

MASTER

Determining the Impact of Business Requirement Changes in Process Models' Graph Representation by NLP Assisted Text Matching

Salmaan, Shabana

Award date:
2020

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain



Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Determining the Impact of Business Requirement Changes in Process Models' Graph Representation by NLP Assisted Text Matching

Master Thesis

Shabana Salmaan



Supervisors:

Prof. dr. ir. Hajo Reijers
Dr. ir. Eduardo Gonzalez Lopez de Murillas

15-August-2020

Abstract

Business process models are often subjected to change rapidly to cope with the requirements needed for organisational growth. These business changes are considered to be a tedious process. It may be helpful to the organisation to search for the changes in its process repository. Earlier methods have been addressed in different perspectives like the text to model alignment and analysing the impact on a specific use case. This thesis describes one such method for an organisation to help business analysts with a semi-automated method to identify all the impacted components in the business processes for a given set of change requirements. To this end, we define a Knowledge Graph implemented using Grakn technology to represent and query different types of business models. By this, business analysts in an organisation can evaluate the impact of an incoming change requirement in a shorter period and more efficiently. Further, we propose methods to qualitatively determine the impact of textual change requirements on an existing business process repository containing several process models. To this end, we first develop a method to match relevant business entities inside the process repository to change requirements. This method involves the study and application of Natural Language Processing techniques to deal with textual documents and short-text fragments used in process model definitions. We use a small annotated dataset to evaluate the performance of several state-of-the-art semantic text similarity methods for this task. Further, we propose a text-distance-based approach for the above semantic similarity problem improving over the state-of-the-art methods' performance. Finally, we determine the impact of a change requirement on a process repository qualitatively by providing the context of the affected matched business entities inside process models and an OCR-assisted visualisation of these matched entities.

Keywords: Process Repository, Text Distance, Semantic Similarity, Change impact assessment, Change Requirements

Dedicated to Yash Kumar Bhati, his family and friends.

Acknowledgements

This report results from my thesis at the Architecture of Information Systems (AIS) group in the Department of Mathematics and Computer Science, Eindhoven University of Technology (TU/e). This thesis is only possible because of my program Erasmus Mundus Big Data Management and Analytics (BDMA) and all the universities and professors involved. I thank everyone for allowing me to be branded for life as a BDMA student. This thesis is a combined effort of myself and a few select people guiding me at every step.

First of all, I want to thank Professor Hajo Reijers for believing in me by assigning me such an exciting project and continually monitoring my approaches and guiding me towards reaching the results. Secondly, I want to thank Eduardo Gonzalez Lopez de Murillas for always being available for guidance, feedback, and technical issues. It would not have been easier without his suggestions and reviews every week until the end of the project. I also want to thank Chris Bergman for helping me throughout the project with their data and business understanding.

This surely could not be possible without the long-distance support from my mother Akthar Begum, and my brother Shakeel Salmaan. A special thanks from the bottom of heart to my dearest friend Rahini Chandrasekaran and her mother Booma Chandrasekaran for kick-starting the idea to pursue masters; and my friends Ashwini Hebbar, Divya Seevaratnam and Suresh Mohan for being there for me. Finally, my heartfelt thanks and love to my partner Amritansh Sharma for the emotional support, his constant guidance with his knowledge and care.

List of Figures

2.1	Sample Process Repository Architecture. Source: [1]	5
2.2	Sample BPMN Diagram. Source: [2]	6
2.3	Example of a Knowledge Graph	7
2.4	Sample OCR: Extracting editable texts from a photo of an old newspaper. Source: [3]	9
3.1	Business Process Change Management	14
4.1	Schematic of the Overall Methodology	15
5.1	Client-Server Architecture for process repository data access	18
5.2	Process repository hierarchy	19
5.3	Relationship between JSON files	20
5.4	High Level Architecture of Grakn knowledge graph creation	21
5.5	JSON objects representation with Grakn variables and attributes	22
5.6	Knowledge graph schema	24
6.1	Text pre-processing schematic	27
6.2	Sample data of textual change requirements with ground truth business entities	29
7.1	Textual change requirements with the Ground Truth and Retrieved top 3 Ranking	33
7.2	Word2Vec - Accuracy and NDCG	36
7.3	GloVe - Accuracy and NDCG	37
7.4	BERT - Accuracy and NDCG	38
7.5	Comparing Text Embedding algorithms to selected textdistance algorithms in our framework using Accuracy and NDCG metrics	47
7.6	Comparing Text Embedding algorithms with and without our text similarity framework using Accuracy and NDCG metrics	48

7.7	Comparing Text Embedding and selected textdistance algorithms, both in our text similarity framework using Accuracy and NDCG metrics	49
8.1	The user has to enter a change requirement (highlighted in red) to obtain the top 5 suggested matches. Then, the user has to select the actual business entity (highlighted in blue). Finally, the selected business entity will be placed in the standard query template to get the impacted business entities. . .	52
8.2	Impacted business entities	53
8.3	BPMN image example from our process repository	55
8.4	Preprocessed example BPMN image	60
8.5	Text detection by bounding box example BPMN image	61
A.1	Sample Analysis Model	73
A.2	Sample Company Map	74
A.3	Sample Document Model	74
A.4	Sample IT System Model	75
A.5	Sample Working Environment Model	75

List of Tables

5.1	Different types of business models loaded in Grakn	25
7.1	Results from NLP State of the Art Methods	39
7.2	Ranks generated for each Change Requirements by NLP State of the Art Methods	40
7.3	Results for textdistance algorithms in our text similarity framework. The best performing algorithm for each category using the top 5 NDCG has been highlighted and the corresponding NDCG score marker with an *.	43
7.4	Ranks generated for each Change Requirements by textdistance algorithms in our text similarity framework.	45
7.5	Median Ranks for the different algorithms	46

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BPM	Business Process Management
BPMN	Business Process Model and Notation
CRD	Change Requirement Document
GloVe	Global Vectors for Word Representation
HTTP	Hyper Text Transfer Protocol
ID	Identifier
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
KG	Knowledge Graph
NDCG	Normalised Discounted Cumulative gain
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
RDBMS	Relational Database Management System
REST	Representational state transfer
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator

Contents

List of Figures	iv
List of Tables	vi
List of Abbreviations	vii
1 Introduction	1
1.1 Problem Context	1
1.2 Research Question	2
1.3 Thesis Outline	3
2 Background	4
2.1 Preliminaries	4
2.1.1 Process Repository	4
2.1.2 Grakn: Knowledge Graph Database	6
2.1.3 Natural Language Processing: Existing Methods	7
2.1.4 Pytesseract : Optical Character Recognition	8
2.2 Related Work	9
2.2.1 Text to Process Matching	10
2.2.2 Change Impact Analysis	11
2.2.3 Process Repository Framework	11
3 Business Understanding	13
4 Methodology	15
5 Graph Representation of Process Repositories	17
5.1 Data Collection	17
5.2 Data Understanding	18
5.3 Implementation	21
5.3.1 Schema Definition in Grakn	22
5.3.2 Data Insertion into Grakn	24

5.4	Summary	25
6	Text Similarity	26
6.1	Methodology	26
6.1.1	State-of-the-art in Text Similarity	26
6.1.2	Framework for Text Similarity applied to Change Requirement Matching	28
6.2	Implementation	30
6.2.1	State-of-the-art in Text Similarity	30
6.2.2	Framework for Text Similarity applied to Change Requirement Matching	31
6.3	Summary	31
7	Text Similarity Evaluation	32
7.1	Objective	32
7.2	Setup	32
7.3	Execution	35
7.4	Results	35
7.4.1	State-of-the-art models	35
7.4.2	Text-distance algorithms in our Similarity Framework .	41
7.4.3	Comparing our framework with the State-of-the-Art . .	46
7.5	Summary	49
8	Impact Assessment	51
8.1	Context Detection	51
8.2	Visualisation Tool for Impact Assessment	54
8.2.1	Data Collection	54
8.2.2	Methodology	54
8.2.3	Implementation	59
8.2.4	Evaluation	62
8.3	Summary	62
9	Conclusion	63
9.1	Summary of Contributions	63
9.2	Limitations and Future Works	64
	Bibliography	66
A	Types of Business Models	73
B	Graql Queries	76

Chapter 1

Introduction

In this work, we present the method and evaluation of a semi-automated method to identify the business entities in a process repository impacted by textual business change requirements. Further, we propose methods to qualitatively determine the impact on these business entities in the context of the business process models that they belong to. This semi-automated method is designed to be effectively used by business analysts, who will save their time that would otherwise be employed in manually identifying the impacted business entities. In this chapter, we will discuss the context of the overall problem is detailed, then we present our research goals in the form of research questions. Finally, the outline of the following chapters is provided.

1.1 Problem Context

Many organisations maintain their graphical representation of their business processes, generally in process models, for various stakeholders to logically and visually interpret what the organisation does [4]. Some examples of business models are process models expressed in Business Process Models and Notation (BPMN), Use-Case Diagrams, Organisational Diagrams, etc.

In order to keep up with the expectation of their clients, to adopt new and evolving technologies, to optimise existing solutions, or to be able to compete, businesses need to make changes to their existing business processes [5]. One of the important steps involved in planning and changing existing business processes is creating a formal textual document containing the change requirements to be implemented, based on the analysis provided by business analysts [6]. A business analyst is a person who has the required domain knowledge and experience to evaluate all kinds of business changes and to make decisions on how to apply these changes to the existing busi-

ness depending upon the level of impact. The business models tend to be huge in most organisations, and business analysts spend enormous amounts of time evaluating the impact of each requirement. To avoid this time consuming task and to identify the impacted business entities more efficiently, we developed a semi-automated system for this task.

1.2 Research Question

This study focuses on how to qualitatively determine the impact of given change requirements on existing business processes. In order to evaluate our system for identifying the matching business entity in existing business processes from the textual change requirement, we will need some ground truth for the same. Further, we have to process and collect the relevant data corresponding to process models in available data formats (such as JSON, XML) that are available in the organisation's process repository in a format that is amenable to our analysis. Next, we wish to match business change requirements to impacted business entities in the process repository using text similarity approaches. Finally, the impacted business entities thus obtained can be shown in the context of the process repository representation. With the availability of data and description of the problem context, we can formally define our research questions below.

RQ1: *Can we design a framework for representing raw process repositories which can be easily queried to obtain relevant information related to different entities in the contained process models?*

We will investigate available techniques in the literature that are used to store business process information. Also, we will compare the related works to our approach.

RQ2: *How well do generic state-of-the-art text matching algorithms perform in matching change requirements to relevant business entities represented as text? Is there is a scope of improving their performance using a different method better suited to the problem domain?*

The objective of this question is to evaluate state-of-the-art text matching methods when applied to our ground truth dataset and possibly develop a new approach, if needed, to improve upon these methods when applied to our dataset.

RQ3: *Can we develop better models in the specific domain of matching change requirements to relevant business entities?*

This will help provide a better solution if needed making the job of the business analyst easier by providing better suggestions for matched business entities.

RQ4: *Can we qualitatively determine the impact of a change requirement through the impacted entities obtained in RQ2 or RQ3 by querying the process repository framework defined in RQ1?*

This would help complete our pipeline for semi-automatically determining the qualitative impact of business change requirements on process models. Note that a quantitative impact analysis requires ground truth data based on business expertise in the context of the business process models and is hence, outside the scope of this project.

1.3 Thesis Outline

The remainder of this report is structured as follows. In Chapter 2, the detailed background and preliminary concepts necessary to understand the rest of the report are explained along with the related work. In Chapter 3, the business understanding is explained, wherein we explain what happens in an organisation when there is a new change requirement. We present our overall methodology in Chapter 4. In Chapters 5, 6 and 8 we answer our research questions stated in Section 1.2 respectively. In Chapter 7, we evaluate our system for matching business entities to change requirements using the ground truth data. Finally, in Chapter 9, we conclude our report by summarising our contributions, the limitations of our work, and the possible directions for future work.

Chapter 2

Background

This chapter presents the relevant work, techniques, and concepts needed to understand the rest of the thesis. Section 2.1 introduces essential concepts, methods, and tools relevant to this project. Next, in Section 2.2, we will present the existing approaches, their scope, and limitations and motivate why we require an improved method to solve the research questions we will be addressing.

2.1 Preliminaries

Before describing our approach in detail, this chapter will introduce the concepts, techniques, and technologies that we will use to answer our research questions. The RQ1 can be approached by understanding the definitions of process repository, business process model, and process collection concepts which are discussed in Section 2.1.1 along with knowledge graph technology for both RQ1 and RQ4 discussed in Section 2.1.2. The RQ2 requires the understanding of the existing Natural Language Processing techniques discussed in Section 2.1.3. Lastly, in Section 2.1.4, we will discuss a concept required for text detection in images to help answer RQ4.

2.1.1 Process Repository

A process repository is a database where an organisation can store and manage their business processes in different hierarchy levels. The process repository is an important module of an organisation's Business Process Management (BPM) architecture as shown in Figure 2.1, a hierarchical structure of an organisation's process models. It is a business blueprint that helps the organisational decision-makers to execute the strategy required to solve a

business problem. The process repositories has its life cycle and are required by the organisation to incorporate more business processes without being deployed in their applications, that is, they have the potential to update dynamically without having to change any data or application code.

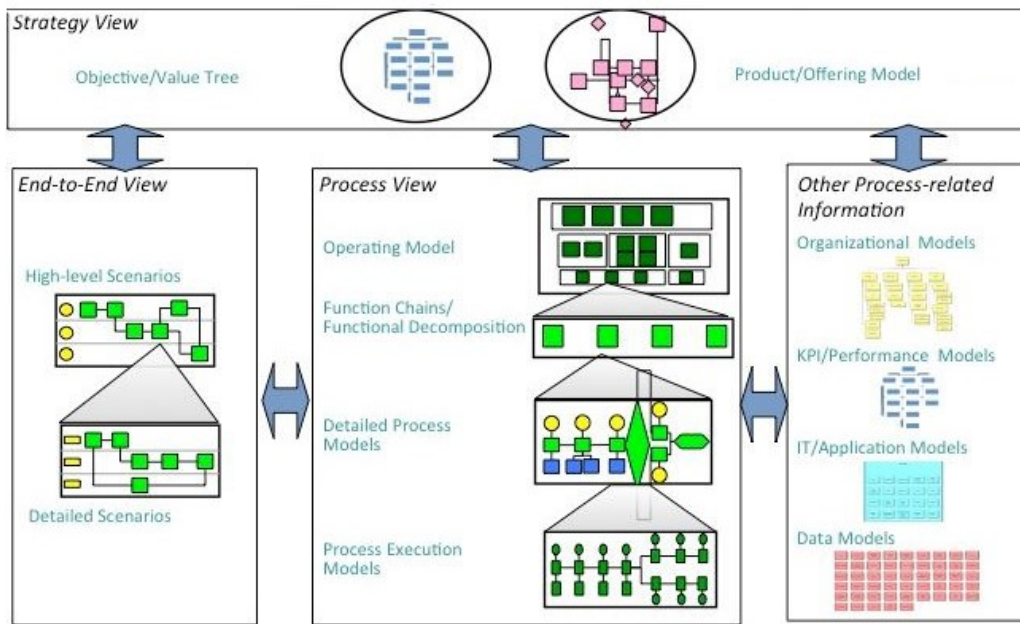


Figure 2.1: Sample Process Repository Architecture. Source: [1]

The following are some benefits of having a process repository [7]:

- Along with process models, process repositories can store their relevant business entities, elements or artifacts that are a set of graphical elements that are required to create models, domain models (a domain that is represented in the form of a conceptual model that incorporates both behaviour and data) and business rules (conditions to consider while making a decision) [8].
- Continuous storing and updating of business processes with a version number.
- A web-based interface to manage, update and deploy business processes without having to change the data or code for the targeted business users and developers.
- Only authorised users would be able to view and/or edit the business processes.

- Synchronisation with software life-cycle development environment.

As stated earlier, a process repository can consist of several business processes. A business process is a sequence of activities or tasks, that are defined by roles, resources, or/and technologies, in which represents a service, application, or product that has a specific business goal for certain customers or users [9] [10] [11]. These business processes can be modeled to visually represent the sequence connection of business objects such as events, tasks, roles, and resources. The purpose is to create visual models to understand and effectively improve the relevant business processes. One such way to model the business processes is by means of the Business Process Diagram or BPMN. It is a graphical notation for creating business process models, similar to sequential flowcharts, that is understood and used by every business oriented organisation as shown in Figure 2.2 [2].

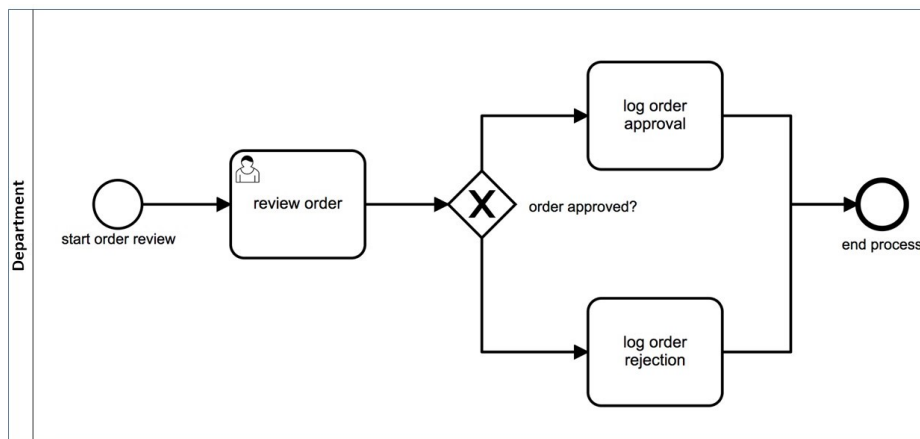


Figure 2.2: Sample BPMN Diagram. Source: [2]

2.1.2 Grakn: Knowledge Graph Database

A knowledge graph (KG) is a representation of collected and interlinked descriptions of data. Figure 2.3 depicts a sample knowledge graph. There are three graph elements that are required to construct a knowledge graph, such as [12]:

- **Entities** distinguish data independently based on categories. For example, an actor or a movie are entities.
- **Attributes** are the fields that have specific information about the entity. For example, the Name of the actor or the movie.

- **Relationships** is the important aspect of a KG where it contributes a link from one to another entity by forming a network that leads to getting the related information. For example, an actor (one entity) acts (relationship) in a movie (another entity).

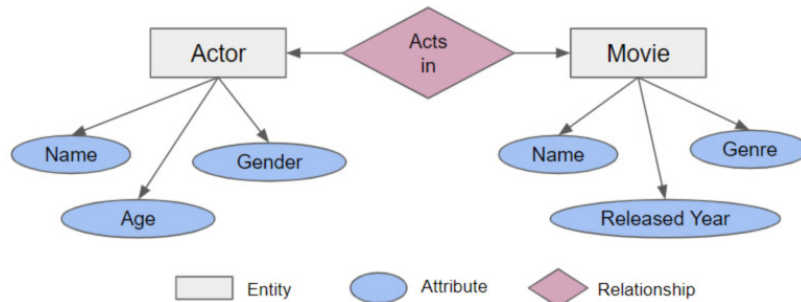


Figure 2.3: Example of a Knowledge Graph

Knowledge graphs combine different characteristics of several data management prototypes, such as:

- **Database:** The data schema can be defined using a data definition language (DDL), changed using a data manipulation language (DML) and queried using a data query language (DQL) [13].
- **Graph:** A graphical structure allows it to be analysed as any other network data structure that has defined relationships [14].
- **Knowledge base:** The stored data has formal semantics, which can be used to interpret the data.

One such KG is Grakn, a database that stores and organises a complex network of data that helps with efficient querying. It allows [15] to model a domain-specific schema using the Entity-Relationship model. A Grakn schema is composed of entity, relationship, and attribute types. Graql is the query language used to perform all types of data definitions, insertion, and manipulation with a Grakn database.

2.1.3 Natural Language Processing: Existing Methods

Natural Language Processing (NLP) is a division of Artificial Intelligence (AI) that uses the natural language to deal with the interaction between

computer programs and humans [16]. NLP's primary purpose is to read, decode, understand, and derive meaning from human languages. One of the NLP applications is to compare texts and determine the similarity between them. This section describes different NLP techniques and algorithms in the context of similarity between texts relevant to understanding the rest of this report.

Word and Sentence Embedding Models

A standard approach to dealing with text data is to convert individual words (or sentences) into a vector of numbers of a fixed size so that these vectors have some semantic correspondence with the meanings of the words (or sentences) to which they correspond [17]. Several different models exist in the literature for this task, including Word2Vec, GloVe, and BERT, as described below.

- **Word2Vec** [18] is a word vectorisation algorithm that takes as input a text corpus and returns as output a vectorised representation of each word in the corpus, each vector having the same size (say N-dimensions). Word2vec embeddings are created by first, training a neural network to predict the context of words in a text corpus. Next, this neural network's hidden layer value for a given word is the embedding vector for that word.
- **GloVe** or Global Vectors for Word Representation [19], developed by Stanford, is another word vectorisation algorithm that uses the word context in a text corpus to obtain the word embeddings by representing the word co-occurrence information as a matrix and minimising the reconstruction loss of a matrix factorisation problem.
- **BERT** or Bidirectional Encoder Representations from Transformers [20] is a state-of-the-art NLP model published by Google AI Language. It performs bidirectional training of Transformer, which uses the word-context to encode text directly into text embeddings (as opposed to converting words into word embeddings in the Word2Vec and GloVe).

2.1.4 Pytesseract : Optical Character Recognition

Optical Character Recognition (OCR) is a program based extraction of data strings from two-dimensional text-based images such as scanned documents as shown in Figure 2.4 [3], soft copy of handwritten letters, a photo with text on billboards, or a subtitle on a video. These extracted strings can be used

to search, edit, store, convert to speech, translate to other languages, and find patterns.

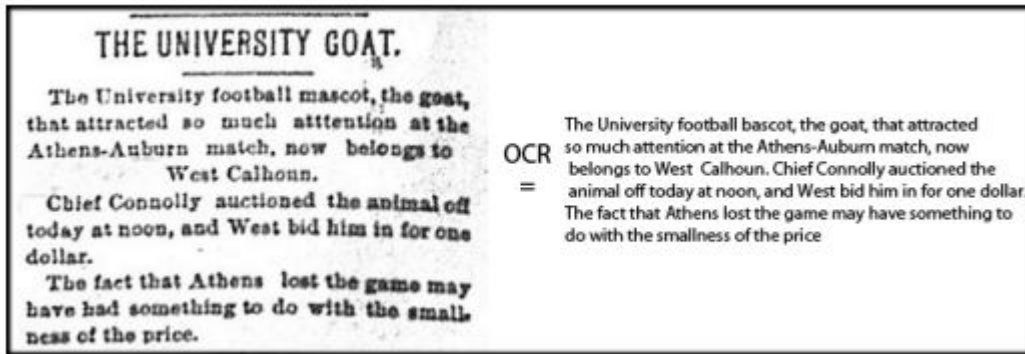


Figure 2.4: Sample OCR: Extracting editable texts from a photo of an old newspaper. Source: [3]

Tesseract is an open-source OCR Engine, accessible under the license of Apache 2.0 [3]. From 2006, Tesseract is sponsored by Google, and it is one of the most accurate known open-source OCR engines. It is compatible with a variety of programming languages and wrapper frameworks. One such Tesseract-OCR Engine wrapper is known as Pytesseract, an open-source OCR library that can be used in Python along with OpenCV (Computer Vision library) [21] and Pillow (Python Imaging Library) [22] which can recognise and read text in images. It can read from all types of images such as jpeg, png, gif, bmp, tiff, etc.

2.2 Related Work

Given the problem context, we present the relevant literature methods that solved a similar problem. First, we have to check how it is solved till the present day. Next, we have to find techniques that are used to solve the problem context. Later, we have to analyse the limitations of the existing methods and their applicability to our problem context.

Before we start with the improved method, it is good to look at previous works. Insights from related works can help a lot in solving the stated research questions. Therefore, we elaborated on a literature study in this section.

The following are the search terms used on Google Scholar, IEEE, and Springer to find relevant research papers required for this thesis:

1. (Compare or match) Process to text (or vice versa)

2. Change impact analysis
3. Process repository storage

Our goal compares and matches a textual change from the requirement document to process models/repository. Therefore we start by doing research on how to compare and match text to models in Section 2.2.1.

Secondly, as a result of this project, we need to determine the impact of change requirements. Therefore we start by doing research on what is known about change impact analysis/assessment in business process management in Section 2.2.2.

Lastly, one of our research questions is to find out how we can represent a process repository to query relevant information. Therefore doing some research in literature from process repository representation and querying can yield insights into what kind of framework we can use, and what kind of querying we can perform. This is done in Section 2.2.3.

As per the contents, we have chosen the following criteria to filter the relevant information:

1. Business-oriented, having a direct practical application
2. Goal of the article must be to find the business process artifact from text and match them against process models

2.2.1 Text to Process Matching

In [23], textual descriptions are compared to the process models using NLP techniques. However, this approach is used only to detect the inconsistencies between texts and process models.

In [24], a novel process model matching approach is presented that depends on the textual descriptions of processes. The main drawback of this model is that it was tested with a textual description generated using Natural Language Generated System (NLGS), which would be far more different from the human-generated text with domain knowledge incorporated in it.

In [25], NLP4BPM is a tool that supports the business organisation to help to maintain consistency between the process representations (texts and models). However, this tool is not openly available.

In [26], a unified format approach is presented to search data in both textual and model-based business descriptions. This approach aims to combine both of the process representations but cannot perform search operations from text to model.

2.2.2 Change Impact Analysis

In [27], a rule-based technique for change impact analysis in software architecture is presented. This technique uses formal semantics on business relations and the change requirements to identify impacted business elements. However, the rules on the “add/delete/update” change types have limitations of identifying the affected entities in real-time and defining these rules are highly complex and different for real-world business.

In [28], an automatic approach is presented to identify the impact of requirements changes on system design by a two-step process: 1. a static slicing algorithm to get the set of impacted process model entities; 2. rank the result set entities according to a quantitative measure designed to predict how likely it is for each entity to be impacted. This seems to be a logical approach to determine the impact, but the impact can only be determined when both the change requirements and business entities are represented in models.

Another approach is proposed in [29]; the impact is estimated by dependency-based analysis in service-based business processes. This approach will only work if a change is between services and their supporting business processes. Additionally, this approach fails to investigate the impact on complex business structures.

Likewise in [30], change impact is estimated by dependency-based analysis for complex process models. Impact business entities are determined by analysing and tracing the dependency relations with other connected business entities. The results presented have two types of information: 1. add/delete/update changes, 2. depth of impact. The intensity of impact estimation could lead to incorrect results for sizeable real-world business models.

[31] is another approach that determines the impact on the business process based on graph-based BPM. By this approach, it is possible to determine the impact on a business process model qualitatively but still uses the rules on the change types like in [27] [30].

2.2.3 Process Repository Framework

In [32], the authors presents a Semantic Business Process Repository (SBPR) for storage and management of semantic business process models with RDBMS with an integrated rule inference engine. This method can be used to achieve automation throughout the BPM life cycle, to store process models, and to query efficiently, but could not find the required business entity in a process model.

In [33], an approach is proposed to represent the business process repository in the object-relational DBMS environment. However, this approach does not store the overall business process model. Still, its subprocess components, i.e., relationships between the processes and their sub-processes, are presented, which could be used only to identify the process's hierarchical structure.

Apart from storing a process repository in an RDBMS, there is a research [34] that stores a process repository in XML, proposed to support the management of business processes. However, this paper fails to test the proposed method in terms of accuracy, confidence, and other possible metrics.

Considering the XML representation of business processes from [34], we realise that they are highly connected and, complex structured. Such data structures are better represented in graphs as per [35]. The framework is a visual query language, especially for querying business processes called BPMN-Q. Still, this framework works only on specific use-cases such as compliance checking, detecting anomalies, and discovering frequent process patterns/anti-patterns. It does not mention querying the business entity from a process model.

Over the years, there have been many developments in the field of graph databases. One such graph database is known as a knowledge graph, and it is, so far, the best way to represent an organisation's process repository [36]. A knowledge graph can store the relationships between the different entities of data comprising the organisation's data. The main advantage of using a knowledge graph for storing a process repository is that a user can query it with the required search input and can obtain results that are semantically and contextually relevant to the organisation's data rather than generic and unnecessary results.

To conclude this section, we have discussed the related work that aims to search the impacted business entity in a process repository and determine the impact for a given textual change. However, each of these works differs from this thesis in some way or another. We observed that there is no complete approach that determines the impact by matching textual change requirements to business entities that are present in a process repository. Hence, we will take inspiration from works related to different components of the system that we will develop in this project from Sections 2.2.1, 2.2.2 and 2.2.3 respectively.

Chapter 3

Business Understanding

Businesses continuously meet various challenges such as new competitors, the need to adapt to new business regulations, keeping up with the changing customer demands, etc. Failure to do so may lead to huge losses or, in some cases, failure in businesses. The management of such changes taking place at an organisational level is referred to as Change Management. The basic flow of change management is depicted in Figure 3.1. It starts by raising a change request, usually from a developer, a client, or a project manager. All these change requests are documented in a Change Requirement Document (CRD).

A CRD is a formal document that contains information that allows a business analyst to assess how a business or technical change impacts a project's existing implementation.

The change assessment is defined by the set of tasks to be performed to analyse the level of impacts on an existing business process. The following are some of the main components presented in CRD for effectively assessing the change requests [37]:

- **The name of the project:** A name that is specific to identify a project, among other projects in an organisation.
- **Request number:** An unique ID that can be referred easily by the employee and helps when getting it to the project log.
- **Requestor:** A client who wants to be satisfied with the project's outcome; a developer who wants the process to be optimised and effectively produces the expected results or a project lead who wants to improve the efficiency of the project.
- **Description of the change:** A clear and concise change description

gives sufficient information allowing the business analyst to understand it at a glance.

- **The reason for the change:** A reasonable precise description that acknowledges the description of the change. Changes happen in every organisation due to client expectations, introducing new technology, etc.

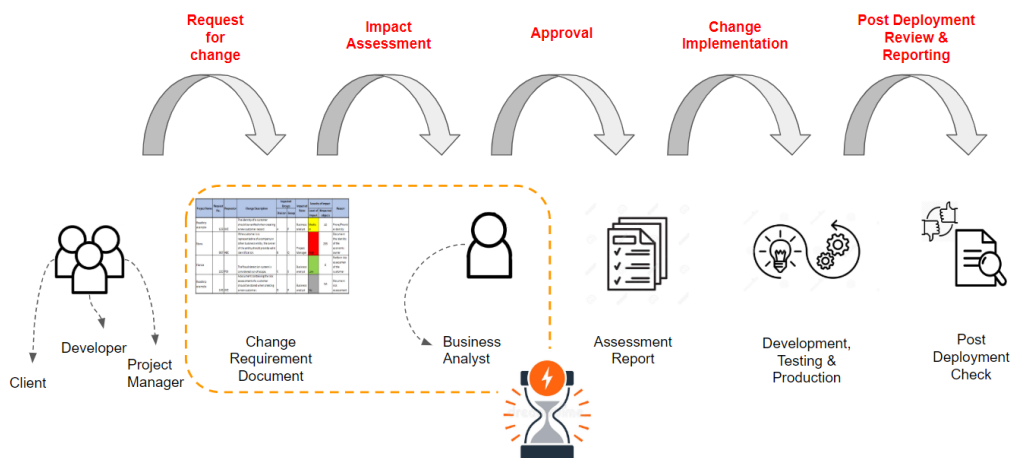


Figure 3.1: Business Process Change Management

Once the CRD is understood, the necessary actions can only be taken after evaluating each change request's level of impact. The impact assessment should be carefully evaluated, as it may lead to a significant loss in an organisation. Keeping the criticality in mind, business analysts spend large amounts of their time manually assessing the several factors in real-world business processes and may yet lead to inefficient results due to human error. This time consuming and ineffective evaluation can be avoided by creating a system that can help business analysts to identify the affected business entities from business processes for the respective change requests. The remaining section of this report describes how such a system can be created.

Further, a change request can be either accepted or rejected depending on its impact level. Accepted change requests are sent to the implementation stage, and checks are done post-deployment to ensure the business complies with the change assessment.

Chapter 4

Methodology

In this Chapter, we present the overall approach we have developed to help answer the research questions stated in Section 1.2. given the problem context (Section 1.1).

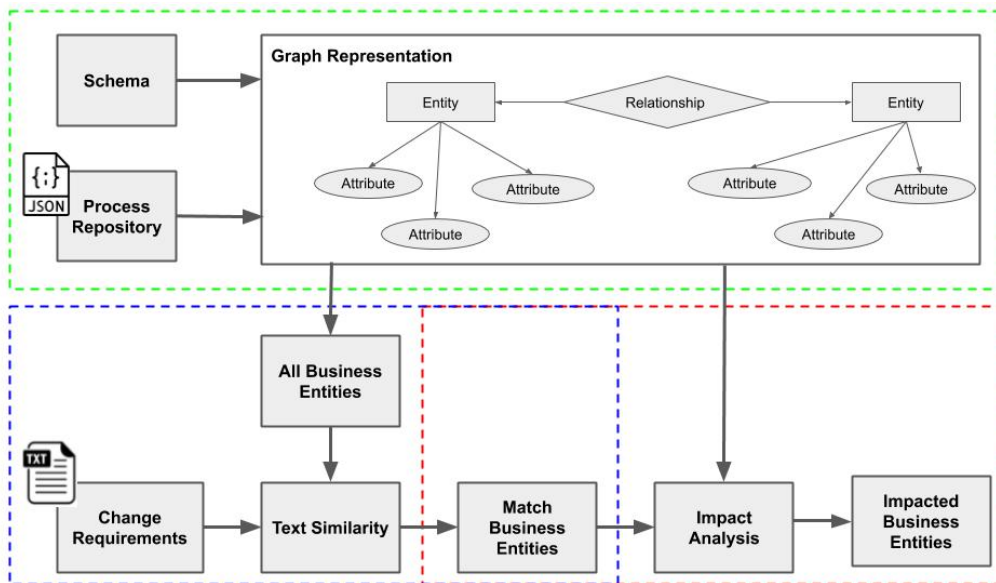


Figure 4.1: Schematic of the Overall Methodology

The overall approach can be divided into 3 parts.

1. We define the schema for our Knowledge Graph database which will be populated using the Process Repository data. This corresponds to the green box in Figure 4.1 and is discussed in detail in Chapter 5.
2. For a given textual change requirement, we will find the text similarity between all the business entities retrieved from the knowledge graph.

From this step, we get the match business entities (entities having the highest text similarity with the change requirement). This corresponds to the **blue** box in Figure 4.1 and is discussed in detail in Chapter 6. The evaluation of different text similarity algorithms is presented in Chapter 7.

3. Finally, the knowledge graph is queried with the matched business entities, in order to determine the impacted business entities. This corresponds to the **red** box in Figure 4.1 and is discussed in detail in Chapter 8.

Chapter 5

Graph Representation of Process Repositories

In this chapter, we address our first Research Question regarding creating a framework for representing, storing, and accessing hierarchical process repository data, including the different business entities it contains. In Section 5.1, we discuss how we obtain the data containing information related to different process models and their respective business entities. A complete understanding of the data is required to store a complete process repository in a knowledge base. In Section 5.2, we will discuss how to interpret the retrieved data considering its hierarchical structure. Lastly, in Section 5.3, we will explain how we created a schema in the Grakn knowledge graph to store the retrieved information.

5.1 Data Collection

This section explains retrieving all the process models and their related objects from a process repository stored on the organisation's server. Figure 5.1 represents the client-server architecture necessary for understanding how to obtain the necessary data in a particular desired format. An authorised **Client**, who is a user can communicate with the server via an **API**. We use a **GET** method for reading the process repository and getting the data from the server. The communication between the client and the server is done using **REST APIs** using **HTTP** requests.

Sending requests to the server using **REST APIs** is similar to providing search requests on the internet. The **web server** reveals itself to a RESTful connection to the world wide web which can be used by external clients and devices to retrieve or manipulate data specific to the process repository. The

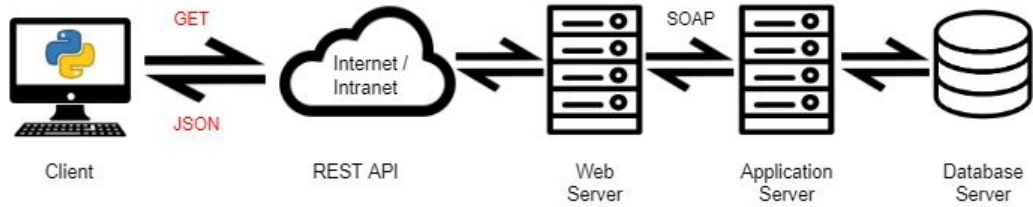


Figure 5.1: Client-Server Architecture for process repository data access

related data is displayed to the client/user in the web browser. To access this related data, a client has to provide the credentials to log into the web client. Later, a service is requested from the web server to the **application server** by SOAP (Simple Object Access Protocol), which is a messaging protocol that exchanges structured data that is stored in the **database server**. Finally, as per the client's preference, the data can be stored locally by the client in JSON or XML format.

5.2 Data Understanding

The hierarchical structure of the process repository which we will work with in this project is depicted in Figure 5.2. The given process repository has seven levels of data hierarchy consisting of business groups (collection of information specific to one business), business models, and business objects. Further, business groups, models, and objects can have sub-groups, sub-models and sub-objects, respectively.

- At the highest level (Level 1) is the process repository. This stores information at the organisation level, e.g. information about the different business clients of the organisation.
- Level 2 (group) contains one or more business projects of the organisation. In our repository, for example, we have the Roadtrip Example group. This group contains the company map diagram (a high-level visual representation of business) of the client to which this group corresponds.
- Level 3 (sub-group) has several departments based on the organisation of the business project. For example, Roadtrip Example is a project with a department named **Documents** that is responsible for collecting and storing customer documents.

- Level 4 (sub-group) includes further sub-divisions of Level 3 sub-groups, if any. For example, for the Process Flow sub-group, we have three sub-divisions, i.e., core, management, and support processes.
- Level 5 (models and sub-models) contain the visual representation of business processes from Levels 3 and 4 in the form of business diagrams, organised into blocks (e.g. Document Model block). Further, sub-model diagrams are also represented by their respective diagrams in this Level. For instance, `Process to create visa` is an example of a model inside the Business Process Diagram block shown in Level 5.
- Level 6 (objects) contains information about the business entities used to create models and sub-models in Level 5. For example, `Check valid ID` activity is a business entity in the `Process to create visa` model.
- Level 7 (sub-objects) has information about the people responsible for the business object. For example, `Organiser` or `Tour guide` is responsible for the task object `Check valid ID`.

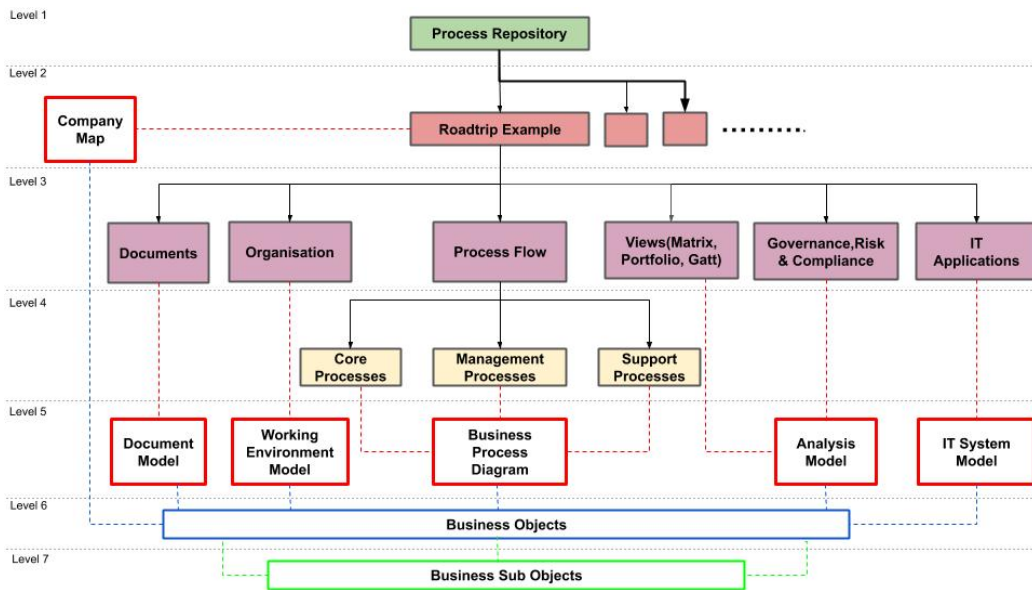


Figure 5.2: Process repository hierarchy

Each level of data is obtained from the server as discussed in Section 5.1 by calling their respective ‘rest_links’ (a unique URL) in the JSON format. The JSON for each level contains the ‘rest_links’ of its lower level. For example, Figure 5.3 shows how each sub-group’s ‘rest_links’ key is available in the higher level group’s JSON file. Hence, we are able

to obtain the entire process repository starting from Level 1 by following these 'rest_links'. Along with 'rest_links', we can also retrieve other attributes such as:

- **ID:** A unique ID that is specific to each business entity. It is important to have an ID since the same object name can be associated with entities in different models.
- **NAME:** This is the name of the business entity.
- **TYPE:** This attribute categorises the business model and objects in their particular type. For example, a model type can be “Use-case diagram” or “BPMN diagram”, whereas an object type can be a Task, Role, etc.
- **DIRECTION:** This attribute contains two possible sub-attributes, INCOMING and OUTGOING, containing the IDs of incoming and outgoing business entities, if any, for a given business entity.

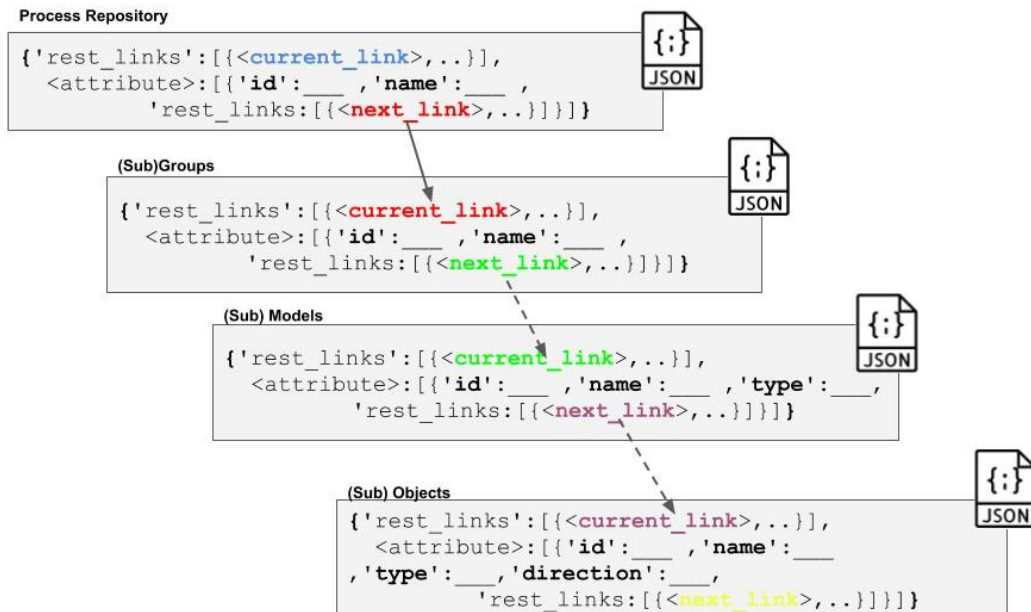


Figure 5.3: Relationship between JSON files

5.3 Implementation

This section describes part of the implementation explaining how a knowledge graph was created and populated to store the entire process repository data. A high-level architecture of the implementation is shown in Figure 5.4.

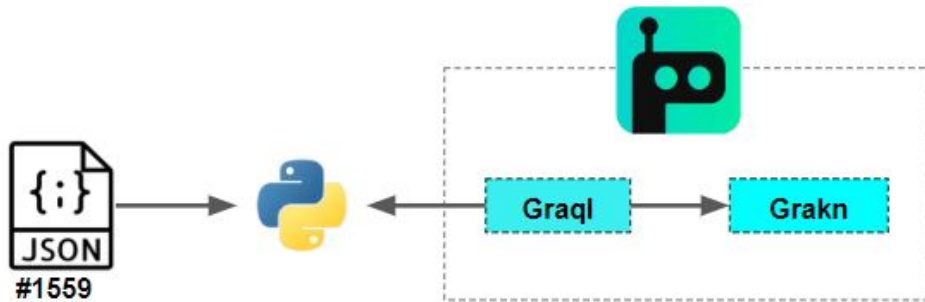


Figure 5.4: High Level Architecture of Grakn knowledge graph creation

The implementation of the knowledge graph representation of the process repository is done using the following technologies:

1. **Grakn:** This is the knowledge graph database used to store the process repository data and to view the created schema in a Grakn workbase.
2. **Graql:** Query language to create the definitions of a knowledge graph schema and insert queries to populate this knowledge graph with the process repository data.
3. **Python:** Programming language used to parse the JSON files and to map the Grakn schema keywords with JSON attributes with Grakn-Client.

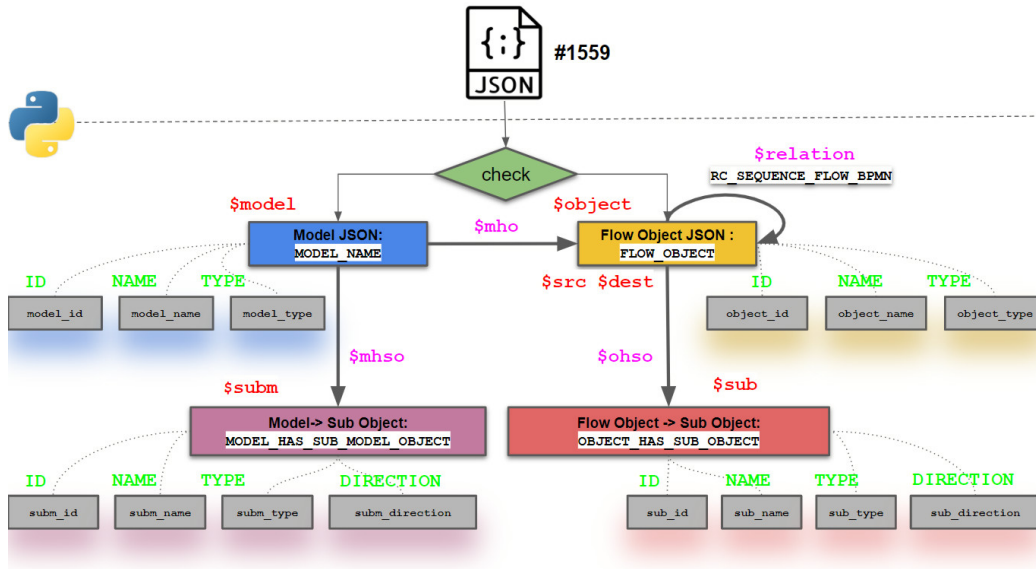


Figure 5.5: JSON objects representation with Grakn variables and attributes

In the Figure 5.5, we represent the 1559 collected JSON files, which are either a Model JSONs or an Object JSONs. Using Python, we have pre-processed all the JSON files to get only the required information that is necessary for our work. The Model JSON contain data related to the models, sub-models, and its objects whereas the Object JSON contain data related to objects and sub-objects. We have defined Graql variables for all the entities (in Figure 5.5, highlighted in red) and relationships (in Figure 5.5, highlighted in pink); and attributes (in Figure 5.5, highlighted in green). Below, we explain how we defined the Grakn database schema and populate it with our process repository data.

5.3.1 Schema Definition in Grakn

The Grakn schema definition is implemented in the following steps.

- Creating Entities First we create four types of Grakn entities using `sub entity`, namely
 - `MODEL_NAME` for models and sub-models
 - `FLOW_OBJECT` for objects
 - `SUB_OBJECT` for sub-objects

- SUB_MODEL_OBJECT for objects belonging to models and sub-models, such as, the output of the process `Process to create visa` (from level 5) is `visa application` or the process owner is `user`

Following is part of Graql schema entity definition showing the `MODEL_NAME` entity definition.

```

1 #ENTITIES
2 MODEL_NAME sub entity,
3     has ID,
4     has NAME,
5     has TYPE,
6     plays MDL,
7     plays MOD;
8     ...

```

- Creating Relations Further, every entity needs to be created with its relevant roles using the `plays` keyword and attributes using the `has` keyword as shown in the `MODEL_NAME` entity definition above. Second, we create four relationships (using `sub relation`) by relating to the entities' roles (using `relates` keyword) namely
 - `MODEL_HAS_OBJECT` relationship between (sub)models and its objects
 - `RC_SEQUENCE_FLOW_BPMN` for preceding and succeeding objects
 - `OBJECT_HAS_SUB_OBJECT` relationship between object and its sub-object information
 - `MODEL_HAS_SUB_MODEL_OBJECT` relationship between (sub)models and its object's information

The following is a part of Graql schema showing the `MODEL_HAS_OBJECT` relationship definition.

```

1 #RELATIONSHIPS
2
3 MODEL_HAS_OBJECT sub relation,
4     relates MDL,
5     relates OBJECT;
6     ...

```

- Adding Attributes to Entities Finally, the attributes of entities are created (using `sub attribute`) along with their `datatype`. The following is a part of Graql schema attribute definition:

```

1 #ATTRIBUTES
2   ID sub attribute,
3     datatype string;
4   ...

```

The complete schema definition is available in Appendix B.

By running the following command in the command prompt, we can load and visualise the created schema in the Grakn workbase tool, as shown in Figure 5.6.

```

1 > grakn server start
2 > grakn console --keyspace process_repo schema --file $(pwd)/
   process_repo_schema.gql

```

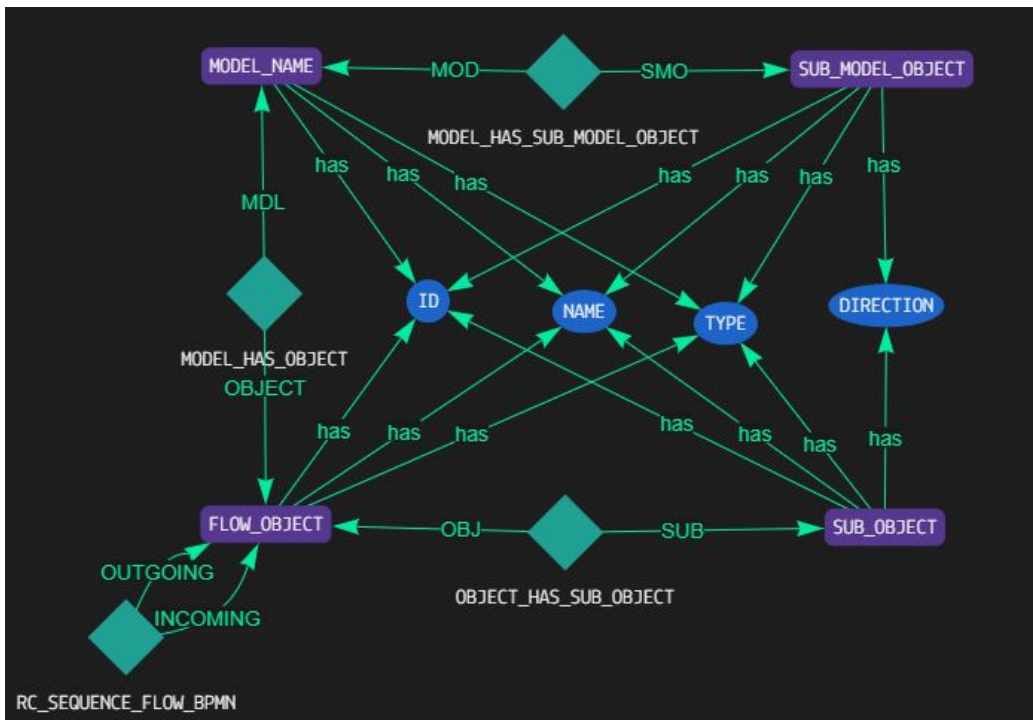


Figure 5.6: Knowledge graph schema

5.3.2 Data Insertion into Grakn

In order to insert the data mined from JSON to Grakn schema using Graql, we need to install Python's `grakn-client` [38] package. The following is the sample code for inserting the data to Grakn. First, we have to insert data

for entities along with their relevant attributes. The complete list of insert queries is provided in Appendix B.

The insert query below in line 3 inserts a model (MODEL_NAME) with a provided ID, NAME and TYPE into the Grakn database.

```

1 #Entities & Attributes insertion into Grakn
2 insert $model isa MODEL_NAME,
3 has ID "{model_id}", has NAME "{model_name}", has TYPE "{
   model_type}";
4 ...

```

The insert query below, on the other hand, inserts the relation between a model (MODEL_NAME) and an object (FLOW_OBJECT) by matching the existing model and object entities using their respective IDs in the database and inserting the MODEL_HAS_OBJECT relationship between them.

```

1 #Relationships insertion into Grakn
2 match $model isa MODEL_NAME, has ID "{model_id}"; $object isa
   FLOW_OBJECT, has ID "{object_id}";
3 insert $mho (MDL: $model, OBJECT: $object) isa
   MODEL_HAS_OBJECT;
4 ...

```

Hence we have inserted 80 different types of business models in Grakn, shown in Table 5.1. Except for the business process diagram, a short description of the other business models are discussed in Appendix A.

Model Type	Number
Analysis Model	37
Business Process Diagram	19
Company Map	9
Document Model	1
IT System Model	2
Working Environment Model	3

Table 5.1: Different types of business models loaded in Grakn

5.4 Summary

In this Chapter, we have described our approach for accessing process repository data stored in a server, creating a Grakn Knowledge Graph schema and inserting the process repository data in it. Next, we will discuss our approach for identifying business entities stored in this database are impacted by given business change requirements.

Chapter 6

Text Similarity

In this chapter, we address our Research Question 2 involving the applicability of Text Semantic Similarity algorithms to the problem of matching change requirements to business objects. We also address Research Question 3 by proposing an algorithmic framework for applying different text similarity (or distance) algorithms tailor-made for text similarity in the context of retrieving relevant business entities from a process repository given an input business change requirement. This corresponds to business change requirements typically in the form of a single sentence and business entities described in the form of a short text (phrases).

6.1 Methodology

In order to be able to compare the different algorithms, we apply the same text-preprocessing function for all algorithms described below for converting change requirement text to a sequence of tokens. This includes punctuation removal, converting to lowercase, tokenisation, stopword removal, and finally lemmatisation.

6.1.1 State-of-the-art in Text Similarity

This section discusses the application of different state-of-the-art word embedding models that we will evaluate later in Chapter 7 for matching business entities to a given change requirement. This section will describe our approach for developing a semantic similarity algorithm using the different state-of-the-art text (or word) embedding models.

1. **Text pre-processing:** There are several steps involved in processing natural language text to obtain relevant information from it in the form

of individual tokens. First, these involve a tokenisation step that removes punctuations in the text and splits text in the form of a sentence (or phrase) into a list of individual words, alternatively referred to as tokens. Next, from this list of tokens, we remove the tokens that are very commonly used (referred to as stopwords) in the language. Examples of stopwords are articles like **the** and prepositions like **of**. This is done because the stopwords usually add additional noise to the sentence embedding being calculated (described in the next step) without adding much value in terms of a semantic understanding of the text’s actual relevant content. Also, all tokens are converted to the lower case. Finally, we can lemmatise individual tokens to normalise different forms of the same root word to its common root form as shown in Figure 6.1.

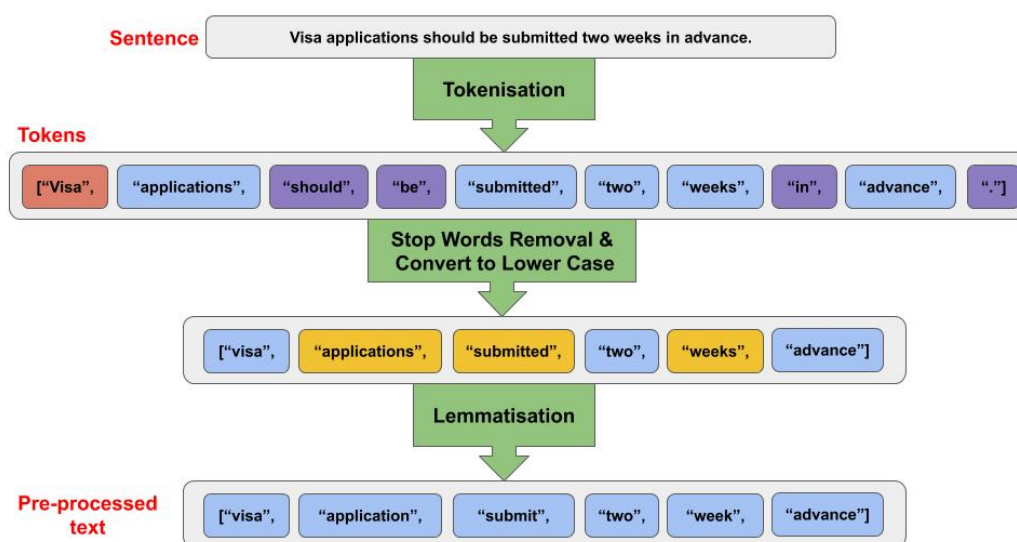


Figure 6.1: Text pre-processing schematic

2. **Generating sentence embeddings:** After pre-processing, an input sentence represented by a sequence of tokens can be converted to a feature vector. For word-embedding models (Word2Vec and GloVe), each token is converted into its corresponding embedding using available pre-trained models. Note that some tokens can be Out-Of-Vocabulary of the word-embedding model. The overall sentence embedding is then calculated as the average of the word-embeddings of the tokens. BERT, a text embedding model, works directly with the sentence inputs to obtain the corresponding sentence embedding. Note that BERT could be

applied to a pre-processed sentence by converting the sequence of pre-processed tokens into a sentence. This is done by merely arranging the tokens in a sequence of words to form a sentence.

3. **Computing Text Similarity:** The text similarity between two texts is calculated as the distance between their sentence embeddings using suitable distance metrics. The Cosine and Euclidean distance between two N-dimensional vectors \mathbf{A} and \mathbf{B} such that, $\mathbf{A} = (a_1, \dots, a_N)$ and $\mathbf{B} = (b_1, \dots, b_N)$ are defined below.

- **Cosine distance:**

$$\text{cosine}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^N a_i b_i}{\sqrt{\sum_{i=1}^N (a_i)^2} \sqrt{\sum_{i=1}^N (b_i)^2}}$$

- **Euclidean distance:**

$$\text{euclidean}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^N (a_i - b_i)^2}$$

6.1.2 Framework for Text Similarity applied to Change Requirement Matching

In this section, we propose a generic framework for semantic similarity from an input text **query** (typically a sentence) to find the ranking of **candidate** texts in decreasing order of similarity or relevance. After performing text pre-processing and tokenisation to a **query** and a **candidate**, we have:

- The set of tokens in the **query**: $Q = \{q_1, \dots, q_{|Q|}\}$
- The set of tokens in the **candidate**: $C = \{c_1, \dots, c_{|A|}\}$

Now, the distance between Q and C is calculated as:

$$\text{distance}(Q, C) = \frac{1}{|C|} \sum_{i=1}^{|C|} \left\{ \min_{j=1}^{|Q|} \{ \text{textdistance}[c_i, q_j] \} \right\}$$

Here, **textdistance** is the placeholder for an algorithm for calculating the distance between a pair of tokens. The formula calculates each token in C , its distance from the nearest token to it in Q using some **textdistance** metric. This distance metric is calculated as the average of these distances. Finally,

given a set of candidates for a given query, the candidates can be ranked in increasing the **distance** metric from the query. The resulting ranking is in order of increasing distance or decreasing similarity/relevance as required.

Further, it is also possible to use word embedding models (i.e. Word2Vec and GloVe) with cosine (or euclidean) distance as the **distance** metric within our framework. These can be compared when used directly as described in Section 6.1.1.

Framework Justification

The explanation for the design of the framework proposed above is justified based on our ground truth data (shown in Figure 6.2).

S. No.	Change Requirements	Ground Truth Business Entity
1	Future vaccinations will be subsidised by the Dutch authorities.	Determine costs and vaccination
2	Credit card expenses should be verified.	Credit card statements and account statements entirely verified?
3	Visa applications should be submitted two weeks in advance.	Apply for visa
4	All participants should carry a first-aid kit.	Prepare first-aid kit
5	Tour guides should warn participants to make a copy of their identification and other important documents.	Make copies of ID cards and travel documents
6	At the end of the trip a questionnaire should be filled in by every participant.	Feedback questionnaire
7	This project is dependent on the new version of the spreadsheet calculator application.	Spreadsheet calculator

Figure 6.2: Sample data of textual change requirements with ground truth business entities

- The framework was designed keeping in mind the ground truth data that we have . It contains change requirements typically in the form of sentence and business entities typically in the form of short text (or phrases).
- The idea of comparing each token in a business entity directly with its nearest token in the change requirement is based on the ground truth data where in most cases, many of the (non-stopword) Ground Truth Business Entity tokens correspond to similar tokens in the change requirement. We also note that the nature of this similarity in many cases is textual as well as semantic. For example, in line 3, “Apply for visa”:
 - **Apply** corresponds to **applications** token in the change requirement.
 - **for** is a stopword

- **visa** corresponds to **visa** token in the change requirement.
- Also, we note that instead of comparing change requirement directly with business entities, it is relevant to compare each token in the business entity with its most similar token in the change requirement. This is because many of the change requirement tokens are irrelevant to the Ground Truth Business Entity and add additional noise to the text similarity calculation if we compare, for instance, the average word vectors of change requirements directly with those of the business entities. For example, in line 3, the (non-stopword) tokens **submitted**, **two**, **weeks**, **advance** are irrelevant as they specify the **change** to be made to the ground truth business entity. Our framework will not consider these tokens for the text similarity calculation in the case of the ground truth entity.
- Finally, it is important to normalise the text similarity score by taking the average of the similarity of each business entity token with its nearest change requirement token.

6.2 Implementation

In this section, we discuss the different Python packages used for implementing the different NLP algorithms and methods described in Section 6.1.

6.2.1 State-of-the-art in Text Similarity

To implement the semantic similarity framework described in Section 6.1.1, we use several NLP packages available for Python for the different tasks.

We use the Spacy Python package [39] for doing stopwords removal and lemmatisation. For Word2Vec, we downloaded the Word2Vec pre-trained model [40] from the Gensim package [41] for loading and accessing the word vectors for different text tokens in Python. For GloVe embeddings, we use the pre-trained embeddings available with the GloVe project [19]. For BERT, we use the pre-trained models from Python’s Sentence Transformer [20] package to encode sentences into sentence embeddings. Finally, we use Python’s Scipy package [42] for calculating Cosine (and Euclidean) distance between sentence vectors.

6.2.2 Framework for Text Similarity applied to Change Requirement Matching

The text similarity framework described in Section 6.1.2 was implemented in Python. We use the same pre-processing steps as above. For the **similarity** function to be used in the framework, we used different types of **textdistance** functions defined in Python's `textdistance` package [43]. These include different categories of **textdistance** functions [44], namely:

- **Simple similarity:** These include simple text comparison algorithms like prefix matching, suffix matching etc.
- **Edit-based similarity:** These algorithms compute metrics based on edit-distance, for example, the number of insert, change or delete operations (on characters) needed to convert one text to another.
- **Phonetic based similarity:** These algorithms define methods and rules to compute distance based on the sound created when texts are spoken verbally.
- **Token-based similarity:** These algorithms compute metrics based on the similarity of sets representing two texts, e.g. set of characters in texts, using set operations (union, intersection, difference).
- **Sequence-based similarity:** These algorithms use metrics based on sequences of characters obtained from the two texts, e.g. the longest common sub-sequence of characters.

6.3 Summary

In this chapter, we have discussed two different approaches for text similarity. The first is based on applying state-of-the-art text embedding methods on the pre-processed text. The second introduces a simple framework particularly in the context of matching change requirements to business entities (both in the form of text) within which different **textdistance** or **word embedding** algorithms can be used. Also, we provide the intuition behind creating this framework in the context of our ground truth data. A detailed evaluation of these two approaches using different performance metrics on ground truth data is provided in Chapter 7.

Chapter 7

Text Similarity Evaluation

In this chapter, we present the evaluation of different state-of-the-art NLP methods used in the semantic similarity framework we describe in Section 6.1.1, namely Word2Vec, GloVe, and BERT applied to our ground truth data. Further, we apply this evaluation strategy to different text-similarity algorithms available with Python’s **textdistance** package, in the framework of text similarity matching applied to match change requirements to business entities, as described in Section 6.1.2.

7.1 Objective

We want to calculate the top-N accuracy and the top-N NDCG scores for the different algorithms described above for different values of N. We want to use these scores to compare the performance of state-of-the-art text vectorisation algorithms with that of different **textdistance** algorithms when implemented in our text-similarity framework.

7.2 Setup

We use a ground truth dataset consisting of 20 change requirements applied to a given process repository consisting of 243 business entities. For a given algorithm, corresponding to each change requirement, we rank the 243 business entities in decreasing order of their similarity score with the textual change requirement. We call this ranking the **retrieved ranking**. Figure 7.1 shows 3 change requirements in our dataset with their corresponding ground truth business entity and the top 3 retrieved ranks (using a **textdistance** algorithm in our text similarity framework).

Change Requirements	Ground Truth Business Entity	Retrieved Ranking		
		1	2	3
Visa applications should be submitted two weeks in advance.	Apply for visa	Visa application	Apply for visa	Requirements - Applications
All participants should carry a first-aid kit.	Prepare first-aid kit	Prepare first-aid kit	Participant candidate	Participants defined
All incidents with respect to the service continuity should be logged and categorised.	Categorise incident	Incident	Log incident	Is it a Service Request?

Figure 7.1: Textual change requirements with the Ground Truth and Retrieved top 3 Ranking

The Median Rank, Accuracy, and Normalised Discounted Cumulative Gain (NDCG) metrics for a given algorithm on the ground truth data are calculated as follows.

- **Median Rank:** For an input change requirement, the **Rank** is defined as the position of ground truth entity in the **retrieved ranking**. For example, for the first change requirement in Figure 7.1, the **Rank** corresponding to the retrieved ranking is 2. Clearly, the lower the **Rank**, the better the **retrieved ranking**. The **Median Rank** is the median of the **Ranks** obtained corresponding to the list of (20) change requirements in our ground truth dataset.
- **Accuracy:** Given the value of N, we consider the **retrieved ranking** as **correct** if it contains the ground truth business entity in one of the top N items. The accuracy of an algorithm is calculated as the fraction of change requirements for which its output is **correct**.
- **Normalised Discounted Cumulative Gain (NDCG):** Accuracy (at N) only considers if the **Rank** of a **retrieved ranking** is at most N or not. It does not consider the actual **Rank** of the ground truth in our ranking if it is at most N. The Normalised Discounted Cumulative Gain (NDCG) is a more refined metric for evaluating recommendations by scoring the **Rank** itself as described below.

- For the list of 243 business entities, we assign a **relevance** score of 1 to the matching ground truth entity and 0 to all the other entities.
- The Discounted Cumulative Gain (DCG) at N for a ranking is defined as:

$$DCG@N = \sum_{i=1}^N \frac{relevance_i}{\log_2(i+1)}$$

- The NDCG at N is calculated as the ratio of the DCG of the **retrieved ranking** ($DCG_R@N$) to that of the **ideal ranking** ($DCG_I@N$), where the **ideal ranking** arranges the entities in decreasing order of relevance. Note that in our case, we do not have a unique **ideal ranking**. However, any **ideal ranking** has a **relevance** of 1 for the first entity and 0 for all others based on our definition of **relevance**. Suppose that for a **retrieved ranking**, the ground truth entity has **Rank=Rank**.

Using the DCG@N formula, we obtain:

$$DCG_R@N = \begin{cases} \frac{1}{\log_2(Rank+1)}, & \text{if } Rank \leq N \\ 0, & \text{otherwise} \end{cases}$$

$$DCG_I@N = \frac{1}{\log_2(1+1)} + 0 = 1$$

- Finally, NDCG for a given Rank corresponding to a **retrieved ranking** is calculated as:

$$NDCG@N = \frac{DCG_R@N}{DCG_I@N} = \begin{cases} \frac{1}{\log_2(Rank+1)}, & \text{if } Rank \leq N \\ 0, & \text{otherwise} \end{cases}$$

Note that the maximum value of NDCG is 1 if Rank=1, and its minimum value is 0 for Rank > N. The average NDCG score (also referred to as NDCG in short) is the average of the NDCG scores corresponding to the 20 change requirements.

7.3 Execution

In this section, we describe the different algorithms which we implemented. Firstly, for each state-of-the-art text embedding model (Word2Vec, GloVe, and BERT), we evaluate 4 different variants described below.

- Using Cosine distance for embedding distance calculation, and with text pre-processing.
- Using Cosine distance for embedding distance calculation, and without text pre-processing.
- Using Euclidean distance for embedding distance calculation, and with text pre-processing.
- Using Euclidean distance for embedding distance calculation, and without text pre-processing.

7.4 Results

In this section, we present the results for the different text embedding algorithms and different **textdistance** algorithms in our text similarity framework using the evaluation metrics described above. Further, we compare and analyse the results obtained from these approaches.

7.4.1 State-of-the-art models

Figures 7.2, 7.3 and 7.4 compare the Accuracy and NDCG scores for the 4 variants of Word2Vec, GloVe and BERT respectively. On the horizontal axis in each plot, the value of N is varied. From these Figures, we make the following observations:

- In all cases, it is clear that all algorithms perform much better on pre-processed data compared to non-preprocessed data.
- Cosine distance is better suited for Word2Vec and GloVe models compared to Euclidean distance. For BERT, Cosine distance and Euclidean distance give similar performance in terms of Accuracy and NDCG.

Hence, for each algorithm, we will discuss only the results for pre-processed text and using Cosine distance compared with **textdistance** algorithms in our Text Similarity Framework.

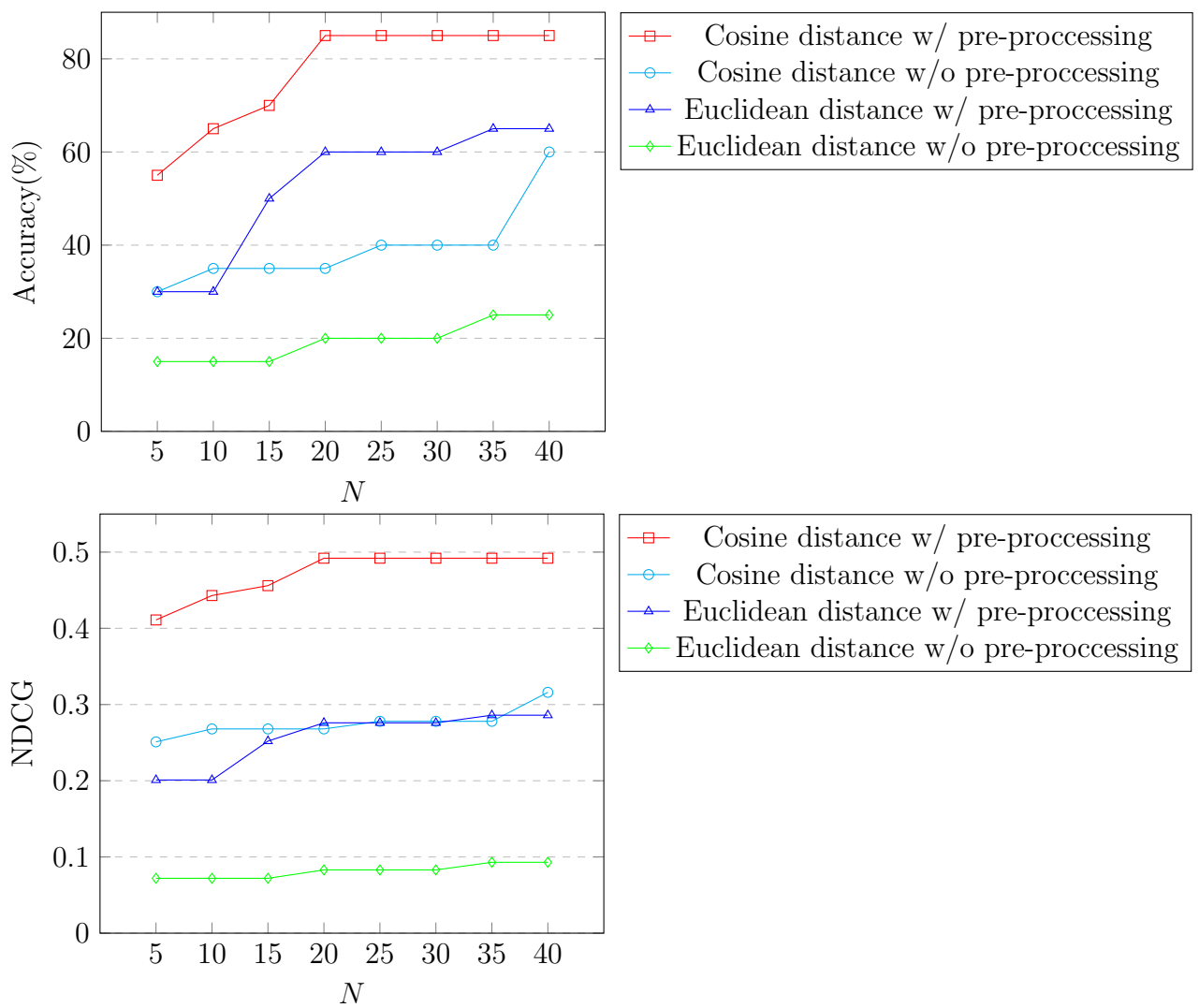


Figure 7.2: Word2Vec - Accuracy and NDCG

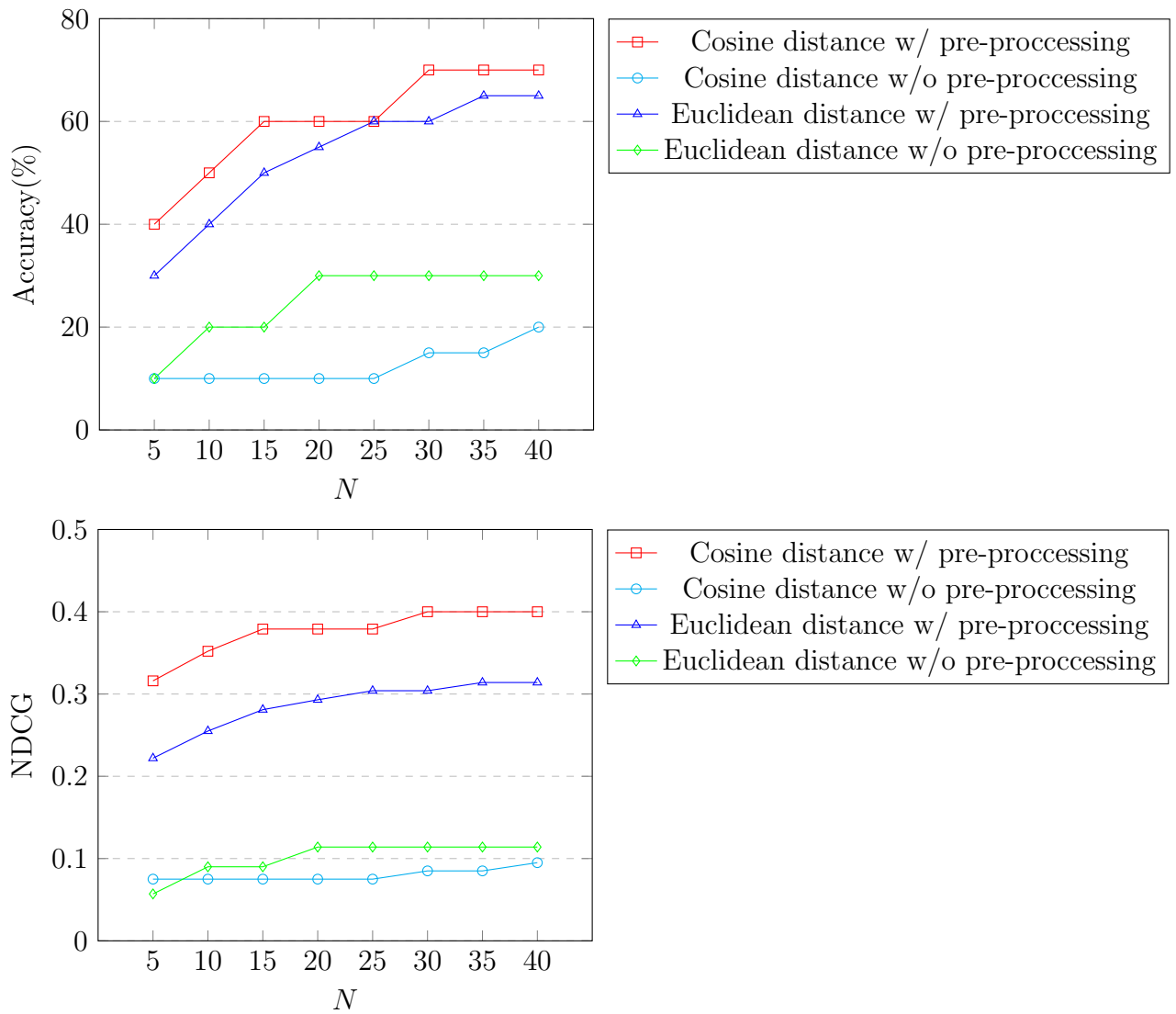


Figure 7.3: GloVe - Accuracy and NDCG

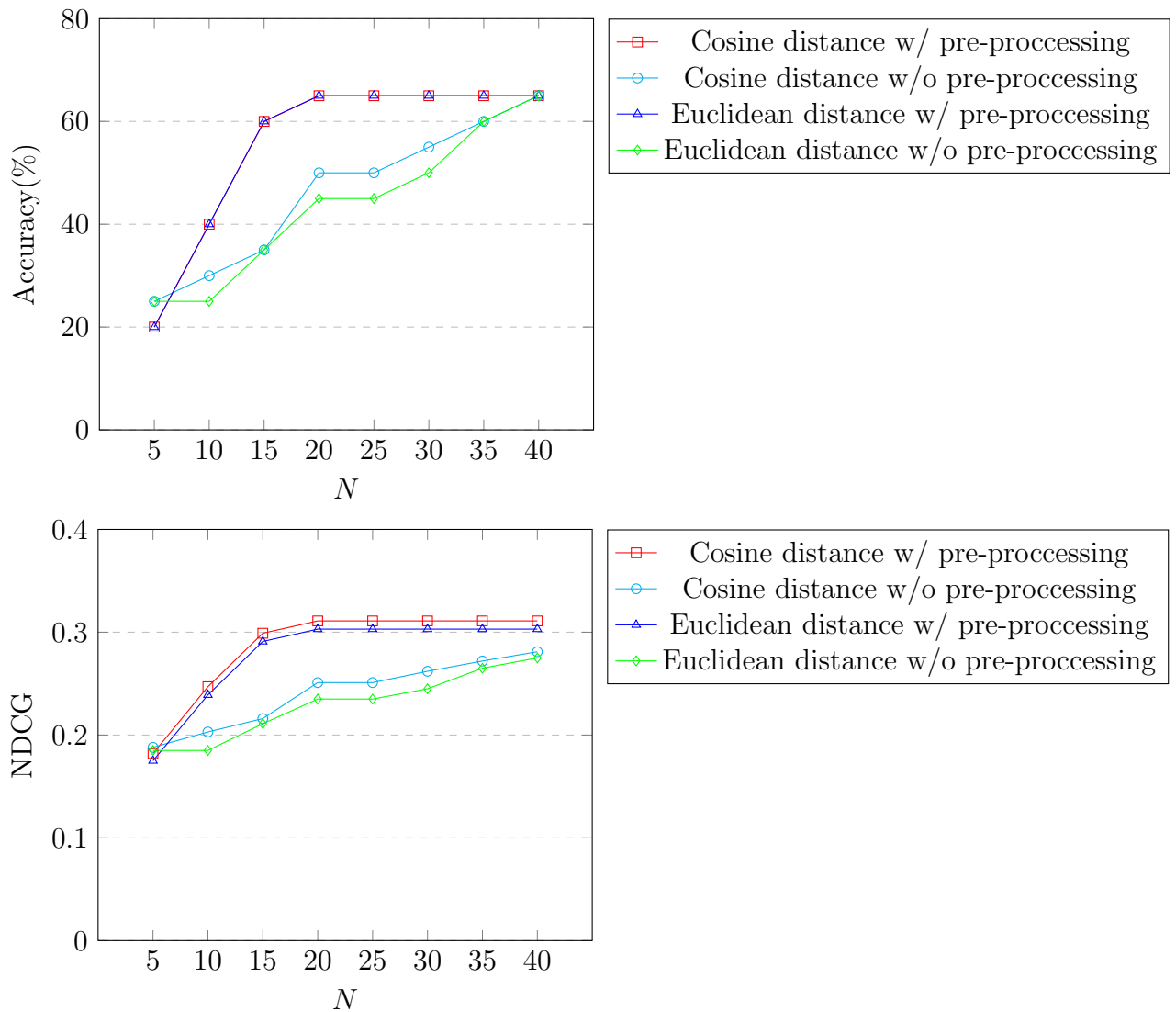


Figure 7.4: BERT - Accuracy and NDCG

Table 7.1 shows the values of Accuracy and NDCG corresponding to the different algorithms, each with Cosine and Euclidean distance, and with and without pre-processing. As **highlighted** in Table 7.1, we note that Accuracy and NDCG scores for each algorithm are highest when using Cosine distance with pre-processed text data (except for Top 5 Accuracy and Top 5 NDCG for BERT, marked by *).

Further, Table 7.2 shows the Ranks corresponding to the 20 different change requirements for these algorithms.

Algorithm	Distance	Preprocess	Accuracy(%) (Top N)							NDCG (Top N)								
			5	10	15	20	25	30	35	40	5	10	15	20	25	30	35	40
GloVe	Cosine	True	40	50	60	60	60	70	70	70	0.316	0.352	0.379	0.379	0.379	0.4	0.4	0.4
		False	10	10	10	10	10	15	15	20	0.075	0.075	0.075	0.075	0.075	0.085	0.085	0.095
	Euclidean	True	30	40	50	55	60	60	65	65	0.222	0.255	0.281	0.293	0.304	0.304	0.314	0.314
		False	10	20	20	30	30	30	30	30	0.057	0.09	0.09	0.114	0.114	0.114	0.114	0.114
Word2Vec	Cosine	True	55	65	70	85	85	85	85	85	0.411	0.443	0.456	0.492	0.492	0.492	0.492	0.492
		False	30	35	35	35	40	40	40	60	0.251	0.268	0.268	0.268	0.278	0.278	0.278	0.316
	Euclidean	True	30	30	50	60	60	60	65	65	0.201	0.201	0.252	0.276	0.276	0.276	0.286	0.286
		False	15	15	15	20	20	20	25	25	0.072	0.072	0.072	0.083	0.083	0.083	0.093	0.093
BERT	Cosine	True	20	40	60	65	65	65	65	65	0.182	0.247	0.299	0.311	0.311	0.311	0.311	0.311
		False	25*	30	35	50	50	55	60	65	0.188*	0.203	0.216	0.251	0.251	0.262	0.272	0.281
	Euclidean	True	20	40	60	f65	65	f65	65	f65	0.175	0.239	0.291	0.303	0.303	0.303	0.303	0.303
		False	25*	25	35	45	45	50	60	65	0.185	0.185	0.211	0.235	0.235	0.245	0.265	0.275

Table 7.1: Results from NLP State of the Art Methods

Method	Distance	Processed	Ranks for 20 Change Requirements																			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GloVe	Cosine	True	11	4	28	6	142	151	144	27	1	1	2	145	66	2	2	12	1	1	60	6
		False	89	1	74	81	179	70	136	167	143	139	96	127	222	44	108	3	28	40	265	261
	Euclidean	True	15	4	25	9	75	34	98	6	1	16	5	201	133	13	2	128	1	1	210	208
		False	257	2	7	103	159	76	232	71	95	16	102	173	220	19	3	41	149	7	258	264
Word2Vec	Cosine	True	18	2	16	6	49	12	18	10	1	1	2	226	46	5	3	4	1	1	2	1
		False	66	107	5	1	85	2	124	135	39	1	38	216	174	40	7	1	25	39	79	1
	Euclidean	True	32	2	15	18	122	17	257	13	3	14	5	165	128	13	3	165	1	1	252	206
		False	258	3	18	66	179	98	256	71	73	3	62	167	234	124	4	46	222	33	259	254
BERT	Cosine	True	64	1	27	56	11	9	45	1	5	3	23	138	223	1	4	3	1	1	41	1
		False	22	1	108	32	128	92	86	7	3	26	18	194	214	1	5	7	1	4	23	1
	Euclidean	True	63	1	31	57	11	9	38	1	4	6	31	154	208	1	4	4	1	1	35	1
		False	19	1	101	33	130	91	70	5	3	46	25	200	217	1	5	7	1	4	22	1

Table 7.2: Ranks generated for each Change Requirements by NLP State of the Art Methods

7.4.2 Text-distance algorithms in our Similarity Framework

Table 7.3 shows the results for the different **textdistance** algorithms (grouped by the algorithm category) applied in our Text Similarity framework. Also, Table 7.4 shows the Ranks corresponding to the 20 different change requirements for these algorithms.

We choose the best performing algorithms for each category based on the Top 5 NDCG scores (marked with *), since we are interested in showing the Top 5 results to the business analyst to choose from in our semi-automatic approach and because NDCG is a more refined metric for evaluating recommendations than Accuracy.

These algorithms are briefly explained below:

- **Simple Prefix:** This algorithm computes text similarity by computing the similarity between prefixes of different lengths in texts.
- **Phonetic MRA:** This algorithm compares two texts by using a set of encoding and comparison rules corresponding to the model to compare two text phonetics [45].
- **Edit-based Jaro:** The Jaro distance metric is an edit-distance based metric between two strings [46].
- **Sequence-based Longest common substring (lcsstr):** This algorithm uses the longest common substring similarity length as a measure for calculating similarity between texts.
- **Token-based Tversky:** This algorithm computes text distance using the Tversky index [47] between two texts by comparing the set of characters they contain.

We will now compare the performance of these algorithms with the text-embedding models.

Category	Algorithm	Accuracy(%) (Top N)								NDCG (Top N)							
		5	10	15	20	25	30	35	40	5	10	15	20	25	30	35	40
Edit based	Damerau-Levenshtein	60	70	80	85	85	85	85	85	0.476	0.507	0.535	0.547	0.547	0.547	0.547	0.547
	Gotoh	55	75	75	80	80	85	85	85	0.429	0.499	0.499	0.511	0.511	0.521	0.521	0.521
	Hamming	70	75	80	85	85	85	85	85	0.466	0.481	0.494	0.506	0.506	0.506	0.506	0.506
	Jaro	65	80	85	85	85	85	85	85	0.506*	0.558	0.572	0.572	0.572	0.572	0.572	0.572
	Jaro Winkler	65	80	85	85	85	85	85	90	0.488	0.539	0.552	0.552	0.552	0.552	0.552	0.562
	Levenshtein	60	80	85	85	85	85	85	90	0.444	0.508	0.521	0.521	0.521	0.521	0.521	0.531
	MLIPNS	45	65	70	80	80	85	85	85	0.341	0.411	0.425	0.449	0.449	0.459	0.459	0.459
	Needleman-Wunsch	60	80	80	80	85	85	85	85	0.455	0.519	0.519	0.519	0.53	0.53	0.53	0.53
	Smith-Waterman	50	70	70	80	85	85	85	85	0.385	0.449	0.449	0.472	0.483	0.483	0.483	0.483
	Strcmp95	70	80	85	85	85	85	85	85	0.501	0.534	0.548	0.548	0.548	0.548	0.548	0.548
Phonetic	Editex	55	75	80	80	80	80	85	85	0.422	0.49	0.503	0.503	0.503	0.503	0.513	0.513
	MRA	60	75	75	75	75	80	80	80	0.461*	0.511	0.511	0.511	0.511	0.521	0.521	0.521
Sequence based	Longest common subsequence	55	75	80	80	85	85	85	85	0.447	0.515	0.528	0.528	0.539	0.539	0.539	0.539
	Longest common substring	70	80	85	85	85	85	85	85	0.485*	0.519	0.533	0.533	0.533	0.533	0.533	0.533
	Ratcliff-Obershelp	60	75	80	85	85	85	85	85	0.469	0.517	0.53	0.542	0.542	0.542	0.542	0.542

Simple	Identity	50	70	75	75	75	75	75	75	0.416	0.482	0.495	0.495	0.495	0.495	0.495	0.495
	Length	0	15	25	35	35	40	50	55	0	0.05	0.075	0.099	0.099	0.109	0.128	0.138
	Matrix	50	70	75	75	75	75	75	75	0.416	0.482	0.495	0.495	0.495	0.495	0.495	0.495
	Postfix	45	70	75	75	75	75	75	75	0.345	0.428	0.441	0.441	0.441	0.441	0.441	0.441
	Prefix	70	70	80	85	85	85	85	85	0.535*	0.535	0.561	0.574	0.574	0.574	0.574	0.574
Token based	Bag	50	65	70	70	70	75	75	75	0.416	0.466	0.479	0.479	0.479	0.49	0.49	0.49
	Cosine	45	70	75	85	85	85	85	85	0.378	0.46	0.473	0.497	0.497	0.497	0.497	0.497
	Jaccard	55	75	85	85	85	85	85	85	0.446	0.513	0.538	0.538	0.538	0.538	0.538	0.538
	Monge_Elkan	55	70	75	80	80	80	80	80	0.407	0.452	0.465	0.477	0.477	0.477	0.477	0.477
	Overlap	40	55	65	65	70	75	90	90	0.288	0.335	0.361	0.361	0.372	0.382	0.412	0.412
	Sørensen–Dice	45	65	70	75	75	75	80	85	0.372	0.441	0.453	0.465	0.465	0.465	0.475	0.484
	Tanimoto distance	0	0	0	0	0	10	15	15	0	0	0	0	0	0.02	0.03	0.03
Tversky	55	75	85	85	85	85	85	85	0.446*	0.513	0.538	0.538	0.538	0.538	0.538	0.538	

Table 7.3: Results for **textdistance** algorithms in our text similarity framework. The best performing algorithm for each category using the top 5 NDCG has been **highlighted** and the corresponding NDCG score marker with an *****.

Category	Algorithm	Ranks for 20 Change Requirements																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Edit based	Damerau Levenshtein	9	11	13	1	17	12	1	4	2	1	2	59	100	4	1	5	1	7	1	1
	Gotoh	16	6	19	1	30	6	1	5	3	1	2	69	148	4	2	6	1	7	1	1
	Hamming	13	10	57	1	18	4	1	4	3	1	2	44	204	5	4	2	1	5	3	1
	Jaro	11	7	110	1	3	6	1	6	3	1	1	46	217	4	1	2	1	4	2	1
	Jaro Winkler	12	6	114	1	4	7	1	7	2	1	3	40	218	4	1	2	1	3	2	1
	Levenshtein	12	9	186	1	10	7	1	4	3	1	2	38	170	4	3	5	1	6	1	1
	MLIPNS	30	17	103	1	17	6	2	6	1	2	6	212	234	2	7	4	1	3	11	1
	Needleman Wunsch	10	5	186	1	21	9	1	4	2	1	2	81	149	5	2	6	1	7	1	1
	Smith Waterman	18	7	125	1	19	2	1	4	3	1	2	191	114	9	3	10	1	21	6	1
	Strcmp95	11	6	120	1	5	10	1	5	4	1	2	52	208	4	1	2	1	3	2	1
Phonetic	Editex	7	9	76	1	32	12	1	5	3	1	2	52	65	3	4	6	1	6	1	1
	MRA	2	10	194	1	28	55	1	6	6	1	2	65	140	5	3	1	1	4	2	1
Sequence based	Longest common subsequence	9	5	183	1	14	21	1	6	3	1	2	82	236	4	1	6	1	7	1	1
	Longest common substring	7	5	212	1	12	2	1	6	4	1	4	42	218	4	3	3	1	5	1	1
	Ratcliff-Obershelp	16	4	199	1	9	15	1	7	3	1	2	64	241	5	1	4	1	7	1	1

Simple	Identity	6	15	101	1	7	2	121	8	8	1	1	198	234	79	2	2	1	4	1	1
	Length	87	6	161	17	251	252	63	35	30	13	33	36	221	19	97	7	9	185	197	15
	Matrix	6	15	101	1	7	2	121	8	8	1	1	198	234	79	2	2	1	4	1	1
	Postfix	7	7	186	1	7	3	89	9	12	1	2	203	90	85	2	3	1	6	2	1
	Prefix	16	11	182	1	14	1	1	3	3	1	3	45	219	2	4	2	1	3	1	1
Token based	Bag	8	4	158	1	55	49	1	6	3	1	1	207	95	13	1	7	1	26	5	1
	Cosine	9	6	77	1	12	19	1	6	4	1	2	215	103	10	1	3	1	18	7	1
	Jaccard	9	6	91	1	14	8	1	6	4	1	1	208	94	4	1	4	1	14	2	1
	Monge_Elkan	3	14	86	1	9	3	17	9	10	1	1	92	226	80	2	3	1	5	2	1
	Overlap	31	30	23	1	3	3	1	15	12	2	7	244	177	9	9	3	2	32	33	1
	Sørensen	9	6	90	1	37	33	1	6	4	1	2	204	94	15	1	5	1	20	6	1
	-Dice	9	6	90	1	37	33	1	6	4	1	2	204	94	15	1	5	1	20	6	1
	Tanimoto	117	34	85	78	165	166	110	59	164	167	111	181	224	72	255	29	28	256	189	236
Tversky	9	6	91	1	14	8	1	6	4	1	1	208	94	4	1	4	1	14	2	1	

Table 7.4: Ranks generated for each Change Requirements by **textdistance** algorithms in our text similarity framework.

7.4.3 Comparing our framework with the State-of-the-Art

The different algorithms for which results are presented in this section are all applied to the same pre-processed text to make a fair comparison of the results.

Firstly, Table 7.5 shows the median rank of all the selected **textdistance** algorithms in our framework. Also, it shows the use of word embedding models (Word2Vec and GloVe) with Cosine distance metric in our text similarity framework. We note that:

- The **textdistance** algorithms used in our framework have lower (hence better) median rank compared to all text embedding methods.
- Using text embedding algorithms in our framework improves the performance in terms of median rank for both Word2Vec and GloVe.

Method	Algorithm	Median Rank
Text Embedding	GloVe	8.5
	Word2Vec	4.5
	BERT	12.5
Text-distance + Our Framework	Simple Prefix	3
	Phonetic MRA	3.5
	Edit-based Jaro	3
	Sequence-based lcsstr	4
	Token-based tversky	4
Text Embedding + Our Framework	Word2Vec	3
	GloVe	3

Table 7.5: Median Ranks for the different algorithms

Below, we will compare the three different **Methods** shown in Table 7.5 as follows.

- **Text Embedding algorithms VS Text-distance algorithms + Our Framework:** In Figure 7.5 we note that all Text-distance algorithms in our framework out-perform the Text Embedding algorithms, particularly for the NDCG metric. In particular, we note that a simple distance metric like the Simple Prefix algorithm applied in our framework is superior to all Text Embedding algorithms.

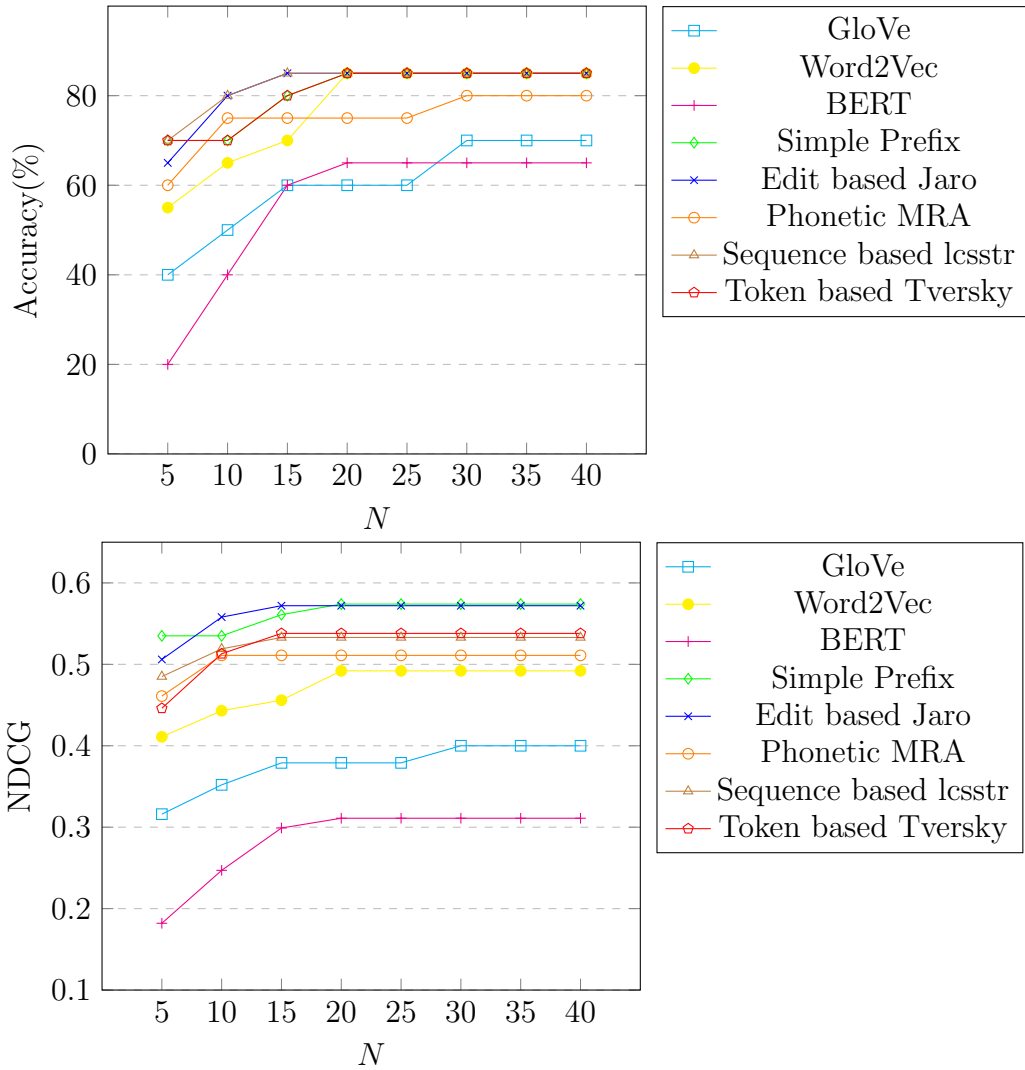


Figure 7.5: Comparing Text Embedding algorithms to selected `textdistance` algorithms in our framework using Accuracy and NDCG metrics

- Text Embedding algorithms VS Text Embedding algorithms + Our Framework:** In Figure 7.6 we note the improvement in performance (particularly for NDCG) of Text Embedding algorithms (Word2Vec and GloVe) when used with our framework. This proves the usefulness of our framework irrespective of the `distance` algorithm used with it. This is because of the nature of the ground truth data as discussed in Section 6.1.2.

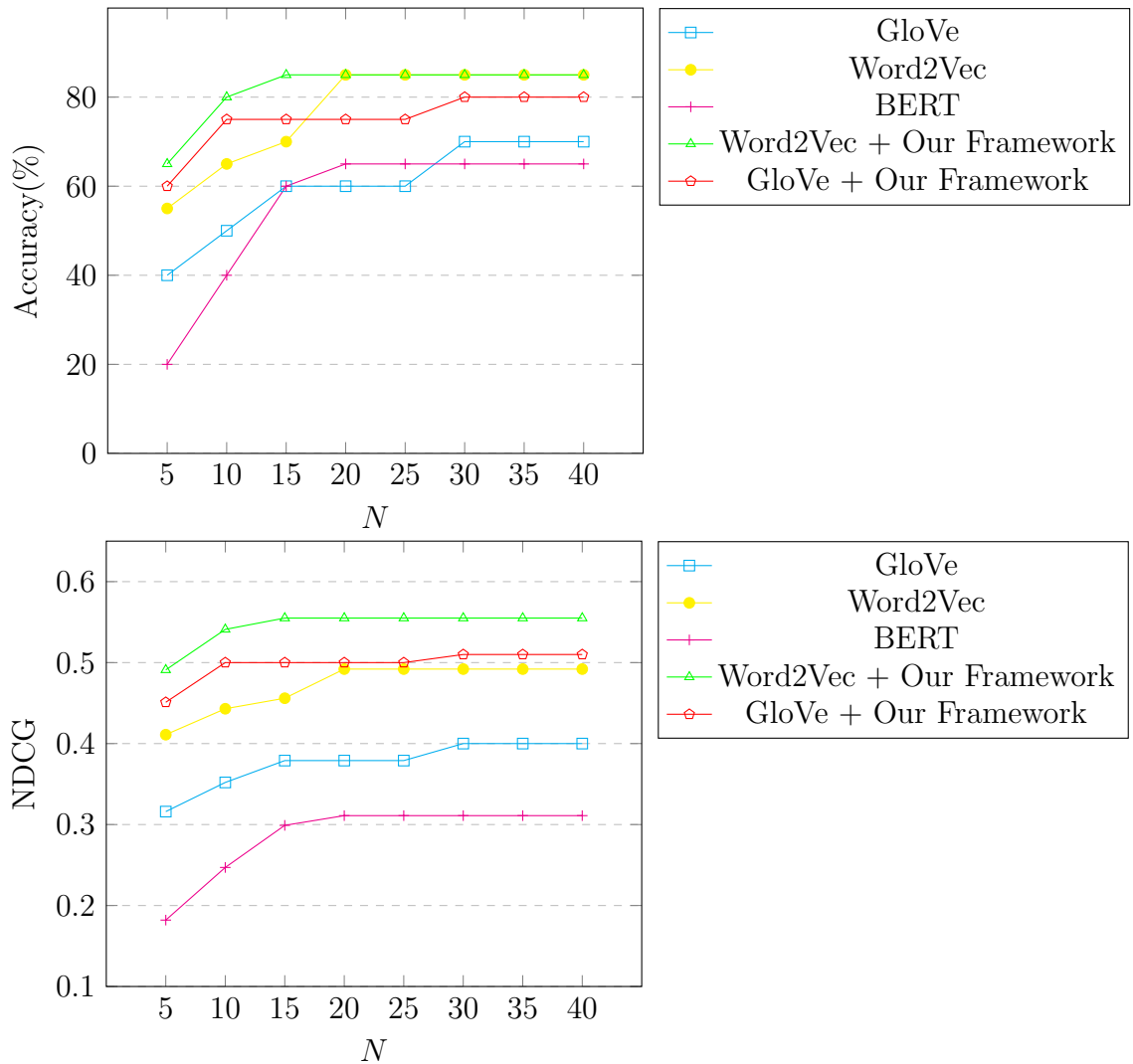


Figure 7.6: Comparing Text Embedding algorithms with and without our text similarity framework using Accuracy and NDCG metrics

- Text-distance algorithms + Our Framework VS Text Embedding algorithms + Our Framework:** In Figure 7.7 we note that the Text-embedding and Text-distance algorithms have similar performance in our framework. This indicates that the choice of **distance** function is not the most important factor for good performance on our dataset. Instead, it is the use of our framework itself that boosts text-similarity performance.

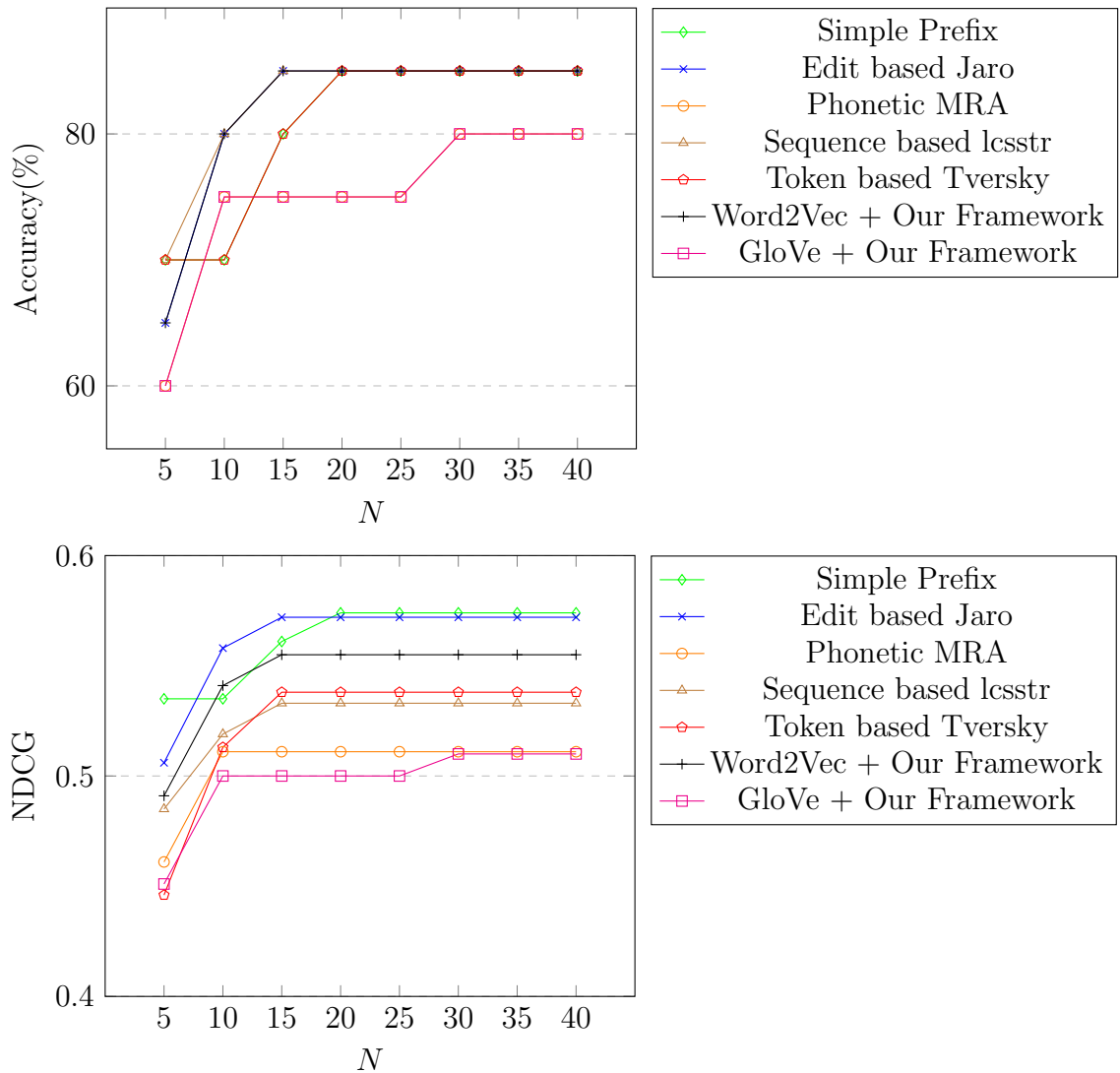


Figure 7.7: Comparing Text Embedding and selected **textdistance** algorithms, both in our text similarity framework using Accuracy and NDCG metrics

7.5 Summary

We have demonstrated the importance of performing text pre-processing before applying text embedding algorithms in the context of text similarity. Similarly, we have shown the importance of using the right distance metric, which in the case of Word2Vec and GloVe, is the Cosine similarity, by comparing it with another standard distance metric, the Euclidean distance.

Finally, given the use of pre-processed text in text embedding models with a Cosine distance metric to get the best performance with these models, we have shown that simple **textdistance** algorithms outperform all state-of-art models for our dataset. We also choose **textdistance** algorithms belonging to different categories like phonetic, edit-based, etc and show that they all outperform the text embedding models. This result shows our text similarity framework's robustness by providing excellent performance for a variety of **textdistance** algorithms. Further, we showed that text embedding models can be used within our framework to boost their performance. Also, we showed that **textdistance** and **text-embedding** algorithms have similar performance on our ground truth database. All the results obtained are justified by the design of our framework in the context of our ground truth data as explained in Section 6.1.2.

In the next Chapter, we will describe two approaches for qualitatively determining the impact of business change requirements on process repository business entities.

Chapter 8

Impact Assessment

In the preceding chapter, we addressed the Research Question involving the matching of Business Change Requirements to Process Objects (represented by short text). This chapter explores different methods for enabling a Business Analyst to qualitatively assess the impact of a matched Business Object in the context of the entire Process Repository. The qualitative impact analysis of a Change Requirement based on the matched business object (or objects) is performed in two ways. First, we provide context on the matched business object (or objects) in terms of the business model, the business role, etc. Secondly, we develop an OCR-based algorithm for detecting text bounding boxes in images to help develop a visualisation aid applied to our Process Repository image dataset.

8.1 Context Detection

We obtained the top N matched business entities from the process repository for given textual requirements in the previous chapter. Using Python with GraknClient [38] and Graql, we have retrieved the related business entities that get impacted for every change requirement. The related business entities can be viewed at four different levels:

1. At **Model** level, we retrieved all the business objects, sub-objects and sub-model objects.
2. At **Object** level, we retrieved the Model it belongs to, related preceding and succeeding object; and related sub-objects.
3. At **Sub-object** level, we retrieved the model and objects it belongs to.

4. At **Sub-model-object** level, we retrieved the model name it belongs to.

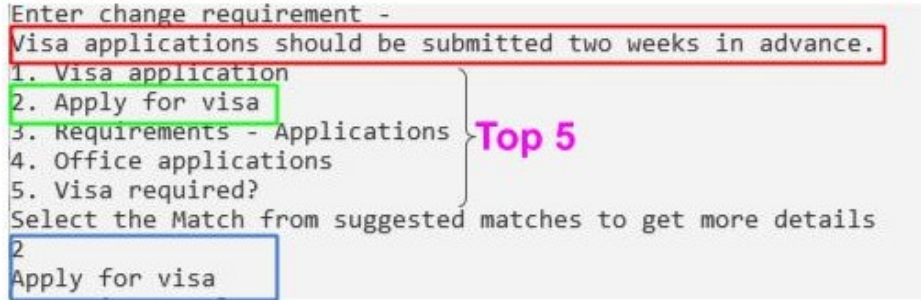


Figure 8.1: The user has to enter a change requirement (highlighted in red) to obtain the top 5 suggested matches. Then, the user has to select the actual business entity (highlighted in blue). Finally, the selected business entity will be placed in the standard query template to get the impacted business entities.

Given a change requirement, the user would need to select the relevant business entity from top N suggested results by a text similarity algorithm from the previous chapter, as shown in Figure 8.1. To do so, the Grakn knowledge graph is queried to retrieve the 243 candidate business entities in our process repository. This need for human intervention makes our system semi-automatic. To determine the context of the chosen business entity in the process repository, we have created different Graql query templates for extracting different context information of interest from the knowledge graph. For example, for a selected business object, we retrieve the model it belongs to, preceding and succeeding objects and its sub-objects as highlighted in Figure 8.2.

We describe the three query templates used and the corresponding context information obtained below.

- The following is the Graql query used to output the related model for the given object name. In Graql, the **MATCH** clause is used to get the data from Grakn. First, we have get the instance of **MODEL_NAME** entity with its attribute **NAME** using **HAS** keyword followed by a variable ($\$modelname$) as in line 1. The same format is used on **FLOW_OBJECT** (line 2) along with the **CONTAINS** keyword to search for the selected object 'Apply for Visa' (line 3). Later, we have to relate the entities **MODEL_NAME** and **FLOW_OBJECT** (line 5) with **MODEL_HAS_OBJECT** to get the related model name for a given object name.

```

Apply for visa
Executing Graql Query:
match $x isa MODEL_NAME, has NAME $modelname; $flow isa C_FLOW_OBJECT, has NAME $y; $y contains "Apply for visa";(MDL: $x, OBJECT: $flow) isa MODEL_HAS_OBJECT; get $modelname;

Models with matched activity are:
1. Make preparations for travel

Executing Graql Query:
match $act isa C_FLOW_OBJECT, has NAME $a; $src isa C_FLOW_OBJECT, has NAME $s; $dest isa C_FLOW_OBJECT, has NAME $d; $a contains "Apply for visa";(INCOMING: $src, OUTGOING: $act) isa RC_SEQUENCE_FLOW_BPMN;(INCOMING: $act, OUTGOING: $dest) isa RC_SEQUENCE_FLOW_BPMN;get $s,$d;

Previous and next activities are:
1. Visa required?, Converging Exclusive Gateway Visa required?
2. Visa required?, Pets to take care of?

Executing Graql Query:
match $x isa SUB_OBJECT, has NAME $sub; $flow isa C_FLOW_OBJECT, has NAME $y; $y contains "Apply for visa"; (SUB: $x, OBJ: $flow) isa OBJECT_HAS_SUB_OBJECT; get $sub;

Sub-objects are:
1. Organiser
2. Visa required?
3. Visa application (rules and fill-in helper)
4. Roadtrip participant
5. Tour guide
6. Visa application
7. Converging Exclusive Gateway Visa required?
8. Visa does not arrive in time
9. Embassy
10. Pets to take care of?

```

Figure 8.2: Impacted business entities

```

1 #Impacted Model
2 match $x isa MODEL_NAME, has NAME $modelname;
3 $flow isa FLOW_OBJECT, has NAME $y;
4 $y contains "Apply for visa";
5 (MDL: $x, OBJECT: $flow) isa MODEL_HAS_OBJECT;
6 get $modelname;

```

- For retrieving the preceding and succeeding object of a given object, we used the following Graql query. All the model objects are stored as FLOW_OBJECT. First, we have to get the instance of FLOW_OBJECT entity with its attribute NAME three times; first, for the preceding object (line 2), then for the succeeding object (line 3) and finally, for searching the selected object (line 1 & 4). Next, we have to relate the entity FLOW_OBJECT with its relation RC_SEQUENCE_FLOW_BPMN twice to get the dependent objects; first to relate preceding object(s) with the selected object (line 6) and secondly, to relate selected object with succeeding object(s) (line 7).

```

1 #Impacted Preceding and Succeeding Objects
2 match $act isa FLOW_OBJECT, has NAME $a;
3 $src isa FLOW_OBJECT, has NAME $s;
4 $dest isa FLOW_OBJECT, has NAME $d;
5 $a contains "Apply for visa";

```



```

6 (INCOMING: $src, OUTGOING: $act) isa
  RC_SEQUENCE_FLOW_BPMN;
7 (INCOMING: $act, OUTGOING: $dest) isa
  RC_SEQUENCE_FLOW_BPMN;
8 get $s,$d;

```

- The last Graql query is to obtain the impacted sub object(s) for the selected activity. First, we have to declare the instance of SUB_OBJECT with its attribute NAME to get the sub object(s) name (line 2) as well as for FLOW_OBJECT (line 3 & 4). Secondly, we have to relate the entities FLOW_OBJECT and SUB_OBJECT with their relation OBJECT_HAS_SUB_OBJECT (line 5) to GET the impacted sub object(s) (line 6).

```

1 #Impacted Sub Objects
2 match $x isa SUB_OBJECT, has NAME $sub;
3 $flow isa FLOW_OBJECT, has NAME $y;
4 $y contains "Apply for visa";
5 (SUB: $x, OBJ: $flow) isa OBJECT_HAS_SUB_OBJECT;
6 get $sub;

```

8.2 Visualisation Tool for Impact Assessment

In this section, we describe the OCR-based algorithm that we developed for detecting text bounding boxes in images to help develop a visualisation aid applied to our Process Repository image dataset.

8.2.1 Data Collection

For Impact Visualisation of a matched In order to visualise the business entities impacted by a change, we require the image of the process model. The image of the process model is retrieved from the Model JSON that was collected as described in Section 5.1, which has an item called IMAGEMAP. The value of the IMAGEMAP item is a base64 encoded string. Using Python, we first convert the encoded string to bytes, then decode it to create an image in the JPEG format.

8.2.2 Methodology

Process Model diagrams can exist in image formats, as shown in Figure 8.3. These diagrams contain information on different business entities in the form of human-readable text.

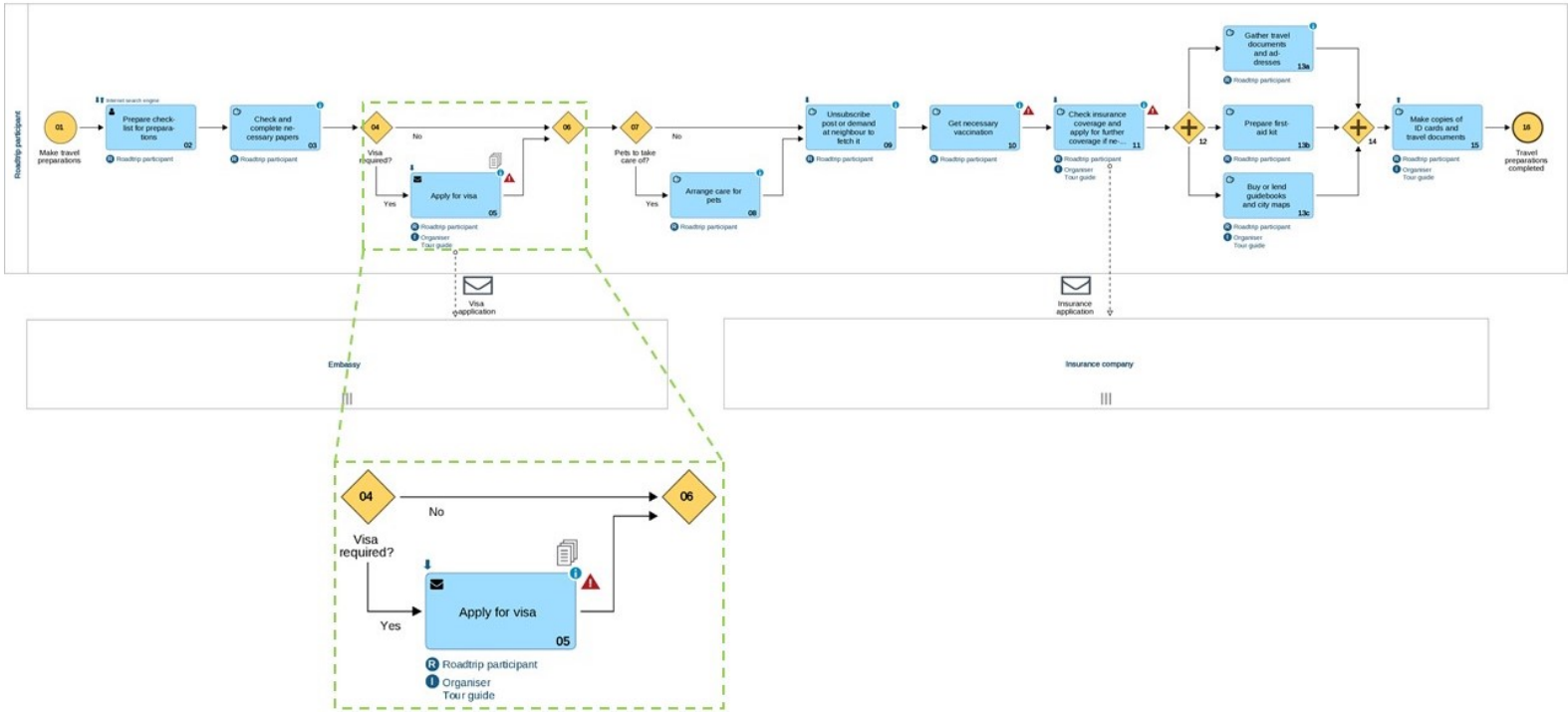


Figure 8.3: BPMN image example from our process repository

In this Section, we describe a method for automatically locating a given text (representing an existing business entity) if present in a process model image by drawing a rectangle around it. To this end, an Optical Character Recognition (OCR) system (like Tesseract) can be used to transcribe this text from a human-readable format in machine-recognised text.

However, these models can recognise fragments of text (each fragment typically representing a single word or token) individually along with their bounding box co-ordinates representing their respective locations in the image. We describe a system based on a **nearest neighbour set search algorithm** that we designed to iteratively combine the relevant fragments obtained through OCR systems based on their textual content and location. We provide a general framework of our approach in this section, followed by a detailed explanation of the specific algorithm applied to work well for our Process Model Image Repository. We describe the approach for solving this bounding box detection problem in a 3-part pipeline below.

- **Preprocessing:** Images may need to be pre-processed to remove, for instance, background colors and other non-text artifacts where possible depending on some prior knowledge on the object of interest. An example of such pre-processing applied to our image dataset is provided in Figure 8.4. This pre-processing should be designed to improve the out-of-the-box OCR systems' performance on the image dataset of interest.
- **OCR for Text Detection:** Apply an OCR system to detect text objects from a pre-processed input image. This is generally in the form of text fragments with corresponding information on each fragment's location in the image in terms of co-ordinates of its bounding box.
- **Locating Input text from OCR Text Fragments:** In this part of the pipeline, we describe the algorithm we designed for detecting the location of an input text in an image from its OCR Text Fragments extracted above.

Suppose we tokenise the Input Text to be found simply by splitting it into individual words. Let us say that the set n tokens of an input text are represented by -

$$X = \{t_i\}, \text{ where } i = 1, \dots, n$$

Let us say that the fragments are represented by a set Y of size m :

$$Y = \{f_j\}, \text{ where } j = 1, \dots, m$$

each fragment f_j containing the following information about the j^{th} fragment.

- **text:** The text content of the fragment
- **left:** The minimum co-ordinate of the part of the image covered by the fragment on the horizontal axis
- **right:** The maximum co-ordinate of the part of the image covered by the fragment on the horizontal axis
- **top:** The minimum co-ordinate of the part of the image covered by the fragment on the vertical axis
- **bottom:** The maximum co-ordinate of the part of the image covered by the fragment on the vertical axis
- **confidence:** The confidence of the OCR detection for this fragment

The Overall Algorithm for drawing the bounding-box around some text in an *image* given X , Y , and a distance *threshold* is given in Algorithm 1. We start by obtaining the initial fragments (C) with the confidence of at least 50% and matching the first token in X in line 1. Next, we initialise a set of clusters as an empty set in line 2. For each initial fragment c in C , we obtain a cluster of fragments starting with a cluster containing only c (line 4 to line 12). This is done by expanding the current cluster by including its neighbouring fragments based on the *nearestNeighboursFromPoints* algorithm. The cluster is expanded until no more fragments are added to it (line 7).

Next, for each cluster corresponding to the initial fragments, we obtain the best cluster as the largest cluster in line 14. The size of this cluster should ideally correspond to the size of X . In lines 16 to 19, we obtain the bounding-box co-ordinates of the best fragment cluster using the extreme (i.e., maximum and minimum where applicable) bounding-box co-ordinates of its fragments. Finally, we call a procedure in line 21 to draw a rectangle in the original image from the bounding-box defined above.

Algorithm 2 takes as input a cluster, the set of tokens X , the set of fragments Y , and the distance *threshold*. It adds to *cluster* all fragments in Y corresponding to some token in X which are within a *threshold distance* of some existing fragment in *cluster* and returns

this expanded cluster. The **distance** between two fragments f_i and f_j is calculated as the smallest Euclidean distance between some bounding-box corner of f_i with some bounding-box corner of f_j .

Algorithm 1 localiseTextFromFragments (*image, X, Y, threshold*):

```

1:  $C = \{y \mid y \in Y \text{ and } y.confidence > 50 \text{ and } y.text = t_1\}$ 
2:  $clusters = \{\}$ 
3: for  $c$  in  $C$  do
4:    $cluster = \{c\}$ 
5:   while true do
6:      $clusterNew = nearestNeighboursFromPoints(cluster, X, Y,$ 
                                                     $threshold)$ 
7:     if  $clusterNew = cluster$  then
8:       break
9:     end if
10:     $cluster = clusterNew$ 
11:  end while
12:   $clusters.add(cluster)$ 
13: end for
14:  $bestCluster = \max\{cluster.size \mid cluster \in clusters\}$ 
15:
16:  $left = \min(\{fragment.left \mid fragment \in bestCluster\})$ 
17:  $right = \max(\{fragment.right \mid fragment \in bestCluster\})$ 
18:  $top = \min(\{fragment.top \mid fragment \in bestCluster\})$ 
19:  $bottom = \max(\{fragment.bottom \mid fragment \in bestCluster\})$ 
20:
21:  $drawRectangle(image, left, right, top, bottom)$ 

```

Algorithm 2 *nearestNeighboursFromPoints(cluster, X, Y, threshold)*

```
1: result = cluster
2: clusterTokens = {c.text | c ∈ cluster}
3: XFiltered = {x | x ∈ X, x ∉ clusterTokens}
4: YFiltered = {y | y ∈ Y, y.text ∉ clusterTokens}
5: for y ∈ YFiltered do
6:   minDistance = min{distance(y, f) | f ∈ cluster}
7:   if y.text ∈ XFiltered and minDistance ≤ threshold then
8:     result.add(y)
9:   end if
10: end for
11: return result
```

8.2.3 Implementation

In this section, we describe the implementation of the methods detailed above. The text detection and localisation algorithm explained in Section 8.2.2 was implemented by using the Tesseract OCR engine developed by Google [3]. We also use Python’s pytesseract package [48], which acts as a wrapper around Tesseract to provide a purely Pythonic implementation of our OCR system.

For pre-processing, we use Python’s OpenCV library [21] to first convert our color image to greyscale, as shown in Figure 8.4. Next, we apply simple thresholding on this greyscale image to obtain the pre-processed image shown in Figure 8.4. We tested that this method works well to identify the different text fragments in our image dataset’s images. Without this pre-processing, only the text fragments on a white background were being detected by Tesseract OCR.

Next, pytesseract OCR is used to obtain the set of text fragments from this pre-processed image. Next, we apply our text detection and localisation algorithm described in Section 8.2.2 to obtain the bounding box of the input text. Again, we use Python’s OpenCV library to draw a rectangle around the obtained text bounding box on the original image, as shown in Figure 8.5.

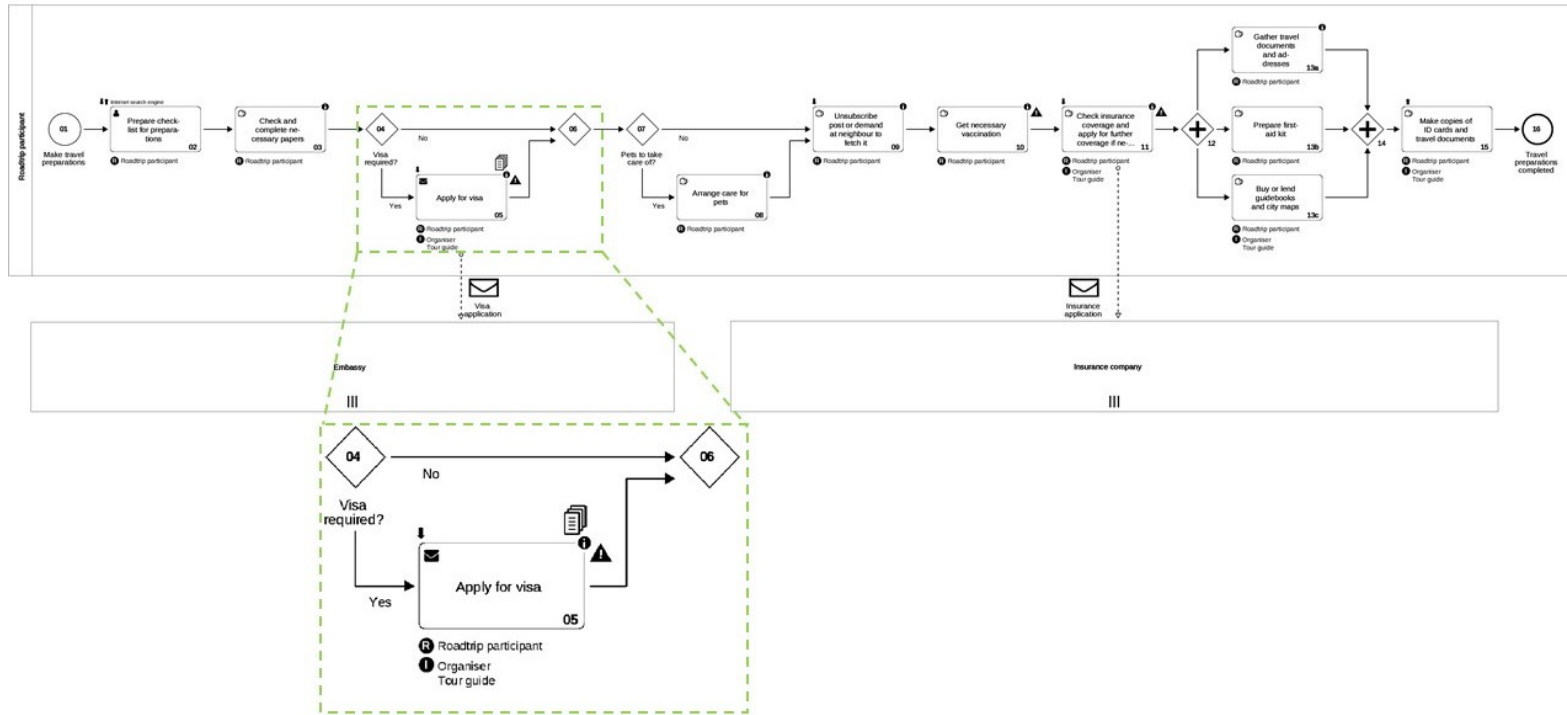


Figure 8.4: Preprocessed example BPMN image

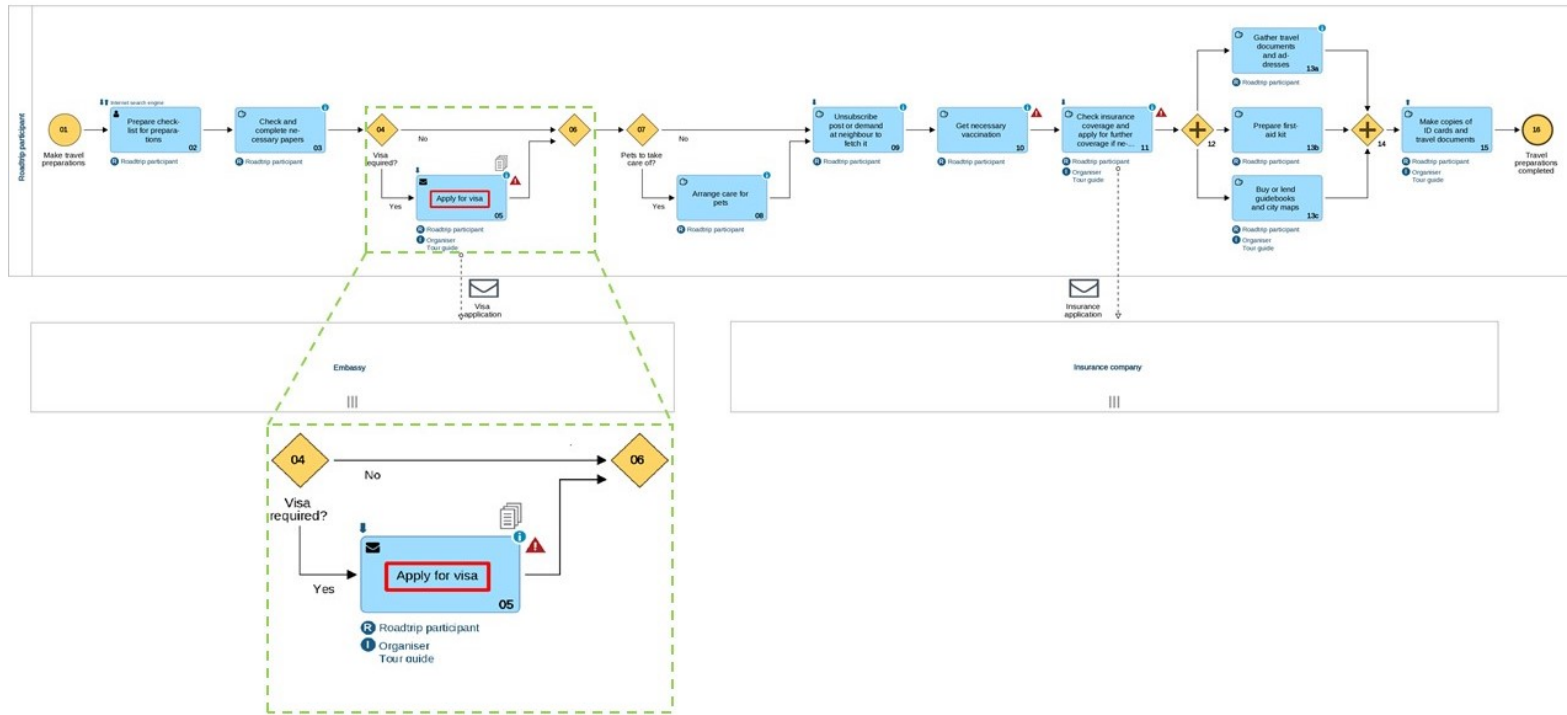


Figure 8.5: Text detection by bounding box example BPMN image

8.2.4 Evaluation

In Figure 8.5, we showed an example of the result of our OCR-based algorithm for drawing a rectangle around the text in a process model image. For evaluating our OCR-algorithm, it is not practical to create ground truth data of rectangles drawn around input texts. Instead, we test our algorithm manually on a total of 20 business entity texts in 5 different process model images. In all 20 cases, our algorithm correctly enclosed the input text with a rectangle (as in the case of Figure 8.5). However, we must note that a limitation of our model is that the image pre-processing as well as the *threshold* parameter in our overall algorithm, Algorithm 1, need to be modified when applied to a different image dataset.

8.3 Summary

In this Chapter, we have described two methods for the qualitative assessment of the impact of a change requirement using the matched (impacted) business entities obtained from Text Similarity algorithms.

Chapter 9

Conclusion

To conclude the work of this master project, first, we summarise how the research questions were answered and how we achieved the research goals mentioned in Section 1.2 of this report. Second, the limitations of the proposed solutions are introduced. Finally, future research possibilities are discussed.

9.1 Summary of Contributions

In this work, a method is presented that aims to semi-automatically match the business entities from a given textual change requirement. Further, we determine the qualitative impact of the change requirement using the context of the matched business entities. The following research questions were answered through the presented method to achieve the given goal:

RQ1: *Can we design a framework for representing raw process repositories which can be easily queried to obtain relevant information related to different entities in the contained process models?*

From the literature review in Section 2.2.3, we found that we can use the knowledge graph as a framework for representing a process repository. So, in Chapter 5, using Grakn (and Graql) knowledge graphs, we represented the process models and their relevant objects from a process repository and stored them simply and efficiently.

RQ2: *How accurate are the state-of-the-art text matching algorithms? How good are they in matching change requirements to relevant entities?*

In Chapters 6 and 7, we describe the implementation and evaluation of the state-of-the-art text similarity algorithms on our ground-truth dataset. Further, we address the shortcomings of these approaches on our dataset in

the following Research Question.

RQ3: *Can we develop a better model in the specific domain of matching change requirements to relevant business entities?*

In Chapter 6, we have developed a framework to identify business entities from given textual change requirements. We then evaluated these approaches and compared their performance with the state-of-the-art algorithms and found superior performance in both accuracy and NDCG using simple **textdistance** algorithms.

RQ4: *Can we qualitatively determine the impact of a change requirement through the impacted entities obtained in RQ2 or RQ3 by querying the process repository framework defined in RQ1?*

The goal of this thesis is to help a business analyst identify the impacted business entities for a given change requirement. In Chapter 8, we semi-automatically determine the qualitative impact of business change requirements on process models using the business entity context and an OCR-based visualisation tool.

To summarise, by answering these research questions, we have developed systems for:

- Representing, storing and querying process repository data in a knowledge graph database.
- Ranking business entity texts in decreasing order of their relevance to a given change requirement, using a generic text similarity framework that outperforms current state-of-the-art methods for semantic similarity base on text embeddings.
- Determining the impact of change requirements by providing context of business entities in process models as well as indicating the location of impacted business entities in their model images.

9.2 Limitations and Future Works

The following are the limitations of our work:

1. We could not process the information about the level in the hierarchical process repository because each group and sub-group had various levels.

2. We cannot provide a fully automatic solution for matching relevant business entities as it would require us to provide a perfect text similarity algorithm always obtaining the matched entity at Rank 1.

The following future works can be considered:

1. We can try to process the level information as it will help the business analyst to find the impacted business entities on that particular level.
2. An ensemble of **textdistance** algorithms applied within our framework can be evaluated and might yield performance metrics improvements.
3. A more significant ground truth dataset can be curated and tested to provide more reliable results.
4. We can compare and evaluate other state-of-the-art NLP techniques, such as, ELMo [49], Transformer-XL [50], GPT-2 [51], ERNIE 2.0 [52], XLNet [53], Compressive Transformer [54], and T-NLG [55].
5. We can enrich the context of matched business entities with more query templates.
6. We can enhance our implementation for entity matching to incorporate domain knowledge about the type of the business entity (e.g. activity) that we are interested in using the knowledge graph.

Bibliography

- [1] BPM-D. *Executive Introduction Part 3 Reading*, 2018 (accessed August 7, 2020). <https://bpm-d.com/executive-introduction-part-3-reading/>.
- [2] Adrián Marcelo Mendoza Mendoza. Bpm “gestión de proyectos de investigación” del centro universitario de investigación científica y tecnológica de la universidad técnica del norte utilizando auraportal. B.S. thesis, 2018.
- [3] Victor Ohlsson. Optical character and symbol recognition using tesseract, 2016.
- [4] William J Kettinger, James TC Teng, and Subashish Guha. Business process change: a study of methodologies, techniques, and tools. *MIS quarterly*, pages 55–80, 1997.
- [5] V Daniel Hunt. *Process mapping: how to reengineer your business processes*. John Wiley & Sons, 1996.
- [6] Joon Park and Seung Ryul Jeong. A study on the relative importance of underlying competencies of business analysts. *THIS*, 10(8):3986–4007, 2016.
- [7] Harry Jiannan Wang and Harris Wu. Supporting process design for e-business via an integrated process repository. *Information Technology and Management*, 12(2):97–109, 2011.
- [8] Marcello La Rosa, Hajo A Reijers, Wil MP Van Der Aalst, Remco M Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. Apromore: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, 2011.
- [9] Mathias Weske. Business process management architectures. In *Business Process Management*, pages 333–371. Springer, 2012.

- [10] Mathias Kirchmer et al. *High performance through business process management*. Springer, 2017.
- [11] Mark Von Rosing, Henrik Von Scheel, and August-Wilhelm Scheer. *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume 1*, volume 1. Morgan Kaufmann, 2014.
- [12] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Introduction: What is a knowledge graph? In *Knowledge Graphs*, pages 1–10. Springer, 2020.
- [13] Ramez Elmasri. *Fundamentals of database systems*. Pearson Education India, 2008.
- [14] Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.
- [15] Antonio Messina, Haikal Pribadi, Jo Stichbury, Michelangelo Bucci, Szymon Klarman, and Alfonso Urso. Biograkn: A knowledge graph-based semantic database for biomedical sciences. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 299–309. Springer, 2017.
- [16] Christopher Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.
- [17] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.
- [18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

- [21] OpenCV. *library of Python bindings designed to solve computer vision problems*, 2018 (accessed June 18, 2020). <https://pypi.org/project/opencv-python/>.
- [22] A et al. Lundh, F Clarck. *Pillow*, 2016 (accessed June 18, 2020). <https://python-pillow.org/>.
- [23] Han van der Aa, Henrik Leopold, and Hajo A Reijers. Comparing textual descriptions to process models—the automatic detection of inconsistencies. *Information Systems*, 64:447–460, 2017.
- [24] Maria Rana, Khurram Shahzad, Rao Muhammad Adeel Nawab, Henrik Leopold, and Umair Babar. A textual description based approach to process matching. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 194–208. Springer, 2016.
- [25] Luis Delicado Alcántara, Josep Sánchez Ferreres, Josep Carmona Vargas, and Lluís Padró. Nlp4bpm: Natural language processing tools for business process management. In *BPM Demo and Industrial Track 2017 Proceedings*, pages 1–5, 2017.
- [26] Henrik Leopold, Han van der Aa, Fabian Pittke, Manuel Raffel, Jan Mendling, and Hajo A Reijers. Searching textual and model-based process descriptions based on a unified data format. *Software & Systems Modeling*, 18(2):1179–1194, 2019.
- [27] Arda Goknil, Ivan Kurtev, and Klaas van den Berg. A rule-based change impact analysis approach in software architecture for requirements changes. *arXiv preprint arXiv:1608.02757*, 2016.
- [28] Shiva Nejati, Mehrdad Sabetzadeh, Chetan Arora, Lionel C Briand, and Felix Mandoux. Automated change impact analysis between sysml models of requirements and design. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 242–253, 2016.
- [29] Yi Wang, Jian Yang, Weiliang Zhao, and Jianwen Su. Change impact analysis in service-based business processes. *Service Oriented Computing and Applications*, 6(2):131–149, 2012.
- [30] Oussama Mohammed Kherbouche, Adeel Ahmad, Mourad Bouneffa, and Henri Basson. Ontology-based change impact assessment in dynamic business processes. In *2013 11th International Conference on Frontiers of Information Technology*, pages 235–240. IEEE, 2013.

- [31] Mourad Bouneffa and Adeel Ahmad. The change impact analysis in bpm based software applications: A graph rewriting and ontology based approach. In *International Conference on Enterprise Information Systems*, pages 280–295. Springer, 2013.
- [32] Zhilei Ma, Branimir Wetzstein, Darko Anicic, Stijn Heymans, and Frank Leymann. Semantic business process repository. *SBPM*, 251:55, 2007.
- [33] Katalina Grigorova and Ivaylo Kamenarov. Object relational business process repository. In *Proceedings of the 13th International Conference on Computer Systems and Technologies*, pages 72–78, 2012.
- [34] Injun Choi, Kwangmyeong Kim, and Mookyung Jang. An xml-based process repository and process query language for integrated process management. *Knowledge and Process Management*, 14(4):303–316, 2007.
- [35] Sherif Sakr and Ahmed Awad. A framework for querying graph-based business process models. In *Proceedings of the 19th international conference on World wide web*, pages 1297–1300, 2010.
- [36] Jitender Aswani, Ryan Leask, and Jens Doerpmund. Representing enterprise data in a knowledge graph, November 27 2014. US Patent App. 13/902,677.
- [37] Lian Wen and R Geoff Dromey. From requirements change to design change: A formal path. In *Proceedings of the Second International Conference on Software Engineering and Formal Methods, 2004. SEFM 2004.*, pages 104–113. IEEE, 2004.
- [38] GraknClient. *python interface which we can use to read from and write to a Grakn knowledge graph.*, 2019. <https://dev.grakn.ai/docs/client-api/python>.
- [39] SpaCy. *Python-based Industrial-Strength Natural Language Processing*, 2015 (accessed June 12, 2020). <https://spacy.io/usage>.
- [40] Márton Miháلتz. *word2vec-GoogleNews-vectors*, 2016 (accessed June 12, 2020). <https://github.com/mmihaltz/word2vec-GoogleNews-vectors>.
- [41] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

- [42] SciPy. *Python-based ecosystem of open-source software for mathematics, science, and engineering.*, 2001 (accessed June 12, 2020). <https://www.scipy.org/>.
- [43] TextDistance. *python library for comparing distance between two or more sequences by many algorithms.*, 2017 (accessed June 12, 2020). <https://pypi.org/project/textdistance/>.
- [44] life4/textdistance. *30+ algorithms python implementation for comparing more than two sequences*, 2017 (accessed June 12, 2020). <https://github.com/life4/textdistance>.
- [45] Wikipedia. *Match rating approach.*, 2009 (accessed August 1, 2020). https://en.wikipedia.org/wiki/Match_rating_approach.
- [46] Wikipedia. *Jaro–Winkler distance.*, 1989 (accessed August 1, 2020). https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance.
- [47] Wikipedia. *Tversky index.*, 1977 (accessed August 1, 2020). https://en.wikipedia.org/wiki/Tversky_index.
- [48] Python-tesseract. *an optical character recognition (OCR) tool for python.*, 2014. <https://pypi.org/project/pytesseract/>.
- [49] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020.
- [50] Leeja Mathew and VR Bindu. A review of natural language processing techniques for sentiment analysis using pre-trained models. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 340–345. IEEE, 2020.
- [51] Virapat Kieuvongngam, Bowen Tan, and Yiming Niu. Automatic text summarization of covid-19 medical research articles using bert and gpt-2. *arXiv preprint arXiv:2006.01997*, 2020.
- [52] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975, 2020.
- [53] Hailong Li, Jaewan Choi, Sunjung Lee, and Jung Ho Ahn. Comparing bert and xlnet from the perspective of computational characteristics. In

2020 International Conference on Electronics, Information, and Communication (ICEIC), pages 1–4. IEEE, 2020.

- [54] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- [55] Min-Yuh Day and Chichang Jou. Universal sentence-embedding models. *Foundations*, 2(2020/03):09, 2020.

Appendix A

Types of Business Models

Analysis Model: A high-level organisation of the objectives, controls, and risks related to the business. This model helps to analyse the data, capacities and the behavior of the business framework and these are converted into the engineering, interface and component-level design in the business design modeling as shown in Figure A.1.

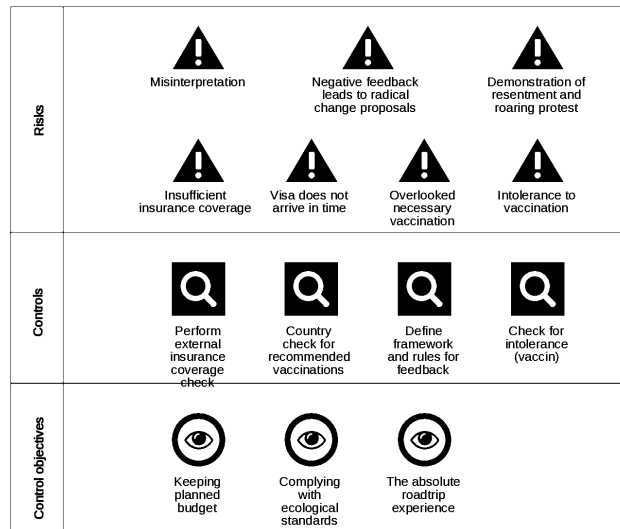


Figure A.1: Sample Analysis Model

Company Map: Graphical visualisation of a business's essential elements to plan, analyse, and develop a business. This business model is the highest level diagram that specifies a high-level description of the business. Those who have domain knowledge or have business expertise can understand this mapping of business vital elements as shown in Figure A.2.

Document Model: This model is designated for textual documents.

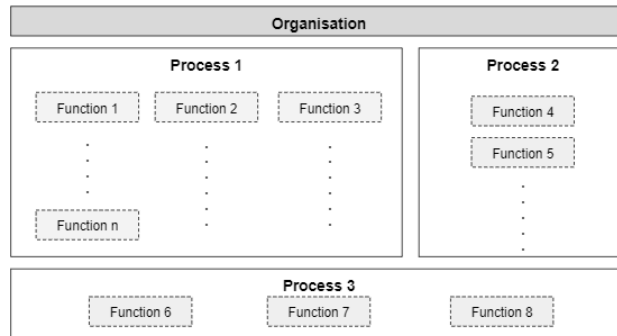


Figure A.2: Sample Company Map

This model has high-level information about the documents related to business procedures, rules, and client's information in the file system. This model categorizes the types of documents in folders and each folder has its related files as shown in Figure A.3.

















Documents for trip preparation	     
Travel documents	
Application forms	   
Contracts and policies	
Check lists	  
Further applicable documents	

Figure A.3: Sample Document Model

IT System Model: Similar to the document model but it organises different technologies, applications, and services based on business functions as shown in Figure A.4.

Working Environment Model: The business process diagram is in the form of a connected structure that represents an organisation's business. It gives a graphical overview of how different businesses are connected and

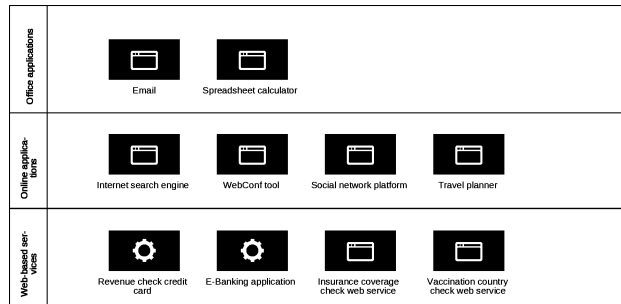


Figure A.4: Sample IT System Model

which role is responsible for these different businesses are depicted in a working environment model as shown in Figure A.5.

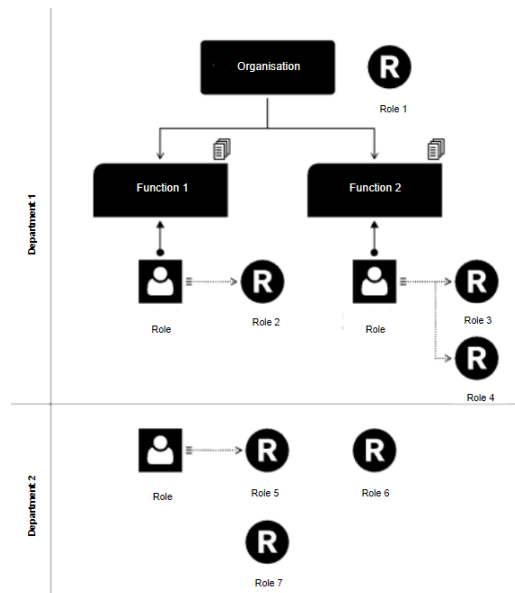


Figure A.5: Sample Working Environment Model

Appendix B

Graql Queries

The following is the complete schema definition used to create a Grakn, a knowledge graph database as shown in Figure 5.6

```
1 #Entities of the business process models
2
3 #models
4 MODEL_NAME sub entity,
5     has ID,
6     has NAME,
7     has TYPE,
8     plays MN,
9     plays MDL,
10    plays MOD;
11
12 #objects of all models
13 FLOW_OBJECT sub entity,
14     plays INCOMING,
15     plays OUTGOING,
16     plays OBJECT,
17     plays OBJ,
18     has ID,
19     has NAME,
20     has TYPE;
21
22 #subobject of object
23 SUB_OBJECT sub entity,
24     has ID,
25     has NAME,
26     has TYPE,
27     has DIRECTION,
28     plays SUB;
29
30 #subobject of model
31 SUB_MODEL_OBJECT sub entity,
```

```

32     has ID,
33     has NAME,
34     has TYPE,
35     has DIRECTION,
36     plays SMO;

```

```

1  #Relationships between each entities
2
3  RC_SEQUENCE_FLOW_BPMN sub relation,
4      relates INCOMING,
5      relates OUTGOING;
6
7  MODEL_HAS_OBJECT sub relation,
8      relates MDL,
9      relates OBJECT;
10
11 OBJECT_HAS_SUB_OBJECT sub relation,
12     relates OBJ,
13     relates SUB;
14
15 MODEL_HAS_SUB_MODEL_OBJECT sub relation,
16     relates MOD,
17     relates SMO;

```

```

1  #Attributes
2  ID sub attribute,
3      datatype string;
4  NAME sub attribute,
5      datatype string;
6  TYPE sub attribute,
7      datatype string;
8  DIRECTION sub attribute,
9      datatype string;

```

The following is the complete Graql INSERT queries used in this work.

```

1  #Entities & Attributes insertion in Graln
2  insert $model isa MODEL_NAME,
3  has ID "{model_id}",
4  has NAME "{model_name}",
5  has TYPE "{model_type}";
6
7  insert $object isa FLOW_OBJECT,
8  has ID "{object_id}",
9  has NAME "{object_name}",
10 has TYPE "{object_type}";
11
12 insert $sub isa SUB_OBJECT,
13 has ID "{sub_id}",
14 has NAME "{sub_name}",

```



```

15 has TYPE "{sub_type}",
16 has DIRECTION "{sub_direction}";
17
18 insert $subm isa SUB_MODEL_OBJECT,
19 has ID "{subm_id}",
20 has NAME "{subm_name}",
21 has TYPE "{subm_type}",
22 has DIRECTION "{subm_direction}";

1 #Relationships insertion into Grakn
2 match $model isa MODEL_NAME, has ID "{model_id}";
3 $object isa C_FLOW_OBJECT, has ID "{object_id}";
4 insert $mho (MDL: $model, OBJECT: $object)
5 isa MODEL_HAS_OBJECT;
6
7 match $object isa FLOW_OBJECT, has ID "{object_id}";
8 $sub isa SUB_OBJECT, has ID "{sub_id}";
9 insert $ohs (OBJ: $object, SUB: $sub)
10 isa OBJECT_HAS_SUB_OBJECT;
11
12 match $src isa FLOW_OBJECT, has ID "{source}";
13 $dest isa FLOW_OBJECT, has ID "{dest}";
14 insert $relation (INCOMING: $src, OUTGOING: $dest)
15 isa RC_SEQUENCE_FLOW_BPMN;
16
17 match $model isa MODEL_NAME, has ID "{model_id}";
18 $subm isa SUB_MODEL_OBJECT, has ID "{sub_object_id}";
19 insert $mshm (MOD: $model, SMO: $subm)
20 isa MODEL_HAS_SUB_MODEL_OBJECT;

```

The following is the complete Graql MATCH queries used to GET information from the Grakn.

```

1 #For a "given model name"
2
3 #model -> sub model object(s)
4 match $model isa MODEL_NAME, has NAME $modelname;
5 $modelname contains "given model name";
6 $subm isa SUB_MODEL_OBJECT, has NAME $submodelname;
7 (MOD: $model, SMO: $subm) isa MODEL_HAS_SUB_MODEL_OBJECT;
8 get $submodelname;
9
10
11 #model -> all objects
12 match $model isa MODEL_NAME, has NAME $modelname;
13 $modelname contains "given model name";
14 $flow isa FLOW_OBJECT, has NAME $objectname;
15 (MDL: $model, OBJECT: $flow) isa MODEL_HAS_OBJECT;
16 get $objectname;
17

```

```

18
19 #model -> all objects -> all sub_objects
20 match $model isa MODEL_NAME, has NAME $modelname;
21 $modelname contains "given model name";
22 $flow isa FLOW_OBJECT, has NAME $objectname;
23 (MDL: $model, OBJECT: $flow) isa MODEL_HAS_OBJECT;
24 match $sub isa SUB_OBJECT, has NAME $subobjectname;
25 (OBJ: $flow, SUB: $sub) isa OBJECT_HAS_SUB_OBJECT;
26 get $subobjectname;

1 #For a "given sub object name"
2
3 #sub_object -> object(s)
4 match $sub isa SUB_OBJECT, has NAME $subobjectname;
5 $subobjectname contains "given sub object name";
6 $flow isa FLOW_OBJECT, has NAME $objectname;
7 (OBJ: $flow, SUB: $sub) isa OBJECT_HAS_SUB_OBJECT;
8 get $objectname;
9
10 #sub_object -> object(s) -> model(s)
11 match $sub isa SUB_OBJECT, has NAME $subobjectname;
12 $subobjectname contains "given sub object name";
13 $flow isa FLOW_OBJECT, has NAME $objectname;
14 (OBJ: $flow, SUB: $sub) isa OBJECT_HAS_SUB_OBJECT;
15 $model isa MODEL_NAME, has NAME $modelname;
16 MDL: $model, OBJECT: $flow) isa MODEL_HAS_OBJECT;
17 get $modelname;

1 #For a "given sub model object name"
2
3 #sub_model_object -> model(s)
4 match $subm isa SUB_MODEL_OBJECT, has NAME $submodelname;
5 $submodelname contains "given sub model object name";
6 $model isa MODEL_NAME, has NAME $modelname;
7 (MOD: $model, SMO: $subm) isa MODEL_HAS_SUB_MODEL_OBJECT;
8 get $modelname;

```