FROM UNRESTRICTED NATURAL LANGUAGE REQUIREMENTS TO DOMAIN MODELS

by

Ulaş Onur Dedeoğlu

B.S., Computer Engineering, Boğaziçi University, 2023

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Software Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisor Asst. Prof. Fatma Başak Aydemir. Her constant encouragement and support allowed me to continue my study even in the hardest times. Her precious guidance and endless patience helped me navigate in academic research which was an unknown territory for me.

I also would like to thank Prof. Şule Gündüz Öğüdücü, and Prof. Tunga Güngör for taking their valuable time to read and review my thesis. I appreciate their helpful remarks and insightful recommendations.

I am thankful to my parents Hatice and Cafer Dedeoğlu and my grand parents Hafize and Halil Püsküllü for supporting to me through all my life and allowing me to chase my dreams. I would like to thank my sister MD. Bilge Eylem Dedeoğlu for being the best sister I could ever ask for. She has been the most influential person in my entire life, and I could never thank her enough for her efforts to support me. İrem Cansu Bozkaya and her endless encouragement, joyful attitude, and most valuable companionship allowed me to endure all hardships during my studies. She gave me hope, cheered me up, shared her experiences and provided me a safe place. I also would like to thank my dear friend Hakkı Tayfun Bulak for bringing me joy and his true friendship.

I would like to thank my Managing Director/Colleague Özgür Kaya for his understanding, for trusting my judgement, allowing me more flexibility and supporting me through my thesis. He guided me in many aspects of my life and cultivated an environment for his team to prosper.

Lastly, I would like to thank the whole Software Engineering program. It has been an excellent opportunity to learn from their expertise and wisdom.

ABSTRACT

FROM UNRESTRICTED NATURAL LANGUAGE REQUIREMENTS TO DOMAIN MODELS

Domain models are used to establish general overview of a software system to ease the communication between the project stakeholders and as various inputs for other software development activities. Due to these benefits, domain model extraction is an important task for both researchers and practitioners of software projects. Domain model extraction process bears challenges such as being labour intensive, requiring extensive communication which is not always possible in real-world projects, and coverage completeness being hard to attain. For these reasons, researchers propose methods to ease and aid the domain extraction process using natural language processing methods. In this study, we propose a fully automated approach to extract domain models from unstructured natural language requirements which combines capabilities of modern language models, a state-of-the-art term ranking algorithm, and a rule based extraction module. We evaluate our proposal with both industrial and educational data sets and perform a quantitative evaluation. The state of the art overperform our approach in the relation detection performance and overall precision of the pipeline. In terms of domain concept coverage and individual concept detection we achieve on par or better overall performance compared to state-of-the-art methods. Our approach perform better in data sets from the industry compared to the students' data sets.

ÖZET

KISITLANMAMIŞ DOĞAL DIL YAZILIM GEREKLILIKLERINDEN ALAN MODELLERINE

Alan modelleri, proje paydaşları arasındaki iletişimi kolaylaştırmak ve diğer yazılım geliştirme faaliyetlerine çeşitli girdiler sağlamak amacıyla bir yazılım sistemine genel bir bakış oluşturmak için kullanılır. Bu faydalar alan modellerini endüstrideki uygulayıcılar ve akademik araştırmacılar için ilgi çekici bir çalışma haline getirmiştir. Alan modellerini tanımlama ve inşa etme faaliyetleri sırasında çeşitli problemler uygulayıcıların karşısına çıkmaktadır. Gerekli olan işgücü maliyetinin fazlalığı, gerçek endüstride her zaman hali hazırda kaynak olarak mevcut olmayan derinlemesine alan uzmanlığı gerekliliği, inşa edilen modellerde kavram bütünlüğünü ve kapsamının sağlanmasının zorluğu gibi problemler bu görevin gerçekleştirilmesini zorlaştırmaktadır. Bu engellerin aşılması ve alan modellerini inşa etmeyi kolaylaştırmak için araştırmacılar doğal dil işleme yöntemleri önermişlerdir. Bu çalışmada modern büyük dil modelleri, bir güncel terim sıralama algoritması ve kural tabanlı çıkarım modülünü bir arada kullanarak tamamen otomatik bir şekilde, sınırlandırılmamış doğal dil ile yazılmış yazılım gerekliliklerini işleyerek alan modeli inşa eden bir yaklaşım öneriyoruz. Bu yaklaşımımızı değerlendirmek için hem öğretim temelli veri setleri hem de endüstri temelli veri setlerini kullanarak sayısal bir analiz gerçekleştirdik. Özgün alan terimlerinin kapsanması ve tekil alan terimlerinin tespit yetenekleri bakımından mevcut teknolojinin seviyesine denk veya daha iyi bir performans elde ettik. Önerdiğimiz yaklaşım, endüstri temelli veri setleri ile daha iyi bir performans gösterdi ancak alan terimlerinin arasındaki ilişkilerin tespiti ve genel sistem kesinliği göze alındığında mevcut yaklaşımlardan daha düşük bir performans gözlemledik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

FN	False Negative
FP	False Positive
TP	True Positive
$\alpha_{threshold}$	Relevance threshold for a token in a given domain

LIST OF ACRONYMS/ABBREVIATIONS

CNN	Convolutional Neural Network
NER	Named-Entity Recognition
NL	Natural Language
NLP	Natural Language Processing
NLP4RE	Natural Language Processing for Requirement Engineering
NLP4SE	Natural Language Processing for Software Engineering
NP	Noun Phrase
POS	Part of Speech
RE	Requirement Engineering
UML	Unified Modeling Language

1. INTRODUCTION

In this chapter, we will provide background knowledge on the domain model extraction task including problem domain and key challenges in Section 1.1, our objectives in Section 1.2, and general overview of our approach in Section 1.3. After providing fundamental knowledge on domain we will share our contributions in Section 1.4, and the organization of the thesis in Section 1.5.

Domain model is a holistic conceptual model of a given domain which represents interconnected concepts and their relations. It represents the abstractions in the system and communicates the coherent entirety of concepts with various stakeholders. By mining the concepts and relations, practitioners can create a baseline domain model which can be used as a communication medium between project stakeholders. Despite the benefits which domain models provide, practitioners may refuse to extract domain models for their software projects since the domain model extraction task being highly laborious for human practitioners.

In software projects, system and user requirements usually specified using requirement statements. Most common requirement statements formats are user stories, declarative shall statements, and unrestricted NL statements. Along with free form text documents which describes the domain elements, these requirement specifications can be used as a source to construct a domain model which will be used to communicate between various parties. There are many studies which tries to extract key concepts from given text input and use these concepts to build diagrams or summarize the domain. In our study we design a fully autonomous pipeline with no informed heuristics, and with unrestricted NL requirement statements. We also extract relations between these concepts to specify interaction between domain elements.

Whether it is key term extraction or model generation, different ML approaches are used by researchers to achieve their tasks. In our pipeline we use a combination

of modern big language models, a recent high performing term ranking algorithm, and lastly rule based heuristics which utilizes syntactic and statistical features of given requirement statements.

1.1. Problem Statement

Model generation is a key study area under NLP4SE domain. The task contains some key challenges itself such as

- Construction of domain models is a manually laborious task.
- Human practitioners may lack domain expertise to provide a correct, complete domain model.
- Construction of domain models may require constant communication between stakeholders, which sometimes may be impossible due to geographical obstacles, unavailability of the staff, and other factors.
- Extraction of model concepts from provided input
- Extraction of concept attributes.
- Extraction of model relations between elements.
- Achieving acceptable performance metrics for specific model.
- Generalization capabilities of the suggested method, such as input compatibility, sensitivity to data set characteristics, and its statistical features.
- Applicability of the suggested method across multiple domains.

Model generation studies also suffer from both the biased nature of preparing ground truth for given data sets, and also the requirements specifications being based on various perceptions. These two challenges create a lack of a general confirmation between all stakeholders of a software project or system. Model generation process is not a binary process where one golden model is generated using same resources.

Existing model generation methods differs from each other and our proposed method in terms of evaluation process, completeness of pipeline, restriction on input

format, and performance. Existing studies lack at least one of the following features; complete extraction of domain concepts and relations, robustness against data set characteristics, fully autonomous process, unrestricted NL input format, avoidance of using informed heuristic, public resources to reproduce their results, or a quantitative analysis of their pipeline.

1.2. Purpose of the Study

In this study we aim to extract domain models described in natural language documents. Similar studies on the subject uses different approaches such as rule based extraction, text ranking, word embedding. We propose an extraction pipeline which utilizes modern language models, term ranking algorithms and heuristic based rules all together. We used spaCy NLP framework which uses state of art methods for various NLP tasks. Using spaCy and Python programming language can provide readability and ability to maintain the tool for third party practitioners. We showed that utilizing modern tools and technologies for NLP tasks in combination with heuristic rules which benefit from linguistic features of English language can assist the domain model extraction to practitioners. And we demonstrated that our tool can decrease the human labour drastically without sacrificing from coverage greatly. We also extend the labeled data sets on the model extraction literature and contribute the community where finding labeled data set is a challenge itself.

1.3. Approach

In this study, we accept the study by Arora [1] as the baseline and derived our pipeline from their approach. Since implementation details of the pipeline proposed by Arora is unavailable, we interpreted the description of his proposal as our fundamental pipeline. Eventually our pipeline transformed greatly and differed from the baseline study. We continued our preliminary research explained in chapter 2 and focused on domain model generation and term extraction.

With the knowledge we gather from studying other studies, we decided to implement additional pipeline components for increased performance and increased ability to generalize. Our overall pipeline components are as follows; We utilize a transformer based pipeline in spaCy. We use noun chunks feature of spaCy to extract noun phrases which provide us a baseline for concept extraction. Then, we perform n-grams extraction from the noun chunks. We build a statistical based large English language model pipeline in spacY. Then, we use this pipeline to access vector representations of tokens in n-grams and calculate a similarity/relativity score for each term. Using these score we build a weighted undirected graph. Lastly, we perform a text rank using the graph and top candidates are mapped into additional concepts. After performing initial concept extraction, we use predefined heuristic rules to extract additional concepts. Relations are also extracted using heuristic rules. Relation's source and target concepts are also extracted and mapped into concepts during relation extraction. Lastly, after extracting all domain model elements in lists, we performed our quantitative analysis to assess the performance of our pipeline.

1.4. Contributions

Our study attempts to address challenges for domain model extraction described in section 1.1. While addressing these challenges we make valuable contributions to the NLP domain in general.

Firstly, we implement a pipeline for domain model extraction using state the of art technologies and selected heuristics compiled from many studies from model extraction, NLP, and information retrieval domains. Our proposed method;

- Supports unrestricted NL text input.
- Uses no informed heuristic.
- Provides both concept and relation extraction.
- Achieves on par or better performance compared to related studies.

- Evaluated quantitatively across multiple domains and with different data sets tagged by internal and external parties with varying backgrounds.
- Fully automated.
- Has public repository.

Secondly, we provide a data set which can be used by researchers as an external resource to avoid tagger bias. Model extraction and NLP4SE studies require extensive data sets to train, test, and evaluate their methods, and currently lack of tagged data set creates a challenge for researchers. We contributed to the literature by tagging 863 requirement statements from different domains and originators, identifying 854 unique and 3277 individual domain concepts, and lastly 1098 individual concept relations. We made our data sets publicly available for other researchers.

Finally, we performed an extensive quantitative analysis of our pipeline which can shed light on various aspects of domain extraction problem. We investigated the performance of our approach for concept and relation extraction. Additionally, we assess the performance of each method and each heuristic to reveal their significance in the pipeline. Modern language model capabilities and heuristic based methods are examined in terms of performance for both unique coverage and individual coverage. We believe our efforts contributed to both domain model extraction literature and also NLP domain in general.

1.5. Organization

This thesis is structured as follows. In Chapter 2 we review earlier studies on NLP , NLP4SE, NLP4RE, key term extraction, and model generation in a top down fashion so that we can describe our problem domain and identify available tools and techniques. Chapter 3 describes the extraction approach in detail, and discuss the process of selection of tools, technologies and heuristics by explaining design decisions. Chapter 4 reports our quantitative evaluation study by describing data sets and its characteristics, explaining performance measurements for both complete pipeline and

also for individual components of the pipeline, and lastly threats to validity of our study. Finally, in Chapter 5 we summarize our findings and share our insights on the future work.

2. RELATED WORK

In this chapter, we will discuss Natural Language Processing and it's sub fields, their brief histories, current state of the art, and their relevance with our proposed method. Firstly we will be discussing NLP in general in Section 2.1, then, key phrase extraction domain will be discussed in detail in Section 2.2. After discussing these topics, we will be directing our focus towards our method by talking about NLP for Software Engineering in Section 2.3, NLP for Requirement Engineering in Section 2.4, and lastly model generation in Section 2.5.

2.1. Natural Language Processing

Natural Language Processing (NLP) is a systematic process which focuses on representation and linguistic analysis of natural language for computer systems. NLP researchers aim to achieve human-like conceptual grasp and performance in their tools and techniques [2]. There are two major study topics under NLP, firstly symbolic NLP techniques which studies the linguistic theory and focuses on heuristic based techniques and semantic networks, secondly statistical NLP techniques which focuses on ML techniques and training of statistical models of a natural language. In the early days, methods mostly relied on rule based symbolic NLP. Linguists and scientists formalized the language specific rules and heuristics. One of the earliest examples of such application of NLP is the 1954 Georgetown-IBM experiment [3]. Lately, statistical methods become more popular than the symbolic methods [4, 5]. This advancement is due to increased amount of available data used for training models and ever-growing computational power used for performing costly training tasks. Today, most of the primal NLP tasks like dependency parsing, tokenization, etc. are performed by statistical methods. Also, In 2018, Young reported that latest deep learning methods are started to being used in NLP scene as well [6]. Particularly transformer-based architectures shown to be promising. Language models like BERT [7], RoBERTa [8], Generative Pre-trained Transformer [9], and Text-to-Text Transfer Transformer [10] have demon-

strated extraordinary performance across many NLP tasks across multiple languages and domains.

Even though a trend transition happened from symbolic to statistical NLP, still both approaches have their pros and cons. Symbolic methods usually lack of generalization and completeness. Additionally they are sensitive to faulty input. On the other hand statistical methods are hard to built accurately since they require massive amounts of data and they are also sensitive to faulty input. NLP researchers strive to fine tune their methods and models to overcome these disadvantages.

Currently NLP is used to handle many tasks and it is divided into sub fields. Sentiment analysis, parsing, information extraction and analysis, named entity recognition, part of speech tagging, text summarizing, question answering are examples of these NLP sub fields. Researchers are exploring methods to obtain multilingual or cross lingual understanding for NLP. Additionally, ethical aspect of the NLP domain is being discussed since the emergence of the research on bias detection using NLP.

A recent study has been performed by Castanha in 2022, where they performed a bibliographic review on the NLP domain, and they reported that NLP research is channeled towards areas such as deep learning based methods, data mining using social media platforms, sentiment analysis, data capturing and learning, and modeling [11]. Deep learning based methods usually focus on more specific tasks like key phrase extraction, and pos tagging. Deep learning based key phrase extraction methods will be discussed in detail in the below subsection 2.2.2. Other trending areas will be discussed in the following sections 2.3, 2.4, 2.5.

In this chapter we are specifically interested in:

- Key phrase extraction because of it's significance for noun phrase extraction
- NLP for Software Engineering since it is a super set domain for our method
- NLP for Requirement Engineering which will demonstrate our end goals

- Model generation domain to discuss available methods and compare them against our proposed method.

2.2. Keyphrase Extraction

Key phrases are phrases which can summarize the contents of a given document and key phrase extraction is the process of automatically extracting these key phrases [12]. Key phrase extraction is an interesting research area since it can be used for various tasks such as semantic indexing of documents, linking semantic concepts within a set of documents, summarizing the content for human readable format and many other labour intensive tasks. Key phrase extraction is also an important tool to extract concepts in a given natural language text input which is one of our main steps to perform domain model generation in this study.

We can group existing key phrase extraction techniques into two major groups, namely, supervised and unsupervised techniques. Unsupervised methods include statistical, graph based, embedding based, and lastly, language models based techniques. Supervised methods are deep learning based and traditional machine learning based methods. This categorization provides a holistic view of the domain an also used in a recent comprehensive systematic literature review on key phrase extraction domain [13]. Below Section 2.2.1 details the related work on unsupervised methods for key phrase extraction and Section 2.2.2 focuses on supervised methods.

2.2.1. Unsupervised Methods

There are four sub categories under unsupervised methods for key phrase extraction.

Firstly we have statistic based methods. These methods will use statistical features of the document like term frequency, inverse document frequency [14], least allowable seen frequency [15], co-occurrence clusters [16], term positions, case statistics, word

relation by neighbouring words, and word occurrence within different sentences [17,18]. In statistical based unsupervised methods, baseline method is tf-idf which stands for term frequency - inverse document frequency. Also KP Miner [15], and Key Cluster [16] are other notable methods for the task. More recently firstly Won's method [18] and then YAKE [17] proved to be a great method for statistical based key phrase extraction combined with context knowledge or other methods from different schools of thought.

Graph based unsupervised methods rely on representing the domain with a graph where nodes represent candidate concepts and each edge is a connection between these nodes. To determine final concepts these nodes will be ranked using graph ranking algorithms. Text Rank [19] is the foundational study for graph based methods. Page Rank [20] and Positional Function [21] are other influential examples of graph based methods. There are different approaches for graph based methods such as weighted edges approach for Single Rank [22], co-occurrence for [23], combinational methods which include combinations of the above approaches [24], [14], there are also approaches where multiple documents are used as source like Expand Rank [25], and using external contexts to improve the performance like Cite Text Rank [14]. While these approaches use statistics to build and utilize graphs, some approaches support graph based extraction using clustering [26, 27] and Latent Dirichlet Allocation [28, 29]. To decrease vagueness and ambiguity of graph based methods, semantics are incorporated into graph based methods [30–34].

Additionally, researchers propose many embedding based unsupervised methods for key phrase extraction. They base their method on representing textual elements using embeddings and co-occurrence matrices such as Latent Semantic Analysis and Latent Dirichlet Allocation. Embedding approaches differ in different studies. Some popular approaches are using word embeddings [35], sentence embeddings [36,37], and word vectors [38]. Researches proposed methods to exploit these different approaches like Embed Rank [39] and Reference Vector Algorithm [40] which stands out among other methods.

Lastly, Language Model Based Methods are available for unsupervised key phrase extraction. Methods based on language models basically assigns statistical probability for any n-gram word sequences. These methods can be based on n-grams [41] or semantic contexts using deep learning [42].

2.2.2. Supervised Methods

Along with unsupervised methods, we also have supervised methods under our belts to extract key phrases from documents. These methods usually trained to classify key phrases in binary fashion. Approaches vary from training a naive Bayes model using statistical features of candidate phrases [42] to feature based linguistic methods [43]. Witten's work is then extended by multiple proposals [44, 45]. More recently, WINGNUS [46] and CeKe [47], in given order, used regular expressions and naive Bayes based model with features utilization and managed to display notable performances. Recently more advanced supervised methods are proposed by Wang [48] and Shen [49]. These methods combine and extend previously discussed methods.

Another school of thought is to approach extraction task as a ranking task in a supervised fashion. This approach aims to rank key phrases non-binarily. Jiang [50] and Zhang [51] proposed methods in this manner.

Lastly, we have supervised methods which does not fall into earlier categories. Bougouin proposed a method which extends the already discussed topic rank method with utilizing additional domain graphs [52]. Yang's more recent work uses a variation of Latent Dirichlet Allocation to extract key phrases those corresponding to a certain event. There are also other researchers [53, 54] with notable work where they propose approaching key extraction as a sequential labeling task.

Lastly, we have deep learning methods which extract key phrases in supervised fashion. Deep learning is a very common approach for extracting key phrases and concepts from text documents. Zhang [55], Meng [56], Chen's [57] methods used recurrent

neural network with differing architectures to extract key phrases. Ye also proposed another deep learning based method to improve extraction performance on unlabeled data since other methods being inferior in this task, by utilizing transfer learning [58].

2.3. Natural Language Processing for Software Engineering

Software engineering discipline is entwined with natural language. There are many tasks under software engineering which requires expression and demonstration using natural language. This organic relation motivated researchers to explore NLP for Software Engineering (NLP4SE). NLP4SE is an emerging research area where practitioners utilize NLP techniques to ease and support the SE practices. NLP4SE domain contains research sub fields like requirements engineering, code completion and code analysis, automated code generation, unit and acceptance test generation, modeling, bug detection, software maintenance and countless more niche fields concerned with requirement engineering and other software engineering related tasks across the full software life-cycle.

There are various methods and methodologies to perform automated analysis of free form text which uses NLP tools and techniques for software engineering. These methods aim to automate software engineering tasks and complement the software development process. These diverse range of methods and methodologies includes machine learning, deep learning, statistical analysis and information retrieval techniques. Researchers tackle NLP4SE challenges using variations and combinations of these tools.

We can broadly classify NLP4SE tasks into 5 major categories; requirements engineering, code analysis/ generation/ summarization, bug detection and resolution, software maintenance and evolution, and lastly language interfacing.

Firstly, NLP4RE area is focused on leveraging NLP advancements to support requirements elicitation, analysis, and domain understanding. NLP4RE researchers try to extract useful information from natural language textual requirements using

methods like named-entity-recognition (NER), sentiment analysis, and key concept extraction. NLP4SE tools provide improved understanding of user-needs, they establish increased understanding among stakeholders, and raises accuracy of specification of the requirements.

Code Anaylsis/ Generation/ Summarization area is concerned with analysis of code, and code comprehension. Researchers utilize NLP to generate code snippets, summarize the code blocks, improve full and partial code search and indexing, and code smells. These tools both ease the development and software maintenance.

Bug detection and resolution methods include methods to analyze bug reports and extract key information within these reports, finds duplicated reports, and even suggests potential fixes for certain type of bugs. It provides developers with the smart debugging options. There are tools to mine software repositories and issue trackers to achieve this task.

Software maintenance and evolution tools which uses NLP techniques involves automatically extracted change logs, release notes, statistical analysis of different versions. These tools assists comprehension of software, eases maintenance tasks, and reduces human labour required for maintenance tasks.

Lastly, NLP4SE researchers explored the natural language interfacing to establish common understanding of the domain between the stakeholders of the software project. These efforts includes, generation of executable code and tests from natural language, collaboration tools for developers, and model generation from textual natural language documents for both requirements engineering and for software design. Natural language interface methods provides value by simplifying the development process and by establishing improved team communication.

Among these 5 categories we are particularly interested in NLP4RE domain and language interfacing, more specifically model generation domain due to it's significant

relevance to our method. These topics will be discussed in detail in the following sections 2.4, 2.5.

2.4. Natural Language Processing for Requirements Engineering

NLP4RE is the systematic process of applying NLP methods on RE related NL text to ease the RE process. It has many different applications such as finding duplicate requirements, aiding the human operator for requirement elicitation, and defect detection. The ultimate goal of NLP4RE is to decrease human labour required for RE tasks and improve RE process.

Since NL is used in requirement engineering so thoroughly and nature of the RE is labour intensive at its current state, various effort has been made to apply natural language processing techniques on requirement engineering related natural language text documents. Earliest research studies are reported as far as early 80's. [59, 60]Using NLP for requirements engineering tasks has been the main method for researchers and industry to ease the process [61]. Latest discoveries in the NLP domain, and increasing availability of data and resources accelerated the research on NLP4RE domain. Additionally founding of a dedicated conference called NLP4RE allowed researchers to collaborate and report latest advancements more coherently.

2.5. Model Generation

Model generation from NL has been an interesting challenge for NLP researchers. The main objective of model generation is to identify concepts and their relations to generate specific diagram to represent varying aspects of the given domain. We identified related studies on model generation in our literature review. These studies aim to generate various models represented in UML such as class diagrams [62–68], sequence diagrams [69, 70], use case diagrams [71–75] and various other model diagrams. These studies used varying tools and techniques on different input formats to achieve their objectives. For our study, our review on model generation topic has been narrowed

down to domain model extraction. We analyzed many different modeling tools and techniques so that we can observe potential techniques for our objectives. Yet, our main objective is to extract the domain model using NLP. Therefore our further investigation channeled into more related papers on the topic. We identified 6 related studies on domain model extraction using NLP tools and techniques. The most related study was performed by Arora [1] which we used as a base line for our tool. In their study, they implemented a pipeline using Stanford CoreNLP and GATE NLP toolkits to extract domain models from unrestricted text documents using parsing pipeline and heuristic rules. They used private industrial data sets and performed an emprical study to evaluate performance of their tools. DoMoBOT [76] is another domain model extraction tool proposed by Saini. They implemented a pipeline using spaCy, GloVe and Scikit which extracts domain model elements using rule based approach, supervised ML models and neural networks. In an earlier study, Saini also proposed a neural network based method [77]. This neural network based method exploited the BiLTSM neural network model implemented with Tensorflow and Keras libraries, and additionally utilized GloVe word embeddings to deal with the short comings of rule based methods. Aydemir and Dalpiaz proposed a method [78] to assist domain modeling process by suggesting missing concepts based on word-embeddings and noun phrase similarities between multiple models. Their pipeline used spaCy, WordNet word embeddings, and Google BERT for semantic similarity. A recent study by Mengyuan proposed a rule based approach [79] which uses same pipeline with Arora's study. Lastly, we found the tool proposed by Deeptimahanti [80] which extracts domain models along with additional models to provide holistic view on a domain. But since this tool is developed in 2009, we exclude this tool and its techniques from our consideration.

3. METHOD

In this chapter, we will explain our proposed method in detail. Firstly, we will briefly explain our method and give an overview of the approach. Then, in Section 3.1 we will describe our concept extraction pipeline in detail. After concept extraction pipeline is explained, we specify the details of post-processing noun chunks extracted in concept extraction pipeline to generate the ranked N-grams in Section 3.2. In Section 3.3, we explain the rule based concept and relation extraction pipeline, heuristics used for extraction task, and our design decisions for the rule-based extraction pipeline. Finally, we explain the implementation of the complete pipeline and describe our system building blocks in Section 3.4.

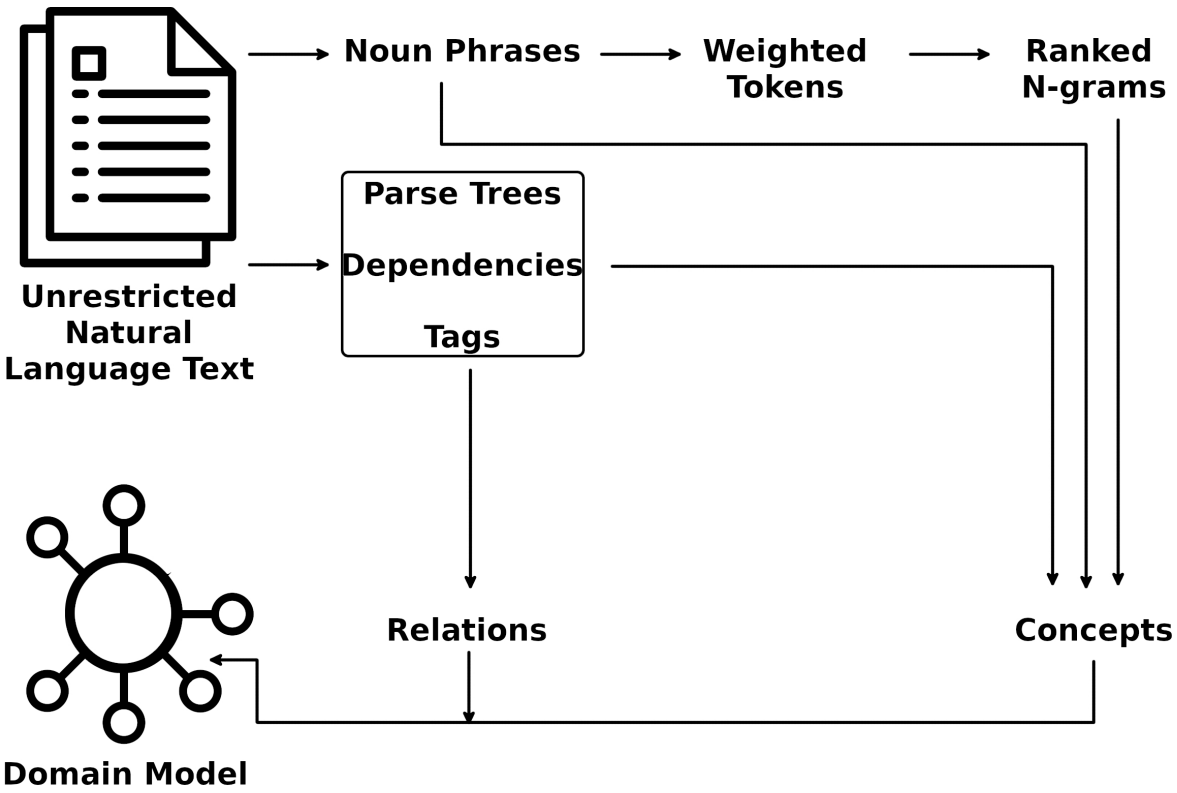

Figure 3.1. Overview of our pipeline.

We approach the domain model extraction as a two-staged problem. Figure 3.1 shows the overview of our method. Firstly, we extract potential key concepts as noun

phrases from the unrestricted text input and extend this initial concept list using Text Ranked [19] Ngrams with our pipeline shown in Figure 3.2. This allows us to obtain the final list of key concepts to be used in the domain model. Secondly, we extract remaining concepts and relations between the finalized key concepts with the pipeline shown in Figure 3.5 by using heuristic rules which utilizes syntactical features and English language models. The input for our tool is an unrestricted text document and after successfully executing the algorithm, the output will be a domain model represented using an UML Class Diagram. The following sections will explain each pipeline and transitional steps in detail.

3.1. Key Concept Extraction Pipeline

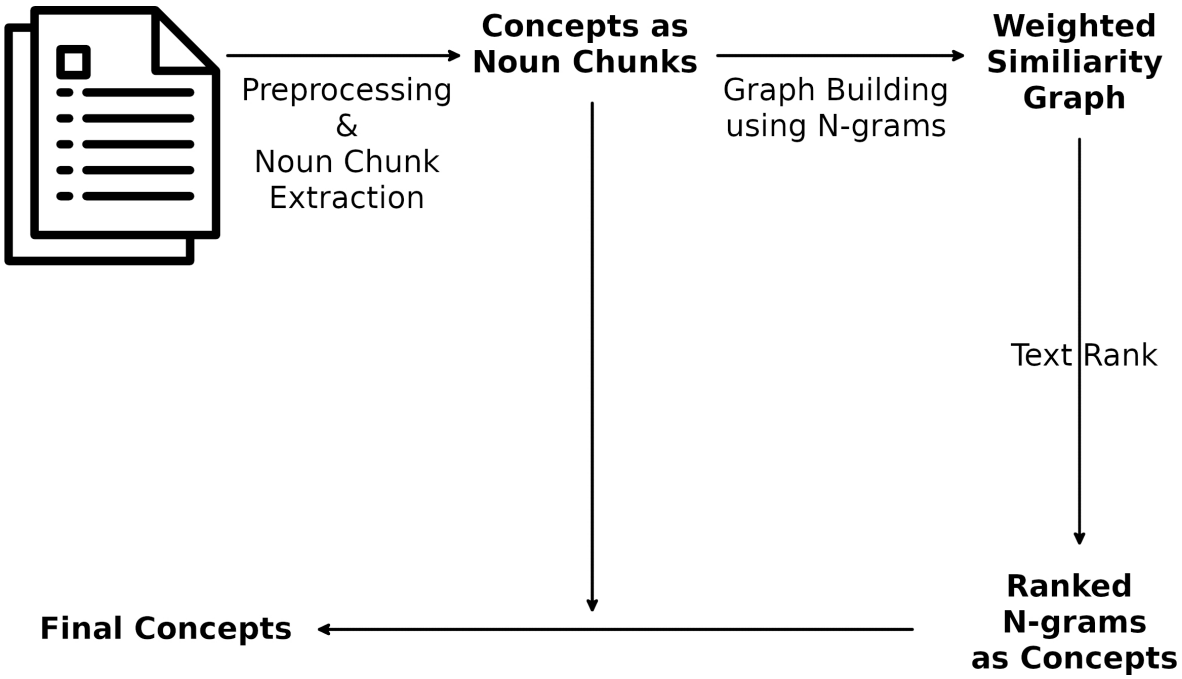

Figure 3.2. Overview of the initial concept extraction pipeline.

When developing our method we specified some key requirements for our tool to be a viable start point for automatic domain model extraction in practical use. These requirements are as follows; Firstly, we need a pipeline which does not require an external corpus or an informed heuristic. Additionally, our method should be independent from the structure of the input document such as requirements structure or length of

the document. Lastly, solution to be developed should be applicable to various application domains. These requirements are established so that, our end product can support requirement engineering activities with minimum effort across multiple projects. To satisfy these requirements, we developed the pipeline shown in Figure 3.2 to address these needs for extracting key concepts within given domain.

First and foremost, we extract noun phrases from the input, which account for almost all of the key concepts within a document according to Justeson [81]. To perform this task, we use spaCy's [82] noun_chunks annotations. Noun chunks annotations are spans corresponding the phrases lead by a noun. Before continuing further with extracted noun chunks to obtain domain concepts, we pre-process the noun chunks so that we can remove syntactic and semantic redundancy. Next subsection will describe the considerations and design decision related to pre-processing the extracted noun chunks. In this pipeline; tokenization, POS tagging, dependency parsing is performed by running spacY's transformer based roBERTa [8] pipeline. This initial noun chunk extraction pipeline provides us the baseline for our method by producing a list of concepts as noun chunks.

3.1.1. Pre-processing Noun Chunks

We investigated alternative pre-processing methods to reduce redundancy of extracted concepts without over filtering or losing meaningful semantic information about the domain since the initial list of extracted noun chunks is the most fundamental resource for obtaining domain concepts. Without such intermediate step, domain model will include syntactic variations of same domain concept along with other noisy elements. As a consequence, this will increase the human labour required to model the domain. There are number of methods we can apply to increase or maintain the concept coverage while decreasing the noisy, redundant concept candidates. These are as follows:

- Punctuation Removal This step removes punctuation characters defined for En-

glish language in default string class of python from the text and replaces them with blank space character. We excluded " \", " /" and " - " characters from this list since, these characters are not used without deliberate purposes such as indicating options, as a part of a model label, and others in a text. We also replaced " ‘ " character with " ’ " character to normalize the usage of apostrophes in the text. Applying this step in our initial extraction module, made no significant difference in recall or precision values with the data sets we evaluated, still, we choose to include it to make our tool more robust for various input texts.

- Non-alpha or Non-alpha numeric token removal This step removes each token which includes non-alpha characters or non-alpha numeric characters. Since we already removed punctuation characters in the previous step, removing non-alpha numeric characters will not have any effects. On the other hand, digits and numbers may provide cardinality information for our domain concepts, yet they are not part of a domain concept itself. Therefore we choose to remove tokens which includes digits as a pre-processing step.
- Text reconstruction One of our main motivations was to develop a tool to work with unrestricted text documents. Therefore, our tool should not rely on any format specific actions. For example, our tool should work with all of the following example formats,

  (i) As a <Role >, I want to <Feature >so that <Benefit >.
  (ii) User shall export data.
  (iii) User is the customer for our Bank.
  (iv) User Dashboard.

  Therefore to make our tool work with available data sets, we performed text reconstruction for user story format to reconstruct test into format (i) seen above for CrowdRE [83] data set.
- Spell Checking This step tries to correct incorrectly spelled, typed words. In the CrowdRE [83] data set, there are many cases which includes such words, yet for our tool, instead of trying to filter such cases, we choose to assume given input set is checked for such cases. Reason for this decision is trying to alter as less input as possible to avoid over filtering domain concepts.

- Co-reference resolution Our preliminary trials with co-reference resolution indicated that this step introduces significant noise to our model yet it makes no significant difference for obtaining domain concepts. This step might be beneficial for extracting dependencies but for solely domain concept extraction, we found it ineffective.
- Stemming Stemming diminishes the words to their root form. But stemming can cause words to lose their semantic meanings. For a domain model, losing or not detecting concepts is more harmful than not being able to filtering redundant concepts. This lead us to opt-out of using stemming.
- Lemmatization Lemmatization is very similiar to stemming, but, it tries to actually transform words into their root form and not just stem them out. Lemmatization increases the accuracy of our model without sacrificing its recall values.

We investigated the effect of applying each step in our pipeline. After careful consideration, we decided to apply punctuation removal, sentence reconstruction, and lemmatization steps to initial list of noun chunks. Each step is evaluated by comparing recall gain versus precision loss. Also some of the steps caused our tool to lose important domain insight after the step is applied, therefore we decided to exclude such steps. Algorithm 1 provides the pseudo-code for the steps up until this point including the extraction of the noun chunks.

3.2. Post-Processing Noun Chunks

After obtaining the initial list of extracted noun chunks, we have our baseline concepts and candidates. By applying post-processing steps, we aim to increase our recall which represents our ability to recognize concepts in the given domain. To achieve this task, we propose a method which utilizes n-grams and a variation of TextRank [19]. N-gram can be defined as continuous sequence of n tokens in a given text. And TextRank is a well known graph based text ranking algorithm. There are many studies which utilizes TextRank for noun chunk, keyword, or concept extraction. At one particular, Zhang's [84] study uses adapted TextRank for relevant terminology

Algorithm 1

Let $S \xrightarrow{denotes} StopWords\{\} \neq \emptyset$

Let $P \xrightarrow{denotes} Punctuation\{\} \neq \emptyset$

Let $R \xrightarrow{denotes} Requirements\{\} \neq \emptyset$

Let $N \xrightarrow{denotes} NounChunks\{\} = \emptyset$

Require: " ' " $\notin P$

for all $r \in R$ **do**

$r =$ As a $< r_{role} >$, I want to $< r_{feature} >$ so that $< r_{benefit} >$

Run pipeline for r

for all $noun\_chunk \in r_{noun\_chunks\{\}}$ **do**

Let $pN \xrightarrow{denotes} PreProcessedNounChunk = \emptyset$

for all $token \in noun\_chunk$ **do**

if $token.isAlpha()$ & $token \notin S$ **then**

$pN = pN \cup token$

end if

end for

$N = N \cup pN$

end for

end for

Figure 3.3. Noun chunk extraction algorithm.

extraction which inspired our method significantly. We assume that extracted multi-word noun chunks is a container of smaller concepts. Based on this hypothesis we propose a post processing method. We suggest that generating n-grams , and ranking this n-grams with adapted TextRank [84] will provide us the candidate concepts. By filtering this list, we can increase our recall with minimum precision loss.

Finally, we implement our post processing method. Firstly, we build an undirected weighted graph for given domain. In this graph, unique words in extracted noun chunks represent vertexes, and similarity scores of word2vec [85] vectors inherited from "en_core_web_lg" spacy model [82] between each unique word above a certain threshold represents the edges between these vertexes. Average weight of all edges belonging a vertex provides the average weight of the vertex, which is a relevance score for given domain. By building this graph, we obtained a relevancy mapping for domain terms.

Finally, we generate all possible n-grams for extracted multi word noun chunks. This step generates a noisy list of candidates, and we proceed with filtering this list of candidates. To filter out the noise, we average each term's score for all generated n-grams to obtain a score for given noun chunk. After ordering this n-grams by average weight of tokens within them, we filter a selected top percentage of n-grams to capture related fractional concepts within already captured multi word noun chunks. Given pseudo code in Algorithm 2 demonstrates the steps of the proposed method.

Algorithm 2

Let $N \xrightarrow{denotes} NounChunks\{\}$

Let $G \xrightarrow{denotes} N-Grams\{\} = \emptyset$

Let $W \xrightarrow{denotes} Undirected\ weighted\ graph$

Let $V \xrightarrow{denotes} Vertexes \subset W$

Let $E \xrightarrow{denotes} Edges \subset W$

Let $S \xrightarrow[denotes]{} SortedN-grams = \emptyset$

Let $\alpha_{threshold} \xrightarrow[denotes]{} Relevance\ threshold\ for\ domain$

for all $n \in N$ **do**

  if $n.word\_count >= 2$ **then**

    $G = G \cup n_{ngrams}$

  end if

end for

for all $g \in G$ **do**

  $g.total\_weight = 0$

  for all $word \in g$ **do**

    $V_{word}.total\_weight = \sum_{i=0}^{V_{word}.edges\_count} E_{word-i}$

    $V_{word}.avg\_weight = V_{word}.total\_weight / V_{word}.edges\_count$

    $g.total\_weight+ = V_{word}.avg\_weight$

  end for

  $g.avg\_weight = \dfrac{g.total\_weight}{g.word\_count}$

  if $g.avg\_weight > \alpha_{threshold}$ **then**

    $S = S \cup \{g, g.avg\_weight\}$

  end if

end for

$S.sort\_by\_weight$

$N = N \cup S.filter\_top\_candidates$

Figure 3.4. Post processing algorithm for extracted noun chunks.

3.3. Concept and Relation Extraction via Heuristics

Our previously explained pipeline generated the linguistic nature of the sentences, corresponding syntactical parse trees, and syntactical dependencies between input tokens along with the concepts as noun chunks within the data set. In the second part of our method, we process this extra syntactic information to extract additional concepts and relations between concepts using heuristic rules. First step to achieve this task is building a master heuristic list [86] and deciding the heuristic rules to be used in our method. Master heuristic list is composed of heuristic rules collected from literature and heuristics suggested or modified by us, details of how this list is build is discussed in related work section of the thesis. Secondly, we choose promising heuristics which are suitable to be used in our pipeline to extract concepts and relations from unrestricted natural language text input. In the following subsection we will discuss these heuristics, and our rationale to include or exclude the heuristic. All of the example parse trees are displayed using displaCy [87].Overview of the process can be seen in the Figure 3.5

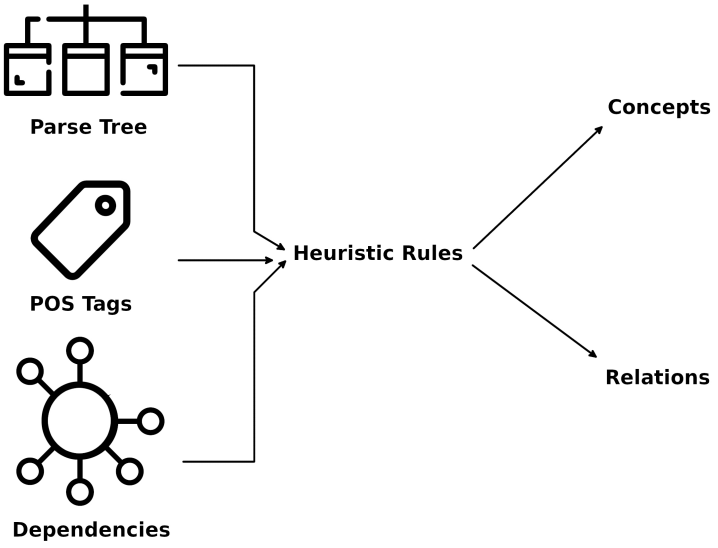

Figure 3.5. Overview of heuristic based extraction pipeline.

3.3.1. Selecting Heuristic Rules for Our Pipeline

The complete list of all identified heuristics in related studies are presented in [86]. In this section we will explain included and excluded heuristics in detail.

3.3.1.1. Heuristics Included in Our Pipeline. This subsection will describe each heuristic, express the motivation for including heuristic in our method, and lastly will demonstrate the heuristic using examples.Following heuristics are all included in our pipeline;

- C6- Subjects in the requirements are concepts. Dependency parsing can be used to extract subjects of sentences. The dependency label 'nsubj' suggests a concept. Subject chunks can reveal most of the key concepts within a domain.
  e.g., My smart home to lock the doors.
  In this example We map "smart home" as a concept since it has "nsubj" dependency.
- C8- If a sentence in (S-V-O) subject (nsubj) and (dobj) are candidate classes. If the verb of a sentence has children tokens with "subj" and "dobj" dependency tags, subject chunks and direct object chunks suggests concepts. We don't differentiate between candidates and actual concepts, instead we map all as concepts in our pipeline.
  e.g., The radio system adjust volume.
  In this example concepts "radio system" and "volume" will be extracted.
- C11- "Is a" relationship suggests candidate classes. This heuristic is implemented implicitly when extracting relations via HR1. <A> is a <B> suggests A and B as concepts. If A and B are not already extracted as a concept until this point, we will map them as concepts.
  e.g., "Premium account is an account."
  Concepts "Premium account" and "Account" will be extracted in this case.
- C13- Noun compounds are taken together to form a concept Noun compounds are combined phrases which has noun word as their head. This heuristic is implemented implicitly by design in our pipeline. We principally extract chunks

(compounds) for all heuristic unless the heuristic explicitly states otherwise.
e.g., Premium account is an account.
In this example we extract "Premium account" and "account" concepts.

- C14- Gerunds in the requirements are concepts Gerunds are forms of verbs those ends with '-ing' and used as nouns. We assume only gerunds which are part of noun chunks suggests concepts. Examples below show extracted concepts with bold italic format.
  e.g., Arriving passenger books transportation,
  Checking is processed by the system.

Relations are actions which connect concepts within the domain. We use parse trees, POS tags, and heuristic rules to extract non-hierarchical relationships between concepts. Following heuristics are used to extract non-hierarchical relationships in our pipeline;

- NHR2- Transivite verbs are associations.
  Transitive verbs are actions which contains a direct object. We extract transitive verbs by finding verb with "nsubj" and "dobj" dependencies.
  e.g., Smartphone app controls temperature.
  In this example Smartphone app and Temperature will be associated with the Control action.
- NHR7- A verb with a preposition is an association. Prepositions indicates spatial and temporal dependencies. This heuristic provides accurate insight for relations according to our observations.
  e.g., Video feed is sent to my smartphone app.
  In this example Video Feed and Smartphone App will be associated with Sent relation and additional to preposition will express Sent (to) association between these concepts.
- NHR10- In noun-noun compound there is a non-hierarchical relationship between prefix and compound. Initially, this heuristic is aimed at specifically noun-noun compounds, which means noun chunks with two noun words. But we tried to

generalize this rule into noun chunk-noun chunk format to be used for any length of noun compounds. This heuristic generated some noisy relations with multi-word concepts, but it yielded promising semantic insight. Therefore, we choose to include the heuristic in our pipeline.

e.g., Customer Account concept will generate "has" relationship between "Customer" and "Customer Account"

- NHR11- <R>of <A>is <B>is likely to be an association. Concepts connected with preposition "of" and supplemented with forms of auxiliary "to" indicates associations. We include this heuristic even though it covers a very specific case therefore it is unlikely to generate a noisy relation set for our pipeline. Relation <R>will be the association between <A>and <B>

  e.g., Security cam of smart home is video feed.

  Concept "Smart home" will be associated with "Video Feed" concept with the "Security Cam" relation.

- NHR13- Relative clause modifiers of nouns (rcmod) suggest associations. Relative Clause defines a dependent clause to express information about a noun. Even though this heuristic generates noisy relations and concepts, we include this heuristic since it provides beneficial semantic links within the domain. This heuristic can be modified to only apply if concepts are already extracted, which is a version that we did not use in our pipeline.

  e.g., "... an app to control the blinds on my windows."

  Relative clause dependency will reveal association between App and Blind with Control action.

- NHR14- Verbal clausal complements (ccomp/xcomp dependencies) suggest associations. "ccomp" and "xcomp" dependencies is used to express dependent clause of a verb or adjective in both Standford CoreNLP dependencies [88] which is the tool of choice for the origin paper for the heuristic [1], and also per spaCy [82]. If subject is controlled meaning that no other interpretation is possible it is a "xcomp" dependency otherwise it is a "ccomp" dependency.

  e.g., "... my smart home to have a certain television show start on my TV ..."

  Concept Smart Home will be associated with Television show with Start relation.

In the above heuristic list, we focused on extracting associations which are basic relations between concepts. Now we focus hierarchical relations such as inheritance, composition, aggregation type of relationships. This relationships both provide technical insight for project team members such as designers, developers, and testers, and also establishes a median to communicate domain structure among most of the stakeholders.

Following heuristic rules are used to extract hierarchical relationships in our pipeline;

- HR1- Verb "to be" and its tenses suggest generalizations where subject is taken as a specialization of the parent object. [HR1 to HR5] all describes the relation between general and specific concepts. We include extraction of this kind of relationship by combining them into one method under the heuristic HR1. In conclusion, we modify HR1 to capture concepts when target structures of these heuristics is encountered in given text. "to be", "is a", "type of", "kind of", "may be", "category of", "can be", "is a .. type/kind/etc .." are relations captured using this heuristic.

  e.g., Paid member is a type of member.

  "Member" concept will be the general concept for specific "Paid Member" concept.
- HR8- Head of noun-noun compound suggests is-a relationship between compound and head. This heuristic seems similar to heuristic NHR10 at first glance, but it extracts hierarchical generalization relation between compound and head unlike NHR10.

  e.g., Customer Account concept will generate "is" relationship between "Customer" and "Customer Account" per NHR10

  e.g., Customer Account concept will generate "is a" relationship between "Customer Account" and "Customer" per HR8.
- HR9- "contain", "is made up of", "include" , [".."] suggest aggregations/ compositions. Heuristics [HR9 to HR14)] besides HR13 all defines aggregation/ composition relationship between two concept. We extend HR9 to unify this cases under one method.

  e.g., Smart home contains speaker TV.

Concept “smart home” and “speaker TV” will be related with an containment relation.

Refinement rules are used to clear noisy output from the domain model. And selected heuristics are explained below. Following heuristic rules are used to refine extracted concepts and relations;

- RF7- Remove redundant candidates from the list. We implicitly use this heuristic by using set() objects as containers in our pipeline.
- RF9- If any two candidate classes are related by “known as”, “same as”, “similar to”, they are same class. “IS” association will be drawn between source and target.

3.3.1.2. Heuristics Excluded from Our Pipeline. Following heuristics are excluded from our pipeline. We only try to explain our reasoning for this exclusion in below. Excluded heuristic rules to reconstruct or normalize the text are described below.

We aim to extract domain models from unrestricted text input, therefore we refused to include any text reconstruction or normalization steps which can potentially alter the syntactic or semantic elements of the text. Only exception for this rule is reconstruction of input as intended by data set gatherers to generate valid input for our tool. An example for this exception is the CrowdRE data set [83]. In the public data set files, each requirement is represented as a row in a .csv file which composed of various columns. A requirement has id, role, feature, and benefit columns which corresponds the elements of a user story format, therefore we transform this input to full sentences instead of seperated columns by reconstructing each record to comply with user story format without modifying any element. This step is only implemented and discussed in the first section, and no heuristic is used for this task.
Excluded heuristic rules to extract concepts are listed below:

- C1- All noun phrases in the requirements are candidate concepts: This heuristic suggests holding noun phrases as candidates at first yet not final concepts.

- C2- If a word is a noun, then it is a potential concepts: Noun words are candidate concepts instead of final concept.

  These two heuristics is responsible for capturing all the candidates in the input data set, and have an intersecting subset as well. Including the said heuristics bears a trade-off between recall and precision compared to accepting noun chunks or noun words as a final concepts. There are other heuristics which contradicts these approaches, and they are the corespondents of this trade-off, such as C4, and C5. After explaining C4 and C5 our motivation for this exclusion will be explained.
- C4- Common nouns (NN) and Proper nouns(NNP) are classes. In C4 we use POS tags NN, and NNP to extract final concepts, not candidates. The term 'class' in this heuristic, translates to term 'concept' for our method.
- C5- A common noun is concept. Like C4 In C5, it is suggested that proper noun (NNP) POS tag suggests a final concept. Compared to C1 and C2, C4 and C5 suggests assuming extracted concepts as final concepts instead of candidate concepts. C1 and C2 uses heuristic C3 to perform transition from candidate concepts to final concepts.
- C3- Recurring NPs are concepts. Candidate concepts generated using C1 and C2 are checked and recurring ones are added to the list of final concepts.

  After careful consideration, we concluded that including heuristic rules into our pipeline to extract concepts, whether as in C1-C2-C3 fashion or as in C4-C5 fashion, is not suitable since they introduce high amount of noise while having insignificant increase in recall values. Transformer based initial noun chunk extraction pipeline, and ranked n-grams generated from noun chunks can capture most of the concepts without high amounts of incorrect concepts. Therefore, we opt out from usage of both using noun/noun-chunk based extraction heuristics and also heuristics based on the separation of candidate concepts and final concepts.
- C7- Subject of a sentence is concept C7 is the duplicate of C6 from different study.
- C9- Objects in the requirements are concepts. This heuristic's target cases are

already covered by C8. Furthermore, accepting all objects generates more noise than actual concepts.

- C10- If the noun is post-fixed by preposition <IN>, then ignore it as a class and map it as a part of a method. We do not extract methods for our domain model yet we can transform this heuristic into simpler form of "If the noun is post-fixed by preposition <IN>, then ignore it as a concept." This heuristics improves the accuracy of our model significantly but it also causes us to lose noticeable recall.
- C12- For a noun phrase (Noun+Noun) if the first noun is already a candidate class, then the second noun is mapped into candidate instances (objects) of that class. This heuristic is class diagram specific, and also it depends on candidate concept versus final concept separation. Therefore we can not directly use it.
- C15- Gerunds in the requirements are concepts. C15 is replica of C14, which was described previously, from another source.

Following heuristic rules to extract non-hierarchical relationships are excluded from our pipeline;

- NHR1- Every verb is a potential relationship. This heuristic is not implemented since its motivation is not explicitly stated in its paper, nor in its source code repository. We assume, this heuristic is a baseline for other heuristics related to extraction using verbs.
- NHR3- A transitive verb indicates a relationship.
- NHR4- Transivitive verbs are association relationships between two candidate classes.

  Heuristics NHR3 and NHR4 are identical to NHR2.
- NHR5- The verb phrase linking the sentence subject and an object forms the relationship between these two. This heuristic is already implemented in C8, NHR2, NHR3, NHR4, and NHR7.
- NHR6- (Noun+Verb+Noun) and (Subject-Predicate-Object) suggest association between candidate classes. This heuristic is a rephrased version of NHR5.

- NHR8- Verb followed by preposition is is a relationship. This heuristic is a duplicate of NHR7.
- NHR9- <VB>+ <IN>or <To>suggests association relationship. This heuristic is a duplicate of NHR7.
- NHR12- Verb phrases with two verbs (VB+VB) suggests association between associated noun phrases. Details of this heuristics is missing from the related paper and its repository is not publicly available. Therefore, we exclude this heuristic.
- NHR15- Non-finite verbal modifiers (vmod dependencies) suggest assciations.
  Vmod dependency is used to express reduced non-finite verbal modifiers in Standford CoreNLP dependencies [88] which is the tool of choice for the origin paper of the heuristic [1]. Vmod is the participial or infinitive form of verbs which leads phrases. In spaCy, "acl" tag and its variations describes both finite and non-finite verbal modifiers. And usually non-finite verbal modifiers in English are not termed relative, therefore plain "acl" stands for non-finite verbal modifiers [89]. After futher investigation of implications of this heuristic we excluded it from the pipeline since it's implementation does not transform precisely from Stanford Core NLP [88] to spaCy [82].

Following heuristic rules to extract hierarchical relationships are excluded from our pipeline;

- HR2- "is a", "type of", "kind of", "may be", ["..."] suggest generalizations.
- HR3- "is category of", "is type of", "is kind of", "may be", "is a" between subject and object suggest generalization.
- HR4- "may be" and "can be" suggest inheritence from object to subject.
- HR5- "is a type of" suggests inheritence from subject to object.
  The four heuristics listed above all describes the relation between general and specific concepts. We include extraction of this kind of relationship by combining them into one method under the heuristic HR1.

In conclusion, we modify HR1 to capture concepts when target structures of these heuristics is encountered in given text.

- HR6- N1 is a N2 and a N3 denotes multiple inheritence.

  We tried to only include general heuristics to capture most significant concepts and relations, specific case rules such as HR6 is not included in our pipeline. The rationale behind this decision is that we believe trying to capture all concepts and relations generates too many incorrect outputs and as a consequence human labour required to organize domain model surpass the convenience gained by automatic extraction. Specifically designed heuristics such as this one can be implemented as an improvement to our pipeline. In general ability to extract compound statements correctly using all of the heuristic discussed would benefit stakeholders greatly. Yet, we were not able to implement a general rule for such cases under unrestricted text constraints.
- HR7- Subject part is parent and object is child for inheritance relations. Inheritance relation is already extracted via corresponding heuristics explained earlier and HR7 is not used for the task.
- HR10- “have”, “hold”, “possess”, “carry”, “involve”, “imply”, “embrace”, “contains”, “consists of”, “comprises of”, “is part of”, “included in”, “belong to”, “divided to”, “has part”, or “is made up of” between subject and object suggest composition or aggregation.
- HR11- “comprises, have, include, possess, contains” suggest composition relationship is from object to subject.
- HR12- “is part of” suggests aggregation from subject to object.
- HR13- Subject part is parent and object part is subclass for the aggregation.
- HR14- Collective nouns suggest aggregation.

  All of the heuristics listed above until HR6, defines aggregation/ composition relationship between two concept. We extend HR9 to unify all cases which we think is necessary under one method. This aggregation/ composition extraction method is implemented in our pipeline under HR9.
- HR15- “require”, “depends on”, “rely on”, “based on”, “uses”, “follows” suggest dependency between subject and object.

We exclude HR15 to extract dependency between two concepts, since it is a hierarchical relation which can not be labeled as inheritance relation nor a aggregation/composition relation.
e.g., Media player depends on sound driver.
"Media player" and "Sound driver" has a dependency relationship.

Excluded heuristic rules to extract attributes of concepts are described below;

We do not extract any attributes in our pipeline. We assume key concept extraction and relations of this concept is key elements of the domain model, and specific implementation details such as individual attributes does not concern all of the stakeholders. Therefore we limit our extraction scope to key domain concepts, significant relations between them, and lastly the cardinalities of the relations. Excluded heuristic rules to extract concept cardinalities are described below;

These rules are used to extract valuable multiplicity information to represent relations precisely. This information is specifically useful for establishing common understanding between designers and project's target stakeholders.
We try to extract cardinalities of individual concepts for each relation in our pipeline instead of considering relation as a whole. For each concept in each relation; we extract concept's individual multiplicity by following heuristics listed below. To perform this task, we unify heuristics which extract singularity, plurality, and lastly exact multiplicities into a combined heuristic of their own. We assume, if we perform this operations on every concept in all relations, It will define cardinality information for each relation as well, therefore we do not extract cardinalities based on relations as a whole. Heuristics which extract relations from coupled concepts are modified to be applied on individual concepts in relations accordingly.For example in the heuristic CAR2 both source concept and target concept is considered to extract many-to-one relation. However in our approach, we check source concept's cardinality in isolation then perform same operation on target concept, together they will define the cardinality of the relation between source and target concept.

Due to time constraints, we failed to implement cardinality extraction pipeline and discarded the above proposed extraction approach and below listed heuristics from our pipeline.

- CAR1- Singular noun + definite article suggets exactly 1.
- CAR2- Singular source concept and plural target concept with definite article suggest many-to-one.
- CAR3- Singular source concept and singular target concept suggest one-to-one.
- CAR4- An explicit number before a concept suggests a cardinality.
- CAR5- Number from <CD>tag suggests multiplicity.
- CAR6- “More than X” suggests X cardinality.
- CAR7- “many”, “each”, “all”, “every”, “some”, “any” suggest many cardinality.
- CAR8- Plural nouns suggest many <NNS>and <NNPS>.
- CAR9- Plural source concept with universal quantifier and plural target concept suggest many-to-one.
- CAR10- Indefinite article suggests exactly one cardinality.
- CAR11- Indefinite article suggests exactly one cardinality. <DT>.
- CAR12- Noun or prepositional phares with singularity suggest one-to-one cardinality.

Following rules are suggested to clear noisy output from the domain model. Yet for the stated reasons we exclude the usage of them from our pipeline.
Excluded heuristic rules to refine extracted concepts and relations are as follows;

- RF1- Candidate classes those occur only one time and has frequency of <2% are ignored.
- RF2- Candidate classes with no attributes are ignored.
- RF3- Candidate classes with no relations are ignored.
- RF4- Candidate classes related to design, location name, people name are ignored.
- RF5- Candidate classes with relationships, attributes are classes.

All of the heuristics above utilizes the design which separates concepts as candidate and final. We opt out of using separation design for our method, therefore, all of the heuristics above are ignored.

- RF6- If a concept is an attribute ignore it as class. We decided not to extract attributes for domain modeling, as a consequence including RF6 would not serve its purpose in our pipeline.
- RF8- If two classes reveal the same information, remove a class. The details and implementation of this heuristic is missing from the source. For this reason, we could not implement and include this heuristic.
- RF10- Classes found in adjective and attribute class list are removed.
  Classes and attributes are not used in our pipeline.
- RF11- If a class takes a list of values, such classes are attributes of a class.
  For the same reasons with RF10, and additionally, lack of examples by the the author of origin paper for the heuristic we could not use RF11 in our pipeline.
- RF12- Remove classes without purpose manually. We do not include any manual steps to our pipeline. RF12 is also excluded for this reason.

3.4. Implementation

Our method is implemented using Python programming language. "spaCy" natural language processing library is used for various tasks within our pipeline. SpaCy is an advanced library for full range of NLP tasks such as tokenization, POS tagging, NER, and etc using state-of-art algortihms and technologies. It is bundled with tokenizer module for many languages and pretrained models with diverse methods. In our pipeline we choose to use "en_core_web_trf" labeled RoBerta [8] based transformer pipeline. Additionally, we use word2vec [85] vectors from the "en_core_web_lg" pipeline to compute a weighted graph which is used to rank generated Ngrams. These Ngrams are additonal sources of concepts in our pipeline as well. Following Figure 3.6 will demonstrate the components of the proposed method's implementation.

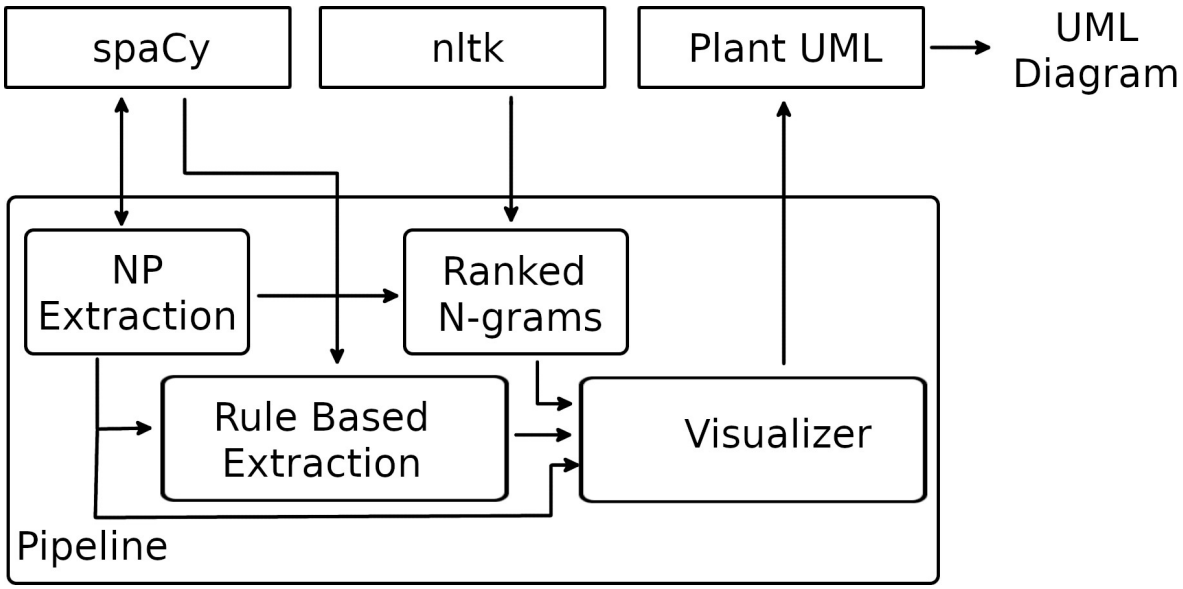

Figure 3.6. Components and structure of the system.

The pipeline shown above in Figure 3.6 is implemented in a linear fashion. It will take unrestricted natural English language text input from given "csv" file and treat each row as a sentence to be processed. Each sentence is kept as a string in a list and later this list is turned into "Doc" file type defined in spaCy library. Then pre-trained transformer based pipeline will be created and word2vec vectors will be included to the transformer based pipeline from the tok2vec based pre-trained CNN pipeline. This extended transformer based pipeline will process the input. Processing is done by various components in the pipeline. Tokenizer component will create the "Doc" object and split the input into tokens. Tagger component will tag the tokens with POS tags. Parser component will extract parse trees for each sentence. Attribute ruler component will tag tokens using rule based approach. Lemmatizer component will extract base forms of tokens. And lastly, entity recognizer component will identify the non-overlapping labelled spans of tokens. Tagger, parser, and entity recognizer components will utilize transformer component while attribute ruler and lemmatizer components employ CNN configuration used in tok2vec based statistical pre-trained models. To elaborate more on the used technologies in the pipeline, we should also note that, the parser component of the transformer based pipeline is a transition based dependency parser which based on a non-monotonic arc-eager transition-system [90] with transition-break system [91]. After applying the described pipeline on the 'Doc'

object, we obtain linguistic, and syntactic information for the input set. Using this information we extract a list of concepts and relations between these concepts. Concepts and relations will be transitioned into domain model by encoding them into required text format for PlantUML [92]. Finally, domain model visualization will be performed using PlantUML and output of the pipeline will be an image file which contains UML Class Diagram to represent domain model for given domain, and the corresponding input text file used to generate this domain model. Flowchart in Figure 3.7 shows the process from input to output in a simpler fashion.

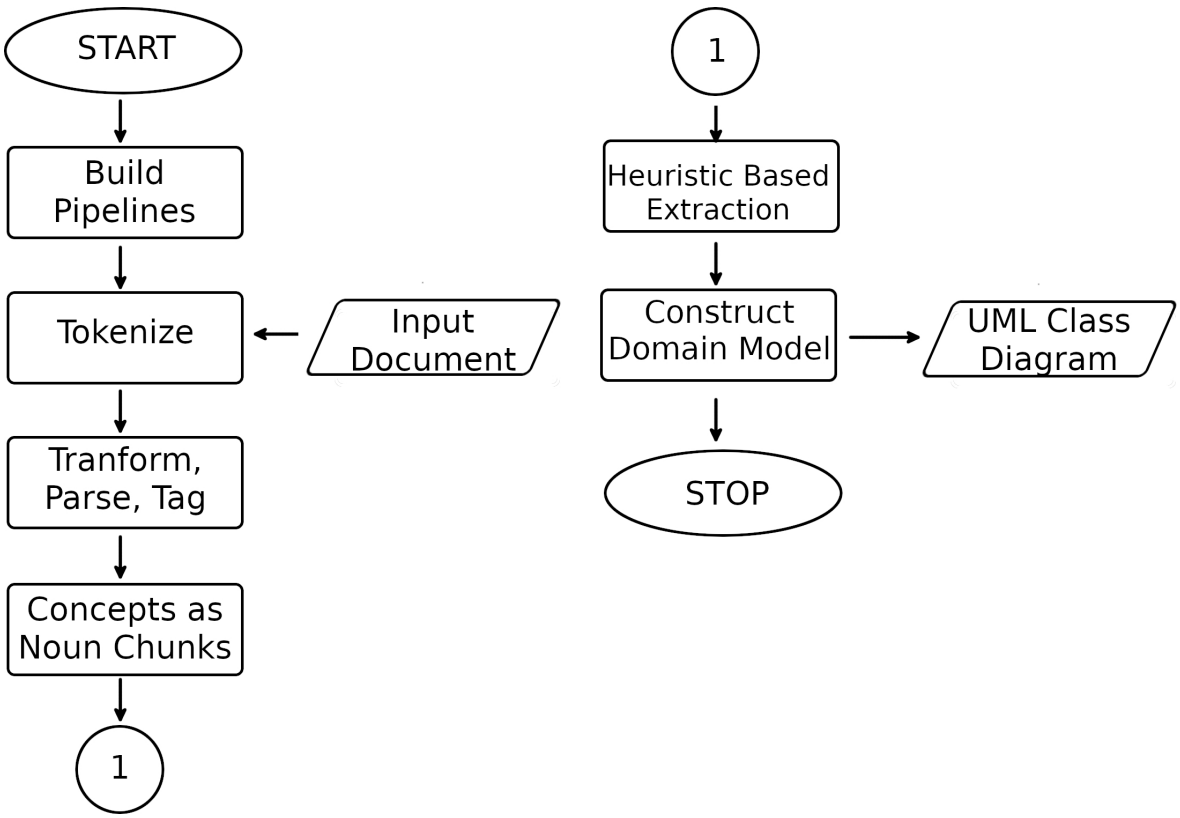

Figure 3.7. Flowchart for the proposed method.

4. EVALUATION

In this chapter we share our evaluation method and techniques. Firstly, we define our research questions which draws an outline for our performance evaluation. In Section 4.1, we describe our data set collection, contents of our data set, and tagging process. Then, in Section 4.2 we provide our performance metrics to evaluate the performance of our method. In Section 4.3 we share our performance results, observations, and findings in great detail. Finally, in Section 4.4 we identify potential threats to validity of our study.

To analyze the performance of our proposed method we carried out a statistical analysis to answer following research questions:

- RQ1 - How effective is our approach? Usefulness of our approach can be measured by calculating recall of our pipeline. Results can give insight about the effectiveness of the proposed method when compared with the notified results of other studies. We measure usefulness of pipeline and additionally individual heuristics by measuring their performance in terms of recall, and precision. Additionally, triggering frequency of each heuristic and ratio of unique to non-unique concepts extraction will reveal the significance of the heuristic for our pipeline. Additional performance measures will reveal how useful of a heuristic for our pipeline. These measures will be described in detail in the following sections.
- RQ2 - Does our approach provides additional benefits compared to existing approaches in practical setting? Each proposed approach has pros and cons over each other. By comparing features of existing approaches we can observe the advantages our proposed approach.

4.1. Preparing the Datasets

Before evaluating our pipeline's performance, we had to establish a ground truth to compare with our extracted domain model and its elements. For this purpose, we performed a preliminary research to obtain public requirement specifications documents and their domain models (in the form design documents, list of concepts or UML diagrams) so that we can obtain an unbiased data set, yet we were unable to find such resources. Most notable data set we found was Amazon Smart Home data set [83], [93] which consist of requirements specified by large group of Amazon staff. This data set is already studied in terms of requirement engineering by researchers [94], [95] and we can access to two different set of domain concepts tagged by the researchers. Statements follow the commonly used user story format. Data set belongs to the smart-home domain and it has 100 labeled requirement statements where we can obtain concepts along with the requirements. However the relations between domain concepts were not included in any of these versions. Amazon Smart Home data set is included in our pipeline evaluation for many reasons; Data set is used in an industrial setting which can demonstrate the applicability of our approach in real world settings. Data set is already labeled by unbiased researchers. And lastly, we have two different reportings on performance metrics for the Amazon data set, which we can compare our performance against.

Additionally, we obtained 8 more data sets from educational setting. These 8 data set obtained from MSc. students of the Boğaziçi University. Student groups provided data sets which include requirement specifications of their term project and its corresponding domain model in the form of UML diagrams. Students are not asked to use any specific format or template, and each group choose the domain they would like to work in. These data sets are labeled with letters from A to I (group F data is discarded).

- Group A: Data set belongs to the Banking Domain and contains 102 requirement statements. Data set mostly contains shall statement requirements.

- Group B: Data set belongs to a Library Application and contains 72 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group C: Data set belongs to the Resource/Product Planning Domain and contains 100 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group D: Data set belongs to the Food Delivery Domain and contains 100 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group E: Data set belongs to the Banking Domain and contains 100 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group G: Data set belongs to the E-commerce Domain and contains 94 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group H: Data set belongs to the E-commerce Domain and contains 100 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Group I: Data set belongs to the E-commerce Domain and contains 95 requirement statements. Data set mostly contains "Actor" shall "Action" formatted requirements.
- Amazon A: Data set belongs to the Smart Home Domain and contains 100 requirement statements. Data set mostly contains user story formatted requirements. Data set is tagged by external researchers [94]. Domain relations are not mined by the researchers, but domain concepts and their source requirements are labeled.
- Amazon B: This data set share exactly same requirements with "Amazon A" data set but it has multiple sources of tagging for domain concepts. Data set is tagged by external researchers [95]. Domain relations are not mined by the researchers, but domain concepts are labeled.

- Amazon C: This data set share exactly same requirements with "Amazon A" data set but it has multiple sources of tagging for domain concepts. Data set is tagged by internal researchers. Domain relations are also mined by the researchers along with the domain concepts and their source requirements.

In total, we have 100 requirements from the industrial setting, and 763 requirements from the educational setting. Amazon Smart Home data set is created with the efforts of 300 staff members and student data sets are created with the efforts of 3 to 4 student groups for each data set. Our data set collection covers 6 different domains and contains sentences with different structures such as shall statements, user story formatted requirements and lastly free-form English statements. No pre-processing or operation has been performed on the data sets by the researchers other than removal of erroneous dot (".") usages. Student models are manually represented in a .csv document for convenience of coding. Afterwards researchers mined the domain concepts and relations before the implementation of the pipeline to stay unbiased during the labeling process and also mapped them into a .csv document to be accessed by the pipeline. All performance metrics are automatically measured at the end of the pipeline for each data set and the results in section4.3 are observed by the researchers. We made data set related files public and it can be accessed online [96].

4.2. Metrics and Tools

In similar studies [1, 84, 94, 95], empirical and statistical evaluation process included recall, precision and F1 score metrics to assess the performances of their tools. We perform our statistical evaluation by measuring these values as well. Additionally, we calculate local average and major average values for these metrics when it is applicable.

- Recall: Represents the percentage meaningful instances extracted among all instances in a set. In our case, this is equivalent to ability to cover existing concepts

in given domain. Recall is calculated using

$$Recall = \frac{TP}{TP + FN} \tag{4.1}$$

or

$$Recall = 1 - \frac{FN}{TP + FN}. \tag{4.2}$$

depending on the case (redundant concept labels).

- Precision: Represents the percentage meaningful instances extracted among all extracted instances from a set. In our case, this metric represents the noise of our domain model. The higher the precision is, the domain model is less noisy. Precision is calculated using

$$Precision = \frac{TP}{TP + FP}. \tag{4.3}$$

- F1 Score: It is the harmonic mean of earlier metrics. In our case, this metric can be used to assess heuristic performance considering the trade off between recall and precision for different heuristics. In addition, we can use F1 score to identify which data sets our pipeline can potentially perform better in future. F1 is calculated using

$$F1 = \frac{2.Precision.Recall}{Precision + Recall} = \frac{2.TP}{2.TP + FP + FN}. \tag{4.4}$$

4.3. Results

In this chapter, data set statistics and the results of the pipeline is explained in detail. We will start the evaluation by explaining the data set metrics and performance evaluation of concept extraction.

Table 4.1 shows the concepts identified in the given data set. “Unique Student Concept” column represents the number of uniquely labeled concepts, which identified by students for the data set using UML diagrams. “Unique Golden Truth Concept” column represents the uniquely labeled concepts mined by the researchers. Lastly, “Individual Golden Truth Concept” column represents individual concepts mined by the researchers. This column does not consider uniqueness of the concepts. Same concept labels can be extracted from different requirement sentences and there will

Table 4.1. Number of identified concepts.

Group ID	Unique Student Concept	Unique Golden Truth Concept	Individual Golden Truth Concept
A	14	75	346
B	23	59	272
C	8	81	359
D	8	82	395
E	16	109	409
G	20	77	386
H	12	101	397
I	23	96	368
Amazon A	0	250	511
Amazon B	0	250	250
Amazon C	0	174	345

be recurring concepts within document.In total we identified 4038 individual domain concepts and 1354 unique domain concepts using the 763 requirement statements from 7 different data set and 9 different ground truth. These ground truth concepts are either tagged by the internal researchers or external researchers and details of this tagging process is described in Section 4.1.

We can observe in Table 4.1 that student domain models have limited scope, and these models hardly contains any domain element. We also observed that students defines implementation elements or database objects rather than actual domain concepts in their domain models. Since evaluation of the pipeline extraction compared to student models will not reveal any meaningful representation of the domain, we excluded student models from the further evaluation process.

Table 4.2 demonstrates the comparison of unique concepts and the golden truth of each data set. The most important metric for a domain modeling pipeline is the major average recall metric since it captures the coverage of domain concepts across multiple domains and varying origins of requirements written by differing experienced practitioners. 74,8% major average recall has been accomplished by our pipeline while being limited to 37,2% precision and 0.49 F1 Score. This recall value is on par with other reported studies [1], [76], [84], [94], [95] on domain modeling, concept extraction and

Table 4.2. Unique concepts extracted versus the golden truth.

Data Set Name	Unique Golden Truth Concepts	Extracted Unique Concepts	TP	FP	FN	Recall	Precision	F1
Amazon A	250	460	214	246	36	0.8560	0.4652	0.6028
Amazon B	250	460	206	365	44	0.8240	0.4478	0.5018
Amazon C	174	460	139	321	35	0.7989	0.3022	0.4385
Group A	75	313	121	192	41	0.7469	0.3866	0.5095
Group B	59	247	93	154	25	0.7881	0.3765	0.5096
Group C	81	441	163	278	51	0.7617	0.3696	0.4977
Group D	82	456	147	309	54	0.7313	0.3224	0.4475
Group E	109	280	109	171	47	0.6987	0.3893	0.5000
Group G	77	340	111	229	45	0.7115	0.3265	0.4476
Group H	101	366	122	244	70	0.6354	0.3333	0.4373
Group I	96	346	126	220	75	0.6269	0.3642	0.4607
					Local Average	0.7436	0.3712	0.4866
					Major Average	0.7478	0.3720	0.4882

key phrase extraction. Precision and F1 scores are lower than other reported studies however, for domain modeling purposes the trade off between recall and precision favor towards to earlier since cost of over looking existing domain concepts is far greater than dealing with noisy domain concepts. With additional steps, the precision performance metric can be improved greatly yet identifying and extracting missed domain concept is very labour intensive for practitioners. Our pipeline can identify every 3 concept out of 4 across multiple domains in free-form text.

Also, it can be observed that recall value goes as far as 85,6% and 82,4% for industrial data sets labeled by external researchers. This metric advances the reported 75,3% [94], 73,2% [95] performance for the same data set, and 82,8% [84] performance for a modern key phrase extraction method.

After examining the unique concept coverage, we proceed with the inspection of individual concept coverage by our pipeline. “Amazon B” data set does not include the source requirement ids along with the concept labels, therefore individual concept extraction performance could not be measured using the data set. In total 34475 individual concepts are extracted by the pipeline, while data set golden truths contained 3788 individual concepts. In Table 4.3, it can be seen that correctly identified individual concept number exceeds the individual concepts labeled by researchers. This is due

Table 4.3. Individual concepts extracted versus the golden truth.

Data Set Name	Golden Truth Concept Count	Extracted Concept Count	TP	FP	FN	Recall	Precision	F1
Amazon A	511	5168	2789	2379	65	0.8728	0.5397	0.6953
Amazon C	345	5168	2070	3098	43	0.8754	0.4005	0.5686
Group A	346	2891	1721	1170	47	0.8642	0.5953	0.7388
Group B	272	1925	972	953	43	0.8419	0.5049	0.6612
Group C	359	3414	1553	1861	68	0.8106	0.4549	0.6169
Group D	395	4606	2426	2180	68	0.8278	0.5267	0.6834
Group E	409	2827	1474	1353	108	0.7359	0.5214	0.6686
Group G	386	2600	1261	1339	108	0.7202	0.4850	0.6354
Group H	397	2850	1551	1299	82	0.7935	0.5442	0.6919
Group I	368	3026	1608	1418	93	0.7473	0.5314	0.6803
					Local Average	0.8090	0.5104	0.6641
					Major Average	0.8086	0.5054	0.6622

to different heuristics identifying same concepts from the same requirements. Because of this recurring concept extraction, we have to calculate recall by subtracting the false negative percentage from golden truth concepts sample size. This recall value demonstrates the actual ability to identify a concept from given requirement. Here we can observe that both recall and precision values improves over a large sample size. Even though major average recall value for unique concept extraction is the key metric for our purposes, major average recall value for individual concept extraction metric may also provide useful insights on performance of our tool. We can claim that in case of a larger sample size of requirements and labeling process by more experienced practitioners, the performance of the pipeline will potentially improve.

To assess performance of individual heuristics for concept extraction task, additional performance measurements for has been made. Firstly, number individual concepts extracted by each heuristic is noted in Table 4.4. Secondly, number of unique concepts extracted by each heuristic is noted in Table 4.5. And lastly, conversion rate from individual concepts identified to unique concepts is measured for each heuristic in Table 4.6. These tables indicate the frequency of heuristic triggering and their ability to identify different concepts in the domain. We observed that noun phrase extraction, heuristics; C6, C8, NHR2, NHR10, and HR8 identified the most concepts. Noun phrase extraction and heuristic NHR10 introduced most unique concept to the extraction pipeline. Further investigation on heuristic's conversion rate indicates noun phrase

extraction, ranked n-gram extraction, heuristics C11, C14, NHR7 seem to have ability to identify unique concept with lowest noise. These results showed that noun phrase extraction is the most useful step to identify domain concepts. This deduction support the claim that noun phrases represents almost all key concepts in a domain [81]. Therefore, further investigation on noun phrase extraction is performed.

Table 4.4. Individual concepts found by each heuristic

Data Set Name	NP	Ranked N-grams	C6	C8d	C8n	C8	C11s	C11a	C11	C14	NHR2s	NHR2d	NHR2	NHR7p	NHR7n	NHR7	NHR10p	NHR10c	NHR10	NHR11	NHR13	NHR14	HR8p	HR8c	HR8	HR9
Amazon A	536	25	110	282	282	564	23	37	60	2	28	28	56	9	9	18	1522	1522	3044	0	14	46	462	231	693	0
Amazon B	536	25	110	282	282	564	23	37	60	2	28	28	56	9	9	18	1522	1522	3044	0	14	46	462	231	693	0
Amazon C	536	25	110	282	282	564	23	37	60	2	28	28	56	9	9	18	1522	1522	3044	0	14	46	462	231	693	0
Group A	402	13	176	167	167	334	18	17	35	0	102	102	204	17	17	34	603	603	1206	0	2	212	182	91	273	0
Group B	315	12	110	93	93	186	13	13	26	2	31	31	62	11	11	22	457	457	914	0	12	56	134	67	201	4
Group C	429	21	140	124	124	248	27	37	64	3	40	40	80	12	12	24	836	836	1672	3	10	244	316	158	474	2
Group D	487	25	174	184	184	368	43	33	76	3	78	78	156	29	29	58	765	765	1530	21	10	310	246	123	369	6
Group E	447	16	181	232	232	464	28	28	56	3	87	87	174	17	17	34	1279	1279	2558	0	6	150	342	171	513	4
Group G	402	19	154	185	185	370	18	11	29	2	93	93	186	22	22	44	649	649	1298	3	6	32	188	94	282	0
Group H	437	25	168	179	179	358	9	16	25	2	95	95	190	31	31	62	608	608	1216	0	12	110	162	81	243	2
Group I	386	13	149	156	156	312	41	37	78	3	73	73	146	19	19	38	752	752	1504	6	14	62	210	105	315	0

Table 4.5. Unique concepts found by each heuristic

Data Set Name	NP	Ranked N-grams	C6	C8d	C8n	C8	C11s	C11a	C11	C14	NHR2s	NHR2d	NHR2	NHR7p	NHR7n	NHR7	NHR10p	NHR10c	NHR10	NHR11	NHR13	NHR14	HR8p	HR8c	HR8	HR9
Amazon A	294	25	60	59	41	100	19	35	54	2	11	27	38	7	3	10	91	118	209	0	12	2	91	120	211	0
Amazon B	294	25	60	59	41	100	19	35	54	2	11	27	38	7	3	10	91	118	209	0	12	2	91	120	211	0
Amazon C	294	25	60	59	41	100	19	35	54	2	11	27	38	7	3	10	91	118	209	0	12	2	91	120	211	0
Group A	181	12	40	72	9	81	14	15	29	0	12	73	85	13	3	16	49	65	114	0	2	13	120	75	195	0
Group B	140	9	51	35	16	51	9	13	22	2	9	23	32	7	3	10	37	46	83	0	8	5	77	49	126	4
Group C	260	21	84	124	29	153	20	32	52	2	22	35	57	12	5	17	74	105	179	3	10	14	158	126	284	2
Group D	255	23	67	184	26	210	34	31	65	2	18	57	75	24	6	30	72	87	159	13	9	14	149	107	256	6
Group E	227	15	56	74	18	92	22	28	50	2	20	69	89	15	5	20	85	104	189	0	5	17	160	111	271	4
Group G	161	19	21	64	12	76	14	9	23	2	4	57	61	19	3	22	51	60	111	2	6	7	108	70	178	0
Group H	205	19	41	67	7	74	7	15	22	2	6	73	79	22	6	28	65	77	142	0	8	8	112	73	185	2
Group I	204	13	56	67	18	85	27	32	59	2	15	63	78	15	5	20	51	72	123	5	13	10	116	82	198	0

Table 4.6. Individual to unique conversion rate of each heuristic

Data Set Name	NP	Ranked N-grams	C6	C8d	C8n	C8	C11s	C11a	C11	C14	NHR2s	NHR2d	NHR2	NHR7p	NHR7n	NHR7	NHR10p	NHR10c	NHR10	NHR11	NHR13	NHR14	HR8p	HR8c	HR8	HR9
Amazon A	0.5485	1.0000	0.5455	0.2092	0.1454	0.1773	0.8261	0.9459	0.9000	1.0000	0.3929	0.9643	0.6786	0.7778	0.3333	0.5556	0.0598	0.0775	0.0687	#DIV/0!	0.8571	0.0435	0.1970	0.5195	0.3045	#DIV/0!
Amazon B	0.5485	1.0000	0.5455	0.2092	0.1454	0.1773	0.8261	0.9459	0.9000	1.0000	0.3929	0.9643	0.6786	0.7778	0.3333	0.5556	0.0598	0.0775	0.0687	#DIV/0!	0.8571	0.0435	0.1970	0.5195	0.3045	#DIV/0!
Amazon C	0.5485	1.0000	0.5455	0.2092	0.1454	0.1773	0.8261	0.9459	0.9000	1.0000	0.3929	0.9643	0.6786	0.7778	0.3333	0.5556	0.0598	0.0775	0.0687	#DIV/0!	0.8571	0.0435	0.1970	0.5195	0.3045	#DIV/0!
Group A	0.4502	0.9231	0.2273	0.4311	0.0539	0.2425	0.7778	0.8824	0.8286	#DIV/0!	0.1176	0.7157	0.4167	0.7647	0.1765	0.4706	0.0813	0.1078	0.0945	#DIV/0!	1.0000	0.0613	0.6593	0.8242	0.7143	#DIV/0!
Group B	0.4444	0.7500	0.4636	0.3763	0.1720	0.2742	0.6923	1.0000	0.8462	1.0000	0.2903	0.7419	0.5161	0.6364	0.2727	0.4545	0.0810	0.1007	0.0908	#DIV/0!	0.6667	0.0893	0.5746	0.7313	0.6269	1.0000
Group C	0.6061	1.0000	0.6000	1.0000	0.2339	0.6169	0.7407	0.8649	0.8125	0.6667	0.5500	0.8750	0.7125	1.0000	0.4167	0.7083	0.0885	0.1256	0.1071	1.0000	1.0000	0.0574	0.5000	0.7975	0.5992	1.0000
Group D	0.5236	0.9200	0.3851	1.0000	0.1413	0.5707	0.7907	0.9394	0.8553	0.6667	0.2308	0.7308	0.4808	0.8276	0.2069	0.5172	0.0941	0.1137	0.1039	0.6190	0.9000	0.0452	0.6057	0.8699	0.6938	1.0000
Group E	0.5078	0.9375	0.3094	0.3190	0.0776	0.1983	0.7857	1.0000	0.8929	0.6667	0.2299	0.7931	0.5115	0.8824	0.2941	0.5882	0.0665	0.0813	0.0739	#DIV/0!	0.8333	0.1133	0.4678	0.6491	0.5283	1.0000
Group G	0.4005	1.0000	0.1364	0.3459	0.0649	0.2054	0.7778	0.8182	0.7931	1.0000	0.0430	0.6129	0.3280	0.8636	0.1364	0.5000	0.0786	0.0924	0.0855	0.6667	1.0000	0.2188	0.5745	0.7447	0.6312	#DIV/0!
Group H	0.4691	0.7600	0.2440	0.3743	0.0391	0.2067	0.7778	0.9375	0.8800	1.0000	0.0632	0.7684	0.4158	0.7097	0.1935	0.4516	0.1069	0.1266	0.1168	#DIV/0!	0.6667	0.0727	0.6914	0.9012	0.7613	1.0000
Group I	0.5285	1.0000	0.3758	0.4295	0.1154	0.2724	0.6585	0.8649	0.7564	0.6667	0.2055	0.8630	0.5342	0.7895	0.2632	0.5263	0.0678	0.0957	0.0818	0.8333	0.9286	0.1613	0.5524	0.7810	0.6286	#DIV/0!

In Table 4.7, it can be seen that noun phrase extraction can alone achieve fairly good results for concept identification in given domain. Noun phrase extraction yields impressive major average recall performance of 71,5% along with better precision (60%) compared to the whole pipeline (37,2%). Additionally, this method is better performing in the industrial setting compared to the student data sets. Modern transformer based spaCy NLP pipeline achieves outstanding results as high as 83%.

Table 4.7. Unique concepts found by noun phrase extraction.

Data Set Name	Unique Concepts	TP	FP	FN	Recall	Precision	F1
Amazon A	294	198	96	52	0.7920	0.6735	0.7279
Amazon B	294	187	107	38	0.8311	0.6361	0.7206
Amazon C	294	136	158	38	0.7816	0.4626	0.5812
Group A	181	117	64	45	0.7222	0.6464	0.6822
Group B	140	84	56	34	0.7119	0.6000	0.6512
Group C	260	150	110	64	0.7009	0.5769	0.6329
Group D	255	136	119	65	0.6766	0.5333	0.5965
Group E	227	155	72	52	0.7488	0.6828	0.7143
Group G	161	105	56	51	0.6731	0.6522	0.6625
Group H	205	110	95	82	0.5729	0.5366	0.5542
Group I	204	123	81	78	0.6119	0.6029	0.6074
				Local Average	0.7112	0.6003	0.6483
				Major Average	0.7148	0.5968	0.6505

Lastly, to assess performance of the relation extraction pipeline, recall, precision and F1 score metrics have been measured against the relation tags defined in golden truth. This comparison checks for the label of source concept, relation type and lastly label of the target concept. Table 4.8 shows the detailed results of the relation extraction pipeline. It can be observed that major average recall value of 36,52% for relation extraction is far smaller than the same metric for concept extraction which is 74,8%. Our pipeline can correctly identify 1 relation for every 3 relation in the data set. There are only one public study on domain model extraction using NLP [1] which reported above 90% recall value for relation extraction. They opted to keep their data set pri-

vate and their evaluation method was through an empirical study compared to our automated statistical evaluation. A quick review of extracted relation set compared to ground truth revealed that many of the "false" relations extracted by our pipeline is actually only vary slightly from the ground truth version. Human intuition would allow us to easily label relations with slight variations as a correct relation for given domain. However, as long as relation is not exactly the same with the predefined golden truth, our performance evaluation pipeline will label the relation as a false positive, therefore low recall performance of relation detection is an exaggerated result.

Table 4.8. Unique relation extraction performance metrics of the pipeline.

Data Set Name	Ground Truth Relations	Relations Found	TP	FP	FN	Recall	Precision	F1
Group A	123	243	55	188	68	0.4472	0.2263	0.3005
Group B	97	140	29	111	68	0.2990	0.2071	0.2447
Group C	112	297	27	270	85	0.2411	0.0909	0.1320
Group D	152	300	63	237	89	0.4145	0.2100	0.2788
Group E	156	329	51	278	105	0.3269	0.1550	0.2103
Group G	150	215	54	161	96	0.3600	0.2512	0.2959
Group H	156	259	71	188	85	0.4551	0.2741	0.3422
Group I	152	244	51	193	101	0.3355	0.2090	0.2576
					Local Average	0.3599	0.2030	0.2578
					Major Average	0.3652	0.1978	0.2568

Table 4.9. Comparison of our study with the state of art.

Study	Evaluation	Objective	Input	Output	Requirement Count	Autonomy	Public	Concept Extraction Recall	Concept Extraction Precision	Relation Extraction Recall	Relation Extraction Precision
Ours	Statistical	Domain Modeling	Unrestricted NL	Domain Model	763	Automated	Yes	0.7478	0.4842	0.3652	0.2568
[1]	Empirical	Domain Modeling	Unrestricted NL	Domain Model	678	Automated	Only Source Code			≈ 0.9	≈ 0.36
[76]	Statistical	Domain Modeling	NL Problem Descriptions	Domain Model		Automated	Private	≈ 0.77	≈ 0.9		
[94]	Statistical	Term Extraction	User Story Requirements	Class Diagram	100	Automated	Yes	0.749	0.7344		
[95]	Statistical	Term Extraction	User Story Requirements	Ranked Term List	100	Automated	Only Data Set	0.732	0.8394		
[84]	Statistical	Term Extraction	User Story Requirements	Ranked Term List	100	Automated	Only Data Set	≈ 0.82	≈ 0.72		

Lastly, to summarize our answer to RQ2 we prepared a comparison Table 4.9 which compares our study with similar studies in holistic fashion. Our pipeline achieves on par performance with similar studies while providing a completely autonomous unrestricted pipeline to extract a more complete domain model. Also we made all of our sources public to provide the ability to track, reproduce, and modify the pipeline for the readers. Additionally, our pipeline has been tested under larger requirement set and more diverse data set collection in terms of domain variety and diversified backgrounds of taggers.

4.4. Threats to Validity

During our research we encountered and identified several challenges and potential threats to the validity of the study.

- Data Set Validity: Identifying domain concepts and relations is a task which requires the interpretation of the researcher. Process of tagging the data set and requirements may be subject to the bias of the tagger. This effect can be observed in our pipeline results by comparing the performance on different data sets tagged by internal and external researchers from different backgrounds. We provide different tags for Amazon data set to give a fair insight on the performance of our pipeline to the readers to overcome this threat. Additionally, source code of our pipeline [97], source requirements, student models, and expert models (ground truth models) [96] are made public to let individuals assess the validity of the data set and true performance of our tool.
- Implementation Validity We tried to access source code for related studies but unfortunately researchers choose to keep their implementation sources private. We contacted the author/researcher of the most similar study [1] on domain model extraction from unrestricted test documents and managed to get their pipeline working in our local environments, but their rule based system uses compiled NLP pipeline components. Therefore we could not examine their implementation of the same heuristics. Lack of access to the implementation of related studies

channeled us to implement each heuristic by interpreting the source study for the heuristic. This interpretation may cause different usage of the heuristic from the intended usage of the heuristic in the source studies. Additionally, spaCy is a massive NLP framework and masterful usage of spaCy requires long term study and our lack of experience with the spaCy NLP collection may cause inefficiencies or faults in our NLP pipeline.

- External Validity Despite our efforts we couldn't manage to access requirements and models used in industrial setting and commercial projects. Additionally our sample size is limited compared to data collections of real world software projects. Lack of large scale data sets may cause performance issues and challenges while transitioning the project setting from educational to industrial.

5. CONCLUSIONS

In this chapter we will summarize our findings and observations. Firstly, we come to the conclusion for performance of our method. Then we interpret individual conclusions for performance of pipeline components based on our results. We analyze potential causes of effectiveness of the approach on different data sets. After performance conclusions are made, observations on the domain model extraction task, effectiveness of rule-based, language model and statistical ranking approaches are explained. And lastly, we draw an outline for future research on the problem with proposed approach.

In this study, we explored the Domain Model Extraction from Unrestricted Text using NLP by identifying key domain model elements with the help of modern NLP tools combined with a rule based approach. We build a resourceful pipeline which utilizes different approaches. The quantitative analysis showed that our pipeline can achieve on par performance with similar studies while utilizing recent technological advancements. While using external golden truth for the same data set our pipeline exceeds the performance of reported studies. We also provided public pipeline resources and performed our analysis using a larger data set to assess the performance of the tool more fairly. We achieved improved recall rate for domain concept extraction, but the precision of the pipeline diminished. Relation extraction pipeline performance turned out to be lower than both our concept extraction performance, and reported relation extraction studies. This shows that rule based approach lacks the generalization to interpret complex relations. We can also say that our implementation of relation extraction pipeline may differ from the intended implementation of these heuristics therefore creating the difference between the performance of our pipeline and reported performance measurements. We also observed that noun phrase extraction alone been the best performing method in the pipeline doubtlessly. spaCy's modern roBERTa based pipeline with its syntactic parsing capabilities and large language model data achieved extra-ordinary performance for key concept extraction task.

Our pipeline achieved its best performance with user story formatted Amazon Smart Home data set. We believe this is due to writing of the requirements in more atomic manner. Student data sets violate many of the best practices of writing software requirements. When investigated manually, it can be seen that many of the requirements are not atomic. Also students used different terms for same concepts without explicitly specifying the relation.This can be overcomed with human intuition but for a rule based pipeline, such requirement statements decreases the performance of the pipeline. Lack of additional industrial project data set unfortunately limited our ability to evaluate our pipeline in a practical environment.

In conclusion, our pipeline achieved acceptable domain model extraction capabilities. It may be used as an assistance tool to extract the domain model and later allowing practitioners to modify the model to fit their domain description. Lack of generalization capabilities of rule-based approaches limited our capabilities as well, and we believe performance of the pipeline can be improved by implementing additional heuristics to increase coverage and precision of the pipeline in future. Our approach which combines modern language model, term ranking algorithms, and heuristic rules is still a promising narrow AI approach to extract domain models despite it is shortcomings.

Future work for this multi-level approach should focus on eliminating the validity threats of this study. Firstly, implementation threats can be eliminated by accessing implementation of other resources to error proof to pipeline. Additionally more heuristic rules can be implemented to cover shortcomings of the pipeline. And lastly, larger data sets which is labeled by external industrial practitioners can provide unbiased assessment of the practical performance of the approach.

Since heuristic based approaches lack generalization to capture more complex syntactic relations, if more complete data sets are provided, a multi layered deep learning approach can achieve better performance which can also be transitioned across multiple domains and multiple project settings.

REFERENCES

1. Arora, C., M. Sabetzadeh, L. Briand and F. Zimmer, "Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation", *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pp. 250–260, Saint-malo, France, 2016.

2. Liddy, E. D., *Encyclopedia of Library and Information Science*, CRC Press, Boca Raton, 2001.

3. Hutchins, W. J., "The Georgetown-IBM Experiment Demonstrated in January 1954", *Machine Translation: From Real Users to Research*, pp. 102–114, Berlin, Heidelberg, 2004.

4. Hirschberg, J. and C. Manning, "Advances in Natural Language Processing", *Science*, Vol. 349, pp. 261–266, 2015.

5. Cambria, E. and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research", *IEEE Computational Intelligence Magazine*, Vol. 9, No. 2, pp. 48–57, 2014.

6. Young, T., D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing", *IEEE Computational Intelligence Magazine*, Vol. 13, No. 3, pp. 55–75, 2018.

7. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2019, `http://arxiv.org/abs/1810.04805`, accessed on April 17, 2023.

8. Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", ArXiv:1907.11692, 2019.

9. Alec, R. and N. Karthik, "Improving Language Understanding by Generative Pre-Training", `https://openai.com/research/language-unsupervised`, accessed on April 17, 2023.

10. Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", 2020, `http://arxiv.org/abs/1910.10683`, accessed on April 17, 2023.

11. Castanha, J., Indrawati, S. K. Pillai, G. Ramantoko and T. Widarmanti, "A Systematic Literature Review on Natural Language Processing (NLP)", *2022 International Conference on Advanced Creative Networks and Intelligent Systems*, pp. 1–6, Bandung, Indonesia, 2022.

12. Hasan, K. S. and V. Ng, "Automatic Keyphrase Extraction: A Survey of the State of the Art", *52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1262–1273, Baltimore, Maryland, 2014.

13. Papagiannopoulou, E. and G. Tsoumakas, "A Review of Keyphrase Extraction", 2019, `http://arxiv.org/abs/1905.05044`, accessed on November 11, 2022.

14. Florescu, C. and C. Caragea, "PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents", *55th Annual Meeting of the Association for Computational Linguistics*, pp. 1105–1115, Vancouver, Canada, 2017.

15. El-Beltagy, S. R. and A. Rafea, "KP-Miner: A Keyphrase Extraction System for English and Arabic Documents", *Information Systems*, Vol. 34, No. 1, pp. 132–144, 2009.

16. Liu, Z., P. Li, Y. Zheng and M. Sun, "Clustering to Find Exemplar Terms for Keyphrase Extraction", *Conference on Empirical Methods in Natural Language*

Processing, pp. 257–266, Singapore, 2009.

17. Campos, R., V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes and A. Jatowt, "A Text Feature Based Automatic Keyword Extraction Method for Single Documents", *European Conference on Information Retrieval*, pp. 684–691, Grenoble, France, 2018.

18. Won, M., B. Martins and F. Raimundo, "Automatic Extraction of Relevant Keyphrases for the Study of Issue Competition", EC:875, 2019.

19. Mihalcea, R. and P. Tarau, "TextRank: Bringing Order into Text", *Conference on Empirical Methods in Natural Language Processing*, pp. 404–411, Barcelona, Spain, 2004.

20. Brin, S. and L. Page, "The anatomy of a large-scale hypertextual Web search engine", *Computer Networks and Integrated Services Digital Network Systems*, Vol. 30, No. 1, pp. 107–117, 1998.

21. Herings, P., G. Laan and D. Talman, "The positional power of nodes in digraphs", *Social Choice and Welfare*, Vol. 24, pp. 439–454, 2005.

22. Wan, X. and J. Xiao, "Single Document Keyphrase Extraction Using Neighborhood Knowledge", *23rd National Conference on Artificial Intelligence*, Vol. 2, p. 855–860, Chicago, Illinois, 2008.

23. Rose, S., D. Engel, N. Cramer and W. Cowley, *Text Mining: Applications and Theory*, John Wiley Sons, Ltd, Padstow, Cornwal, 2010.

24. Danesh, S., T. Sumner and J. H. Martin, "SGRank: Combining Statistical and Graphical Methods to Improve the State of the Art in Unsupervised Keyphrase Extraction", *4th Joint Conference on Lexical and Computational Semantics*, pp. 117–126, Denver, Colorado, 2015.

25. Wan, X. and J. Xiao, "Single Document Keyphrase Extraction Using Neighborhood Knowledge", *23rd National Conference on Artificial Intelligence - Volume 2*, pp. 855–860, 2008.

26. Bougouin, A., F. Boudin and B. Daille, "TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction", *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, Nagoya, Japan, 2013.

27. Boudin, F., "Unsupervised Keyphrase Extraction with Multipartite Graphs", *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 667–672, New Orleans, Louisiana, 2018.

28. Liu, Z., W. Huang, Y. Zheng and M. Sun, "Automatic Keyphrase Extraction via Topic Decomposition", *Conference on Empirical Methods in Natural Language Processing*, pp. 366–376, Cambridge, MA, 2010.

29. Sterckx, L., T. Demeester, J. Deleu and C. Develder, "Topical Word Importance for Fast Keyphrase Extraction", *24th International Conference on World Wide Web*, pp. 121–122, 2015.

30. Wang, R., "Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors", 2015, `https://api.semanticscholar.org/CorpusID:14116376`, accessed on January 1, 2023.

31. Shi, W., W. Zheng, J. X. Yu, H. Cheng and L. Zou, "Keyphrase Extraction Using Knowledge Graphs", *Data Science and Engineering*, Vol. 2, No. 4, pp. 275–288, 2017.

32. Bojanowski, P., E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information", *Transactions of the Association for Computational*

Linguistics, Vol. 5, pp. 135–146, 2017.

33. Yu, Y. and V. Ng, "WikiRank: Improving Keyphrase Extraction Based on Background Knowledge", ArXiv:1803.09000, 2018.

34. Mahata, D., J. Kuriakose, R. R. Shah and R. Zimmermann, "Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings", OSF:10.31219/osf.io/j76y3, 2018.

35. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", ArXiv:1301.3781, 2013.

36. Lau, J. H. and T. Baldwin, "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation", *1st Workshop on Representation Learning for NLP*, pp. 78–86, Berlin, Germany, 2016.

37. Pagliardini, M., P. Gupta and M. Jaggi, "Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features", *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 528–540, New Orleans, Louisiana, 2018.

38. Pennington, J., R. Socher and C. Manning, "Glove: Global Vectors for Word Representation", *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

39. Bennani-Smires, K., C. Musat, A. Hossmann, M. Baeriswyl and M. Jaggi, "Simple Unsupervised Keyphrase Extraction using Sentence Embeddings", *22nd Conference on Computational Natural Language Learning*, pp. 221–229, Brussels, Belgium, 2018.

40. Papagiannopoulou, E. and G. Tsoumakas, "Local word vectors guiding keyphrase extraction", *Information Processing Management*, Vol. 54, No. 6, pp. 888–902, 2018.

41. Tomokiyo, T. and M. Hurst, "A Language Model Approach to Keyphrase Extraction", *ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pp. 33–40, Sapporo, Japan, 2003.

42. Merity, S., N. S. Keskar and R. Socher, "An Analysis of Neural Language Modeling at Multiple Scales", ArXiv:1803.08240, 2018.

43. Hulth, A., "Improved Automatic Keyword Extraction given More Linguistic Knowledge", *Conference on Empirical Methods in Natural Language Processing*, pp. 216–223, USA, 2003.

44. Nguyen, T. D. and M.-Y. Kan, "Keyphrase Extraction in Scientific Publications", *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pp. 317–326, Berlin, Heidelberg, 2007.

45. Medelyan, O., E. Frank and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction", *Conference on Empirical Methods in Natural Language Processing*, pp. 1318–1327, Singapore, 2009.

46. Nguyen, T. D. and M.-T. Luong, "WINGNUS: Keyphrase Extraction Utilizing Document Logical Structure", *5th International Workshop on Semantic Evaluation*, pp. 166–169, Uppsala, Sweden, 2010.

47. Caragea, C., F. A. Bulgarov, A. Godea and S. Das Gollapalli, "Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1435–1446, Doha, Qatar, 2014.

48. Wang, L. and S. Li, "PKU_ICL at SemEval-2017 Task 10: Keyphrase Extraction with Model Ensemble and External Knowledge", *11th International Workshop on Semantic Evaluation*, pp. 934–937, Vancouver, Canada, 2017.

49. Shen, J., Y. Qu, W. Zhang, Y. Yu, S. McIlraith and K. Weinberger, "ACM In-

ternational Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers", New York, NY, USA, 2018.

50. Jiang, X., Y. Hu and H. Li, "A Ranking Approach to Keyphrase Extraction", *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 756–757, Boston, MA, USA, 2009.

51. Zhang, Y., Y. Chang, X. Liu, S. D. Gollapalli, X. Li and C. Xiao, "MIKE: Keyphrase Extraction by Integrating Multidimensional Information", *ACM on Conference on Information and Knowledge Management*, pp. 1349–1358, Singapore, Singapore, 2017.

52. Bougouin, A., F. Boudin and B. Daille, "Keyphrase Annotation with Graph Co-Ranking", ArXiv:1611.02007, 2016.

53. Gollapalli, S. D., X. Li and P. Yang, "Incorporating Expert Knowledge into Keyphrase Extraction", *AAAI Conference on Artificial Intelligence*, San Francisco, USA, 2017.

54. Bingel, J. and A. Søgaard, "Identifying beneficial task relations for multi-task learning in deep neural networks", ArXiv:1702.08303, 2017.

55. Zhang, Q., Y. Wang, Y. Gong and X. Huang, "Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 836–845, 2016.

56. Meng, R., S. Zhao, S. Han, D. He, P. Brusilovsky and Y. Chi, "Deep Keyphrase Generation", *55th Annual Meeting of the Association for Computational Linguistics*, pp. 582–592, Vancouver, Canada, 2017.

57. Chen, J., X. Zhang, Y. Wu, Z. Yan and Z. Li, "Keyphrase Generation with Correlation Constraints", ArXiv:1808.07185, 2018.

58. Ye, H. and L. Wang, "Semi-Supervised Learning for Neural Keyphrase Generation", *Conference on Empirical Methods in Natural Language Processing*, pp. 4142–4153, Brussels, Belgium, 2018.

59. Chen, P. P.-S., "English sentence structure and entity-relationship diagrams", *Information Sciences*, Vol. 29, No. 2, pp. 127–149, 1983.

60. Abbott, R. J., "Program Design by Informal English Descriptions", *Communications of the ACM*, Vol. 26, No. 11, p. 882–894, 1983.

61. Li, K., R. G. Dewar and R. J. Pooley, "Requirements Capture in Natural Language Problem Statements", 2003, `http://www.macs.hw.ac.uk/cs/techreps/docs/files/HW-MACS-TR-0023.pdf`, accessed on January 1, 2023.

62. Ibrahim, M. and R. Ahmad, "Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques", *Second International Conference on Computer Research and Development*, pp. 200–204, Washington, USA, 2010.

63. Dahhane, W., A. Zeaaraoui, E. H. Ettifouri and T. Bouchentouf, "An Automated Object-Based Approach to Transforming Requirements to Class Diagrams", *Second World Conference on Complex Systems*, pp. 158–163, Agadir, 2014.

64. Athiththan, K., S. Rovinsan, S. Sathveegan, N. Gunasekaran, K. S. A. W. Gunawardena and D. Kasthurirathna, "An Ontology-based Approach to Automate the Software Development Process", *IEEE International Conference on Information and Automation for Sustainability*, pp. 1–6, Sri Lanka, 2018.

65. Allala, S. C., J. P. Sotomayor, D. Santiago, T. M. King and P. J. Clarke, "Towards Transforming User Requirements to Test Cases Using MDE and NLP", *IEEE 43rd Annual Computer Software and Applications Conference*, Vol. 2, pp. 350–355, Mil-

waukee, USA, 2019.

66. Abdelnabi, E. A., A. M. Maatuk, T. M. Abdelaziz and S. M. Elakeili, "Generating UML Class Diagram Using NLP Techniques and Heuristic Rules", *20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, pp. 277–282, Monastir, Tunisia, 2020.

67. Nasiri, S., Y. Rhazali, M. Lahmer and N. Chenfour, "Towards a Generation of Class Diagram from User Stories in Agile Methods", *Procedia Computer Science*, Vol. 170, pp. 831–837, 2020.

68. Bashir, N., M. Bilal, M. Liaqat, M. Marjani, N. Malik and M. Ali, "Modeling Class Diagram Using NLP in Object-Oriented Designing", *National Computing Colleges Conference*, pp. 1–6, Taif, Makkah, 2021.

69. Elallaoui, M., K. Nafil and R. Touahni, "Automatic Generation of UML Sequence Diagrams from User Stories in Scrum Process", *10th International Conference on Intelligent Systems: Theories and Applications*, pp. 1–6, Rabat, Morocco, 2015.

70. Jahan, M., Z. S. H. Abad and B. Far, "Generating Sequence Diagram from Natural Language Requirements", *29th International Requirements Engineering Conference Workshops (REW)*, pp. 39–48, Notre Dame, IN, USA, 2021.

71. Wautelet, Y., S. Heng, D. Hintea, M. Kolp and S. Poelmans, "Bridging User Story Sets with the Use Case Model", *International Conference on Conceptual Modeling*, pp. 127–138, Gifu, Japan, 2016.

72. Vemuri, S., S. Chala and M. Fathi, "Automated Use Case Diagram Generation from Textual User Requirement Documents", *IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, Windsor, Ontario, Canada, 2017.

73. Elallaoui, M., K. Nafil and R. Touahni, "Automatic Transformation of User Stories

into UML Use Case Diagrams Using NLP Techniques", *Procedia Computer Science*, Vol. 130, pp. 42–49, 2018.

74. Hamza, Z. A. and M. Hammad, "Generating UML Use Case Models from Software Requirements Using Natural Language Processing", *8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pp. 1–6, Manama, Bahrain, 2019.

75. Joanna Santos, C. S., S. Moshtari and M. Mirakhorli, "An Automated Approach to Recover the Use-case View of an Architecture", *IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 63–66, Salvador, Brazil, 2020.

76. Saini, R., G. Mussbacher, J. L. Guo and J. Kienzle, "DoMoBOT: An AI-Empowered Bot for Automated and Interactive Domain Modelling", *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pp. 595–599, Fukuoka, Japan, 2021.

77. Saini, R., G. Mussbacher, J. L. Guo and J. Kienzle, "A Neural Network Based Approach to Domain Modelling Relationships and Patterns Recognition", *IEEE Tenth International Model-Driven Requirements Engineering*, pp. 78–82, Zurich, Switzerland, 2020.

78. Aydemir, F. B. and F. Dalpiaz, "Supporting Collaborative Modeling via Natural Language Processing", *Conceptual Modeling: 39th International Conference*, p. 223–238, Vienna, Austria, 2020.

79. Mengyuan, Y., W. Lisong, K. Jiexiang, G. Zhongjie, H. Wang, W. Yin and C. Buzhan, "Automatic Generation Method of Airborne Display and Control System Requirement Domain Model Based on NLP", *IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, pp. 1042–1046, Chengdu, China, 2021.

80. Deeptimahanti, D. K. and M. A. Babar, "An Automated Tool for Generating UML Models from Natural Language Requirements", *IEEE/ACM International Conference on Automated Software Engineering*, pp. 680–682, Auckland, New Zealand, 2009.

81. Justeson, J. S. and S. M. Katz, "Technical terminology: some linguistic properties and an algorithm for identification in text", *Natural Language Engineering*, Vol. 1, No. 1, p. 9–27, 1995.

82. Honnibal, M., I. Montani, S. Van Landeghem and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python", Zenodo:10.5281/zenodo.3358113, 2020.

83. Murukannaiah, P. K., N. Ajmeri and M. P. Singh, "Acquiring Creative Requirements from the Crowd: Understanding the Influences of Individual Personality and Creative Potential in Crowd RE", *20th IEEE International Requirements Engineering Conference*, pp. 176–185, Beijing, 2016.

84. Zhang, J., S. Chen, J. Hua, N. Niu and C. Liu, "Automatic Terminology Extraction and Ranking for Feature Modeling", *IEEE 30th International Requirements Engineering Conference*, pp. 51–63, Melbourne, Australia, 2022.

85. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", ArXiv:1301.3781, 2013.

86. Dedeoğlu, U. O., "Master Heuristic List", 2023, `https://doi.org/10.5281/zenodo.8301488`, accessed on January 1, 2023.

87. AI, E., "Displacy.js An Open Source NLP Visualizer", 2016, `https://explosion.ai/blog/displacy-js-nlp-visualizer`, accessed on February 1, 2023.

88. Manning, C., M. Surdeanu, J. Bauer, J. Finkel, S. Bethard and D. McClosky,

"The Stanford CoreNLP Natural Language Processing Toolkit", *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, Baltimore, Maryland, 2014.

89. Nivre, J., M.-C. de Marneffe, F. Ginter, J. Hajič, C. D. Manning, S. Pyysalo, S. Schuster, F. Tyers and D. Zeman, "Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection", *12th Language Resources and Evaluation Conference*, pp. 4034–4043, Marseille, France, 2020.

90. Honnibal, M. and M. Johnson, "An Improved Non-monotonic Transition System for Dependency Parsing", *Conference on Empirical Methods in Natural Language Processing*, pp. 1373–1378, Lisbon, Portugal, 2015.

91. Nivre, J. and J. Nilsson, "Pseudo-Projective Dependency Parsing", *43rd Annual Meeting of the Association for Computational Linguistics*, pp. 99–106, Ann Arbor, Michigan, 2005.

92. Team, P., "PlantUML", 2019, `https://plantuml.com/`, accessed on February 1, 2023.

93. Murukannaiah, P. K., N. Ajmeri and M. P. Singh, "Toward Automating Crowd RE", *IEEE 25th International Requirements Engineering Conference (RE)*, pp. 512–515, Lisbon, Portugal, 2017.

94. Gemkow, T., M. Conzelmann, K. Hartig and A. Vogelsang, "Automatic Glossary Term Extraction from Large-Scale Requirements Specifications", *IEEE 26th International Requirements Engineering Conference*, pp. 412–417, Banff, Alberta, Canada, 2018.

95. Mishra, S. and A. Sharma, "Automatic Word Embeddings-Based Glossary Term Extraction from Large-Sized Software Requirements", *Requirements Engineering: Foundation for Software Quality*, pp. 203–218, Pisa, Italy, 2020.

96. Dedeoğlu, U. O., "Data Set files for Domain Extractor", 2023, `https://gitlab.com/uodedeoglu/thesis-dataset-related`, accessed on January 1, 2023.

97. Dedeoğlu, U. O., "Source code repository for Domain Extractor", 2023, `https://gitlab.com/uodedeoglu/domain-extractor`, accessed on January 1, 2023.

98. Ferrari, A., L. Zhao and W. Alhoshan, "NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies", *IEEE/ACM 43rd International Conference on Software Engineering*, pp. 322–323, Madrid, Spain, 2021.