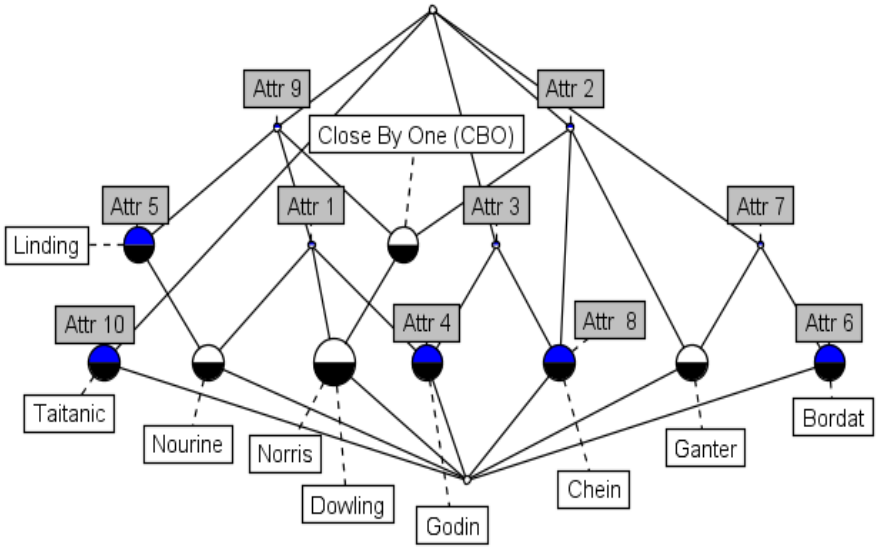


Figure 3.4 – *Concept Lattice of the context defined in Table* 3.2

Bordat follows a tree based top-down approach for retrieving and sorting domain concepts. Repetitive concept generation can be avoided by efficiently searching tree based structure of concepts.

NextClosure is proposed by Ganter [138] which computes all intents and extents, and pseudo-intents of a formal context in a certain linear order called *lectic order*. The main advantage of *NextClosure* is to apply an efficient canonicity test to extract only subset of formal concepts.

Close by One (CbO) [161] follows a similar approach as *NextClosure* for selecting a subset of concepts. Its time complexity is however higher than *NextClosure* due to an intermediate step

to compute closures more effectively. Different versions of this Algorithm have been proposed as *FCBO* (Fast Close By One), *PFCBO* (Parallel Fast Close By One) [165] which are used for improving canonicity test and speeding up the process of formal concepts extraction.

Chein has been introduced in [159], where the authors proposed an incremental Algorithm which iteratively generates a new layer of concepts based on previously constructed pairs of extent-intent. In each iteration, the intent of new concept is calculated based on intersection of intents of two existed concepts.

The ***Nourine*** Algorithm has also been proposed as a semi-incremental Algorithm which is based on a lexicographic tree with nodes and edges labeled by concepts and attributes respectively [166]. *Nourine's* Algorithm consists of two sub-algorithms. First, concepts are incrementally constructed in the form of a concept tree; the tree structure is then used to construct the line diagram or the lattice.

Dowling proposed [154] another incremental Algorithm for computing what they called knowledge spaces from two branches of knowledge. The authors argued that their proposed approach is suitable for integrating different knowledge spaces into a common knowledge space.

In [164], algorithmic complexity of mentioned approaches is studied for various size of formal contexts. However, different implementations of various algorithms have their own advantages. It has been shown that *NextClosure* is one of the most promising methods for constructing domain concept as it spends much less implementation time compared to its counterparts due to efficient canonicity test in each iteration.

As mentioned above, one of the main characteristics of *NextClosure* is generating concept intents and extents in lexicographic ordering. This specific mathematical ordering prevents the Algorithm to check whether each of 2^n subset of attributes could be an intent or not. Therefore, it takes much less time than other approaches for extracting concept intents. The following definitions and theorems (originally introduced in [138]) are crucial for understanding the approach of this Algorithm:

Definition 8 *Suppose that* $Y = \{1, ..., n\}$ *is the set of positive integer numbers from* 1 *to* n*, then lexicographic ordering among* $A, B \subseteq Y$ *with* $i \in \{1, ..., n\}$ *is defined as following:*

$$A \leq B \text{ iff } A <_i B \text{ for some } i, \text{ where}$$

.

$$A <_i B \text{ iff } i \in B - A \text{ and } A \cap \{1, ..., i-1\} = B \cap \{1, ..., i-1\}$$

Definition 9 *For* $A \subseteq Y$ *and* $i \in \{1, ..., n\}$*, the closure operator among* A *and* i *is defined as following:*

$$A \oplus i := \{((A \cap \{1, ..., i-1\}) \cup \{i\})^d\}^u$$

Theorem 3 *The lexicographic successor of B, where $B \subseteq Y$ -also called the least intent greater than B- is defined as follows:*

$$B^+ = B \oplus i \text{ where } i \text{ is the greatest one with } B <_i B \oplus i$$

An approach for implementing *NextClosure* is presented in Algo. 1. In general, the Algorithm attempts to discover all closures by exploiting lectic ordering. Starting from an empty set $\varnothing$, in each iteration, a new candidate is compared with previous concept (based on Definition 8) and if the condition is fulfilled, a new concept is generated by closure operator of Definition 9.

For the sake of clarity and in order to demonstrate how the Algorithm works, consider the formal context of animals and plants from Table 3.1. The Algorithm always starts with applying closure operator on a null attribute set and generates the base intent for all subsequent formal concepts ($A = \{\varnothing^d\}^u = \varnothing$). Now, we should calculate $A^+ = \varnothing^+ = \varnothing \oplus i$ where i is the largest one with $A <_i A \oplus i$. For doing so, we start the test for:

$$i = \{\textit{Animal, Plant, Lives on Land, Lives in Water}\}$$

and test whether $A <_i A \oplus i$. From the base concept ($Y = (\{\varnothing\}, \{\varnothing\})$), we take the attribute set, $Y = \{\varnothing\}$ and calculate $Y \oplus \{Lives \quad in \quad Water\}$ (*'Lives in Water'* has the highest position in columns). First we compute $\{\varnothing\} \cap \{Animal, Plant, Lives \quad on \quad Land\} = \{\varnothing\}$, then we append $\{Lives \quad in \quad Water\}$ and generate $\{\varnothing\} \cup \{Lives \quad in \quad Water\} = \{Lives \quad in \quad Water\}$. Hence, $\{\varnothing\} \oplus \{Lives \quad in \quad Water\} = \{\{Lives \quad in \quad Water\}^d\}^u = \{Lives \quad in \quad Water\}$. According to feasibility condition in Definition 8,

$$\{\varnothing\} <_{Lives \quad in \quad Water} \quad \{Lives \quad in \quad Water\}$$

Therefore, the set $\{Lives \quad in \quad Water\}$ is added as the intent of next concept in lectic ordering of set of concepts. Iterating over this process, *NextClosure* determines all the formal concepts of the context in Table 3.1. The Algorithm stops when the extracted set of intents is equal to Y in given formal context.

The implementation of this approach using native Python operators is given in Algo. 2. It is an almost textual translation of the mathematical definition. This implementation is very compact and it is particularly well suited for doing theoretical work.

Lexical ordering of the concepts as well as operational complexity of union and intersection operators in the $\oplus$ operator for generating the next intent from the previous intent could exponentially increase computational complexity of *NextClosure*. On the other hand, processing a large and dense data table requires fast algorithms, methods and tools that enhance the performance of the existing algorithms.

There are multiple dimensions that have to be considered. Very often, data analysis on large datasets requires some ad-hoc solutions. Therefore, key parts of algorithms should be easily implementable or at least easily adjustable. In naïve implementation of *NextClosure*, applying union and intersection operators on the sets of integers is not very efficient from an execution

Algorithm 1 *Generating formal concepts using NextClosure Algorithm* [138], [167]

Input: $\langle X, Y, I \rangle$ # a formal context
Output: $\{(A_i, B_i)\}$ # formal concepts with lexic. ordered intents $B_1 < ... < B_p$

1: $B_1 = (\emptyset^d)^u$ # the least intent
2: $i = 1$
3: **while** $B_i \neq Y$ **do**
4: $B_i = (B_i)^+$ # replace the current intent B_i with the next intent B_i^+ (see below)
5: $A_i = (B_i)^d$
6: $i++$
7: **end**

$B^+ = B \oplus k$ where $k \in \{1, ..., n\}$ is the greatest one for which $B <_k B \oplus k$
$B \oplus k = \left((B \cap \{1, 2, ..., k-1\} \cup \{k\})^d\right)^u$
$B <_k B' \iff k \in B' \setminus B$ and $B \cap \{1, 2, ..., k-1\} = B' \cap \{1, 2, ..., k-1\}$

Algorithm 2 *Set-based computation of next intent from previous intent in NC.*

```
1: def NextIntent(B): # returns the next intent B+ from the previous intent B
2:   for i from n-1 to 0:
3:     res = set(range(i)).intersection(B) # computes B ∩ {0,1,...,i-1}
4:     B' = Up(Down(res.add(i))) # returns B ⊕ i (full detail is skipped)
5:     if LexoLessThan(B,B',i):  return B' # return B' if B <_i B'
   ------------------------------------------------------------------------
6: def LexoLessThan(B,B',i): # check whether B <_i B'
7:   if i not in B'-B: return False # to see if i ∈ B' \ B
8:   h = set(range(i))
9:   if h.intersection(B) == h.intersection(B'): return True
10:  else:  return False
```

time point of view.

To tackle this problem, we proposed a new two-phased approach called *parallelization of binary based NextClosure*([58]) which reduces the computational cost of implementing the *NextClosure* using binarization and parallelization of codes in the Python environment. The first step (binarization of *NextClosure*) deals with implementing the binarized version of the original. The second aspect (parallelization of binary version) is to show how *NextClosure* can be sped up using parallelization techniques by activating all processing cores of modern CPUs.

3.3.2 Binary-based Implementation of *NextClosure*

The binary based implementation of *NextClosure* is proposed which speeds up the process of *NextClosure's* concept forming operators based on set-notations without changing the underlying logic of the Algorithm [58]. In the binary based implementation, we use another property of Python namely integers of arbitrary length. We can easily see that the set-operators can be implemented as logical operators on bit-string which are representing set membership relations. In this implementation an object $x \in X$ having attributes $y \in Y$ is represented by a bit-string where each 1 represents the presence of an attribute and 0 the non-presence. Therefore, an object $x \in X$ can be represented as a long integer in Python. This way we can speed up the execution of *NextClosure* by skipping unnecessary computations that are imposed by the nature of set-based calculations.

The main idea is very simple and works as follows. The first step is to interpret each binary row/column of the formal context $\mathbb{K}$ as an integer number expressed in the base 2 numeral system. For instance, considering the third object (corresponding to the third row of Table 3.1) is identified by $[1,0,1,0]$, and we view it as number $1+4=5$ (reading from left to right provides $2^0+2^2=5$), because the attributes of this object include 0 and 2. Note that in contrary to the standard notion, in which the attributes are indexed from 1 to n, in the binary version we index the attributes from 0 to $n-1$ as it makes computation easier.

A formal context $\mathbb{K}=(X,Y,I)$ in the binary-based approach is transformed into two vectors of rows and columns with m and n elements, respectively. In vector of rows, each element corresponding to one row of the table $\mathbb{K}$ indicated by a number. For example, the number 2^n-1 corresponds to an object which has all the attributes and number 0 corresponds to an object which does not have any attributes. Similarly, each element of columns, corresponds to a column of table I, lies in the range 0 and 2^m-1.

Such a representation makes many computations easier. For instance, the intersection of the current intent B with $0,...,i_1$ can be easily implemented in binary representation by calculating the remainder of the division of B by 2^i. The union operator $\cup$ with an attribute $\{i\}$ which is currently not present can also be easily implemented by adding 2^i. The Algo. 3 summarizes the functions used for the implementation of binary based *NextClosure*.

Algorithm 3 *Functions used in NextClosure - Binary approach - Python notation.*

```
1: def Up(A): # up operator, A is a number representing a subset of objects.
2:   r = 2**n - 1 # assume that objects within A have all attributes in common
3:   for j from 0 to m-1: # iteration over rows of the table
4:     if A & 1 « j: # if j-th bit of A is 1 (meaning that A contains j-th object)
5:       r = r & rows[j] # logical "and" between r and j-th row of table
6:   return r
   ------------------------------------------------------------------------
7: def Down(B): # down operator, B is a number representing a subset of attributes.
8:   r = 2**m - 1 # assume that attributes within B have all objects in common
9:   for i from 0 to n-1: # iteration over columns of the table
10:    if B & 1 « i: # if i-th bit of B is 1 (meaning that B contains i-th attribute)
11:      r = r & cols[i] # logical "and" between r and i-th column of table
12:  return r
   ------------------------------------------------------------------------
13: def LexoLessThan(B,B',i): # checks whether B <_i B' (lexographic inequality)
14:  if i==0:  return True
     # whether B ∩ {0,1,...i-1} = B' ∩ {0,1,...i-1}
15:  if (B % 2**i) != (B' % 2**i):  return False
     # if i belongs to B' but not to B
16:  if not (B & 1 « i) and (B' & 1 « i):  return True
17:  else:  return False
   ------------------------------------------------------------------------
18: def Oplus(B,i): # for the computation of B ⊕ i where B indicates subset of attributes
19:  if i==0 :  r = 0
20:  else:  r = B % 2**i # computing intersection B ∩ {0,1,...,i-1}
21:  r = r + 2**i # union with {i}
22:  return Up(Down(r))
   ------------------------------------------------------------------------
23: def NextIntent(B): # the next intent that is lexocographically bigger than B
24:  for i from n-1 to 0:
25:    B' = Oplus(B,i) # computing B ⊕ i, starting from biggest i
26:    if LexoLessThan(B,B',i) # if B <_i B' then the next intent is B'
27:      return B'
   ------------------------------------------------------------------------
28: def NextClosure(): # returns all the intents of a formal context
29:  res = [ ] # an empty list of intents
30:  Y = 2**m - 1 # corresponding to the set of all attributes
31:  B = Up(Down(0)) # the least intent corresponding to ((∅)^d)^u
32:  res.Append(B) # store the least intent
33:  while B != Y do: # until the current intent is not the set of all attributes
34:    B = NextIntent(B) # replace the current intent with the next one
35:    res.Append(B) # store the next intent
36:  endwhile
37:  return res
```

3.3.3 Parallel Implementation of Binary-based *NextClosure*

Parallel computing is gaining high interest from different researchers and there is a huge shift in many hardware manufacturers to improve computing power of their devices by developing processors with multiple cores. Therefore, many researchers in different domains are developing new ideas on re-designing algorithms which could fully utilize the power of multi-core systems.

Parallelization of formal concepts extraction has been also widely studied in the past years. In [168], authors proposed a parallel Algorithm based on the closure search space partitioning for computing formal concepts. In their proposed method, dividing the closure search space into several subspaces is based on some predefined criteria. An intermediate structure is also employed in order to recognize the valid subspaces (i.e. the subsets having higher probability to generate formal concepts) and efficiently search for closures. The parallelization is then applied on independent task of searching formal concepts in subspaces. The authors only provided the mathematical formalism of their approach and no implementation is presented.

In [169], a new *NextClosure*-based Algorithm called *ScalingNextClosure* has also been proposed to extract the formal concepts in a partitioned search space. Their proposed Algorithm has two main steps: 1) determining potential partitions, 2) generating formal concepts within each partition using the original *NextClosure* Algorithm. They have argued that the since concept forming process could be independently applied on each partition, the parallelization of formal concepts extraction can be directly achieved by decomposing the search space. However, their work does not consider the evaluation of context density in the final performance and not enough information is provided regarding number of computers being used, environmental configuration and etc.

A complementary work of [169] is done in [170] in which the authors exhibit a new strategy for creating a formal context by considering workload balancing among nodes. Their proposed Algorithm calculates the workload for each node based on the number of attributes in the formal context as well as the maximum size of each partition.

A similar work is also in [171] in which a parallel version of the Algorithm being presented in [172] and [173] is proposed. Their proposed Algorithm is very similar to the famous *Close-by-One* Algorithm ([174]) with a recursive procedure for computing formal concepts. In the proposed recursive approach, each node generates a disjoint set of formal concepts by simulating the sequential procedure. However, the authors have not mentioned which data structures are used for implementing the Algorithm.

On the other hand, authors in [175] have presented a slightly different approach for constructing concept lattice based on divide and conquer strategy. Their strategy is to generate a full lattice by merging partial lattices (applying Cartesian product on each pair of partial lattices). At the core of their strategy, a new technique has also been proposed to identify invalid nodes and they claimed that this approach largely outperforms classical *NextClosure* in the sense

that it could rapidly identify invalid nodes. However, no result has been shown on how their Algorithm behaves compared to original *NextClosure*.

The main contribution of our work in parallelizing concept formation using *NextClosure* relies on simulating a binary-based version of *NextClosure* on each available core of a multiprocessor machine. However, although it might not be the best parallel implementation, it fulfills the required flexibility for the algorithms to maintain the possibility of their easy adjustment to new situations. It has been shown that parallel computation can significantly speed up the execution even further, especially for large datasets. Finally, the effect of the number of processors being involved in processing on the time spent for the execution is studied. We show that the time required for the execution decreases as the number of cores increases. This is particularly important for the analysis of Big Data.

To show the implementation of a parallel version of the binary-based *NextClosure* Algorithm, we decided to illustrate how to parallelize the *up* operator. This parallelization is straightforward as discussed in the following.

Algorithm 3 shows that the *up* operator requires an iteration over all the objects corresponding to the rows of the binary table for a given formal context. For large datasets, this iteration could be very time-consuming. Using all available cores can speed up this operation (that is frequently used in the *NextClosure* implementation) by simply asking each core to iterate over a subset of rows. In other words, we split the binary representation in equal chunks and send them to workers. Once the processing of each worker is finished, the results of all workers are merged to return the final output. This simple technique can significantly seed up the execution time. The Python code for how the *up* operator can be parallelized in the binary-based version is given in Algorithm 4. Although the full details of the implementation are not provided, it gives an idea of how to parallelize the *up* operator in the *NextClosure* Algorithm.

Algorithm 4 *Parallelization of the up operator in the binary-based code.*

```
1: def UpParallel(worker, A):                          ▷ each worker is assigned to a core
2: Results = [worker.apply_async(Up, args=(A, RowIndexStart, RowIndexEnd))
   for core in range(NumberOfCores)]   ▷ each worker is responsible for processing a
   chunk of table
3:   Y = 2**n - 1
4:   for res in Results:
5:     Y = Y & res.get()                          ▷ merging the results obtained by workers
6:   return Y
   ____________________________________________
7: with multiprocessing.Pool(NumberOfCores, Func, FuncArg) as workers:
8:   NextClosure(workers)                         ▷ execute NextClosure using all workers
```

The following comparisons are conducted to show the behaviour of the proposed approach for a wide range of context size. For the sake of implementation, a machine with the following

specifications is used: *1x AMD64* with *8 core Processor @ 3.11 GHz, RAM: 16 GB, OS: Windows 10 Pro.*

Figure 3.5 shows that the time required for the execution using the binary-based approach is less than set-based version. Moreover, the difference between corresponding execution times increases as the number of objects increases. That means the significance of binary computation in reducing the required time for the *NextClosure* execution is highlighted especially for large datasets with large numbers of rows/columns. We further compared our binary code with the *concepts* module written in Python ([176]) that is frequently used to analyse formal contexts. Table 3.3 reports the average time spent for executing *NextClosure* on different-sized datasets. The results are averaged over 5 different simulations. It clearly shows that the binary-based approach is much faster than the concepts module, even for small datasets. These numbers should not be considered as an absolute comparison, because both implementations are aiming at very different targets. The intention is only to show that our basic implementation is quite efficient, which also underlines the importance of the parallel results, since we compare them with a quite efficient sequential Algorithm.

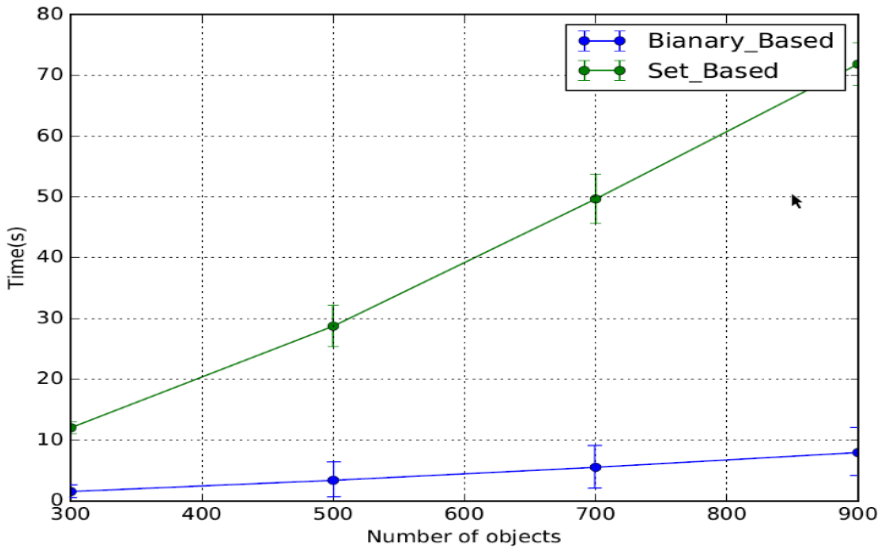


Figure 3.5 – *The binary-based execution (blue curve) of the NextClosure Algorithm requires less time (vertical axis) than its set-based version (green curve). The number of attributes in this simulation is set to* 15 *and the number of objects (horizontal axis) is varied from* 300 *to* 900. *Each simulation is repeated* 5 *times with different datasets to make sure that the result is not biased by the choice of the binary table. The dots on the curves indicate the mean spent time and the error bars indicate the corresponding standard deviation.*

The next comparison, shown in Figure 3.6, is conducted to investigate whether parallel programming can speed-up the *NextClosure* implementation and if yes, under which conditions. The expectation was that if the number of rows is sufficiently large, parallelization has a positive effect in reducing the time required for *NextClosure* executions. This, however, might not be the case if the dataset is too small. The reason is that task distribution between the

Table 3.3 – *Average time required for executing NextClosure using binary-based and Python module* `concepts`.

Spent time [s] for executing *NextClosure*		
Table size	Binary-based	`concepts` Module
10 × 5	0.0004	0.0013
20 × 10	0.0055	0.0257
50 × 10	0.0180	0.2309
100 × 10	0.0365	1.0131
200 × 12	0.2316	12.368
300 × 15	1.2937	140.37

available cores is also time-consuming. Therefore, it might not be reasonable to spend some time for task distribution over available cores when a sequential approach can handle the task itself in a very short amount of time. The same argument can be used for explaining why we did not parallelize the down operator since the number of attributes in our examples is not big enough. The hypothesis is tested by executing the *NextClosure* Algorithm on databases with 15 attributes; but with several number of rows ranging from 3000 to 9000. As depicted in Figure 3.6 the average time spent for executing *NextClosure* using the proposed parallel approach (with all the 8 cores of the machine) is less than for the best sequential approach when the number of rows is bigger than roughly 4200. When the number of rows is small, the sequential approach takes less time in average.

Finally, in order to see how the time required for the *NextClosure* execution varies as a function of the number of cores involved in processing, we applied *NextClosure* using different numbers of cores. Figure 3.7 shows that when the number of rows corresponding to the number of objects in a dataset reaches a certain level, then the time required for *NextClosure* decreases as the number of cores increases. This makes sense because parallelization in general speeds up the execution. This statement is, however, not always true. In the case when the number of objects is small the execution time goes up with the number of cores. This is because that the task distribution among the cores is time-consuming and this time actually increases with the number of cores.

3.4 A Proposed Hybrid Approach for Constructing Concept Lattice

In addition to formal concepts, concept lattices are frequently used for knowledge representation [177] and data mining [178] especially for transforming data into human understandable structures.

Many studies have been conducted on the construction of concept lattices from binary relation. Essentially not all of the proposed algorithms are originally designed for computing both concepts and the Hasse diagram. *Bordat's* [158] and *Godin's* [179] algorithms, for instance, are

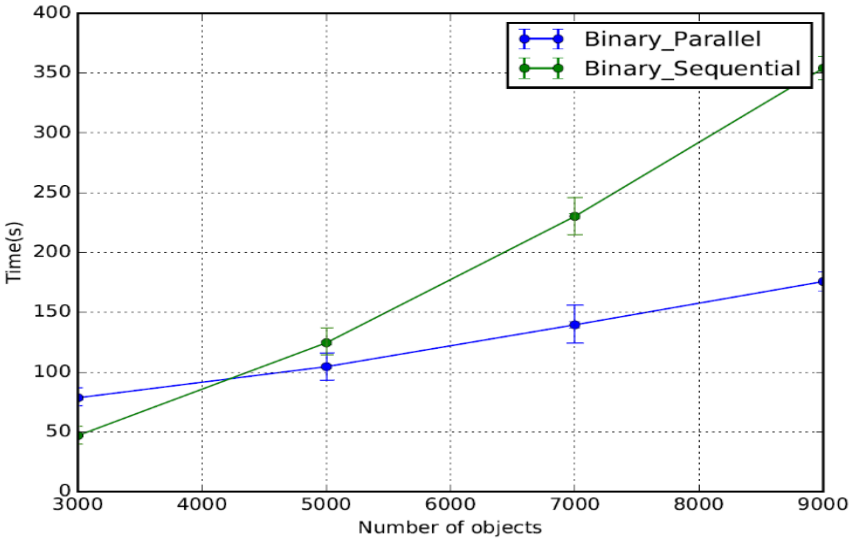


Figure 3.6 – *Parallel versus sequential execution of the NextClosure Algorithm. The time (vertical axis) required for the parallel execution of NextClosure (blue curve) is less than the time needed for our best sequential implementation (green curve) if the number of objects is sufficiently big (here* > 4200*). The figure depicts the average time spent for executing the NextClosure Algorithm on databases with* 15 *attributes and different number of objects (horizontal axis). Each database is created artificially as a binary table whose elements are independently chosen as* 0 *and* 1 *with probability* 0.5*. Each simulation is repeated for* 5 *times. Error bars indicate the standard deviation of the corresponding spent times.*

designed to generate both the formal concepts and the concept lattice. *NextClosure*, on the other hand, requires an additional effort to build the line diagram of generated concepts.

In [166] the authors proposed an efficient Algorithm called *Nourine* Algorithm, which can be used for building lattices in a general framework. The authors argued that this model improves algorithms [159, 157, 158, 179, 155] for Galois lattice generation as well as [180] for building Dedekind-MacNeille completion and [181] for maximal antichain formation from a partially ordered set.

However, the *Nourine* Algorithm is suitable for both, generating formal concepts and building concept lattices, and is fast in building concept lattices compared to many other algorithms, *NextClosure* is still the most sophisticated method for concept generation due to it's deeper insight to avoid redundant generation.

To utilize the benefits of the *NextClosure* and *Nourine* algorithms together, we propose a hybrid approach which combines the advantages of *NextClosure* (for generating formal concepts) and of the *Nourine's* Algorithm (for building concept lattice) [57]. It has been shown through the implementation that especially for a dense formal context, such a hybrid model usually outperforms pure *NextClosure* and pure *Nourine*.

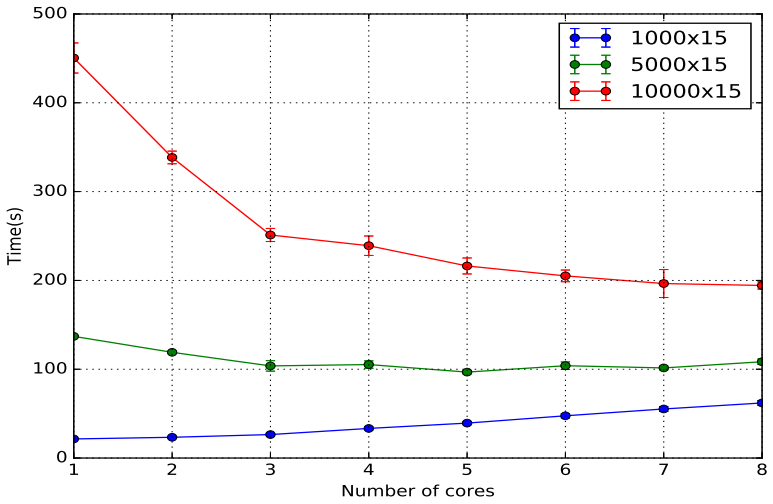


Figure 3.7 – *Parallel execution of the NextClosure Algorithm using different number of cores. Three databases with* 15 *attributes and different number of objects are chosen. Increasing the number of cores reduces the time spent when the number of rows is sufficiently large (red curve corresponding to* 10,000 *rows). If the number of rows is not big enough (blue curve with* 1,000 *rows), the time spent increases with the number of cores. This is because the time required for task distribution over the cores itself is not negligible. For the intermediate number of rows (green curve with* 5,000 *rows) the time spent lies between the two extremes. Once again each dataset is created artificially as a binary table whose elements are independently chosen as* 0 *or* 1 *with probability* 0.5*. Each simulation is repeated for* 5 *times. Error bars indicate the standard deviation of the spent time.*

Nourine is an incremental Algorithm for building concept lattice of a given formal context $\mathbb{K} = (X, Y, I)$ in two main steps:

1. Generating the family $\mathscr{F} = \{(F_i, \gamma(F_i))\}$ in a lexicographic tree format (Algo. 5): This Algorithm receives as the input a *basis* $\mathscr{B} = \{X \setminus y^d \mid y \in Y\}$ as a set of subsets of objects, and returns a set of pairs $\{(F_i, \gamma(F_i))\}$ which is isomorphic to the set of formal concepts $\{(A_i, B_i)\}$, calculated by *NextClosure*. Algo. 5 returns the list of all *extent complements* $X \setminus A_i$ and the list of all *intents* B_i named F_i and $\gamma(F_i)$, respectively.

2. Computing covering relations of elements $\mathscr{F}$ Algo. (6): In the second step, Algorithm benefits from the lexicographic tree of the family $\mathscr{F}$ from the previous step to calculates the adjacency lists $ImSucc$ of the covering graph. It has been proven that the computed adjacency lists of the covering graph by *Nourine* is isomorphic to the Galois lattice of a formal context.

Let us consider the given context of Table 3.1. For simplicity, we renamed rows and columns and the alternative table has been generated as shown in Table 3.4. Based on definition,

Algorithm 5 *Generating formal concepts using Nourine Algorithm* [166], [182]

Input: $\langle X, Y, I \rangle$ # a formal context
Output: $\{(F_i, \gamma(F_i))\}$ # a family of pairs describing formal concepts

$\mathscr{B} = \{X \setminus y^d \mid y \in Y\}$ # the basis: a set of subsets of X
$\mathscr{F} = \{\emptyset\}$ # the root of lattice
for each B in $\mathscr{B}$ **do**
  for each F in $\mathscr{F}$ **do**
    $F' = F \cup B$
    if $F' \notin \mathscr{F}$
      $\mathscr{F} = \mathscr{F} \cup \{F'\}$ # append F' to $\mathscr{F}$
      $\gamma(F') = \gamma(F) \cup \{y_B\}$ # note that $B = X \setminus (y_B)^d$ and $y_B \in Y$
    end
  end
end

Algorithm 6 *Building concept lattice using Nourine Algorithm* [166, 182]

Input: $\mathscr{F} = \{F_i\}$, $\{\gamma(F_i)\}$, $\mathscr{B}$ # basis $\mathscr{B}$ and the family of pairs $\{(F_i, \gamma(F_i))\}$
Output: $\{ImSucc(F_i)\}$ # immediate successors of $\{F_i\}$, i.e., $ImSucc(F_i) \rightarrow F_i$

for each F in $\mathscr{F}$ **do**
  $count(F) = 0$
  for each $B \in \mathscr{B} \setminus \mathscr{B}_F$ **do** # we define $\mathscr{B}_F = \{X \setminus y^d \mid y \in \gamma(F)\}$
    $F' = F \cup B$
    $count(F')++$
    if $|\gamma(F')| = count(F') + |\gamma(F)|$
      $ImSucc(F) = ImSucc(F) \cup \{F'\}$
    end
  end
end

I	a	b	c	d
1	x		x	
2	x		x	
3		x	x	
4		x	x	
5	x			x
6		x		x
7		x	x	x

Table 3.4 – *Animal and planet formal context 3.1 with simplified name of rows and columns.*

$\mathscr{B} = \{a = \{3467\}, b = \{125\}, c = \{56\}, d = \{1234\}\}$. Considering: $\mathscr{F}_{\mathscr{B}} = \left\{\bigcup_{B \in I} B | I \subset \mathscr{B}\right\}$, the family: $\mathscr{F} = \{\{\}, \{3467\}, \{125\}, \{1234567\}, \{56\}, \{34567\}, \{1256\}, \{1234\}, \{123467\}, \{12345\}, \{123456\}\}$ is then calculated for the given basis. As explained, each element $\mathscr{F}_{\mathscr{B}}$ is represented by a pair $(F, \gamma(F))$ where $\gamma(F) = \{B \in \mathscr{B} | B \subseteq F\}$. For instance, $\gamma(\{34567\} = \{ac\})$ where a, c are elements of basis $\mathscr{B}$. For the purpose of the tree based representation of the extracted family, for each element $F = \{34567\}$ of $\mathscr{F}_{\mathscr{B}}$ with $a < c$, the Algorithm associates a path from the root to a marked node in the tree. Each marked node is considered as a fix point in the corresponding Galois lattice. The tree representation of the constructed family is shown in Figure 3.8.

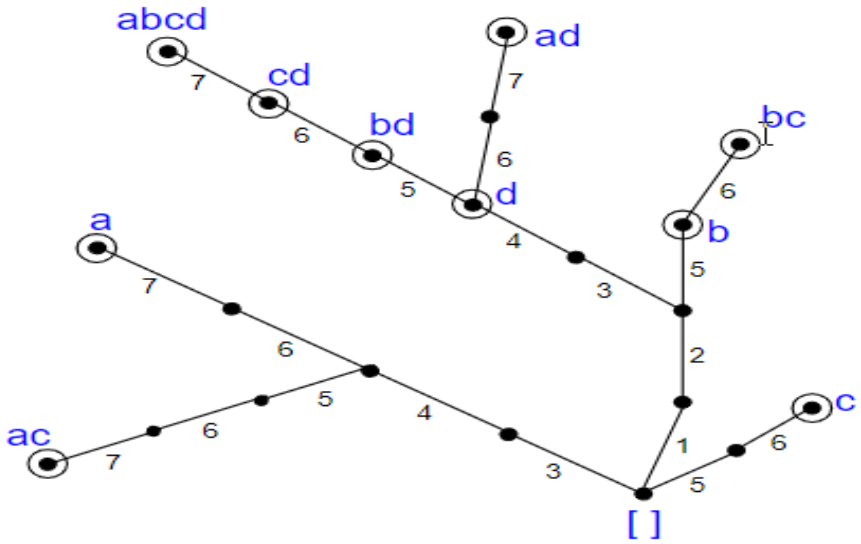


Figure 3.8 – *Nourine lexicographic tree of generated family $\mathscr{F}_{\mathscr{B}}$. For each $F \in \mathscr{F}_{\mathscr{B}}$ a path from root to a circled node is associated and tagged with values of F. Its corresponding circled node is also tagged with $\gamma(F)$.*

The corresponding concept lattice of $(\mathscr{F}_{\mathscr{B}}, \subseteq)$ is generated based on computing the adjacency matrix which stores information on the selected covering element for each $F \subset \mathscr{F}_{\mathscr{B}}$. For each element of $\mathscr{F}_{\mathscr{B}}$, one successor is chosen from list of candidates (all belong to $\mathscr{F}$) that satisfies the following condition in Theorem 4 ([166]):

Theorem 4 *Let $F, F' \in \mathscr{F}$ such that $F \subset F'$. Then F' is immediate successor of F $(F \prec F')$ if and only if $\{B_1 \setminus F = B_2 \setminus F | \forall B_1, B_2 \in \Delta(F', F)\}$.*

In the theorem above, for each pair of elements (F', F) from $\mathscr{F}_{\mathscr{B}}$, we have $\Delta(F', F) = \gamma(F') \setminus \gamma(F)$.

The corresponding Galois lattice of the lexicographic tree presented in Figure 3.8 is shown in Figure 3.9. Note that the generated Galois lattice by *Nourine* is isomorphic to the lattice in Figure 3.2 such that the intent of each node in Figure 3.9 is equivalent to its corresponding node in Figure 3.2. *Nourine's* auxiliary tree structure enormously speeds up the process of constructing the concept lattice for the extracted concepts of the given formal context.

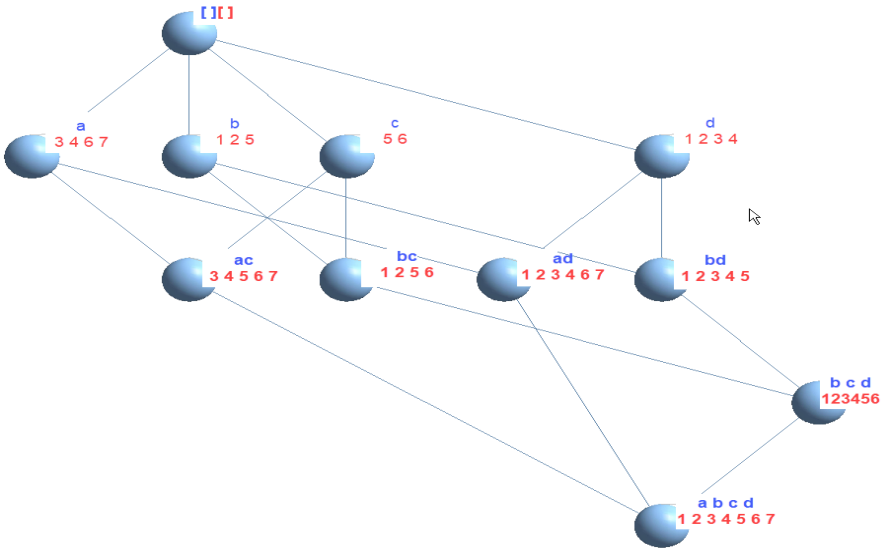


Figure 3.9 – *Nourine's Generated Galois lattice of the lexicographic tree in Figure 3.8. Note that this lattice is isomorphic to the lattice of Figure* 3.2

As discussed above and shown in Algorithm 1, *NextClosure* computes all fixpoints of an arbitrary closure operator in a lexicographic order. It is however, not designed to generate the corresponding concept lattice. We have extended the original *NextClosure* Algorithm to generate the corresponding concept lattice of a lexicographically ordered set of intents. For doing so, an adjacency matrix M (with size $p \times p$ where p is size of extracted intents) is created to indicate which concepts are adjacent. We set $M(i, j) = 1$ if $C_i \rightarrow C_j$ and $M(i, j) = 0$ otherwise.

In the following we assume that all formal concepts of a given context are available and so we have access to the list of all intents $\{B_1, ..., B_p\}$. If the formal concepts are generated by the *NextClosure*, the list of intents are lexicographically ordered. Without loss of generality, we can assume that the intents are sorted as follows:

$$B_1 < B_2 < ... < B_p,$$

where B_1 and B_p correspond to the attributes of *the most general* concept C_1 and *the most specific* concept C_p, respectively. Thanks to the lexicographically ordered intents, a concept C_i is *never* a super-concept of C_j (i.e., $C_j \not\prec C_i$), if $j < i$. This is because B_i is never a subset of B_j when B_j is lexicographically less than B_i (i.e., $B_j < B_i \Rightarrow B_j \not\supset B_i \Rightarrow C_j \not\prec C_i,\ \forall\, j < i$). (Further details about lexicographically ordered sets and their properties is presented in [167].)

A direct consequence of the statements above is that the entries of the adjacency matrix $M(i, j)$ for all $j < i$ are 0. In other words, the adjacency matrix M is an *upper-triangular* matrix. Note that if we did not have access to the list of lexicographically ordered intents, we could not easily argue that the lower triangular entries of the adjacency matrix M are all zero. This feature significantly speeds up generating concept lattices using naive methods, because we do not need to spend time for calculating half of the entries of the adjacency matrix M. However, even by using this trick we will show later that it does *not* outperform *Nourine's* Algorithm for building concept lattice.

Algorithm 7 illustrates how concept lattices can be built using lexicographically ordered intents calculated by *NextClosure*. In summary, if $j \leq i$, we set $M(i, j) = 0$ and for the rest of the entries in the adjacency matrix (i.e., for $j > i$), we set $M(i, j) = 1$ (i.e., $C_i \rightarrow C_j$) only if three conditions are simultaneously fulfilled:

1. $j > i$
2. $B_i \subset B_j$
3. No other B_m exists such that $B_j \subset B_m \subset B_j$

Although generating formal concepts is significantly sped up by lexicographically ordered intents (as it is the case in *NextClosure*), naive methods for building concept lattices (from those calculated formal concepts), e.g. Algorithm 7, is not the most efficient method. This is because *NextClosure* is originally designed for generating formal concepts and not for building concept lattices. In contrast to *NextClosure*, *Nourine's* Algorithm was originally designed for building lattices including the concept lattice, although it can also be used for generating formal concepts.

NextClosure relies on a list of intents which are lexicographically ordered, the *Nourine* Algorithm on the other hand, does not provide an ordered list of intents at all. Therefore, the *Nourine's* Algorithm is not as fast as *NextClosure* for generating formal concepts. However, due to the maintenance of an auxiliary tree structure for representing the formal concepts, *Nourine* is one of the most efficient algorithms that currently exist for building concept lattices (see [166, 182] for more information on *Nourine's* Algorithm).

Our proposed hybrid model (Algorithm 8) benefits from the advantages of both *NextClosure* and *Nourine*. It generates the set of formal concepts $\{(A_i, B_i)\}$ using the *NextClosure* 1 and then

Algorithm 7 *Extended NextClosure for constructing concept lattice using lexicographically ordered intents. NextClosure.*

Input: a list of lexicographically ordered intents $B_1 < B_2 < ... < B_p$
Output: a $p \times p$ adjacency matrix M, where $M(i, j) = 1$ indicates $C_i \rightarrow C_j$

$M(i, j) = 0,\ \forall i, j$
for $i = 1 : p$ **do**
  $temp = [True, ..., True]$ # a list of p Boolean $True$
  for $j = i + 1 : p$ **do** # we only need to check for $j > i$
    if $temp[j] == False$ # there was an $m < j$ such that $B_i \subset B_m \subset B_j$
      jump to the next j
    if $B_i \not\subset B_j$
      jump to the next j
    for $k = j + 1 : p$ **do**
      If $B_j \subset B_k$ # if the condition is true then $B_i \subset B_j \subset B_k \Rightarrow M(i, k) = 0$
        $temp[k] = False$ # to prevent unnecessary check for $M(i, k)$ with $k > j$
    end
    $M(i, j) = 1$
  end
end

transform it to a set of $\{(F_i, \gamma(F_i))\}$ using $F_i = X \setminus A_i$ and $\gamma(F_i) = B_i$. Then it applies *Nourine's* Algorithm 6 on the set $\{(F_i, \gamma(F_i))\}$ to build the concept lattice.

Algorithm 8 *Generating formal concepts and the concept lattice using hybrid Algorithm.*

Input: $\langle X, Y, I \rangle$ # a formal context
Output: a $p \times p$ adjacency matrix M, where $M(i, j) = 1$ indicates $C_i \rightarrow C_j$

$M(i, j) = 0\ \forall i, j$
Generate formal concepts $\{(A_i, B_i)\}$ using *NextClosure* Algorithm 1
Create a set $\{(F_i, \gamma(F_i))\} = \{(X \setminus A_i, B_i)\}$
Find $\{ImSucc(F_i)\}$ using *Nourine* Algorithm 6
$M(k, i) = 1$ for all $F_k \in ImSucc(F_i)$ # F_k is connected to F_i, i.e., $F_k \rightarrow F_i$

We first illustrate that the concept lattices generated by Algorithm 7 (Figure 3.10) and Algorithm 6 (Figs 3.11) for a given formal context $\langle X, Y, I \rangle$ (Table 3.5) are isomorphic. Each formal concept in these two figures are represented by an ellipse. Note that each red ellipse in the lattice of Figure 3.11 corresponds to a blue ellipse in the lattice of Figure 3.10 with $(F_i, \gamma(F_i)) = (X \setminus A_i, B_i)$.

We then compare the time spent for generating formal concepts and building concept lattices using the different algorithms mentioned before. For this purpose we created a variety of artificial formal contexts with different size and sparsity (see Table 3.6).

Figure 3.12 clearly shows that *NextClosure* requires much less time than *Nourine* for generating

Table 3.5 – *A formal context with the object set* $X = \{0,1,2,3,4\}$ *and the attribute set* $Y = \{a,b,c,d,e\}$. *For this formal context, the corresponding* basis *that is given as input to the Nourine Algorithm is* $\mathscr{B} = \{X \setminus y^d \mid y \in Y\} = \{\{0,2,3\},\{1,3,4\},\{1,2,3\},\{0,3,4\},\{1,2,4\}\}$.

I(x,y)	**a**	**b**	**c**	**d**	**e**
0	0	1	1	0	1
1	1	0	0	1	0
2	0	1	0	1	0
3	0	0	0	0	1
4	1	0	1	0	0

Table 3.6 – *List of formal contexts, created artificially, that are used for the comparisons. Here, Density determines the fraction of* 1 *in the corresponding binary table.*

Formal Contexts			
Name	Objects	Attributes	Density
D1	25	26	50%
D2	26	17	76%
D3	500	15	30%
D4	1000	20	15%
D5	1000	15	60%
D6	2688	30	11%
D7	500	15	60%
D8	600	15	50%

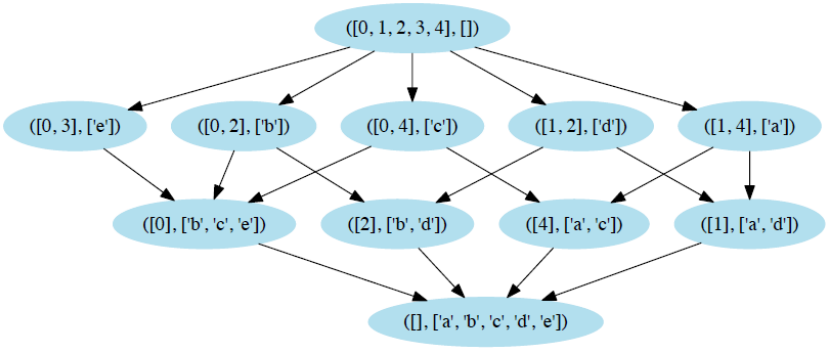


Figure 3.10 – *The concept lattice in Table 3.5 generated by Algorithm 7 using lexicographically ordered intents derived using the NextClosure Algorithm 1. Each concept C_i is represented by a blue ellipse and contains its corresponding (extent, intent), i.e., (A_i, B_i). The very top ellipse corresponds to the most general concept and the very bottom one corresponds to the most specific concept.*

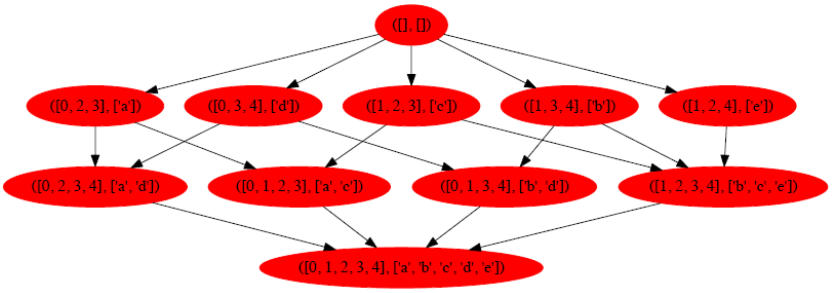


Figure 3.11 – *The concept lattice in Table 3.5 generated by Nourine's Algorithm [166]. Each concept C_i is represented by a red ellipse and contains its corresponding $(F_i, \gamma(F_i))$. Note that this lattice is isomorphic to the lattice of Figure 3.10. Note also that each red ellipse in this lattice corresponds to a blue ellipse in the lattice of Figure 3.10 with $F_i = X \setminus A_i$ and $\gamma(F_i) = B_i$.*

formal concepts. This difference gets bigger when datasets are getting bigger.

Figure 3.13 shows the time spent for building the concept lattices using the *Nourine* Algorithm 6 and the *Extended NextClosure* Algorithm 7 (that uses formal concepts calculated by *NextClosure*). Results indicate that *Nourine* performs much better than its counterpart. Note that the time reported for building concept lattices is only for the calculation of the adjacency matrix and not drawing the graph.

Although *NextClosure* performs faster than *Nourine* for generating formal concepts, *Nourine* builds the concept lattice much faster than *Extended NextClosure*. Therefore, the *pure Nourine* (algorithms 5, 6) outperforms the *pure NextClosure* (algorithms 1, 7). The *hybrid* model (Algorithm 8, or equivalently algorithms 1, 6), however, can compete with the *pure Nourine* (see Figure 3.14). That means, although as our results show the hybrid model not always outperforms *pure Nourine*, we can see that for large datasets with high density (i.e., a large percentage of 1 in the binary table of the corresponding formal context), the hybrid model

surpasses the pure *Nourine* Algorithm. Therefore, in the context of Big Data and depending on the characteristics of the formal context (such as the density), it might be worth to first generate formal concepts using the *NextClosure* Algorithm, and then to build the concept lattice using *Nourine's* Algorithm. Our prediction is that for very big datasets with large density, the hybrid model surpass the pure *Nourine* Algorithm while for very large datasets with low density, the pure *Nourine* Algorithm outperforms the hybrid model.

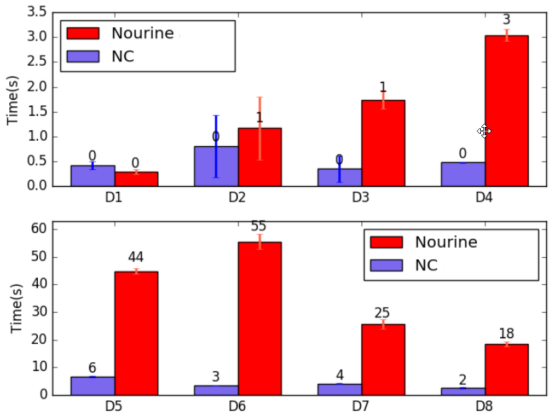


Figure 3.12 – *The average time spent for calculating formal concepts using the NextClosure Algorithm 1 (blue) and the Nourine Algorithm 5 (red) for different datasets in Table 3.6. Error bars indicate the standard deviation (each implementation is executed* 5 *times with a different random dataset).*

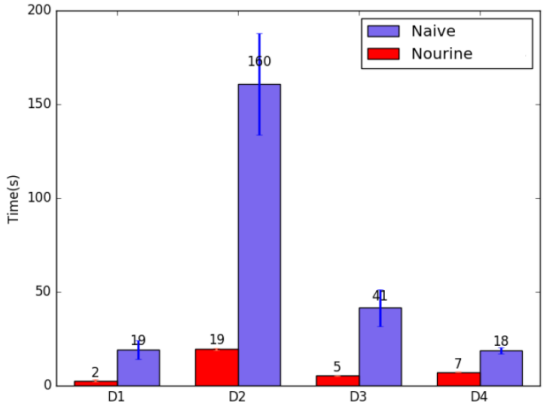


Figure 3.13 – *The average time spent for building the concept lattice using the Nourine Algorithm 6 (red) and the Naïve Algorithm 7 (blue) for some of the datasets in Table 3.6. Error bars indicate the standard deviation (each implementation is run* 5 *times with a different random dataset).*

3.5 Conclusion

In this chapter we described one of the major components of the proposed ontology learning pipeline from textual data. We demonstrated how FCA can be used to extract formal concepts

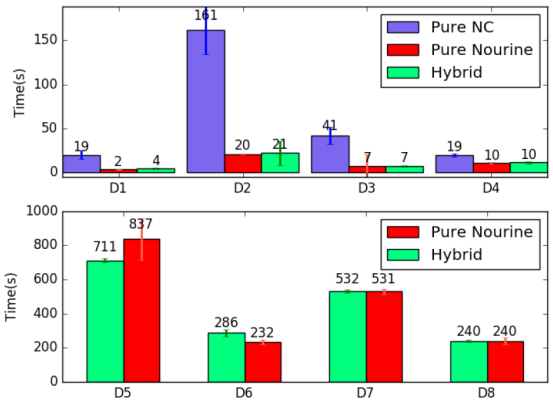


Figure 3.14 – *The average time spent for both generating formal concepts as well as building concept lattices using pure NextClosure (Algorithms 1, 7) (blue), pure Nourine (Algorithms 5, 6) (red) and our proposed hybrid Algorithm 8 (green), for different datasets in Table 3.6. Time for pure NextClosure is not reported for datasets* **D5**-**D8** *because it was shown to be more time consuming than pure Nourine and the hybrid model for large datasets.*

as well as the hierarchy between them. The next step was then to further process the concept lattice in order to get a more suitable reduced form (i.e., GSH) to be used as the cornerstone of the ontology (which will be described in next chapter as the third phase of our processing pipeline).

The key methodology we used in this chapter was FCA, where we described two of the most well-known algorithms for extracting formal concepts as well as building concept lattices such as *NextClosure* and *Nourine*. We further proposed solutions to speed up the processing time using binarization and parallelization of the *NextClosure* Algorithm. We also proposed a hybrid model where we could benefit from the advantages of both *NextClosure* and *Nourine* for the purpose of formal concept extraction, and lattice generation, respectively. Throughout this chapter, we also reviewed previous works on FCA and its significance for addressing many challenge in the domain of data analysis and knowledge representation.

4 Ontology Engineering Based on Prominent Mapping Rules

"You can get significant analytical lift when you combine information from the variety of sources available to you"

- Seth Grimes

4.1 Overview

In this chapter, the aim is to propose a methodology for building an ontology using Galois Sub Hierarchy (GSH) [145],[146],[147] , [148],[149] that was derived from a formal context. In the last two chapters, it has been shown how the most relevant statements within a text corpus can be extracted, and how a formal context can be further transformed into a GSH, respectively.

To build an ontology from a GSH, we need to identify and provide all other components whose presence is needed for having a well-formed ontology. In this chapter we will start with reviewing some background information about ontology, and then we will introduce our methodology for how to convert a GSH structure to an ontology based on a set of proposed mapping rules. The built ontology is then further completed using triplets of interest that have been extracted from text.

4.2 Ontology: Background and Context

4.2.1 An Introduction to Ontology

The term ontology is raised from philosophy and explains the nature of reality to address what exists in a specific domain [Wikipedia]. In computer science, however, it means a model for describing a domain consisting different concepts, properties and relationships between the

concepts in the form of directed graphs.

Among other definitions, [183] has defined ontologies as a *"formal, explicit specification of a conceptualization"*. There exists two major components in this definition: *specification* and *conceptualization*. "Explicit specification of a conceptualization" means that an ontology is a formal description of the main concepts and relationships that could exist for an agent or among a community of agents.

The word *conceptualization* represents the intentional semantic structure which encodes the implicit knowledge constraining the structure of a piece of a domain. *Specification*, on the other hand, is used for making ontological commitments that are defined as an agreement to use a vocabulary with which queries and assertions are exchanged among agents and grants the consistency for communication. This definition also requires the name of the concepts and also how they are connected to each other to be explicitly defined with a rich formal language such as *Web Ontology Language (OWL)* [184].

Hierarchical classification of domain specific items in the form of subsumption relations (also called taxonomical relations) generates the backbone of any ontology. The usefulness of ontology, is however, beyond structuring the concepts in a hierarchical manner and is to express a set of axioms and restrictions which enable inference of additional knowledge and implicit relationships. Nevertheless, the term ontology is most often confused with taxonomy and thesaurus. This misunderstanding is mainly caused by diverse approaches for metadata management in a specific domain. To resolve the ambiguities between different terms and in order to create a shared vocabulary for better communication and collaboration, the authors in [185] classified different types of knowledge base systems based on their inherent structure and types of content.

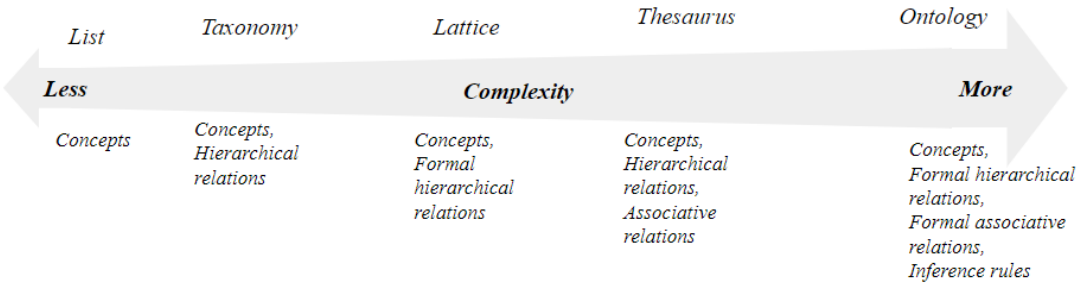


Figure 4.1 – *Structural complexity of Knowledge Base Systems.* ([185])

Knowledge based systems may have different levels of complexity. As illustrated in Figure 4.1, *lists* are at the simplest end of the spectrum as they solely contain linearly organized items and their corresponding attributes. *Taxonomies* on the other hand, have a hierarchical structure being composed from several is-a abstractions among domain items and their attributes. Therefore, as a restriction, they may not necessarily comprise a comprehensive domain knowledge. A formal representation of a taxonomy by describing domain objects in

terms of their corresponding attribute set is called *lattice*. High level mathematical rigour of the lattice enables a taxonomy to be manipulated using FCA. A detailed overview around FCA and lattices is provided in previous chapter. *Thesaurus* could be considered as an extension of a taxonomy in which further enhancements are offered such as integration of semantic relationships as well as a definition of equivalence of specified domain items. It performs beyond hierarchical relationships and is intended to contribute to the reasoning power that is to be built into the applications. *Ontologies* are located at the rightmost side of the spectrum, as they are more complex than their counterparts. They extend knowledge base systems beyond controlled vocabularies, make intensive use of axioms for specification, and enable software agents to exchange, share, reuse and reason about the concepts and relations based on inference rules in the form of meta-relations, constraints, and conditional rules.

Despite the fact that a complete set of inference rules and a collection of structured information is highly required from computers to be able to conduct automatic reasoning, fully automated *ontology learning* is still not an easy task in real world problems. Hence, the majority of ontology learning systems mainly rely on creating lightweight ontologies for a wide variety of applications such as knowledge management, data integration and document retrieval [186], [187].

4.2.2 Ontology Learning Languages

Expressing domain concepts, relationships and axioms in a formal machine readable language is a crucial task in ontology learning. In this regard, an ontology language is required to encode the captured information about a specific domain into ontology and must be amenable for automated processing. *Description Logic (DL)* [188] is a family of knowledge representation languages that is mostly used for modelling ontologies, because it provides a logical formalism for Web Ontology Languages as standardized by *World Wide Web Consortium (W3C)* [189]. It helps to model the relationship between three kinds of entities in a domain of interest. These entities are respectively called concepts, roles and individual names. Concepts represent sets of individuals, roles represent binary relationship between the individuals, and individual names represent a single individual in that domain. Description logic is in fact a prominent paradigm for knowledge representation and reasoning.

We clarify the statements above by providing an example. Assume that we have an ontology that is defined in the context of animals and their types of species. It uses concepts such as *Birds* to represent the set of all birds and *Mammals* to represent the set of all individuals classified as mammal. For instance, *elephant* and *eagle* can be represented as individuals belonging to the class *Mammals* and *Birds*, respectively. Moreover, a role such as *isEating* is used to represent the relationship between *animals* and their *foods*. For example *isEating(elephant,grass)* indicates that *elephant* (as an instance of *Mammals*) is related to *grass* (as an instance of Foods) via relation *isEating*.

Description Logic consists of a set of statements called Axioms. There are three important

groups of axioms in DL called *(i) assertional axioms (ABox)*, *(ii) terminological axioms (TBox)* and *(iii) Relational axioms (RBox)*. ABox axioms represent the knowledge about individuals. This knowledge consists of the concepts that individuals belong to, and their relationships with other individuals. Concept assertions are among the most common ABox axioms. In our example *Mammals(elephant)* asserts that the individual named *elephant* is an instance of the concept *Mammals*. Using Role Assertion, we can define relations among named individuals. For instance *isEating(elephant,cabbage)* states that *"elephant isEating cabbage"*. Likewise, TBox axioms describe relationship between defined concepts. The fact that all *Mammals* are *Animals* is expressed by the concept inclusion $Mammals \subseteq Animals$ which means concept *Mammals* is subsumed by concept *Animals*. Using these kinds of formatting for representation of knowledge simplifies inferring further facts about named individuals (i.e. *"elephant is an Animal"* can be implicitly deduced, although it may not be explicitly expressed earlier. RBox axioms are used for modeling relationships between the roles.

Although DL covers variety of constructors for building complex concepts from simpler ones, it provides a few number of constructors for creating complex roles such as role inclusion, role equivalence and disjointness. But on the contrary, it has been used for development of other languages with richer axioms such as *Web Ontology Language (OWL)*. OWL can be considered as the most important application of DL [184], [190] which exploits the strength of DL such as practical reasoning techniques and adds new set of axioms like the range and domain to the ontology properties. Another interesting feature of OWL, that is missing in DL, is called *Keys* which can be used for data integration. Similar to key constraints in databases, we can specify that two named individuals are supposed to be equal if they agree on certain property values and class memberships by using those keys. Moreover, OWL provides a mean for naming an ontology and interchanging ontological axioms between two ontologies. A characteristic which is not present in DL at all.

OWL contains three main sub languages called *OWL Lite*, *OWL DL* and *OWL Full* [184]. OWL species have different levels of semantic expressivity, various set of constructs and reasoning complexity. As discussed in [184], OWL Lite supports only simple constraints such as binary (0 and 1) cardinality and can be used to express taxonomy in a domain. OWL DL on other hand, supports high level of expressiveness while preserving computational completeness and decidability which enables a reasoner to infer new implicit knowledge from data. However, it has restrictions such as an ontology class can not play the role of class (as subclass of other classes) and an individual (instance of another class) at the same time. OWL Full is suitable for users who need maximum expressiveness and also syntactic freedom with no computational guarantees. Thus, no reasoner can support complete reasoning over all features of OWL Full since there is no guarantee over completeness of OWL Full implementations.

Choosing among OWL sub languages is highly dependent on the application domain and may vary across domain requirements. The major difference among OWL Lite and OWL DL lies in the level of expressiveness constructs. On the other hand, depending on the requirement for meta modelling over ontological classes, users may or may not move from OWL DL to OWL

Full.

Definition of the main components of an ontology is the same over all OWL sub languages. An OWL ontology consists of:

- *Classes*: which describe the main concepts in a domain. An OWL ontology class may have multiple subclasses representing more specific domain concept.
- *Properties*: OWL properties are either *object properties* to describe relations between classes and relate an object to another object, or *datatype properties* to describe individuals of classes and relate domain objects to data type values.
- *Restrictions*: are defined as formal descriptions indicating the conditions to be met by asserted expressions in order to get accepted by the reasoner.
- *Axioms*: are a set of constrains describing ontological classes and valid types of relationships among them. They must be defined with a high level of expressiveness in order to permit additional knowledge extraction by the reasoner and to add semantics into ontology.

4.2.3 Reasoning in Ontology

Reasoning over ontologies and knowledge bases is the main motivation for formal representations of a domain of interest. By reasoning, we mean deriving implicit facts which are not explicitly expressed in an ontology or a knowledge base. In the following, we have summarized the major tasks that must be done by a reasoner [191]:

- *Feasibility of ontology concepts*: determines if the descriptions of the extracted concepts are not contradictory, i.e., checking whether an individual can exist as an instance of a defined concept. A concept is unsatisfiable, if no individual can exist that would be an instance of that concept.
- *Subsumption of concepts*: determines whether concept C subsumes concept D, i.e., whether the description of C is more general than the description of concept D.
- *Consistency of ABox with respect to TBox*: determines whether individuals in ABox do not violate descriptions and axioms described by TBox.
- *Individuals verification*: checks whether the individuals are instances of concepts.
- *Retrieval of individuals*: finds all individuals that are instances of a concept.
- *Realization of an individual*: finds all concepts that an individual belongs to, especially the most specific ones.

Many reasoning Algorithms have been designed which satisfy the aforementioned points by checking deductive consequences or deriving all deductive consequences from the inserted axioms [192, 193, 194]. Many studies have been conducted on comparing the performance of these Algorithms and discussing around their specification [195] which is not in the scope of this work.

One of the interesting yet complex concepts within the context of ontologies, is the use of *open world reasoning*. In most of the ontologies and particularly in OWL, Open World Assumption (OWA) refers to the statement that *"if something is not known in the ontology, it must be considered as missing rather than false"*. By contrast, reasoning with Close World Assumption (CWA), is about returning *"No"* whenever some part of information is not provided. Let's clarify both concepts with the example of: *"S1: Andreas is an employee of Roche."* and assume the following statement is true:*"S2: A person can only be employee of one company."*. Now consider a new statement as *"S3: Andreas is an employee of Novartis."*. In a CWA system, this generates an error because of what we have previously stated in *S2* that a person can be employee of one company and we assume that *Roche* and *Novartis* are different companies. In an OWA system, this however, would infer a new statement as *"S4: If a person can only be employee of one company, and if Andreas is an employee of both Roche and Novartis, the Roche and Novartis must be the same company."*

OWL uses OWA reasoning with negation as unsatisfiability. Hence, it considers the ontology as a knowledge base with incomplete information in which inferring new information is possible. However it is not always possible to entirely define the concepts and relations from the very beginning especially for data domains such as people, companies, universities, etc. [196]. By making the effort in describing domain concepts as complete as possible (using *primitive* and *defined* class types which will be discussed later), we can avoid any inconsistency in inferred information.

4.2.4 Ontology Learning from Text

The term *Ontology Learning* is the process of extracting the main concepts and relationships in a specific domain. Many studies have been conducted in this domain. In [13], the authors summarized the main characteristics of various ontology learning systems with detailed information around their corresponding Algorithms. The major findings of their study are highlighted as: 1) the majority of ontology learning systems benefit from a base ontology rather than building their own ontology from scratch, 2) one of the major challenges is extracting ontological relations based on NLP techniques and 3) NLP techniques could provide considerably promising results for domain concept extraction. Another report has been published by the OntoWeb Consortium [11] around 36 ontology learning methodologies from unstructured text. Another survey report is published in [12] with a slightly different approach. In this study, the authors provide a three dimensional framework for classifying around 50 ontology learning systems. These dimensions are built based on what to learn (based on domain application),

from where to learn, and how to learn the ontology (if ontology is constructed from scratch or generated by leveraging pre-built ontologies). This study highlights mainly taxonomic relation extraction and does not cover that much of non-taxonomic relation extraction among ontological concepts. Moreover, most of the analyzed systems were employing a semi-automated method for ontology construction. In [197], the authors introduced the concept of *ontology learning layer cake* and compare ontology learning techniques of some articles in constructing different layers.

Open issues and challenges in constructing ontologies are discussed in [198], in which the authors proposed some improvements for preventing human involvement in the entire process. In this paper, the significance of logic based techniques has been studied for the purpose of forming axioms in ontology. The study of ontology learning approaches from unstructured and semi-structured data has been conducted in [34]. In this article, the authors argued that NLP techniques could function efficiently in ontology construction from unstructured data. Nevertheless, in semi-structured data, data mining techniques are also required. For ontology evaluation, they described five levels namely: lexical (vocabulary), hierarchical, contextual, syntactic, and structural levels. The last comprehensive study which we found was performed in 2018 in [15]. In this article, the authors categorized ontology learning techniques and their corresponding Algorithms into three classes namely linguistics, statistical and logical. Based on their study, data mining, machine learning and information retrieval are categorized as statistical techniques for extracting domain specific concepts and associations among them. On the other hand, NLP plays its role in almost every level of ontology learning (term extraction, concept formation, relationship extraction, axiom generation, etc.) by providing linguistic techniques.

Our proposed method, takes advantage from both linguistic as well as mathematical techniques for constructing domain ontology from scratch. As shown in Figure 4.2 and explained in the previous chapters, powerful NLP techniques have been involved in almost every step of ontology construction from pre-processing of text documents to concept and relation extraction. On the other hand, statistical techniques such as FCA and clustering play complementary roles in assembling additional pieces of the puzzle including taxonomic relationship among domain concepts. Once a comprehensive knowledge about an unfamiliar domain has been acquired through the previous steps, we have defined and implemented a rich set of mapping rules to interpret the output of each step into ontological axioms by means of selected ontology learning module (*OWLReady* for Python). Indubitably, further treatments are also required to evaluate constructed ontology and to ensure its completeness and correctness.

In the following sections, we deep dive into the applied approach as well as the defined mapping rules for translating the output of each box into the ontological language. Our approach starts from understanding the scope of an unfamiliar domain to importing detailed specifications of its corresponding ontology. Since the major distinction of our proposed method relies on taking advantage of FCA for generating the backbone of the desired ontology, we will mainly focus on converting GSH into ontological classes and their corresponding

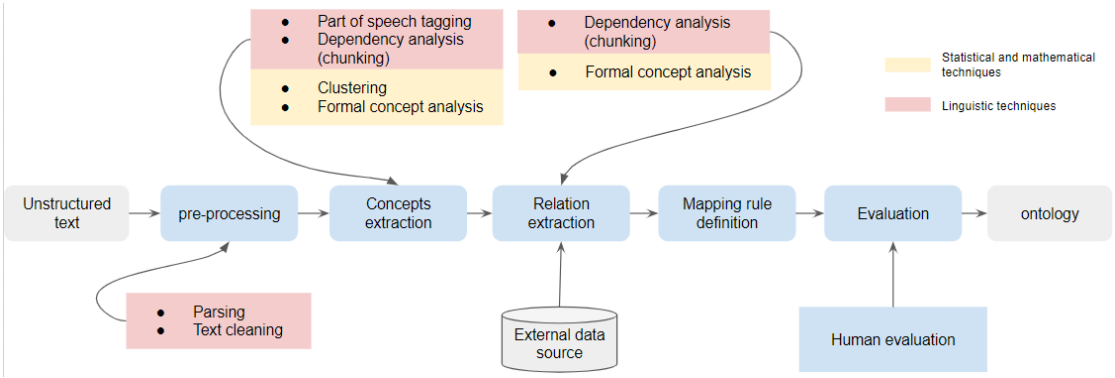


Figure 4.2 – *Ontology learning pipeline.*

sub-concept/super-concept hierarchy (i.e. taxonomical hierarchy).

4.3 Towards Building an Ontology

The very first step for learning an ontology from any text corpus is to grab a comprehensive knowledge about its domain. We could determine the scope of our domain by sketching a list of questions such as:

1. Which are the main concepts in the domain?
2. Through which types of relationship are these concepts connected?
3. Which types of questions must be answered via the generated ontology?
4. Who are the main users of the ontology?

We applied our proposed approach to a case study in the domain of Market and Competitive Intelligence (MCI). Hence, answering the aforementioned questions could bring us to the point that the final ontology should represent extensive knowledge about name of the companies, name of the people, as well as some information regarding which products have been launched by each company. The end users were employees in the section of reporting of the company with the task of producing reports to the senior managers with regular updates about what is happening outside the company. The outcomes of our analysis are published in [54] and [55].

Judging from the list of questions, the final ontology will include information on various companies and their products. We can further extend the work by incorporating other pieces of data such as information about employers and employees, classification of products based on their type which enable us to generate appropriate answers to the question like *"what is the list of products being launched/produced/announced by a company?"*

4.3.1 Comprehensive Overview of Construction Approach

Exploring domain concepts is the first step towards building the backbone of any ontology which could be performed using a *top-down*, *bottom-up* or a *combined* approach. However, there is no defined rule indicating which of these three methods operates better, and it entirely depends on the application domain as well as on the personal view of the ontology designer. In the top-down approach, for instance, the most general concepts are primarily extracted. Further specialization of any of the high level classes is then performed in order to display a comprehensive view about the domain of interest. On the contrary, the bottom-up approach starts with defining the most specific domain concepts and proceeds with grouping those classes in order to produce more general classes. Finally, as the name indicates, a combined development process makes use of both the top-down and bottom-up approaches by first defining the most prominent concepts and then generalize and specialize them appropriately.

Insertion of Text Mining Outcomes into Ontology

Since our initial data is in the form of an unstructured text corpus extracted from multiple unknown sources (e.g. web based content through RSS feed, published annual reports, etc...), we decided to investigate the primary domain concepts (such as *Companies, Products, People, Locations*) by employing both NLP and clustering techniques. Apart from the main concepts, this approach also yields a complete list of instances within each category as well as relationships among them. Clearly, extracted concepts are interpretable into ontological classes. On the other hand, a set of instances along with their relationships construct the most significant components of an ontological ABox which is a list of individuals within each class as well as non-taxonomical relationships among them.

Figure 4.3 demonstrates how the outcomes of analysing text documents could be interpreted into the main components of an ontology. A complete description of our approach for categorizing groups of individuals (subjects and objects) into multiple classes is provided in Section 2.3.6.

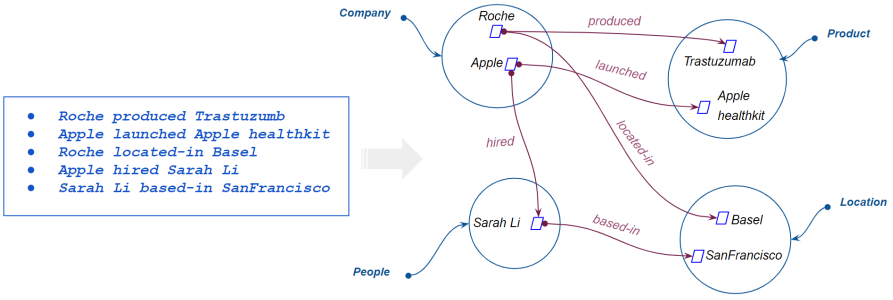


Figure 4.3 – *Example of high level ontological classes, non-taxonomical relationships and individuals within each class. Clustering subject and objects yields ontological classes and their corresponding individuals. Non-taxonomical relationships are generated through analyzed triplets with NLP techniques.*

Nevertheless, this analysis yields no hierarchical relationships among classes. Moreover, one of the main requirements by external users was to obtain some extra information about instances of class *Products* which are either in the group of drugs or devices (e.g: *Trastuzumab* as *drug* and *Apple healthkit* as a *device*). Therefore, further separation of individuals of class *Products* must be performed by benefiting from an extra data source.

In our context, asserting corresponding classes of *Companies, Products, People, Locations* into an ontology and further appending the class *Products* into more sub classes, introduces the challenge of re-adjusting the asserted relationships among individuals of class *Products* with other classes. Therefore, a top-down approach has been followed for **exploring** high level ontological concepts, while **construction** of ontology is performed in a bottom-up manner. For doing so, once we obtained a comprehensive knowledge about which potential classes could exists in the final ontology, we took a reverse approach for constructing the ontology, concentrating on a specific class (*Products*) first, and then asserting other ontological classes, individuals and non-taxonomical relationships. The procedure is explained in the following subsections.

Contribution of External Data Source in Ontology Construction

Note that our primary attempt was to generate a domain ontology from scratch and not to re-use any built ontologies such as ICD [1] and SNOMED [2] within our ontology construction pipeline. Conducting our study in the domain of pharmaceutical companies brought us the opportunity of getting familiar with a rich data source called *FDA dataset*. FDA is the abbreviation of *"Food and Drug Adminstration"* and is a federal agency of the United States Department of Health and Human Services with multiple responsibilities such as protecting and assuring the quality of newly launched products [3]. The information of any officially launched device and drug products has been kept in a database which is publicly accessible through an API called *OpenFDA* [4] [199]. *OpenFDA* contains multiple endpoints for accessing high quality information of drugs, devices, foods, etc. Nevertheless, for the purpose of extracting information for instances of class *Products*, we have used *Drug* and *Device* end points only. Dataset of each endpoints is available in the format of JSON being stored in multiple pdf files, which are updated regularly. For each product, its corresponding data contains two main sections: (1) *meta*: containing metadata about each product such as disclaimer, link to data license, last-updated date, etc. and (2) *results*: an array of extra features assigned to each product based on the type of endpoint. Figure 4.4 demonstrates how the output looks for *"TRASTUZUMAB"* as one of the extracted instances from our analyzed text. For the purpose of simplicity, we have selected a few features and we have removed long descriptions from each field.

[1] https://www.who.int/classifications/icd/en/
[2] http://www.snomed.org/
[3] https://www.fda.gov/
[4] https://open.fda.gov/about/

```
{
 "meta": {
  "disclaimer": "Do not rely on openFDA to make decisions regarding medical care...",
  "terms": "https://open.fda.gov/terms/",
  "license": "https://open.fda.gov/license/",
  "last_updated": "2019-04-20",
  "results": {
   "skip": 0,
   "limit": 1,
   "total": 4
  }
 },
 "results": [
  {
   "drug_interactions": [
    "7 DRUG INTERACTIONS No formal drug-drug interaction studies with KADCYLA have been
conducted..."
   ],
   "geriatric_use": [
    "8.5 Geriatric Use of 495 patients who were randomized to KADCYLA in the randomized trial ..."
   ],
   "description": [
    "..."
   ],
   "mechanism_of_action": [
    "12.1 Mechanism of Action ..."
   ],
   "indications_and_usage": [
    "1 INDICATIONS AND USAGE KADCYLA ..."
   ],
   "set_id": "23f3c1f4-0fc8-4804-a9e3-04cf25dd302e"
   ],
   "storage_and_handling_table": [
    "<table width=\"90%\">..."
   ],
   "pregnancy": [
    "..."
   ],
   "nursing_mothers": [
    "8.2 Lactation Risk Summary..."
   ],
   "spl_product_data_elements": [
    "KADCYLA ADO-TRASTUZUMAB ..."
   ],
   "boxed_warning": [
    "WARNING:..."
   ],
   "adverse_reactions_table": [
    "<table width=\"90%\" ID=\"table6\"> <caption>..."
   ],
   "openfda": {
    "product_ndc": [
     "50242-088",
     "50242-087"
    ],
    "generic_name": [
     "ADO-TRASTUZUMAB EMTANSINE"
    ],
    "brand_name": [
     "KADCYLA"
    ],
    "manufacturer_name": [
     "Genentech, Inc."
    ],
    "unii": [
     "SE2KH7T06F"
    ],
    "rxcui": [
     "1658084",
     "1658087"
    ],
    "substance_name": [
     "ADO-TRASTUZUMAB EMTANSINE"
    ],
    "product_type": [
     "HUMAN PRESCRIPTION DRUG"
    ],
    "route": [
     "INTRAVENOUS"
    ],
    "application_number": [
     "BLA125427"
    ]
   },
   "adverse_reactions": [
    "6 ADVERSE REACTIONS ..."
   ],
   "clinical_studies": [
    "14 CLINICAL STUDIES..."
   ],
   "overdosage": [
    "10 OVERDOSAGE ..."
   ]
  }
```


Figure 4.4 – *OpenFDA features for TRASTUZUMAB.*

Concepts Derivation Using FCA

Further separation of the extracted products is performed through FCA. Therefore, a formal context has been constructed with rows as name of the products and columns as selected fields from *OpenFDA*. It must be noted that not all of the fields are relevant (such as *"overdosage, unii, application-number, product-ndc, etc."*), hence, a human based feature selection process is applied. The list of selected features along with their description as well as corresponding values are summarized in Figure 4.5

Selected Endpoint	Feature Name	Description	Potential Values
Drug endpoint	substance_name	The list of active ingredients of a drug product	SEPIA OFFICINALIS JUICE, KEROSENE, AMANITA MUSCARIA VAR. MUSCARIA FRUITING BODY, SODIUM CHLORIDE, FUMARIA OFFICINALIS FLOWERING TOP, ALUMINUM OXIDE, POTASSIUM ARSENITE ANHYDROUS
	brand_name	Brand or trade name of the drug product.	Ibuprofen, Amoxicillin, Famotidine, etc
	product_type	Specifies type of product	HUMAN OTC DRUG HUMAN PRESCRIPTION DRUG
Device endpoint	medical_specialty	Regulation Medical Specialty is assigned based on the regulation (e.g. 21 CFR Part 888 is Orthopedic Devices) which is why Class 3 devices lack the "Regulation Medical Specialty" field. Two letters indicating the medical specialty panel responsible for reviewing the product. See link for further detail.	AN = Anesthesiology CV = Cardiovascular CH = Clinical Chemistry DE = Dental EN = Ear, Nose, Throat GU = Gastroenterology, Urology HO = General Hospital HE = Hematology IM = Immunology MI = Microbiology NE = Neurology OB = Obstetrics/Gynecology OP = Ophthalmic OR = Orthopedic PA = Pathology PM = Physical Medicine RA = Radiology SU = General, Plastic Surgery TX = Clinical Toxicology
	life_sustain_support_flag	An indicator that the device is essential to, or yields information that is essential to, the restoration or continuation of a bodily function important to the continuation of human life.	Y = Device is used for life sustaining purposes. N = Device is not used for life sustaining purposes

Figure 4.5 – *Selected fields from Drug and Device endpoints of OpenFDA.*

Once feature selection is done, the corresponding formal context is constructed for those drug/device individuals being extracted from text. A general view of constructed formal context is provided in Figure 4.6. The structure of formal context of Figure 4.6 is called *many-valued context* which occurs frequently in many real applications. The binary table structure of a *many-valued context* $\mathcal{K} = (X, Y, W, I)$ consists of a set of objects X, a set of attributes Y with their corresponding values W and the binary relations I in which each relation $I(x, y, w) = 1$ is read as *"object x has the attribute y with value equal to w"*.

Given that FCA Algorithms are only designed for analyzing binary tables with one-valued attributes, a transformation process is required for the table of Figure 4.6. The word *Conceptual scaling* is originally defined by Ganter in [200] as the process of transforming any many-valued context into a one-valued formal context. In the context of conceptual scaling, a separate scale is defined for each many-valued attribute of the table. We should recall the following definitions due to [200]. The first definition corresponds to the formal definition of *scale* as follows:

	Substance Name			Brand Name			ProductType		Product_ Code			Medical Spaciality			Life Sustain Support Flag	
	Sodium chloride	Aluminium oxide	...	SILICEA	Mezereum	...	HUMAN OTC DRUG	HUMAN PRESCRIPTION DRUG	brt	brw	...	CN	TX	. . .	Y	N
device-01									X			X				X
drug-01	X	X			X			X							X	
drug-02	X			X			X				X					
device-02											X		X			X
device-03										X				X		X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.	...	...

Figure 4.6 – *Multi valued context of FDA drug/device binary table.*

Definition 10 *A **scale** for a many-valued attribute y of a many-valued context $\mathcal{K} = (X, Y, W, I)$, is a formal context $\mathbb{S}_y := (X_y, Y_y, I_y)$ such that $\{y(x)|x \in X\} \subseteq X_y$.*

This implies that for any attribute y of a many-valued context, the formation of a new formal context (named scale) is required whose objects are attribute values, and whose attributes are the new attributes that will replace y in the derived one-valued context. The objects and attributes of the scale are called *scale values* and *scale attributes* recursively. For the sake of clarity, the nominal scale of the attribute *ProductType* is constructed in Figure 4.7:

$\mathbb{S}$ ProductType	Product Type_ HUMAN OTC DRUG	Product Type_ HUMAN PRESCRIPTION DRUG
HUMAN OTC DRUG	X	
HUMAN PRESCRIPTION DRUG		X

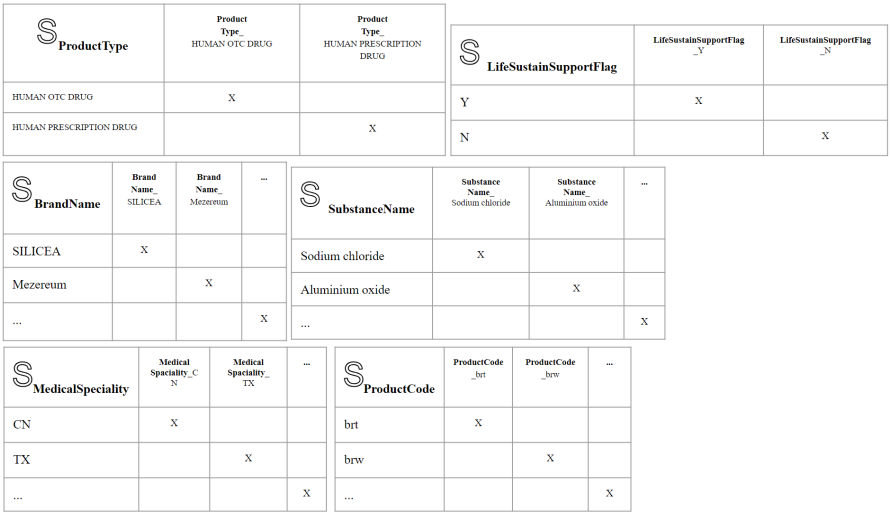
Figure 4.7 – *Scale of the attribute ProductType of table in Figure 4.6.*

To achieve a one-valued context, scaling of all the attributes of the many-valued context is required. The combination of all the scales is called a *scale schema* and is formally defined as follows:

Definition 11 *A **scaling schema** for a set Y of many-valued attributes is family of scales for all attributes of Y: $\mathbb{S}_y|y \in Y$.*

The scale schema of the multi-valued context of Figure 4.6 is shown in Figure 4.8:

The goal of the last step of conceptual scaling is to obtain *derived context* by applying scale schema to the many-valued context. The derived one-valued context of Figure 4.9 is gained by



$\mathbb{S}_{\text{ProductType}}$	Product Type_ HUMAN OTC DRUG	Product Type_ HUMAN PRESCRIPTION DRUG
HUMAN OTC DRUG	X	
HUMAN PRESCRIPTION DRUG		X

$\mathbb{S}_{\text{LifeSustainSupportFlag}}$	LifeSustainSupportFlag _Y	LifeSustainSupportFlag _N
Y	X	
N		X

$\mathbb{S}_{\text{BrandName}}$	Brand Name_ SILICEA	Brand Name_ Mezereum	...
SILICEA	X		
Mezereum		X	
...			X

$\mathbb{S}_{\text{SubstanceName}}$	Substance Name_ Sodium chloride	Substance Name_ Aluminium oxide	...
Sodium chloride	X		
Aluminium oxide		X	
...			X

$\mathbb{S}_{\text{MedicalSpeciality}}$	Medical Spaciality_CN	Medical Spaciality_TX	...
CN	X		
TX		X	
...			X

$\mathbb{S}_{\text{ProductCode}}$	ProductCode _brt	ProductCode _brw	...
brt	X		
brw		X	
...			X

Figure 4.8 – *Nominal scaling schema for table in Figure 4.9.*

replacing each value by the corresponding row of the respective scale. The formal definition of *Derived context* is also provided in :

Definition 12 *From a scaling schema* $(\mathbb{S}_y | y \in Y)$ *for the many-valued context* $\mathcal{K} = (X, Y, W, I)$*, the* ***derived context*** (X, N, J) *is defined by*

$$N := \bigcup_{y \in Y} \{y\} \times Y_y$$

and

$$xJ(y,n) \Longleftrightarrow n \in Y_y \text{ and } y(x) I_y n$$

	Substance Name_ Sodium chloride	Substance Name_ Aluminium oxide	...	Brand Name_ SILICEA	Brand Name_ Mezereum	...	Product Type_ HUMAN OTC DRUG	Product Type_ HUMAN PRESCRIPTION DRUG	Product Code _brt	Product Code _brw	...	Medical Spaciality _CN	Medical Spaciality _TX	...	Life Sustain Support Flag _Y	Life Sustain Support Flag _N
device-01									X			X				X
drug-01	X	X			X			X							X	
drug-02	X			X			X									
device-02													X			X
device-03										X				X		X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 4.9 – *Single valued context of FDA drug/device binary table.*

By constructing the formal context in a standard format, our data gets ready to be analyzed by FCA with the goal of generating sub-concept and super-concept relationships among formal concepts through GSH. Constructing a GSH is just the beginning of the long journey of building the desired ontology. The following section demonstrates the conversion of any GSH

into ontological classes and taxonomical relationships. For the purpose of simplicity, we will discuss our method on the toy example of animals and plants formal context (Table 3.1) and its corresponding GSH of Figure 3.3b.

It has been discussed that FCA formalizes the notion of a concept and a concept hierarchy by means of concept lattice. Ontologies, on the other hand, are frequently used to represent conceptual knowledge of an application domain in a structured way. Although the notion of a concept as set of objects sharing a set of attributes as well as their conceptual hierarchy is fundamental for both FCA and ontologies, the description of concepts and the way they are obtained are significantly different.

In the following, we discuss our mapping Algorithms being applied on the extracted lattice (in the form of GSH) to turn it into a domain ontology. The constructed ontology from the GSH is represented with the OWL-DL. Our method consists of the two following steps:

1. Formal specification of ontological classes
2. Asserting of taxonomic relationship among ontological concepts
3. Concept intents and extents to ontology properties and individuals

In the following, for each step, we have provided an example with detailed explanation.

4.3.2 Formal Concepts to Ontological Classes

As mentioned in section 3.2.4, GSH provides a high fidelity with respect to the original knowledge and also preserves an abstract view of concepts. Being shown in the Hasse diagram of Figure 3.3b, the process of omitting empty concepts from Figure 3.3a produces a modified conceptual hierarchy including either object concepts (simplified form of extents) or attribute concepts (simplified form of intents).

Both types of concepts in GSH contain information about a domain of interest. For instance, looking at the attribute concepts in Fig. 3.3, *concept 4* indicates a group of individuals with the capability of *living in water* and *concept 3* presents a group of individuals *live on land*. On the other hand, although *concept 8* is introduced without any feature, it must be considered as a new ontological class being implicitly retrieved from initial data along with its corresponding objects (i.e., *Oak, Potato*) as group of individuals inheriting specifications of their parent concepts (*Plant* and *LiveOnLand*).

Therefore, all fixed points of GSH are preserved in the final ontological hierarchy because they correspond to the domain concepts and carry explicit or implicit information of initial data set. Referring to the GSH of Figure 3.3b, for each fixed point, there will be an asserted class in the final ontology.

4.3.3 Concepts Hierarchies to Taxonomic Relationships

Sub-concept or super-concept hierarchy among two formal concepts is identified as *"is-a"* relationship between their corresponding ontological concepts. Figure 4.10 demonstrates a simple taxonomy of the GSH presented in Figure 3.3b. In OWL, subsumption (or taxonomic relations) mean inclusion of necessary implication from existing relations. For instance, in Figure 4.10, we can deduce that *"all instances of concept 6 are also instances of concepts 2 and 3."*

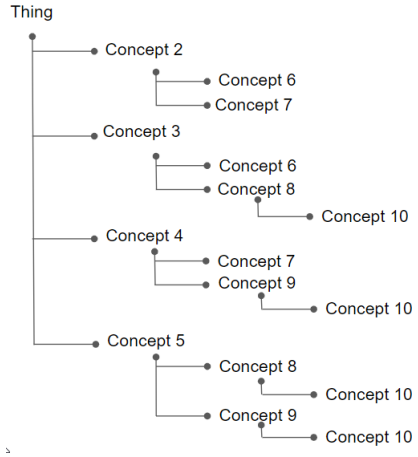


Figure 4.10 – *Desired hierarchy in final ontology created from GSH of Figure 3.3b.*

There exist two types of concepts in the GSH called *single inheritance* (concepts with only one super class) and *multi inheritance* (concepts that form *polyhierarchies* and have multiple super classes). As an example, in figure 3.3b , *concept 6* is a subclass of of both *concept 2* and *concept 3*, meaning that all instances of *concept 6 (Cat and Dog)* are also instances of *concept 2* (as *Animal*) and *concept 3* (*Living on Land*).

Establishing taxonomical relationship is straightforward for single parent concepts. Nevertheless, manual assertion and maintenance of poly-hierarchies structure introduce many limitations to ontology design. For instance, in a dynamically changing domain, adding a new class requires readjustment of the existing hierarchies as well as forming additional relationships which might be missed by the ontologies. Therefore, an automated reasoner is required to restore polyhierarchies by inferring new subsumptions and semantic information, maintaining multiple inheritance, and verifying the consistency of the asserted ontological concepts, based on the precise descriptions inserted within the ontological classes. *OWL* constructors enable ontologiests to specify classes with high level of expressiveness; universal restriction (only), existential restriction (some), number restriction (min, max, exactly) and Boolean operators (or, and, not) can be combined to build rich expressions. Besides prior capabilities, *OWL* also supports defining *primitive* or *defined* classes. *Primitive class* refers to a class which has only necessary conditions. Conversely a *defined class* has at least one necessary and sufficient condition. For instance, in Figure 4.11, *"A"* and *"B"* are asserted as

primitive ontological classes. On the other hand class *"C"* which is subclass of *"A"* is inserted as group of individuals having *"has-part"* relation with individuals of class *"B"*. Thus *"C"* is a defined ontological class with *necessary and sufficient* condition which allows deduction in two direction:

- If an individual is in class *"C"*, then it is also an individual of class *"A"* and it has a relationship *"has_part"* with one of individuals in class *"B"*.
- If an individual sits in class *"A"* and also has a relationship *"has_part"* with a member of class *"B"*, then it is definitely an individual of class *"C"* as well.

Thus for an individual to be accepted as an instance of class *"C"*, it is both necessary and sufficient to be a subclass of *"A"* and to have a relation called *"has_part"* with **at least one** (or some) individuals of *"B"*.

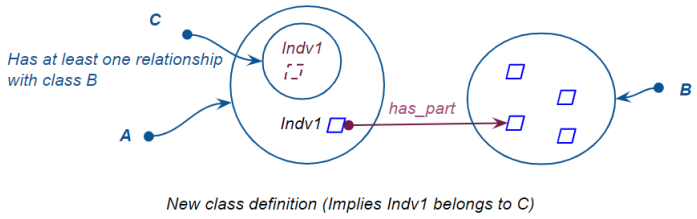


Figure 4.11 – *Defined class with necessary and sufficient condition.*

We have treated the problem of automatic poly hierarchies assertion into an ontology by applying an introduced design technique called *Normalization* (or *untangling*) by [201]. Normalization is the process of decomposing poly hierarchies into simple trees with independent disjoint skeleton taxonomies. This is accomplished by coding subsumption relationships with rich set of definitions and restrictions rather than a simple assertion of parent-child relationships. For doing so, one single hierarchical path among the child class and one of its super classes (called *target class*) is retained and the path from child to its other super classes are *dissolved*. Dissolving class paths have the following requirements:

1. Adding the restrictions of other super classes (not the *target class*) to the specification of the multi-inheritance ontology class.
2. Removing any subsumption of the target class under other super-classes from the specification of the target class.

For clarifying the normalization method, let us apply the above statements on *concept 6* in the example of Fig. 3.3b which is subsumed under two super classes (i.e. *concept 2* and *concept 3*). Without normalization, class assertion in OWL looks like the following:

"*concept 6* EquivalentTo (*concept 2* and *concept 3*)"

However applying normalization changes the class expression such that *concept 6* will primarily belong to one of its super classes, say *concept 2*. Thus, it replaces the hierarchical subsumption under *concept 3* (indicated through a dotted arrow in Fig. 4.12b) by adding the restrictions of *concept 3* to the specification of *concept 6*. Algorithm 9 demonstrates the

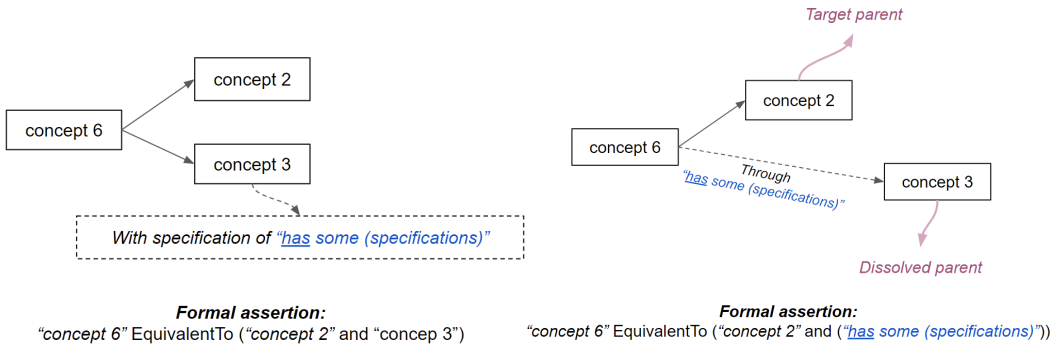


Figure 4.12 – *Dissolving poly hierarchy among classes through normalization, (a) Poly hierarchy before normalization, (b) Poly hierarchy after normalization.*

fundamental steps of our proposed method for asserting both single and multi parent concepts into the domain ontology. Our normalization has been applied through the famous ontology design pattern called *Value Partitions* which is a proven solution published by "The Semantic Web Best Practices Working Group" [202]. By creating *Value Partitions*, we can refine the description of various classes in order to cleverly resolve the multiple inheritance issues explained above.

The following Algorithm executes the ontology development process from GSH. Considering the fact that in each iteration, all formal concepts have an equal possibility of being selected and asserted into an ontology, this Algorithm implements a *combination* development process ([203]) based on the type of input concept. We might start with a top-level concept such as *concept 2* and to relate it to the most general concept in ontology called *Thing*. Conversely, if the input concept is located in the lowest level of the lattice, it gets asserted in the level of the most specific classes (the leaves of the ontology hierarchy). Algorithm 9 inserts different type of node into the ontology. In the following, further explanations are provided for each line of the proposed Algorithms:

Algorithm 9 *Embed a lattice node (node_name) into ontology.*

```
1: procedure NODE_EMBEDDING(node_name , ONTO = onto)
2:     successor_list ← successors(node_name)
3:     if Len(successor_list) == 0 then
4:         Add-to-Onto(node_name)
5:         return True
6:     end if
7:     for successor in successor_list do
8:         if ONTO[successor] is None then
9:             Node_Embedding (successor)
10:        end if
11:    end for
12:    if len(successor_list) == 1 then
13:        direct_parent = successor_list[0]
14:        if ONTO[direct_parent] is None then
15:            Node_Embedding(node_name = direct_parent)
16:        end if
17:        Add-to-Onto (node_name,direct_parent)
18:        return True
19:    else
20:        direct_parent = successor_list[0]
21:        node_vps_list = [ ]
22:        for successor in successor_list[ 1 : ] do
23:            successor_vp_name = successor +"_VP"
24:            if ONTO[successor_vp_name] is None then
25:                Add-to-Onto (successor_vp_name,"ValuePartition")
26:                Primitive-to-Defined (node_name = successor, new_vp = succes-
    sor_vp_name)
27:            end if
28:            return True
29:            node_vps_list ← node_vps_list.append(successor_vp_name)
30:        end for
31:    end if
32:    if ONTO[direct_parent] is None then
33:        Node_Embedding (node_name = direct_parent)
34:    end if
35:    Add-to-Onto(node_name, direct_parent, node_vps_list)
36:    return True
37: end procedure
```

Algorithm 10 *Add new node to ontology.*

```
1: procedure ADD-TO-ONTO(node_name, DirPrt = None, IndirPrtsVPs = [] , ONTO=onto)
2:     if DirPtr is None then
3:         direct_parent = Thing
4:     else
5:         direct_parent = ONTO[DirPrt]
6:     end if
7:     if IndirPrt is None then
8:         with ONTO:
9:             types.new_class(node_name, (direct_parent, ))
10:        return True
11:    else
12:        vp_classes = []
13:        for ipvp in IndirPrtsVPs do
14:            vp_classes ← ONTO[ipvp]
15:        end for
16:        with ONTO:
17:            types.new_class(node_name, (direct_parent, ),
18:            exec_body=lambda ns:  ns.update({"defined_class":  True,
19:            "has_vp":  vp_classes})
20:    end if
21:    return True
22: end procedure
```

Algorithm 11 *Convert primitive class into Defined class.*

```
1: procedure PRIMITIVE-TO-DEFINED(class_name, new_vp, ONTO=onto)
2:     if ONTO[new_vp] in ONTO[class_name].has_vp then
3:         pass
4:     else
5:         ONTO[class_name].equivalent_to.append(
6:             ONTO["has_vp"].some(ONTO[new_vp]))
7:         return True
8:     end if
9: end procedure
```

Procedure of Node_Embedding (new_node) described in Algo. 9:

- Input parameter → *node_name* (a single fix point from GSH)
- Extract a list of successors of *node_name* and keep them in *successor_list* → (Line 2)
- If *successor_list* is empty, *new_node* gets appended into the ontology as sub class of *Thing* (the most general concept in any ontology) → Lines (3-6)
- Verify that all super classes of *new_node* (listed in *successor_list*) have been already asserted into the ontology. For any not-yet-inserted super class *successor*, recursively call the Algorithm with *successor* as input parameter (*Node_Embedding(successor)*) → Lines (7-11)
- If *new_node* is a single parent concept, check if its parent is already asserted into the ontology. If not, create the corresponding parent ontological class and directly connect *new_node* to its only successor. → (Lines 12-18)
- If *new_node* is a multi-parent concept, the first successor is selected as its direct parent. → (Line 20)
- Initialize a list *node-vps-list* to keep subclasses of value partition class for all indirect parents of *new_node*. → (Line 21)
- Continued in the next page (↓)

Node_Embedding (new_node) Algorithm procedure (continue):

- For each indirect parent of *new_node*, the Algorithm verifies if its corresponding class has been asserted under *ValuePartition* class. If not, a new subclass of *ValuePartition* class is created and the primitive indirect parent class is converted into a *defined* ontological class respectively. As an example, considering *concept 2* and *concept 3* as direct and indirect parents of *concept 6*, respectively. The Algorithm constructs *concept3-VP* as a subclass of the *ValuePartition* class and converts the primitive class of *concept 3* into a *defined* class (using *Primitive-to-Defined()* function) by asserting *"has_VP some concept3-VP"* to class definition of *concept 3*. → (Lines 22-34)
- *node-vps-list* is appended by the recently imported subclass of *"ValuePartition"* class. → (Line 29)
- Once the multiple hierarchy is resolved, the Algorithm embeds the *new_node* along with its corresponding direct and indirect parent(s) into the ontology using *Add-to-Onto()* function. (e.g.: *concept 6* will be added with the following expression: *"class concept6(concept2): equivalent-to [concept2* & *has_VP.some(concept3_VP)]")* → (Lines 32-36)

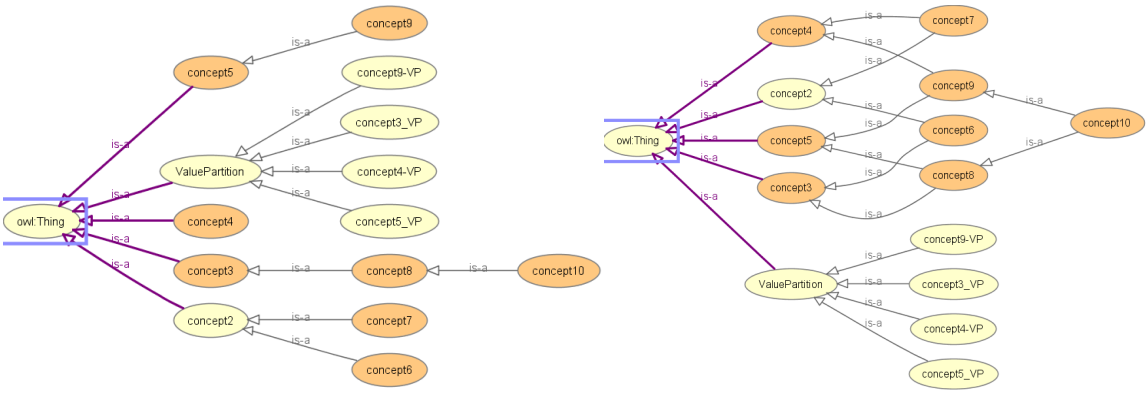
Procedure of *Add-to-Onto* described in Algo. 10:

- Algorithm 10 accepts four input parameters. The name of the new node to be added (*node_name*), its respective direct parent (*DirPrt*), the name of its corresponding indirect parents value partition class (if there exists any, *IndirPrtsVPs*) and the name of the ontology *ONTO*.
- The Algorithm behaves differently based on whether some values are provided for the input parameters *DirPrt* and *IndirPrtsVPs*. We forced the presence of the arguments by assigning the most general class in the ontology as the default value for *DirPrt* (*None* means *Thing*) and an empty list for *IndirPrtsVPs* indicating that there is no indirect parent by default.
- If *IndirPrtsVPs* is empty for the new node, it must be interpreted as a single parent ontological class. Therefore it gets connected to *DirPrt* through OWLReady syntax. → (Lines 7-10)
- Otherwise (where the node has multiple parents), append all value partition classes of *IndirPrtsVPs* into the pre-initialized list called *vp_classes*. → (Lines 11-15)
- The last step is to assert the new node along with the defined restriction into ontology. → (Lines 16-19)

Procedure of converting Primitive to Defined class described in Algo. 11:

- Algorithm 11 accepts three input parameters, the asserted *primitive* ontological class name (*class_name*), the restriction to be added into *"equivalent to"* section of ontology class in order to convert it into *defined* class, and the ontology name *ONTO*.
- The Algorithm first verifies that the input restriction is not asserted into the ontology. → (Lines 2-3)
- If the above condition is not fulfilled, the Algorithm expands the *"equivalent-to"* attribute with the restriction on *"has_vp"* relation. → (Lines 4-7)

Figure 4.13 demonstrates the generated ontology with the asserted and inferred taxonomical (*IS-A*) relationships. Node colors indicate types of ontological classes. *Primitive* classes are highlighted in yellow and *defined* classes are highlighted in orange. As shown in sub-figure 4.13a, the initial ontology is automatically constructed as a simple tree such that classes in the asserted hierarchy have no more than one superclass. In sub-figure 4.13b, however, the reasoner could automatically identify the inferred hierarchies and correctly classified the inheritance structure for the concepts with poly hierarchy.



(a) (b)

Figure 4.13 – *Asserted ontological classes and relationships from GSH, (a) Ontology classes with asserted relationships, (b) Ontology classes with inferred relationships.*

4.3.4 Concept Intents and Extents to Ontology Properties and Individuals

The power of the our proposed approach is to use FCA in the core of the ontology development process. In the previous sections, a combined approach is discussed for extracting and asserting the essential elements of the desired ontology. In this section, we will further discuss the power of FCA in augmenting the constructed ontology and enhancing class definitions.

In the proposed approach, although attribute concepts and object concepts are treated in the same way while being asserted as ontology classes, their encompassed information must be imported differently as explained below:

- **Concept intents**: If the respective node of the asserted ontological class enfolds any attribute(s) (like *Animal* in *Concept 2*), it must be considered as class properties in ontology. Class properties describe the internal structure of an ontology class. OWL models these properties as annotation properties which provide various pieces of information or meta-data about that specific class. It must be considered that class properties are applied to the class rather than to instances of the class and could not be used in any semantic interpretation from the OWL point of view. In order to attach semantics into an annotated class attribute, *Meta Modelling* is introduced by [204] and also discussed in [205] as the process of separating the conceptual model (terminological box - *TBox*) from data model (Assertional box - *ABox*).

  For the purpose of clarification, we discuss the following example which is originally presented in [206]. Let us consider the conceptual hierarchy shown in Figure 4.14. In this hierarchy, the concept *Bird* represents the set of all *birds* and has a subclass of *Eagle* which states the fact that all eagles are also considered as birds. This is what has been denoted as Class View in the TBox represented in Fig. 4.14. On the other hand, an individual *Harry* is assigned as an instance of the concept *Eagle* which constructs a data

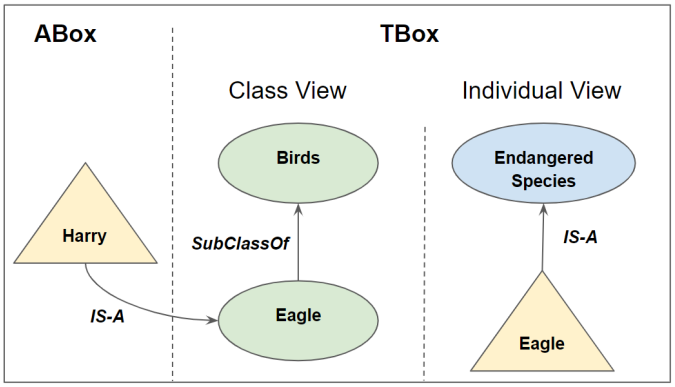


Figure 4.14 – *Meta Modelling.*

model of our knowledge base. The combination of asserted conceptual knowledge (e.g., Eagle) and data model (e.g., Harry) allows implicit knowledge extraction by inferring the fact that *Harry* is also a *Bird*.

In some cases we may want to also characterize or make assertions about eagles such as "*Eagles are among endangered species*". In other words, although *Eagle* has been defined as a class, we may also be interested in using *Eagle* also as an individual. In order to formally represent this statement, a new ontological concept *EndangeredSpecies* is created which contains all species that are endangered. This class may contain individuals such as mammals or eagles (*EndangeredSpecies* class is now a meta-class as its individuals are also classes).

In this example, the word *Eagle* is asserted as an ontological class (defined as a subclass of *Birds* with an individual *Harry*), as well as an individual (belonging to the class *EndangeredSpecies*). We could have both as separate views in the TBox. Figure 4.14 represents the structural and conceptual description of TBox in which *EndangeredSpecies* is in fact a meta-concept for *Eagle*.

Meta modelling is named as *punning* in the context of OWL and refers to the possibility of assigning the same *IRI* for the same names with different natures. For instance, applying meta modelling on the previous example, results in the assignment of the same *IRI* for *Eagle* as a class and *Eagle* as an instance of class *EndangeredSpecies*.

Despite the fact that meta modelling is a conscious solution to handle semantic attachment to imported class properties, it is not supported by *OWLReady*. In *OWLReady*, the given entity must be a class, an individual, an object property, a data property or an annotation property and it cannot have more that one of the above types. Hence the same name must not be used for two distinct entities, even if they have two different types (e.g. class vs. property). On the other hand, without invoking the OWL punning process, the asserted annotation properties do not participate in any inference or reasoning process and they act as additional information or meta data for that class.

To allow class properties to be engaged in the reasoner's inference process, we decided

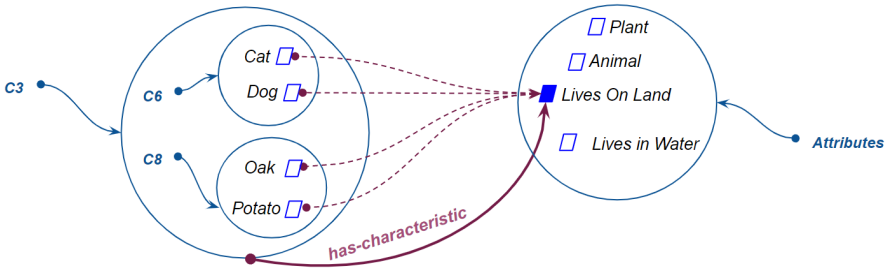


Figure 4.15 – *Property value constraint associated with class C3.*

to use the concept of *Blank Node (bn)* constructs [207], [208] which is represented by a restriction in OWL (noted as $\exists R.\{i\}$ in description logic, where R is a property and i is an individual or a literal). For example, the following representation indicates that all individuals belonging to corresponding ontological class of *"concept 3"* in Figure 3.3b have characteristic of *"Lives On Land"* (i.e. $C3 \sqsubseteq \exists has-characteristic.\{Lives\ \ On\ \ Land\}$):

(C3, rdfs:subclassOf, -bn)
(-bn , rdf:type, owl:Restriction)
(-bn , owl:onProperty, has-charactristic)
(-bn , owl:hasValue, Lives On Land)

or alternatively: (where *"onto"* is the name of constructed ontology):

(C3 a owl:class
rdfs:subClassOf [a owl:Restriction ;
owl:onProperty onto:has-characteristic;
owl:hasValue onto:Lives On Land])

The representations above, make *"C3"* a subclass of an anonymous class defined by the property restriction class. In order to enable the aforementioned property value constraint on any class (such as *"C3"*), a separate ontological class must be created which retains all attributes of the initial formal context 3.1 as its instances. This is a valid insertion due to the presence of all distinct attributes of the formal context in constructed GSH (i.e. the minimum size of GSH is equal to the number of distinct objects and attributes of the formal context). As shown in Figure 4.15, once property value restriction is explicitly asserted into an ontology (highlighted arrow line named *"has-characteristic"* from class *"C3"* to individual *"Lives On Land"* from class *"Attributes"*),the reasoner can infer that the property must also be satisfied by all of its instances and also the instances of its sub-classes (dotted lines from instances of class *"C6"* and *"C8"* to *"Lives On Land"*).

The blank node constructs could be considered as *class attributes* in the object oriented programming paradigm. *OWLReady* as an ontology-oriented programming interface significantly supports the discussed assertion by representing the blank node as a class attribute. This is shown below:

C3.has-characteristic.append('Lives on Land')

- **Concept extents**: FCA enables us to generate new ontological classes which are not explicitly defined within the initial data. For instance, we can interpret the existence of *"concept 6"* as a set of instances who are *"Animals"* and also *"Live on land"*. Attaching extents to *object concepts* could be considered as individuals of their corresponding ontological classes.

  The following flowchart depicted in Fig. 4.16 demonstrates how the asserted ontological classes are fed by the attached extent/intent information to their corresponding node in GSH. *extents* and *intents* have been coded differently in the source file of GSH graph which simplifies distinction among them. Algorithms 12 and 13 are used for asserting class properties and individuals, respectively.

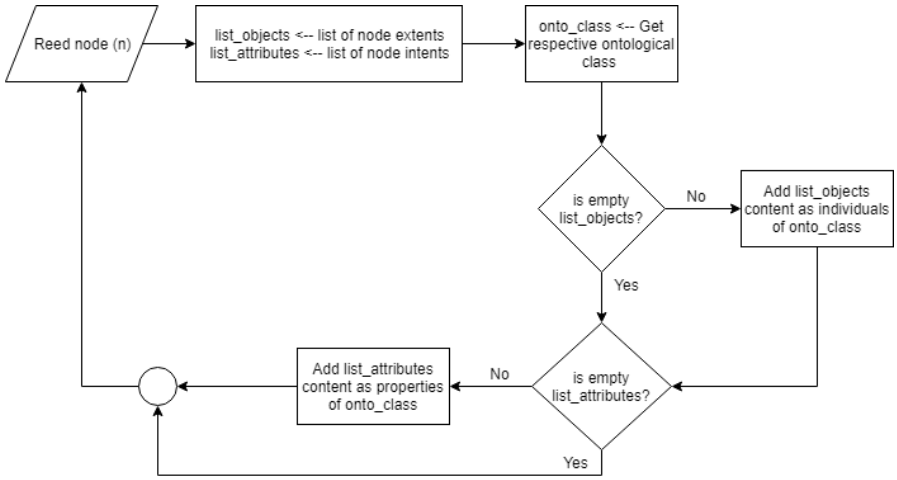


Figure 4.16 – *Augmenting asserted ontological class with concept information.*

Algorithm 12 *Assert class properties to corresponding ontological classes of attribute concepts.*

```
1: procedure ASSERT-PROPERTIES(node_name, ONTO=onto)
2:     atts_list ← list of attributes being attached to node_name
3:     for att in atts_list do
4:         atts_name ← "Att_" + att
5:         if ONTO[atts_name] in ONTO[node_name].has_characteristics then
6:             pass
7:         else
8:             ONTO[node_name].equivalent_to.append(
9:                 ONTO["has_characteristics"].some(ONTO[att_name]))
10:        end if
11:    end for
12: end procedure
```

Algorithm 13 *Assert class individuals to corresponding ontological classes if object concepts.*

```
1: procedure ASSERT-OBJECTS(node_name, ONTO=onto)
2:     objects_list ← list of objects being attached to node_name
3:     for obj in objects_list do
4:         obj_name ← "Obj_" + obj
5:         if obj_name in ONTO[node_name].instances() then
6:             pass
7:         else
8:             ONTO[node_name](obj_name)
9:         end if
10:    end for
11: end procedure
```

4.3.5 Indicate Ontology Class Disjointedness

As mentioned above in the context of Open World Assumption (OWA), OWL considers the ontology as an incomplete knowledge base. This implies that if something is not explicitly stated in the ontology, it can happen and should not be considered as *false* by the reasoner. Therefore, in order to avoid any inconvenient result, impossible facts must be clearly asserted into the ontology.

More specifically, this applies into the *disjointedness* relationship among the ontological classes. In OWL, the default assumption is that all classes are overlapping unless disjointedness axioms are asserted among them. In other words, in OWL, an instance can belong to multiple classes, and if there is no notion of disjointness defined between those classes, OWL will not detect that disjointness automatically.

Thanks to representations of concepts in GSH, we can determine pairs of disjoint concepts (which have been asserted as ontological classes) by means of *meet or infimum denoted by* $\bigwedge$ and *join or supremum denoted by* $\bigvee$ from Definition 5. As explained above, FCA generates a complete lattice in which each subset of concepts possesses a meet and join. GSH, on the other hand, is a partial lattice such that not all pairs have a meet or join (or both). Based on the definition provided in [209], for a given lattice L and its pairs of elements a and b, we can say that a and b are *"disjoint"* if and only if $a \bigwedge b = \emptyset$ (a and b have no *meet*).

Table 4.1 demonstrates pairwise *meet* comparisons of all fix points of GSH in Figure 3.3b. This is a symmetric matrix, therefore its upper diagonal has been left empty. For any pair of concepts, if there is no greatest common subconcept in the GSH, disjointedness axioms must be inserted among their corresponding ontological classes.

Let us consider *"concept 6"* and *"concept 10"* in Table 4.1 (to be asserted as disjoint classes in ontology). If we assume that these two classes are not disjoint, thus it is possible for an instance to be attached to both classes of *"concept 6"* and *"concept 10"*. On the other hand, observing their corresponding information in Figure 3.3b, *"concept 6"* is representing the

meet (/\)	*concept 2*	*concept 3*	*concept 4*	*concept 5*	*concept 6*	*concept 7*	*concept 8*	*concept 9*	*concept 10*
concept 2									
concept 3	concept 6								
concept 4	concept 7	-							
concept 5	-	concept 8	concept 9						
concept 6	concept 6	concept 6	-	-					
concept 7	concept 7	-	concept 7	-	-				
concept 8	-	concept 8	concept 10	concept 8	-	-			
concept 9	-	concept 10	concept 9	concept 9	-	-	concept 10		
concept 10	-	concept 10	concept 10	concept 10	-	-	concept 10	concept 10	

Table 4.1 – *Meet comparison matrix of concepts of GSH in Figure 3.3b. Highlighted cells in red indicate disjoint concepts (ontological classes).*

group of animals that live on land with two attached instances *(Cat , Dog)*. However, *"concept 10"* represents group of individuals which are plant and have the capability of living in both water and land (with an attached instance of *"Reed"*) which is in contrary to the specifications of *(Cat , Dog)* since none of the instances of *"concept 6"* can live in water. Thus, disjointedness axioms must be asserted among ontological classes of *"concept 6"* and *"concept 10"*.

Therefore, one additional step is required to explicitly state disjointedness among the already asserted ontological classes from the previous steps. The pseudo Algorithm for asserting disjointedness is provided in Algorithm 14 which accepts two input parameters, GSH as well as constructed ontology. This Algorithm computes the corresponding list of predecessors for each fix point of GSH. The result of calculation (*"pred"*) is kept in a Python dictionary (line 2). Lines 3-8 apply a pairwise comparison among all nodes using a pre_defined function (`"get_common()"`). Then, disjointedness is asserted among any two nodes having no common predecessors using *"AllDisjoint()"* function of OWLReady (line 6).

Algorithm 14 *Add disjointedness axioms among ontological classes.*

```
1: procedure MakeDisjoint(Graph = gsh , ONTO=onto)
2:     preds ← predecessors (Graph)
3:     for node1 , node2 in preds do
4:         if not common(node1,node2) then
5:             with ONTO:
6:                 AllDisjoint([ ONTO[node1] , ONTO[node2]])
7:         end if
8:     end for
9: end procedure
```

4.3.6 Expanding the Set of Classes and Individuals

As mentioned earlier, we followed a bottom up approach for asserting ontological classes and their corresponding individuals. Once the assertion of class *Products* is performed, the corresponding classes of *Companies, People, Locations* are also introduced to the ontology.

Based on our explanations in Section 2.3.6, subjects and objects of triples are clustered and

further refined to ensure that the most accurate individuals have been located within each cluster. Algorithm 15 is defined for the purpose of importing these classes along with their individuals into the ontology. Detailed procedure of this Algorithm has been explained in the following box:

Algorithm 15 *Import classes and individuals.*

```
1:  procedure DICT-TO-ONTO(input_dic , ONTO = onto)
2:      for k,v in input_dic.items() do
3:          if ONTO[k] is None then
4:              Add-to-Onto(k)
5:          else
6:              for ind in v do
7:                  if ind not in ONTO[k].instances() then
8:                      ONTO[k](ind)
9:                  end if
10:             end for
11:         end if
12:     end for
13: end procedure
```

Description of Dict-to-Onto procedure proposed in Algo. 15:

- Algorithm requires one input parameter as a simple python dictionary with name of classes as keys and their corresponding list of individuals as values. (e.g., input-dic = {$'Companies'$ = [$'Roche'$,$'Apple'$,...],$'People'$ = [$'SarahLi'$,...],$'Locations'$ = [$'Basel'$,$'SanFrancisco'$,...]})
- An iterator meets all keys and values of the input dictionary. For each key, it asserts an ontological class as subclass of the most general class *Thing* using the function of Algorithm 10. → (lines 3-5)
- New individual will be asserted into corresponding ontological class. → (lines 5-12)

4.3.7 Non-taxonomical Relationships from SVO Triplets

As mentioned above, there exist two types of properties in OWL named *object property* and *data type property*. The role of *object properties* is to connect two individuals of any ontology class(es) to each other. The connections among individuals of Figure 4.3 - such as *'based-in'* among *'Roche'* from class *'Companies'* and *'Basel'* from class *'Locations'* - are examples of potential *objects properties* in the ontology. In contrast, the usage of a *data type property* is to attach literal data (e.g., strings, numbers, datetypes, etc.) to the asserted individuals within an ontology. For instance, *'hasRevenuee'* could be a *data type* attached to one of the individuals of class *'Companies'* as a number indicating revenue of a company.

Thanks to extracted subject-verb-object (SVO) triplets, we can further improve our ontology by asserting non-taxonomical relationships in the form of *object properties* and *data type properties*. Regarding *object property* assertion, for every *SVO*, the corresponding individuals of its *subject* and *object* are identified and then associated in the ontology. For example, for the triplet of *'(Apple, hired , Sarah Li)'*, the directional relation *'hired'* from *'Apple'* to *'Sarah Li'* will be created. Moreover, right after adding each *object property*, we also add its corresponding inverse relation (i.e., 'inverse-hired' in the previous example). Such directional relation is necessary to capture both types of relation that can be derived from one verb, i.e., *hiring* someone or *being hired* by some organization.

For the purpose of asserting *data type properties*, our method acts as following. If for any of the SVO triples, the corresponding class of subject is detected but no individual name is matched with its respective object, a new *data type property* with the type of *string* will be added to the corresponding individual of the subject. Even though, this might not be a 100% accurate approach for obtaining ontological *data type properties*, it guarantees all extracted information from text analysis has been captured and asserted into the ontology. There is always a way to improve an engineered ontology by incorporating human knowledge. Thus verifying the correctness and consistency of asserted information will be achieve by applying a classifier (reasoner) as well as benefiting from a subject matter expert.

Both *object properties* and *data type properties* can be created in *OWLReady* by defining a new class and specifying its domain and range. Algorithm 16 is defined for asserting the information of each set of triplets into the ontology.

Algorithm 16 *Assert object/data type properties.*

```
1: procedure TRIPLETS-TO-PROPERTIES(xml_triplets , ONTO=onto)
2:     tree ← ET.parse('xml-file.xml')
3:     root ← tree.getroot()
4:     for svo in root.findall('svo') do
5:         subj ← svo.find ('subject').text
6:         pred ← svo.find ('predicate').text
7:         obj ← svo.find ('object').text
8:         subj_class ← onto.subj.__class__
9:         obj_class ← onto.obj.__class__
10:        if subj_class and obj_class then
11:            pred_name ← pred + "_OP"
12:            if ONTO[pred_name] is None then
13:                with ONTO:
14:                    types.new_class (pred_name , (ObjectProperty, ),
15:                    exec_body = lambda ns: ns.update ({"domain":
16:                    sub_class , "range":obj_class})
17:            end if
18:            ONTO[subj].ONTO[pred_name].append(ONTO[obj])
19:        else
20:            if ONTO[subj] is not None and ONTO[obj] is None then
21:                pred_name ← pred + "_DP"
22:                with ONTO:
23:                    types.new_class (pred_name , (DataProperty, ),
24:                    exec_body = lambda ns: ns.update ({"range":[str]})
25:            end if
26:            ONTO[subj].ONTO[pred_name].append(obj)
27:        end if
28:    end for
29: end procedure
```

Description of the procedure for converting Triplets to Properties proposed in Algorithm 16

- The Algorithm requires the xml file as well as name of the the constructed ontology as input parameter.
- As shown in Figure 2.13, each triple is recorded in a branch called *'SVO'* with sub branches of *'subject', 'predicate',* and *'object'*. In order to get access to the name of each branch, the content of the xml file is parsed into tree format and its root is preserved in *"root"* varaible. → (lines 2,3)
- An iterator iterates over triplets and for each triplet, its corresponding subject, predicate and object part are extracted. The corresponding ontological class of subject and object is also extracted. → (lines 4-9)
- Then the Algorithm verifies if subject and object are both presented in the ontology. In this case, the predicate must be asserted as an object property class among subject and object. In order to identify object properties from data type properties, we decided to append *"_OP"* to the name of the object property classes and *"_DP"* to end of data type property classes. → (lines 10-11)
- If the corresponding object property class is not previously asserted into the ontology, the next lines will be executed. OWLReady allows dynamic assertion of ontological classes using *"types.new_class()"* clause. For asserting an object properties, we should assigns its domain and range classes. (here *"subj_class"* and *"obj_class"* respectively). → (lines 12-17)
- Once the object property class is asserted or found in the ontology, the next line constructs the relationship among *"subj"* and *"obj"* as individuals of classes *"subj_class"* and *"obj_class"* respectively. → (line 18)

Triplets-to-Properties Algorithm procedure (continue):

- The next lines will be executed if only corresponding individual of *"subject"* is found in the ontology (i.e. there is no equivalent individual for *objects* in the ontology). As explained above, in order to transfer the whole extracted information from the text into the desired ontology, such *objects* are asserted as data type properties to their corresponding *subject*. Although OWLReady supports many other data types such as *int, float, bool, etc.*, for the sake of simplicity, we fixed the range of all data types as *"str"*. → (lines 20-25)
- the last line constructs the desired relationship among *"subj"* and *"obj"* of triplet with *"pred"* being changed to *"pred_DP"*. → (line 26)

4.4 Validating and Querying the Ontology

In the previous sections, we have discussed the implementation details of the proposed techniques for automatically constructing the core pieces of a domain ontology. In this section, we are going to discuss around the correctness and completeness of the designed ontology based on an ontology classifier (reasoner) and user defined queries.

As mentioned above, the implementation of the proposed pipeline has been performed using an ontology programming interface for Python, called *OWLReady*, which allows the ontology designer to manipulate and modify its components within the programming environment. Moreover, *OWLReady* syntax is very similar to a well-known ontology editing tool called *Protégé* [5] and dot notation. For the sake of visualization and querying, we imported our ontology into *Protégé*. It comprises well-known query languages such as *SPARQL* and *DL Query* which enable manipulating and retrieving information from asserted and classified ontology, respectively.

The constructed ontology using the proposed pipeline contains 359 classes, 726 individuals, 127 object properties and 5 data type properties. Figure 4.17 shows a screenshot of *Protégé* environment where the constructed ontology has been imported. For the purpose of simplicity, not all the classes are expanded and only relationships among high level classes and their sub classes are shown.

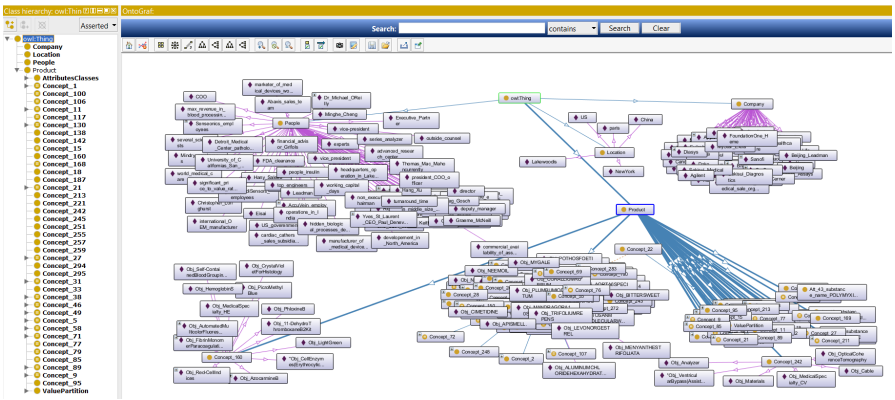


Figure 4.17 – *The screenshot of constructed ontology from (subject, verb, object). Not all classes and sub classes are expanded.*

Since the core content of this ontology has been extracted from real documents containing information about competititors in the pharmaceutical areas, we assigned artificial names (all started by *"Concept_"* term) to all subclasses of class *product*, to protect confidential information. The information about 187 products, including both drugs (medicines) and devices, are extracted from *OpenFDA* database. GSH of single valued context of the size $187 * 907$ is then constructed using *Galois sub-hierarchy builder* [6] tool. The dot file of the corresponding GSH lattice has been extracted and analysed by the *pygraphviz* library in

[5] https://protege.stanford.edu/
[6] https://sourceforge.net/projects/gshbuilder/

Python. The generated GSH consists of 232 nodes and 314 edges. Note that the number of nodes in the GSH is not necessarily equal to the number of rows (i.e., the number of objects in single-valued context). The maximum number of nodes in the GSH is equal to the sum of the number of objects and attributes. Subsequently, the set of defined mapping rules are applied to the analysed GSH as well as to the extracted triples of the form *(subject, verb, object)*.

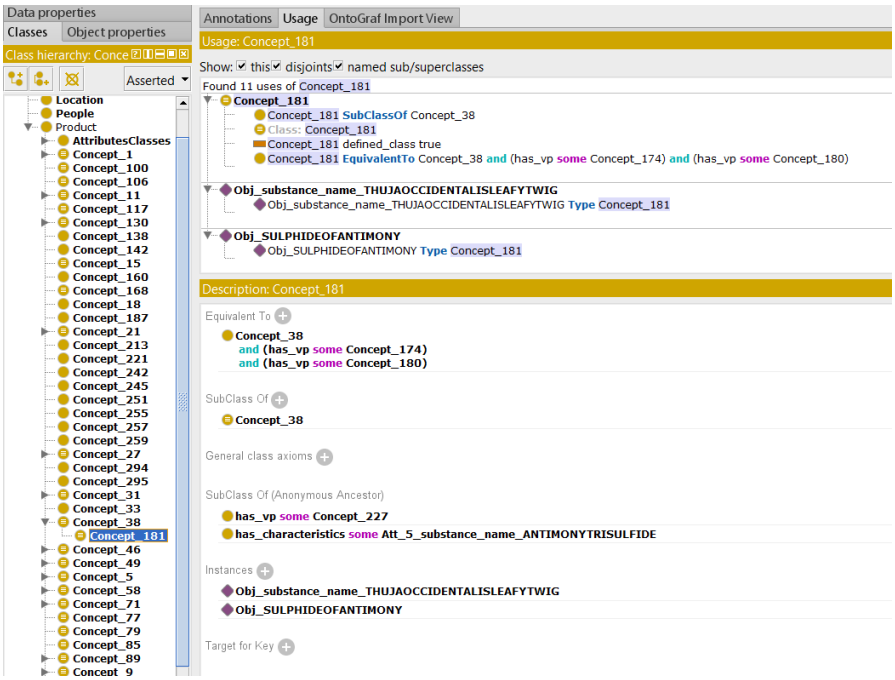
One of the crucial tasks of any ontology designer is to validate its correcteness and completeness. Inconsistency in an ontology can be caused by any invalid asserted facts into ABox or TBox including instantiating an unsatisfiable class, instantiating disjoint classes, conflicting assertions, conflicting axioms with nominals or no instantiation possible. We have validated the constructed ontology by running `"sync_reasoner()"`- as the *OWLReady* classifier - which is capable of re-classifying ontology classes and instances using *HermiT* reasoner [193].

4.4.1 Querying the Constructed Ontology

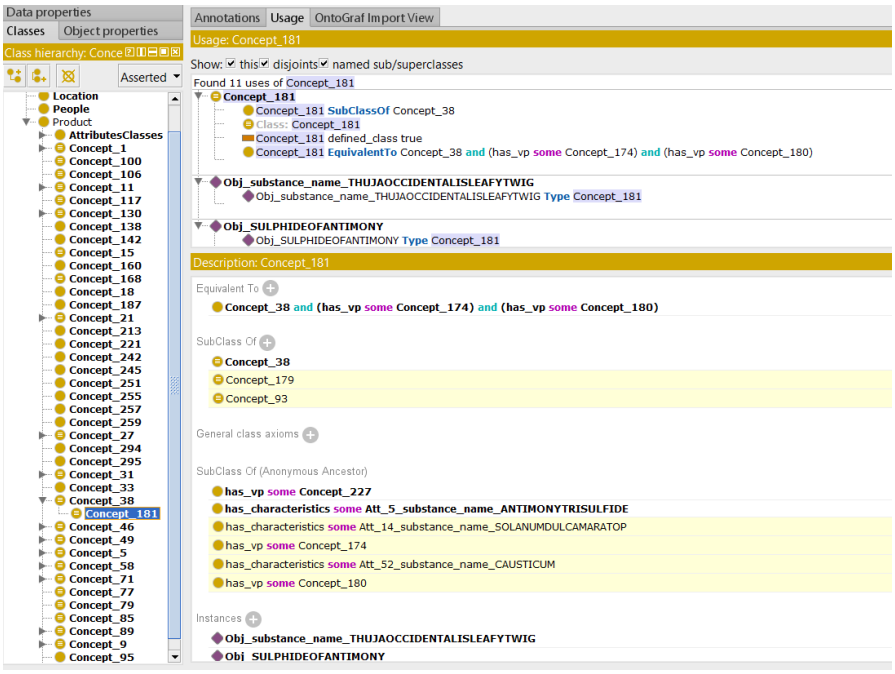
Querying the ontology and assessing the results could also verify the correctness of the information into the ontology. To do so, we imported the *.owl* file of the constructed ontology into *Protégé*. As shown in Figure 4.18, running the reasoner on the imported *owl* file, properly re-classifies the instances and classes without any inconsistency report. *"Concept_181"* is chosen as an example to clarify how the reasoner could infer not explicitly asserted relationships based on class definition. In Figure 4.18a, it has been explicitly asserted that *"Concept_181"* is a direct subclass of *"Concept_38"* and also an indirect subclass of *"Concept_174"* and *"Concept_180"*. Another specification of *"Concept_181"* is that it represents a class in which all its individuals have *"ANTIMONYTRISULFIDE"* as their substance. This characteristic has been inherited from its direct superclass *"Concept_38"*. On the other hand, some extra information is shown in the highlighted rows of Figure 4.18b which indicates that all individuals of *"Concept_181"* do not only have *"ANTIMONYTRISULFIDE"*, but also contain *"SOLANUMDULCAMARATOP"* and *"CAUSTICUM"* being inherited from its indirect superclasses *"Concept_93"* and *"Concept_179"*, respectively. The inferred hierarchy of *"Concept_180"* is also shown in Figure 4.19

Query-answering is also considered an alternative approach for validating the annotated axioms within an ontology. *Protégé* supports querying the imported ontology in multiple ways. Among all *SPARQL* is frequently used for retrieving data from RDF models. It however, does not support OWL entailments and inferring and only provides the asserted statements. On the other hand, *DL Query* requires simple formulation of OWL expressions for inferring implicit information from explicitly annotated statements. In the following, multiple examples of implemented queries are shown. The corresponding result of each query is also provided with some explanations.

- **Query-01: (DL Query)**
  - **Question:** Which products (as well as their individuals - if asserted any) have medical specification of *"Hematology"* or *"Immunology"*?



(a) Asserted information of *"Concept_181"*. It is directly connected to *"Concept_38"*



(b) *"Concept_181"* after re-classification by reasoner. The highlighted rows are inferred based on asserted information in the definition of *"Concept_181"*

Figure 4.18 – *The result of applying reasoner in order to re-classify ontology classes and individuals.*

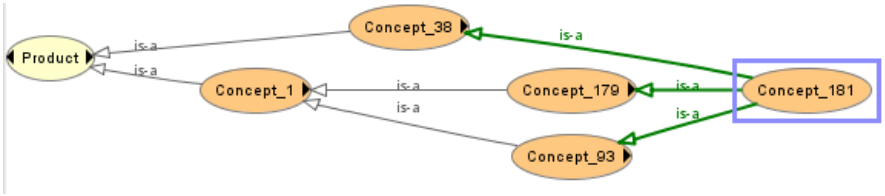


Figure 4.19 – *"Concept_180" is a multi-parent ontological class. By runnign the reasoner, it inherits all specification of its super classes as shown in Figure 4.18b .*

  - **Answer:** The result of query is provided in Figure 4.20. The ontology returns the correct output. Although, the medical speciality of *Hematology (HE)* and *Immunology (IM)* is not asserted in individual levels, they have been displayed in the result since they inherited all asserted and inferred properties of their ontological classes.

- **Query-02: (SPARQL Query)**
  - **Question:** List of people being hired by companies?
  - **Answer:** The result of query is provided in Figure 4.21. As mentioned in previous chapters, the non-taxonomical relationships among individuals have been extracted from text. Considering extracted information from documents, the result are sound and accurate.

- **Query-03: (DL Query)**
  - **Question:** Find all classes and individuals having the type of *"Human OTC Drug"*?
  - **Answer:** *"Human OTC Drug"* has been explicitly asserted as characteristics of classes *"Concept_27"* and *"Concept_142"*. Running the reasoner and executing the query, could correctly classify new inferred relationship. As the result, not only individuals of *"Concept_142"* have been displayed (*Obj_Flasher*, *Obj_InfraredHematomaDetector*, *Obj_Shell*, *Obj_Temography*, *Obj_Trabeculotome*, *Obj_Trephine*), but also instances of sub_classes of *"Concept_27"* and *"Concept_142"* are shown. This is due to the fact that all sub_classes of an ontological class (as well as their individuals), inherit the properties of their super_classes.

- **Query-04: (SPARQL Query)**
  - **Question:** List of companies having collaboration with each other?
  - **Answer:** All companies having *"collaborated"* as an object-property among them are shown. This information is imported from list of extracted triplets.

As shown above, both validation approaches have proven the correctness and completeness of the designed ontology. In a nutshell, the reasoner is used to check whether the constructed ontology is logically consistent and it is well represented in OWL. Moreover, we followed a query answering approach to verify the coherence of the content of the ontology with domain

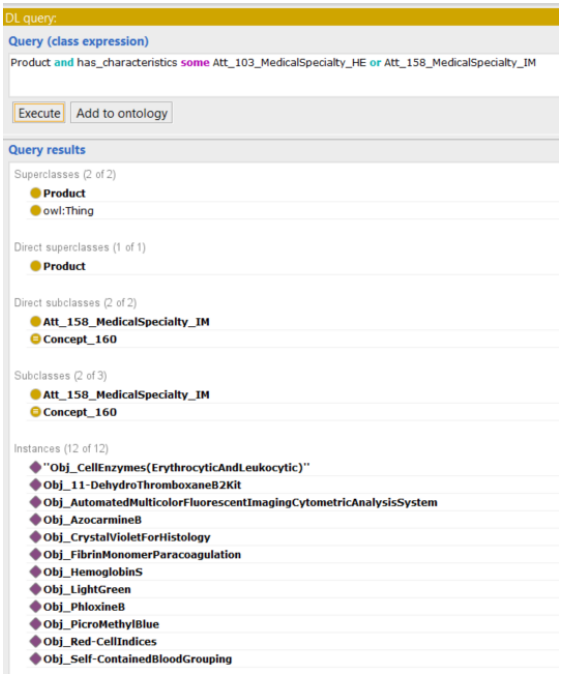


Figure 4.20 – *List of products with medical speciality of Immunology or Hematology.*

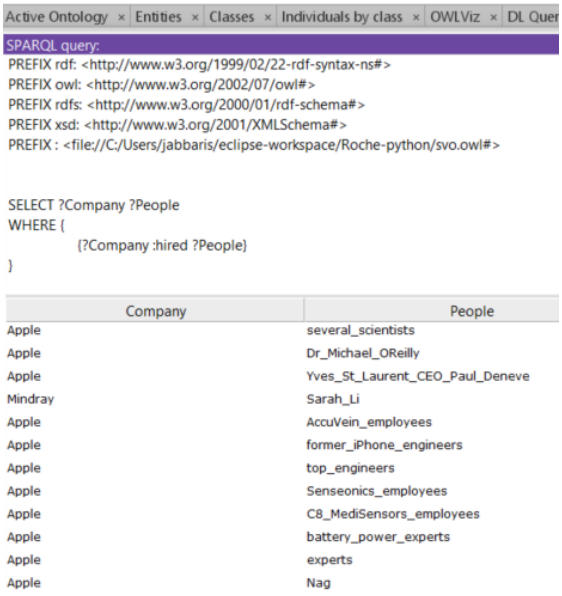


Figure 4.21 – *List of people being hired by companies. Query returns exactly the same subjects and objects having non-taxonomical relationship of "hired" within xml file.*

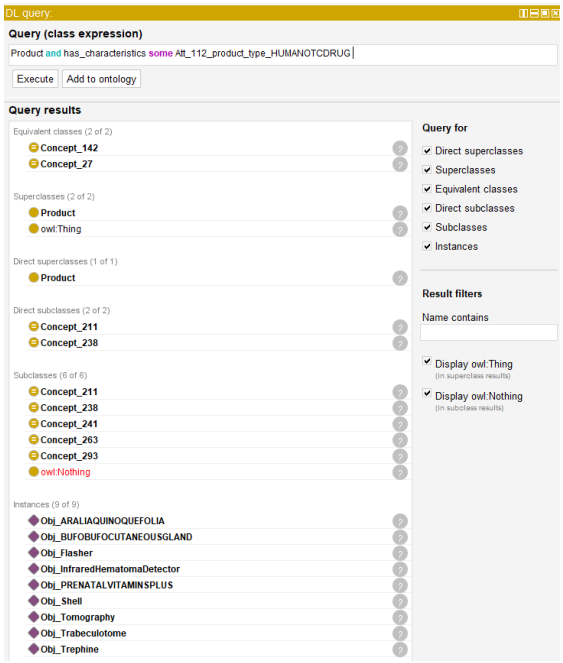
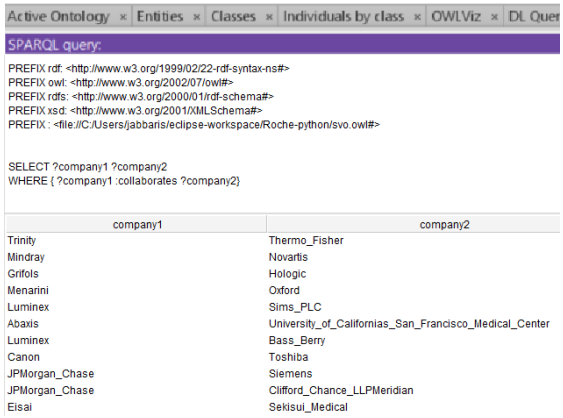


Figure 4.22 – *Products and their individuals with characteristics of "Human OTC Drug".*



company1	company2
Trinity	Thermo_Fisher
Mindray	Novartis
Grifols	Hologic
Menarini	Oxford
Luminex	Sims_PLC
Abaxis	University_of_Californias_San_Francisco_Medical_Center
Luminex	Bass_Berry
Canon	Toshiba
JPMorgan_Chase	Siemens
JPMorgan_Chase	Clifford_Chance_LLPMeridian
Eisai	Sekisui_Medical

Figure 4.23 – *Companies and their collaborators.*

knowledge by testing it against multiple test cases. This not only proved the ability of the generated ontology in information retrieval and knowledge extraction, but also confirmed the accurate formal structure such as hierarchical and non-hierarchical taxonomies, asserted and inferred relationships, annotated domain axioms, etc.

4.5 Conclusion

In this chapter, we started by reviewing the background of ontology, and then we described how to build an ontology using a precise set of prominent mapping rules. We showed how a GSH structure, derived from analyzing *OpenFDA*, can be transferred into different components of ontology.

We then augmented the cornerstone of the constructed domain ontology using triplets of interest extracted from texts. Our proposed pipeline is then validated by applying rules for building a real ontology in the domain of pharmaceuticals. Moreover, we could extract valuable insights by querying the engineered ontology.

Our proposed method can be generalized regardless of the domain of interest, and it can be considered as a promising approach for (semantic) knowledge extraction from textual data.

5 Conclusions

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

- Alan Turing, *Computing Machinery and Intelligence 1950*

Knowledge representation in the form of standard knowledge bases such as ontologies is crucial for extracting explicit and implicit information from unstructured textual data. Most previously proposed ontology learning methods and systems suffer from the lack of access to a comprehensive repository of a domain knowledge, as well as being too specific to a given domain of interest (i.e., the lack of ability to easily generalize to other contexts). Our proposed approach in this thesis for ontology learning from textual data demonstrates an end to end pipeline for addressing the aforementioned challenges to some extent by utilizing two powerful data analysis methods known as Natural Language Processing (NLP) and Formal Concept Analysis (FCA).

In particular, the following difficulties for ontology learning from unstructured textual data have been identified and addressed in the previous chapters. The major challenges are: 1) The inability of current techniques to analyse textual data and extract domain specific named entities, 2) Selecting a proper knowledge base to enhance missing information within original data, 3) Determining a robust and powerful technique for constructing the backbone of the ontology, 4) defining a set of mapping rules which guarantees the correctness of asserted facts into the ontology, and 5) evaluating the accuracy and completeness of the engineered ontology based on the provided information. To prove that the proposed pipeline works well, we applied our Algorithms to a toy example (in Chapter 4) by providing a set of mapping rules to construct a consistent ontology. The consistency and correctness of the generated ontology based on the proposed mapping rules have been verified by running the reasoner (classifier) as well as querying asserted information.

5.1 Summary of Contribution

The major contribution of this dissertation in the filed of ontology learning from unstructured data is summarized as follows:

Chapter 2 addresses the first two problems by benefiting from NLP techniques to extract essential information in the form of short sentences being coded as triplets of *subject-verb-object*. Extracted triplets provide a twofold solution. Firstly they enable us to pull out high level ontological classes and their respective individuals. Secondly they empower the constructed ontology by delivering a complete list of non-taxonomical relationships among the asserted individuals (ABox),

Chapter 3 provides a comprehensive explanations around Formal Concept Analysis (FCA) as a powerful mathematical technique used for developing the backbone of our desired ontology. Our major contribution in the domain of FCA relies on analysing the existing Algorithms for producing domain concepts as well as their corresponding lattice diagram. Our emphasis was on accelerating two famous FCA Algorithms called *NextClosure* and *Nourine* by parallelizing their binary based implementation. Moreover, a new hybrid approach is proposed for generating both formal concepts and a concept lattice. It has been shown that out method could noticeably outperform the original version of the Algorithms depending on density of provided input binary table.

Chapter 4 summarizes all necessary steps towards the creation of a domain ontology with an adequate quality. A set of mapping rules has been defined and implemented each for completing a specific piece of ontology engineering pipeline. The first set of mapping rules are entirely dependent on the output of FCA. They mainly address the development procedure of taxonomical relationships among ontological classes, assertion of the main characteristics and properties of each class, as well as declaring their corresponding individuals. Multiple Algorithms have been proposed for assembling various pieces of the ontology construction puzzle. Resolving poly-hierarchy among ontological classes and sub classes has been issued by means of normalization or untangling. Untangling of multi-parent classes is achieved by initiating all classes in a tree-based format and letting the reasoner infer new relationships among classes. This allows automatic readjustments of sub/super class relationships once a new update occurs inside the ontology with no human effort. Another major contribution is to define an Algorithm which allows to characterize or make assertions about a specific ontology class. Although *meta-modeling* (*punning* in OWL) has been proposed as a solution for this problem, it is not supported by the utilized ontology oriented programming paradigm (*OWLReady*). Therefore, a new design approach has been proposed based on the concept of *blank nodes* and defining more restrictions in the ontology class definition. One additional Algorithm is defined which tackles the problem of asserting FCA objects into ontological instances. This is the final step of converting ALL extracted information by FCA into an ontology. It must be clarified that the correctness of all aforementioned Algorithms is verified by running on a toy example. They have been further applied on a real world data set being

pulled from an open source database (*OpenFDA*).

The constructed ontology has been augmented by benefiting from text-based triplets being retrieved from textual documents. A set of high level classes are also asserted which provided enhancements in ontological TBox. On the other hand, by having triplets of the form *subject-verb-object*, we could magnify ABox axioms by asserting both *subject* and *object* as instances and *verb* as relationship among them. It must be noted that the automatic assertion of inverse relationships is also made among *object* and *subject* as *"object verb-inverse subject"*.

All proposed methods and Algorithms of this thesis have been applied on a real case study from a pharmaceutical company where final goal was to extract knowledge about Market and Competitive Intelligence (MCI). In this case study a group of pdf documents, with a set of controlled vocabularies are provided as input, where we applied the standard NLP techniques to extract domain specific concepts and individuals. Moreover, additional analyses have been applied on specific concepts and their corresponding individuals and a bottom up approach has been conducted for designing and constructing the final ontology. More specific classes and relationships are also asserted to generate the backbone of the ontology. Assertion of extra concepts, individuals and their relationships further improved the quality of the engineered ontology, which tremendously helped in increasing the quality of the information extracted from the engineered ontology-based knowledge base.

5.2 Discussion and Future Work

Nowadays, new data are being produced more than any time before, and so many new scientific and non-scientific disciplines have been proposed to the globe. It is often the case that someone wants to discover a new domain but he lacks a basic knowledge about that context. Building ontologies from unstructured data is a good starting point for deriving a set of taxonomies and concept hierarchies for a given unknown domain.

Besides building the cornerstone of a knowledge model, our proposed model can be used to classify new words into one of the existing ontological concepts (or a perhaps a new one) depending on the characteristics or attributes that the new words share with the existing ones (i.e., previously learned words). Moreover, we can use this method for indexing documents to be used by smart search engines (where documents are tagged by not only their own keywords, but also by other words that are somehow linked to the content of the documents in a semantic way). Another application of the proposed method for learning ontology is to derive implication rules and semantic knowledge. The possibility to further adjust the engineered ontology and to incorporate subject matter expert knowledge to come up with a better ontology is also appealing.

So far, we mainly discussed about some of the potential applications of the proposed method. We would also like to mention some of the future works to be done for the purpose of improving the proposed method. The first topic is around NLP, and how to improve text preprocessing

so that more meaningful triplets are extracted from the corpus. Dealing with huge number of documents is also another challenge. Using entity detection (e.g., which word is disease or medication) for building entity-specific ontologies (such as ontologies for diseases and medications, separately) and then connecting those ontologies via relational formal contexts (such as the one that links diseases to medications) also seems to be a right approach for learning more accurate and complete ontologies in a given domain of knowledge (such as medicine). As another example for future topic, we can also mention how to effectively update an ontology that has been created some time ago after receiving more documents in a given corpus. Adding more and more automation for handling big and complex graphs (as a consequence of a big corpus) is also a topic of interest for research.

The power of FCA has been proven in creating the cornerstone of desired ontology. Therefore, taking advantage of other types processing infrastructures that are readily available can speed up the implementation of its Algorithms even further and open the door for FCA to become more widespread also in the context of Big Data.

On the whole, the aforementioned questions and suggestions provide some interesting insights into future research directions for ontology learning from any unstructured textual data.

Bibliography

[1] Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. Background and foreground knowledge in dynamic ontology construction: Viewing text as knowledge maintenance. In *Proceedings of the Semantic Web Workshop*, 2003.

[2] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. Catriple: Extracting triples from wikipedia categories. In *Asian Semantic Web Conference*, pages 330–344. Springer, 2008.

[3] B López et al. Automatic discovery of synonyms and lexicalizations from the web. *Artificial intelligence research and development*, 131:205, 2005.

[4] Ratanachai Sombatsrisomboon, Yutaka Matsuo, and Mitsuru Ishizuka. Acquisition of hypernyms and hyponyms from the www. In *Proceedings of the 2nd International Workshop on Active Mining*, 2003.

[5] Rada Mihalcea. Using wikipedia for automatic word sense disambiguation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 196–203, 2007.

[6] Marco Baroni and Silvia Bernardini. Bootcat: Bootstrapping corpora and terms from the web. In *LREC*, page 1313, 2004.

[7] Serge Sharoff. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98, 2006.

[8] Rudi L Cilibrasi and Paul MB Vitanyi. The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3):370–383, 2007.

[9] Violeta Seretan, Luka Nerima, and Eric Wehrli. Using the web as a corpus for the syntactic-based collocation identification. 2004.

[10] Hsin-Hsi Chen, Ming-Shun Lin, and Yu-Chuan Wei. Novel association measures using web search with double checking. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1009–1016. Association for Computational Linguistics, 2006.

[11] Asunción Gómez-Pérez, David Manzano-Macho, et al. A survey of ontology learning methods and techniques. *OntoWeb Deliverable D*, 1(5), 2003.

[12] Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. The state of the art in ontology learning: a framework for comparison. *The Knowledge Engineering Review*, 18(4):293–316, 2003.

[13] Ying Ding and Schubert Foo. Ontology research and development. part 1-a review of ontology generation. *Journal of information science*, 28(2):123–136, 2002.

[14] Lucas Drumond and Rosario Girardi. A survey of ontology learning procedures. *WONTO*, 427:1–13, 2008.

[15] Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018:bay101, 2018.

[16] Michele Missikoff, Roberto Navigli, and Paola Velardi. Integrated approach to web ontology learning and engineering. *Computer*, 35(11):60–63, 2002.

[17] Roberto Navigli and Paola Velardi. Semantic interpretation of terminological strings. In *Proc. 6th Int'l Conf. Terminology and Knowledge Eng*, pages 95–100, 2002.

[18] Paola Velardi, Paolo Fabriani, and Michele Missikoff. Using text processing techniques to automatically enrich a domain ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 270–284. ACM, 2001.

[19] Paola Velardi, Roberto Navigli, Alessandro Cucchiarelli, and Francesca Neri. Evaluation of ontolearn, a methodology for automatic learning of domain. *Ontology Learning from Text: Methods, evaluation and applications*, 123:92, 2005.

[20] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[21] Philipp Cimiano and Steffen Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005.

[22] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In *Ecai*, volume 321, page 27, 2000.

[23] Alexander Maedche and Steffen Staab. The text-to-onto ontology learning environment. In *Software Demonstration at ICCS-2000-Eight International Conference on Conceptual Structures*, volume 38, pages 890–930. sn, 2000.

[24] Alexander Maedche and Raphael Volz. The ontology extraction & maintenance framework text-to-onto. In *Proc. Workshop on Integrating Data Mining and Knowledge Management, USA*, pages 1–12, 2001.

[25] Günter Neumann, Rolf Backofen, Judith Baur, Markus Becker, and Christian Braun. An information extraction core system for real world german text processing. *arXiv preprint cmp-lg/9706023*, 1997.

[26] Mirna El Ghosh, Hala Naja, Habib Abdulrab, and Mohamad Khalil. Ontology learning process as a bottom-up strategy for building domain-specific ontology from legal texts. In *ICAART (2)*, pages 473–480, 2017.

[27] David Faure and Claire Nédellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *LREC workshop on adapting lexical and corpus resources to sublanguages and applications*, volume 707, page 30. Citeseer, 1998.

[28] David Faure and Claire Nedellec. Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system asium. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 329–334. Springer, 1999.

[29] David Faure and Thierry Poibeau. First experiments of using semantic knowledge learned by asium for information extraction task using intex. In *Ontology Learning ECAI-2000 Workshop*, pages 7–12. Citeseer, 2000.

[30] Udo Hahn and Martin Romacker. Content management in the syndikate system–how technical documents are automatically transformed to text knowledge bases. *Data & Knowledge Engineering*, 35(2):137–159, 2000.

[31] Udo Hahn and Martin Romacker. The syndikate text knowledge base generator. In *Proceedings of the first international conference on Human language technology research*, pages 1–6. Association for Computational Linguistics, 2001.

[32] Ana Oliveira, F Câmara Pereira, and Amílcar Cardoso. Automatic reading and learning from text. In *Proceedings of the international symposium on artificial intelligence (ISAI)*. sn, 2001.

[33] Francisco Câmara Pereira, A Oliveira, and Amílcar Cardoso. Extracting concept maps with clouds. In *Proceedings of the Argentine Symposium of Artificial Intelligence (ASAI)*, 2000.

[34] Maryam Hazman, Samhaa R El-Beltagy, and Ahmed Rafea. A survey of ontology learning approaches. *International Journal of Computer Applications*, 22(9):36–43, 2011.

[35] Lucia Specia and Enrico Motta. A hybrid approach for relation extraction aimed at the semantic web. In *International Conference on Flexible Query Answering Systems*, pages 564–576. Springer, 2006.

[36] Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *IJCAI*, pages 659–664. Citeseer, 2005.

[37] Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, pages 82–86. Morgan Kaufmann Publishers Inc., 2002.

[38] Francesc Ribas. On learning more appropriate selectional restrictions. *arXiv preprint cmp-lg/9502009*, 1995.

[39] Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1320–1327, 2016.

[40] Thushari Atapattu, Katrina Falkner, and Nickolas Falkner. A comprehensive text analysis of lecture slides to generate concept maps. *Computers & Education*, 115:96–113, 2017.

[41] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304, 2005.

[42] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics, 2006.

[43] Gerhard Wohlgenannt and Filip Minic. Using word2vec to build a simple ontology learning system. In *International Semantic Web Conference (Posters & Demos)*, 2016.

[44] Martin Kavalec and Vojtech Svaték. A study on automated relation labelling in ontology learning. *Ontology Learning from Text: Methods, evaluation and applications*, (123):44–58, 2005.

[45] Rosie Jones. *Learning to extract entities from labeled and unlabeled text*. PhD thesis, Citeseer, 2005.

[46] Barbara Rosario. *Extraction of semantic relations from bioscience text*. PhD thesis, Citeseer, 2005.

[47] MAEF Belal, H Abdel-Galil, and YM Saber. Ontology extraction from text: Related works between arabic and english languages. *Int. J*, 4(8), 2016.

[48] Alvaro L Fraga and Marcela Vegetti. Semi-automated ontology generation process from industrial product data standards. In *III Simposio Argentino de Ontologías y sus Aplicaciones (SAOA)-JAIIO 46 (Córdoba, 2017).*, 2017.

[49] Ishara Sandun, Sagara Sumathipala, and Gamage Upeksha Ganegoda. Self-evolving disease ontology for medical domain based on web. *International Journal of Fuzzy Logic and Intelligent Systems*, 17(4):307–314, 2017.

[50] Jean Petit, Jean-Charles Boisson, and Francis Rousseaux. Discovering cultural conceptual structures from texts for ontology generation. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 0225–0229. IEEE, 2017.

[51] David Sánchez and Antonio Moreno. Learning non-taxonomic relationships from web documents for domain ontology construction. *Data & Knowledge Engineering*, 64(3):600–623, 2008.

[52] Minghua Pei, Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. Constructing a global ontology by concept mapping using wikipedia thesaurus. In *22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008)*, pages 1205–1210. IEEE, 2008.

[53] Nicolas Weber and Paul Buitelaar. Web-based ontology learning with isolde. In *Proc. of the Workshop on Web Content Mining with Human Language at the International Semantic Web Conference, Athens GA, USA*, volume 11. Citeseer, 2006.

[54] Simin Jabbari and Kilian Stoffel. A methodology for extracting knowledge about controlled vocabularies from textual data using fca-based ontology engineering. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1657–1661. IEEE, 2018.

[55] Simin Jabbari and Kilian Stoffel. Fca-based ontology learning from unstructured textual data. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 1–10. Springer, 2018.

[56] Simin Jabbari and Kilian Stoffel. Ontology extraction from mongodb using formal concept analysis. In *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*, pages 178–182. IEEE, 2017.

[57] Simin Jabbari and Kilian Stoffel. A hybrid algorithm for generating formal concepts and building concept lattice using nextclosure and nourine algorithms. 2016.

[58] Simin Jabbari and Kilian Stoffel. Parallel execution of binary-based nextclosure algorithm. In *International Workshop on Algorithms for FCA and Data Mining*, 2016.

[59] Sophia Ananiadou and John McNaught. *Text mining for biology and biomedicine*. Citeseer, 2006.

[60] Sophia Ananiadou and Goran Nenadic. Automatic terminology management in biomedicine. *Text mining for biology and biomedicine*, 685, 2006.

[61] Michael Krauthammer and Goran Nenadic. Term identification in the biomedical literature. *Journal of biomedical informatics*, 37(6):512–526, 2004.

[62] Jong C Park and Jung-Jae Kim. Named entity recognition. *Text mining for biology and biomedicine*, pages 121–142, 2006.

[63] Asma Ben Abacha and Pierre Zweigenbaum. Means: A medical question-answering system combining nlp techniques and semantic web technologies. *Information processing & management*, 51(5):570–594, 2015.

[64] Jose M Fernandez, Robert Hoffmann, and Alfonso Valencia. ihop web services. *Nucleic acids research*, 35(suppl_2):W21–W26, 2007.

[65] Dietrich Rebholz-Schuhmann, Harald Kirsch, Miguel Arregui, Sylvain Gaudan, Mark Riethoven, and Peter Stoehr. Ebimed—text crunching to gather facts for proteins from medline. *Bioinformatics*, 23(2):e237–e244, 2007.

[66] Andreas Doms and Michael Schroeder. Gopubmed: exploring pubmed with the gene ontology. *Nucleic acids research*, 33(suppl_2):W783–W786, 2005.

[67] Thomas Goetz and Claus-Wilhelm von der Lieth. Pubfinder: a tool for improving retrieval rate of relevant pubmed abstracts. *Nucleic acids research*, 33(suppl_2):W774–W778, 2005.

[68] Hans-Michael Müller, Eimear E Kenny, and Paul W Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS biology*, 2(11):e309, 2004.

[69] Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 2010.

[70] Buzhou Tang, Yonghui Wu, Min Jiang, Yukun Chen, Joshua C Denny, and Hua Xu. A hybrid system for temporal information extraction from clinical text. *Journal of the American Medical Informatics Association*, 20(5):828–835, 2013.

[71] Katherine P Liao, Tianxi Cai, Guergana K Savova, Shawn N Murphy, Elizabeth W Karlson, Ashwin N Ananthakrishnan, Vivian S Gainer, Stanley Y Shaw, Zongqi Xia, Peter Szolovits, et al. Development of phenotype algorithms using electronic medical records and incorporating natural language processing. *bmj*, 350:h1885, 2015.

[72] Karla L Caballero Barajas and Ram Akella. Dynamically modeling patient's health state from electronic medical records: A time series approach. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78. ACM, 2015.

[73] Ergin Soysal, Jingqi Wang, Min Jiang, Yonghui Wu, Serguei Pakhomov, Hongfang Liu, and Hua Xu. Clamp–a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association*, 25(3):331–336, 2017.

[74] Ewoud Pons, Loes MM Braun, MG Myriam Hunink, and Jan A Kors. Natural language processing in radiology: a systematic review. *Radiology*, 279(2):329–343, 2016.

[75] Peter Jackson and Isabelle Moulinier. *Natural language processing for online applications*. John Benjamins, 2007.

[76] Stephen Soderland. Learning to extract text-based information from the world wide web. In *KDD*, volume 97, pages 251–254, 1997.

[77] Isaac Martín de Diego, Alberto Fernández-Isabel, Felipe Ortega, and Javier M Moguerza. A visual framework for dynamic emotional web analysis. *Knowledge-Based Systems*, 145:264–273, 2018.

[78] Vandana Korde. Information extraction for personalised services based on conference alerts. *International Journal of Data Mining, Modelling and Management*, 8(1):93–105, 2016.

[79] Jerry R Hobbs. The generic information extraction system. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993.

[80] Jerry R Hobbs. Information extraction from biomedical text. *Journal of biomedical informatics*, 35(4):260–264, 2002.

[81] William Nash Locke and Andrew Donald Booth. *Machine translation of languages: fourteen essays*. Published jointly by Technology Press of the Massachusetts Institute of . . . , 1955.

[82] Warren Weaver. Translation. *Machine translation of languages*, 14:15–23, 1955.

[83] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[84] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2), 1990.

[85] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[86] Joseph Weizenbaum. Computer power and human reason: From judgment to calculation. 1976.

[87] Jennifer Hill, W Randolph Ford, and Ingrid G Farreras. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in Human Behavior*, 49:245–250, 2015.

[88] Anirudh Khanna, Akshay Garg, Akshita Bhalla, et al. Designing natural language processing systems with quickscript as a platform. In *Progress in Advanced Computing and Intelligent Engineering*, pages 305–312. Springer, 2018.

[89] Cyril Joe Baby, Faizan Ayyub Khan, and JN Swathi. Home automation using iot and a chatbot using natural language processing. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6. IEEE, 2017.

[90] Sameera A Abdul-Kader and JC Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7), 2015.

[91] Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185, 2017.

[92] James R Meehan. Tale-spin, an interactive program that writes stories. In *Ijcai*, volume 77, pages 91–98, 1977.

[93] Wendy G Lehnert. The process of question answering. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE, 1977.

[94] Robert Wilensky. Understanding goal-based stories. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE, 1978.

[95] Wendy G Lehnert. Plot units and narrative summarization. *Cognitive science*, 5(4):293–331, 1981.

[96] William F Clocksin and Christopher S Mellish. *Programming in Prolog: Using the ISO standard*. Springer Science & Business Media, 2012.

[97] Roger C Schank and Robert P Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

[98] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[99] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[100] Brian Roark. Probabilistic top-down parsing and language modeling. *Computational linguistics*, 27(2):249–276, 2001.

[101] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.

[102] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387, 2016.

[103] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM, 2017.

[104] Nitin Indurkhya and Fred J Damerau. *Handbook of natural language processing*, volume 2. CRC Press, 2010.

[105] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[106] Geoffrey Nunberg. *The linguistics of punctuation*. Number 18. Center for the Study of Language (CSLI), 1990.

[107] Bernard EM Jones. Exploring the role of punctuation in parsing natural text. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 421–425. Association for Computational Linguistics, 1994.

[108] nltk tokenizer. http://www.nltk.org/api/nltk.tokenize.html, 2018. Accessed: 2018-11-17.

[109] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[110] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 168–175. Association for Computational Linguistics, 2002.

[111] Dekang Lin. Principar—an efficient, broad-coverage, principle-based parser. *arXiv preprint cmp-lg/9407024*, 1994.

[112] Dekang Lin. Dependency-based evaluation of minipar. In *Treebanks*, pages 317–329. Springer, 2003.

[113] Daniel DK Sleator and Davy Temperley. Parsing english with a link grammar. *arXiv preprint cmp-lg/9508004*, 1995.

[114] Lluís Màrquez and Horacio Rodríguez. Part-of-speech tagging using decision trees. In *European Conference on Machine Learning*, pages 25–36. Springer, 1998.

[115] Thorsten Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics, 2000.

[116] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, 1996.

[117] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992.

[118] Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*, 2016.

[119] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[120] Pranjal Awasthi, Delip Rao, and Balaraman Ravindran. Part of speech tagging and chunking with hmm and crf. *Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006*, 2006.

[121] Hugo CC Carneiro, Felipe MG França, and Priscila MV Lima. Multilingual part-of-speech tagging with weightless neural networks. *Neural Networks*, 66:11–21, 2015.

[122] Igor Aleksander, Massimo De Gregorio, Felipe Maia Galvao França, Priscila Machado Vieira Lima, and Helen Morton. A brief introduction to weightless neural systems. In *ESANN*, pages 299–305. Citeseer, 2009.

[123] Elaheh Sadredini, Deyuan Guo, Chunkun Bo, Reza Rahimi, Kevin Skadron, and Hongning Wang. A scalable solution for rule-based part-of-speech tagging on novel hardware accelerators. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 665–674. ACM, 2018.

[124] 1999.

[125] Wikipedia contributors. Hoffmann-la roche — Wikipedia, the free encyclopedia, 2019. [Online; accessed 2-September-2019].

[126] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[127] Samet Atdağ and Vincent Labatut. A comparison of named entity recognition tools applied to biographical texts. In *2nd International conference on systems and computer science*, pages 228–233. IEEE, 2013.

[128] Simone Marinai. Metadata extraction from pdf papers for digital library ingest. In *2009 10th International Conference on Document Analysis and Recognition*, pages 251–255. IEEE, 2009.

[129] Chris Mattmann and Jukka Zitting. *Tika in action*. Manning Publications Co., 2011.

[130] Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *LREC2012*, 2012.

[131] fedelopez77. fedelopez77/langdetect, Oct 2017.

[132] Nickdavidhaynes. nickdavidhaynes/spacy-cld, Oct 2018.

[133] Jacob Perkins. *Python text processing with NLTK 2.0 cookbook*. Number 1. Packt Publishing Ltd, 2010.

[134] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[135] Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics, 1996.

[136] Advaith Siddharthan. An architecture for a text simplification system. In *Language Engineering Conference, 2002. Proceedings*, pages 64–71. IEEE, 2002.

[137] Rudolf Wille. *Introduction to formal concept analysis*. Fachbereich Mathematik, Technische Hochschule Darmstadt, 1996.

[138] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[139] Bernhard Ganter, Gerd Stumme, and Rudolf Wille. *Formal concept analysis: foundations and applications*, volume 3626. springer, 2005.

[140] Ben Dushnik and Edwin W Miller. Partially ordered sets. *American journal of mathematics*, 63(3):600–610, 1941.

[141] Garrett Birkhoff. *Lattice theory*, volume 25. American Mathematical Soc., 1940.

[142] Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In *International Conference on Formal Concept Analysis*, pages 314–339. Springer, 2009.

[143] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[144] Robert Godin and Hafedh Mili. Building and maintaining analysis-level class hierarchies using galois lattices. In *ACM SIGplan Notices*, volume 28, pages 394–410. ACM, 1993.

[145] Marianne Huchard and Hervé Leblanc. Computing interfaces in java. In *Automated Software Engineering, 2000. Proceedings ASE 2000. The Fifteenth IEEE International Conference on*, pages 317–320. IEEE, 2000.

[146] Anne Berry, Marianne Huchard, Amedeo Napoli, and Alain Sigayret. Hermes: an efficient algorithm for building galois sub-hierarchies. In *CLA: Concept Lattices and their Applications*, pages 21–32. Universidad de Malaga, 2012.

[147] Wiebke Petersen. A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 53:296–308, 2004.

[148] Marianne Huchard, Hervé Dicky, and Hervé Leblanc. Galois lattice as a framework to specify building class hierarchies algorithms. *RAIRO-Theoretical Informatics and Applications*, 34(6):521–548, 2000.

[149] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. Extraction of feature models from formal contexts. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, page 4. ACM, 2011.

[150] Gabriela Arévalo, Anne Berry, Marianne Huchard, Guillaume Perrot, and Alain Sigayret. Performances of galois sub-hierarchy-building algorithms. In *International Conference on Formal Concept Analysis*, pages 166–180. Springer, 2007.

[151] Hervé Dicky, Christophe Dony, Marianne Huchard, and Thérese Libourel. Ares, adding a class and restructuring inheritance hierarchy. In *BDA*, pages 25–42, 1995.

[152] Anne Berry, Marianne Huchard, Ross M McConnell, Alain Sigayret, and Jeremy P Spinrad. Efficiently computing a linear extension of the sub-hierarchy of a concept lattice. In *International Conference on Formal Concept Analysis*, pages 208–222. Springer, 2005.

[153] Alain Sigayret. *Data mining: une approche par les graphes*. PhD thesis, 2002.

[154] Cornelia E Dowling. On the irredundant generation of knowledge spaces. *Journal of Mathematical Psychology*, 37(1):49–62, 1993.

[155] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine learning*, 24(2):95–122, 1996.

[156] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational intelligence*, 11(2):246–267, 1995.

[157] Eugene M Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.

[158] Jean-Paul Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, 96:31–47, 1986.

[159] Michel Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, pages 21–25, 1969.

[160] Bernhard Ganter. Two basic algorithms in concept analysis. In *International conference on formal concept analysis*, pages 312–340. Springer, 2010.

[161] Sergei O Kuznetsov. A fast algorithm for computing all intersections of objects from an arbitrary semilattice. *Nauchno-Tekhnicheskaya Informatsiya Seriya 2-Informatsionnye Protsessy i Sistemy*, (1):17–20, 1993.

[162] Christian Lindig. *Algorithmen zur Begriffsanalyse und ihre Anwendung bei Softwarebibliotheken.* PhD thesis, Braunschweig University of Technology, Germany, 1999.

[163] MI Zabezhailo, VG Ivashko, SO Kuznetsov, MA Mikheenkova, KP Khazanovskii, and OM Anshakov. Algorithms and programs of the jsm-method of automatic hypothesis generation. *Automatic Documentation and Mathematical Linguistics*, 21(5):1–14, 1987.

[164] Sergei O Kuznetsov and Sergei A Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.

[165] Petr Krajca, Jan Outrata, and Vilem Vychodil. Advances in algorithms based on cbo. In *CLA*, volume 672, pages 325–337, 2010.

[166] Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Information processing letters*, 71(5-6):199–204, 1999.

[167] Radim Belohlavek. Introduction to formal concept analysis. *Palacky University, Department of Computer Science, Olomouc*, 47, 2008.

[168] Hong Qi, Dayou Liu, Chengquan Hu, Ming Lu, and Liang Zhao. A parallel algorithm based on search space partition for generating concepts. In *Proceedings. Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004.*, pages 241–248. IEEE, 2004.

[169] Huaiguo Fu and Engelbert Mephu Nguifo. Partitioning large data to scale up lattice-based algorithm. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 537–541. IEEE, 2003.

[170] Nilander RM de Moraes, Luis E Zárate, and Henrique C Freitas. A distributed algorithm for formal concepts processing based on search subspaces. In *ICEIS (1)*, pages 105–111, 2010.

[171] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for fca. In *CLA*, volume 2008, pages 71–82. Citeseer, 2008.

[172] Jan Outrata and Vilem Vychodil. Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. *Information Sciences*, 185(1):114–127, 2012.

[173] Vilem Vychodil. *A new algorithm for computing formal concepts*. na, 2008.

[174] Sergei O Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 384–391. Springer, 1999.

[175] Petko Valtchev and Vincent Duquenne. Towards scalable divide-and-conquer methods for computing concepts and implications. In *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR)*, pages 3–6, 2003.

[176] xflr6. xflr6/concepts, Jul 2019.

[177] Rudolf Wille. Lattices in data analysis: How to draw them with a computer. In *Algorithms and order*, pages 33–58. Springer, 1989.

[178] Zaki Parthasarathy Ogihara, MJ Zaki, S Parthasarathy, M Ogihara, and W Li. New algorithms for fast discovery of association rules. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*. Citeseer, 1997.

[179] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Learning algorithms using a galois lattice structure. In *[Proceedings] Third International Conference on Tools for Artificial Intelligence-TAI 91*, pages 22–29. IEEE, 1991.

[180] Bernhard Ganter and Sergei O Kuznetsov. Stepwise construction of the dedekind-macneille completion. In *International Conference on Conceptual Structures*, pages 295–302. Springer, 1998.

[181] Guy-Vincent Jourdan, Jean-Xavier Rampon, and Claude Jard. Computing on-line the lattice of maximal antichains of posets. *Order*, 11(3):197–210, 1994.

[182] Lhouari Nourine and Olivier Raynaud. A fast incremental algorithm for building lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):217–227, 2002.

[183] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[184] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[185] Vreda Pieterse and Derrick G Kourie. Lists, taxonomies, lattices, thesauri and ontologies: paving a pathway through a terminological jungle. *KO KNOWLEDGE ORGANIZATION*, 41(3):217–229, 2014.

[186] John Davies, Dieter Fensel, and Frank Van Harmelen. *Towards the semantic web: ontology-driven knowledge management*. John Wiley & Sons, 2003.

[187] Christiaan Fluit, Marta Sabou, and Frank Van Harmelen. Supporting user tasks through visualisation of light-weight ontologies. In *Handbook on ontologies*, pages 415–432. Springer, 2004.

[188] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[189] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089*, 2012.

[190] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer. *W3C recommendation*, 27(1):123, 2009.

[191] Marek Obitko. Reasoning, 2016.

[192] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[193] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: an owl 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

[194] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *International joint conference on automated reasoning*, pages 292–297. Springer, 2006.

[195] Sunitha Abburu. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, 57(17), 2012.

[196] Hilary Putnam. The meaning of 'meaning'. *Philosophical papers*, 2, 1975.

[197] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

[198] Lina Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007.

[199] Taha A Kass-Hout, Zhiheng Xu, Matthew Mohebbi, Hans Nelsen, Adam Baker, Jonathan Levine, Elaine Johanson, and Roselie A Bright. Openfda: an innovative platform providing access to a wealth of fda's publicly available data. *Journal of the American Medical Informatics Association*, 23(3):596–600, 2015.

[200] Bernhard Ganter and Rudolf Wille. Conceptual scaling. In *Applications of combinatorics and graph theory to the biological and social sciences*, pages 139–167. Springer, 1989.

[201] Alan L Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 121–128. ACM, 2003.

[202] Semantic web best practices and deployment working group, 2001.

[203] Michael Grüninger and Mark S Fox. Methodology for the design and evaluation of ontologies. 1995.

[204] Franz Baader and Philipp Hanschke. *A scheme for integrating concrete domains into concept languages*. Deutsches Forschungszentrum für Künstliche Intelligenz, 1991.

[205] Boris Motik. On the properties of metamodeling in owl. *Journal of Logic and Computation*, 17(4):617–637, 2007.

[206] Christopher A Welty and David A Ferrucci. What's in an instance. Technical report, Technical report, RPI Computer Science, 1994.

[207] Alejandro Mallea, Marcelo Arenas, Aidan Hogan, and Axel Polleres. On blank nodes. In *International semantic web conference*, pages 421–437. Springer, 2011.

[208] Aidan Hogan, Marcelo Arenas, Alejandro Mallea, and Axel Polleres. Everything you always wanted to know about blank nodes. *Journal of Web Semantics*, 27:42–69, 2014.

[209] Ralph Abraham, Jerrold E Marsden, and Jerrold E Marsden. *Foundations of mechanics*, volume 36. Benjamin/Cummings Publishing Company Reading, Massachusetts, 1978.