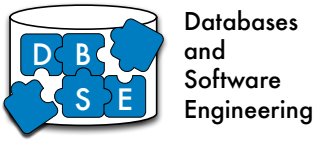
University of Magdeburg

School of Computer Science



Master's Thesis

Clustering And Keyword Based Feature Extraction Using Textual Requirements

Author:

Nikhil PrakashRao Raikar

March 28th, 2019

Advisors:

Prof.Dr.rer.nat.habil.Gunter Saake
Department of Databases and Software Engineering

Dr.-Ing. Sandro Schulze
Department of Databases and Software Engineering

M.Sc. Yang Li
Department of Databases and Software Engineering

Raikar, Nikhil PrakashRao:
Clustering And Keyword Based Feature Extraction Using Textual Requirements
Master's Thesis, University of Magdeburg, *[2018]*.

Declaration of Authorship

I, ***Nikhil Prakashrao Raikar***, declare that this thesis titled, ***"Clustering And Keyword Based Feature Extraction Using Textual Requirements"*** and the work presented in it are my own. I confirm that:

- This work was done while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

Acknowledgements

I would first like to render my warmest gratitude to my main mentor and thesis supervisor *Yang Li*, who made this work possible. His friendly guidance, expert advises have been invaluable throughout all stages of the work. He gave me complete freedom to explore and work towards my goal and he was always there to help me by pointing me in the right direction whenever he felt I lacked it, for his support I am forever grateful.

I would wish to extend my gratitude to *Dr.-Ing. Sandro Schulze*, who has given his valuable time as one of the main reviewers for this work, and I am gratefully indebted for his worthy comments on this thesis.

I would gratefully acknowledge *Prof. Dr. rer. nat. habil. Gunter Saake*, for his valuable time as one of the main reviewers and also for providing me this opportunity of writing this thesis under his main supervision within his working group.

Finally, I must express my very profound gratitude to god, my parents, my siblings, my family members and sincere friends for all their prayers and moral support throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Author
Nikhil Prakash Rao Raikar

Abstract

Software Product Line (SPL) are a set of systems sharing a common set of features which satisfy a common goal. A feature is a representation of a product in SPL. A feature model in SPL is a representation of all these features and their relationships. Designing a feature model from a set of unstructured raw requirement specifications is one of the traditional ways to visualize large amount requirement specifications, to process requirements faster and also one of the best practices incorporated in SPL businesses to keep track of features and to reuse or extend certain features easily. The creation of these feature models helps business owners in making important decisions such as creation of new software assets in a cost effective way by selecting components which can be reused, finding better trace ability links to other artifacts by analyzing variability information, providing systematic integration of single software systems and also helping design teams to processes design decisions and to develop new assets faster. But manually creating a feature model from scratch has its own advantages and its disadvantages. Accuracy, semantic understanding of the requirements and creation of feature model accordingly by an human expert leading to a better understand ability of the design generation are few of its advantages. exponential increased in cost, more time consumption and flexibility constraints are in increased proportions. These are important since these disadvantages can, to any software product line business be very disadvantageous and expensive to cover. In this thesis, we introduce an approach to reduce the manual intervention needed in extraction of features and relationships between features from a set of raw unstructured requirements. Our approach takes into account unstructured natural language texts, preprocesses requirements from it, vectorize them using various different approaches and with the help of clustering algorithms and Natural Language Processing (NLP) techniques we extract meaningful features and their relationship information. We also evaluate our approach with few of the other requirements and techniques to find a best possible solution.

Contents

List of Figures

List of Tables

List of Code Listings

List of Acronyms

BOW	Bag of Words
HAC	Hierarchical Agglomerative Clustering
IDC	Incremental Diffusive Clustering
IDF	Inverse Document Frequency
LDA	Latent Dirichlet Allocation
LSA	latent semantic analysis
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NNP	Proper Noun
POS	Parts Of Speech
SPK	Spherical K-means Clustering
SPL	Software Product Line
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TR	Textrank
VSM	vector space models

1. Introduction

SPL practice is a set of tools and techniques designed to create similar software systems with the use of a shared set of software assets [35]. They help SPL business by helping the service provider to reuse certain systems to create similar software. Feature model is one of the branches of software product line which helps in representing all the requirements of a SPL in terms of features. Each feature of a feature model represents the requirements and functionalities of that specific feature and provides information about it. The visual representation of these feature models can be seen as a feature tree.

Over the following years, these feature models have been growing rapidly with the increase and demands of the requirements growing in the businesses. Various approaches have been formulated in the designing phase of these feature models [49]. Often times these models are created by human interpreters who are knowledge expert in their own respective fields. Although manual creation of feature models have their advantages, few drawbacks which exist tend to impact the process greatly. Few such being a lot of manual work, it is expensive and it is time-consuming. One of the most laborious steps in the manual design process is to extract meaningful features and generate a feature model from a large set of raw unstructured requirements.

The main focus of this thesis has been to design an approach which can reduce the manual work for extracting features and their relationships from raw requirements. The concept has evolved over the years with various techniques due to its usefulness.

1.1 Motivation of this Thesis

Since manual creation of feature models can have its advantages, keyword extraction using clustering based approach is aimed to reduce major drawbacks created with manual work, can significantly decrease the time and accelerate the feature model generation processes. There are studies conducted previously that propose methods to create and to find relevant features and feature models. The accuracy being recorded very low, these systems have been highly dependent on certain approaches incorporated for a specific task.

The main focus of this thesis has been to develop an approach that can extract relevant, meaningful features and relationships which can further help in designing a feature model. The set of relevant features generated after incorporating our approach can be assessed to other similar studies that meet our criteria. To reduce the time and effort required for feature model generation, the suggested approach will support extracting a good set of features and analyze the set of features and extract relationship information from them.

1.2 Goal of this Thesis

The goal of this work is divided into the following tasks:

- Based on the findings of the initial literature survey, the goal of this thesis is to analyze existing approaches that support extraction of features and relationships using requirements. The aim is to study the methodologies proposed previously and identify areas that demand further improvement.

- Propose an approach to successfully extract meaningful features from the raw requirements, based on certain criteria which are termed important by the clustering process. At first, the meaningful features are extracted from raw processed requirements by using a developed vectorization process which takes advantage of a combination of ranking criteria and similarity measure. The vectorized requirements are then fed it to a clustering algorithm. The output of the process is set of grouped requirements which are similar not just in terms of word-level similarity but also in terms of semantic meanings and from these clustered requirements features are extracted using different patterns,frequency recognition and rule-based approach. The relationship detection between features are then extracted by analyzing certain word level dependencies by making use of frequencies, similarities and also making use of Orthogonal variability models.

  By proposing this approach, we answer three of these research questions

  - *RQ1: How can we design an approach which can produce reasonably accurate features?*
    For the SPL business, it becomes very important to extract a good set of meaningful features in-order to determine the usefulness of the processes, extraction of good features gives a bigger advantage in-terms of expanding the services and also moving the process further. Hence it is very essential to design an approach which can extract a good set of features from unstructured documents.
  - *RQ2: How can we achieve variability completeness?*
    From the set of meaningful features can we design an approach which can retrieve relationships between features in a reasonably accurate way?
  - *RQ3: How can we evaluate the obtained features and results ?*
    From the set of extracted features and their relationships, what measures can be taken to validate the results? How to validate the results obtained.

- Evaluate the proposed approach to investigate the validity of the results and flexibility of the procedure. The results obtained using the suggested approach is examined by comparing it with different approaches, state of the art vectorization processes, different clustering algorithms, ground truth and also applying different statistical measures.

1.3 Structure of the Thesis

To present our work, we structure this thesis as follows.

In Chapter 1, we explain a short introduction about what *feature tree* is and how it can help the software product line businesses. The motivation part of our introduction consists of short background which explains few drawbacks of the present approaches and how we can establish an approach to overcome them. It also briefly explains the goals we set in the process and how well we planned to achieve it. The structure of the thesis is elaborated in this following chapter.

In Chapter 2, we explain in detail the *background research* on the related ongoing work we planned to implement. This chapter explains the entire background knowledge required before going through the implementation of our approach. It also explains in detail, the required concepts in natural language processing and different ideas which are implemented in clustering process.

In Chapter 3, we explain in detail different *approaches and the concepts* which are incorporated by different authors regarding our work. We also explain different possible approaches which have been used for different types of scenarios which exist in our work.

In Chapter 4, we explain the *methodology*. We explain how the approach was formulated. We show how we carried out the whole process. We also introduce different techniques of improving accuracy with regards to extracting a good set of features.

In Chapter 5, we show our *base implementation process* and briefly explain how we extracted features from a set of product descriptions and also extracted relationship information from them.

In Chapter 6, we evaluate our approach against *different methods and result*s. We then compare it with state-of-the-art approaches and discuss its findings.

We finally conclude our work in Chapter 7 of this thesis, not only summarizing but also we propose various methods that can be implemented or extended with this work.

2. Background

In this chapter, we go through the basics required to understand this work. In the beginning, we explain background knowledge about software product lines. Subsequently, we will go in depth about how a software product line engineering works, its advantages and its disadvantages and explain few of the related concepts involved. At the end of the chapter, we provide few insights into NLP and its methods which we have used in our work.

2.1 Software Product Line Engineering

SPL are a set of *software assets* that manage a set of common software systems satisfying a particular market. They are built from a common set of core assets. They help the business and the organization by achieving high-quality products with a significant decrease in costs and time. [39]

The main aim of SPL is to *reuse large software assets* to increase the productivity of software systems and thereby decrease the cost of developing those assets from scratch. SPL gives a visual representation of a domain of products, those representations can be further used to identify the commonalities and variable parts between the products. The fact that this process has grown with mass popularity with so many organizations incorporating this process has to be mainly due to the fact that these organizations have seen massive improvements in their productivity, product quality, time to market and customer satisfaction with this approach. The reuse of assets which happens in this line of business will be termed as core assets, mainly because they contain artefacts which are very expensive to develop. These core assets include requirements, architectures, main domain models which can be reused in several products. This would give the flexibility for any organization to not program complex structures within the organization from scratch, rather adopt the reuse approach and customize it accordingly, this is one of the biggest advantages of incorporating software product lines.[49]

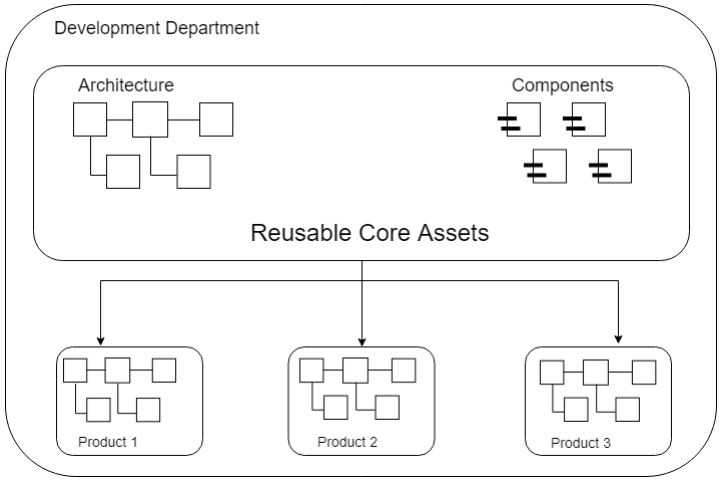


Figure 2.1: Software Product Lines, adapted from [39]

Figure 2.1 shows a overview of how a software product line engineering works and core assets which are always reused to generate new features and create new feature models.

2.1.1 Variants

Variants in SPL deals with the *commonalities existing* in the software product line systems. They on a larger extent are related to the evolution of software assets. Variants are managed as a product line systems which capture the commonalities as well as differences in the software systems. A product often exists with many variants and will be managed by SPL process. From Figure 2.1 we can see that "product 1", "product 2" and "product 3" are all variants of the main Product which uses a common architecture with minimal changes between them.

2.1.2 Features

Features in a Software Product lines describe the functionality of the product. It exhibits what the functionality is and how it is connected to the product. Having many features gives a software product line the flexibility to reuse and extend it easily.

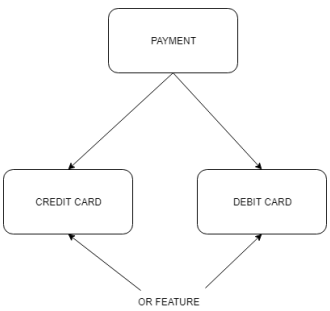


Figure 2.2: Example of a Feature

Figure 2.2 shows an example payment feature having two child nodes.

2.1.3 Feature Model

A feature model shows a *visual representation of all the features* and its connected nodes. From Figure 2.2 you can see the payment node consists of two child nodes "Credit Card" and "Debit Card" and can be visually interpreted as a "Payment" feature model because it explains about all of the features. In simpler terms, we can define a feature model as a model containing the root node (main feature) and all of the child nodes(sub features) which the feature is related or connected to.

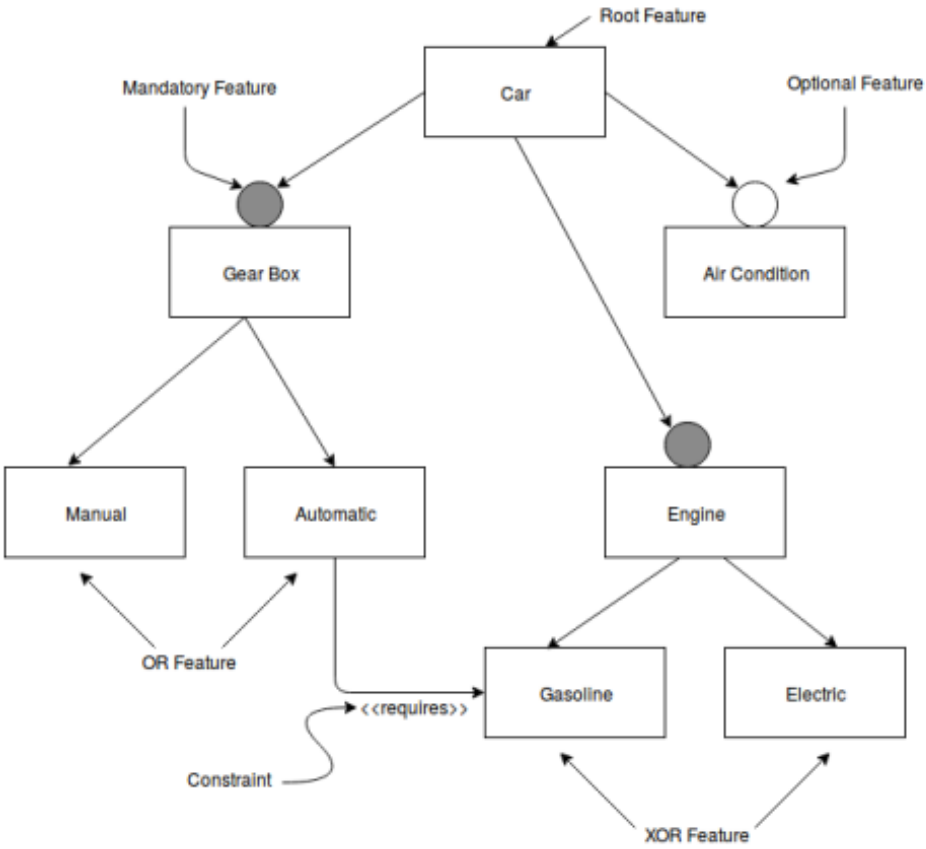


Figure 2.3: Feature Model describing a car SPL,adapted from

The feature model represented in Figure 2.3 shows a complete representation of a feature. The example figure shown above is of a feature of a car. A car has many functionalities each functionality of a car are represented as a feature in the figure. The label representation denoted as a mandatory feature explains that the feature is required by the root feature. The label representation denoted as optional feature explains that if the feature is omitted from the feature model it will not have a bigger impact in making functional decisions to the model.

A feature model is also referred to as a "Feature Tree" or a "Feature diagram" contaning visual representation of all the features in increasing level of details. It summarizes the whole solution by including all of the proper features and their variabilities information(how features are related to each other)

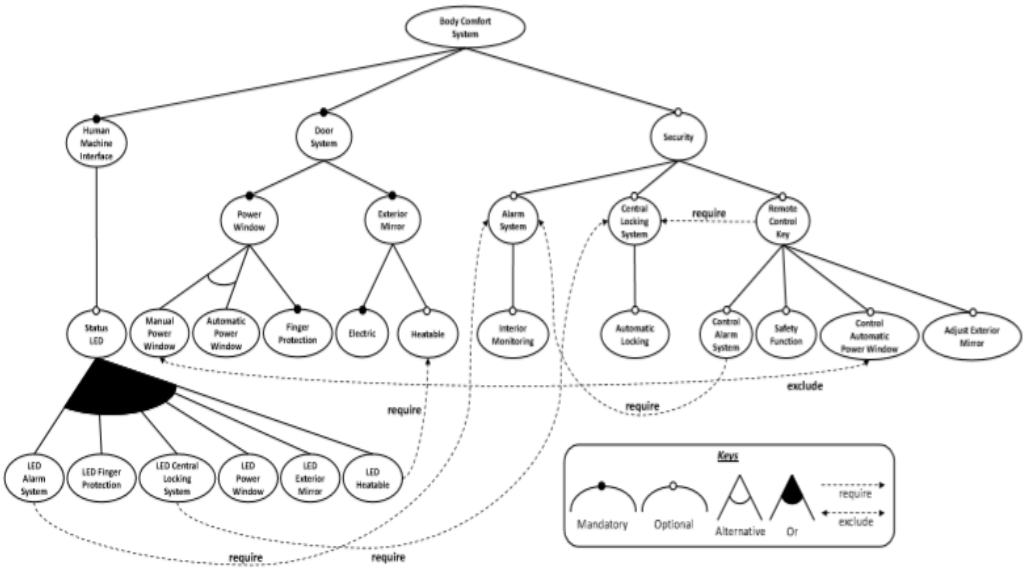


Figure 2.4: Example Feature Tree with variability information

The Figure 2.4 shows an example feature tree with various functionalities grouped in order to represent a whole model. It also visually depicts all the variability information(explained in the next section).

2.1.3.1 Variability Information

The variability information in a feature tree represents the relationship between features and will give us a representation of how a feature is connected (related) to the rest of the features in the Feature Tree. Relationships between features are expressed in various ways,

- [M]Mandatory: Represents, a feature being highly important and needed for the SPL process.
- [O]Optional: Represents, a feature of very low importance (exclusion of this feature will not affect the SPL process)
- [A]Alternative: Between two feature represents, only one sub-feature can be selected.
- [R]OR: Between two features represents, one or more of sub-features can be selected.
- [E]Exclude: Selection of one sub-feature does not require a selection of another feature.
- [I]Include: Selection of one sub-feature requries a selection of another.

All of the variability information can be seen in Figure 2.4.

2.2 Natural Language Processing

A human can write or express his interests, in his own natural language using his own set of sentences either restricted or not restricted by the set of vocabularies. To analyze this and interpret it to make a computer learn and perform complex tasks in quick succession is a challenging task. NLP techniques are one of these techniques which interpret human annotated texts to help computers understand and perform operations on them [54]. Using these techniques we can automate so many things like speech recognition, data summarizations, and we can also find out semantic meanings from it [23].

2.2.1 Text Processing

To find good semantics from a human written language so as to make a computer interact and perform operations on it, we first have to process those texts in such a way that those texts are free from all the noise (stop words such as "the","a","an" and words which are not useful for our tasks) which and can be easily processed. With this our NLP techniques can be easily applied and can be used to obtain good set of results. To process the texts in a way that those texts are free from all the noise, we first have to preprocess them, few of the techniques used in NLP to preprocess are listed below.

2.2.1.1 Stop Words

Words that frequently occur in the document which adds no value to the semantics of the document but are commonly used to build sentences are in no way helpful. Henceforth, removing those words might be one of the first tasks in NLP. Stopwords can be words such as "the", "is", "an", "a". Removing such words can help us process and parse our texts faster.

2.2.1.2 Tokenization

Tokenization is a process of classifying a set of a string of characters [57]. The resulting process are a set of tokens which are then passed further to some form of processing. By tokenizing a set of input stings each token class represents a possible lexeme which can be further used in the semantic analysis[2]. An example of a tokenization process can be seen in Figure 2.5.

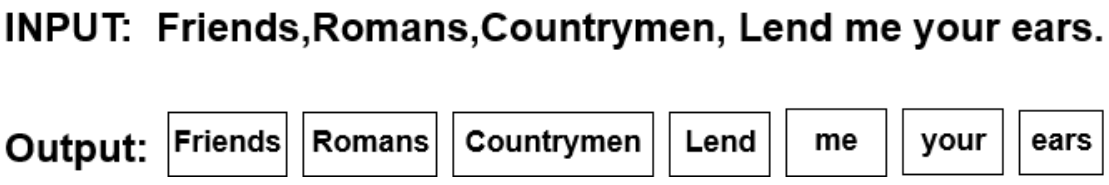


Figure 2.5: Tokenization, adapted from [31]

2.2.2 Word Normalization

Once the words are tokenized, to further process the tokens properly we need to take care of several words which are derived from each other, this form is called inflected language, because the language we speak contains words which are derived from one or the other words. When these words are used often the usage of speech changes. For example, “plays”,“ played”,“playing” all come from the same root word “play”, this degree of inflexion might be higher or lower depending on the words used, hence its always better to bring their root form of the word and use them. Stemming and lemmatization are two techniques which do exactly the same. They take the word and convert them to their root form. If there are words such as “played”, “playing”, “plays” they all will be converted to their root form “PLAY”. The Figure 2.6 depicts a visual representation of this process.

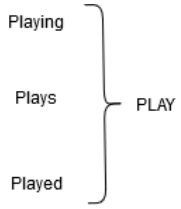


Figure 2.6: Example of Word Normalization

2.2.3 Parts Of Speech

Tokenization plays an important role in treating each word as separate entity, but we cannot get the semantic or the syntactic meanings from just tokenizing the set of words[29]. To get that actual meaning of sentence we need to incorporate certain techniques which provides us with the semantic and syntactic meanings in each and every sentences of a corpus, in that way we can achieve our results efficiently. To do this we incorporate parts of speech tags for each and every tokenised word which helps us by identifying the semantic relations[30]. Few of the parts of speech tags used in our work include,

- [CC]Coordinating Conjunction
- [DT]Determiner
- [JJ]Adjective
- [NN]Noun
- [NNP]Proper Noun
- [VBN]Verb, Past Participle
- [VBG]Verb, Present Participle

Parts of speech can be very important since it can handle both semantic and syntactic analysis. Given a word, obtaining a specific meaning can greatly improve the results.

2.2.4 Parsing

Parsing helps in identifying the semantic meanings of a sentence [33]. It creates some form of internal representation which helps in defining the structure and relation between different sentences. We can then use the parser to create a parse tree that depicts the semantics and syntactic relation between the sentences. A parser is often preceded with the tokenization process to get well structured textual data. The tokenized text is sent to the parser to check whether the tokens form allowable expressions in accordance with the context-free grammar. The parse is then compiled, interpreted or translated to form meaningful output. In NLP parsing is mainly used to break a sentence according to its grammar. A sentence constitutes of a set of noun phrases and a set of verb phrases which separated can help us achieve, analyze or find patterns in the data. A parser is mainly used in this context in our work.

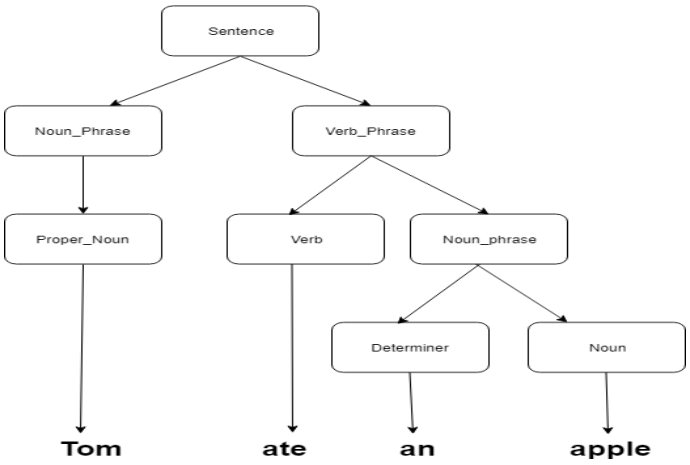


Figure 2.7: Parse Tree

An example of a parse tree can be seen below in the Figure 2.7 which shows a breakdown of all the words in a sentences according to its noun and verb phrases.

2.3 Keywords Extraction

Keywords also called key terms are often used to describe a subject of corpus which it represents. Its often tasked to describe the most relevant information contained in the corpus. Keywords are often chosen from set of words which are mentioned in the original set of collection of texts. There are various ways to extract keywords from the textual requirements [37],methods can be supervised,unsupervised and semi-supervised. Unsupervised methods can be further classified into graph-based[10], ensemble-based methods. We have focused on semi-supervised and unsupervised keywords extraction in our thesis.

2.3.1 Graph-based Ranking Models

An unsupervised graph-based ranking model for text mining, Textrank (TR)[41] has been successfully used in many NLP tasks. Graph-based ranking algorithms have

been successfully used in many applications such as social networks, citation analysis and to analyze and find patterns in the world wide web[40]. They have been crucial in web search technology by providing ranking mechanisms that rely on collective knowledge rather than individual contents. Graph-based ranking algorithms decide on the importance of a vertex(node) in its graph by acting on the global information which has been recursively computed rather than the local vertex which contains specific information regarding that specific vertex. Having this idea of graph-based ranking, TR was designed where the knowledge drawn from the whole corpus is used in making or selecting decisions.

2.3.1.1 Textrank for Keyword Extraction

TR as a graph-based ranking model written based on googles Page Rank [13] algorithm which uses the concepts of term weights to extract keywords form a set of corpus. Page rank uses number of links, similarities and quality of pages to give weights to the page to term how important a page is for its system, similarly TR provides weights to keywords and extracts a set of important keywords from a given natural language text. TR is completley unspervised and uses undirected and weighted graphs to model a set of important keywrods. A vertex of each keyword extracted is created. These verticies are linked to each other with an edge. Weights assigned to each edge dentoes a strength of the connection. An intial value of 1.0 is set to all edges and the algorithm is made to run till the edge weights reach a convergence level. An example of the output of the aglorithm after several iterations can be seen in Figure 2.8.

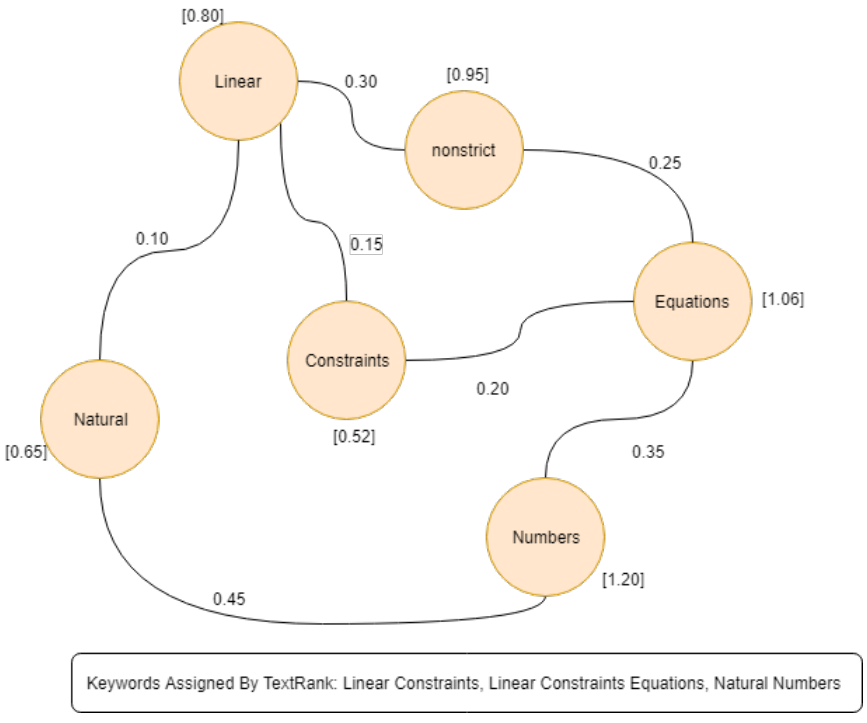


Figure 2.8: Textrank,adapted from [41]

2.4 Similarity Measures

A similarity measure measures the similarity between two data objects. It describes how related or similar the two data objects are in an n-dimensional space. The similarity is highly domain and application dependent, henceforth is very subjective. All the values are normalized in the range [0,1] before calculating the similarity between them so that one object won't dominate the other there by not influencing the distances. Similarity values range form 0 to 1[56]. There are various similarity measures which can tell us how similar two data objects are, but here in our work, we have made use of two main similarity measures which we will explain below.

2.4.1 Cosine Similarity

Cosine similarity is a similarity metric which envisions the features(represented as vectors to form a vector-space model) in our context as points in n-dimensional space. It finds the normalized dot product of two vectors passed by determining the cosine of the angle between them. The two vectors compared are projected in the positive space where the cosine of the angle between them ranges from 0 to 1 [1]. If two vectors tend to have the same orientation between them in the projected positive space their similarity between them will be 1, else the similarity value decreases. It is presented in the following equation as

$$similarity(A, B) = \cos(\theta) = \frac{A \times B}{|A| \times |B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sum_{i=1}^{n} A_i^2 \times \sum_{i=1}^{n} B_i^2} \tag{2.1}$$

The cosine distance can be visualized by the below figure Figure 2.9 which shows two vectors placed in a two-dimensional space, the distance between them can be expressed as the cosine of the angle between the two points.

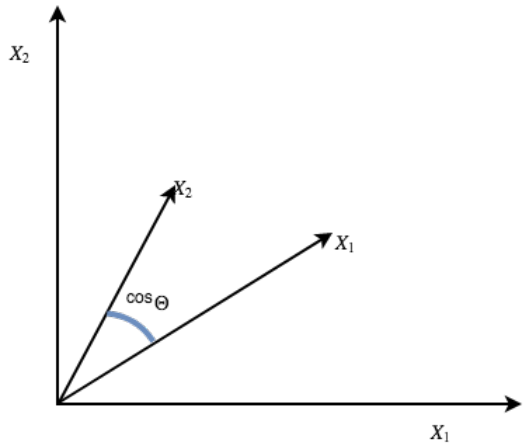


Figure 2.9: Example of Cosine Similarity

2.4.2 Euclidean Distance

Euclidean Distance is One of the most commonly used distance measure which takes into account the simple distance between two vectors in a vector space model [3]. It works best when the data is dense or continuous. It is represented in the following equation as

$$Euclidean(A, B) = \sqrt{\sum_{i=1}^{k}(X_i - Y_i)^2} \tag{2.2}$$

The Euclidean distance can be visualized by the representation in the below figure Figure 2.10 where we can see the two points X1 and X2 plotted in a two-dimensional space. The Euclidean distance between these two points is taken as the length of points connecting them. It can be found by applying the Pythagorean theorem[32].

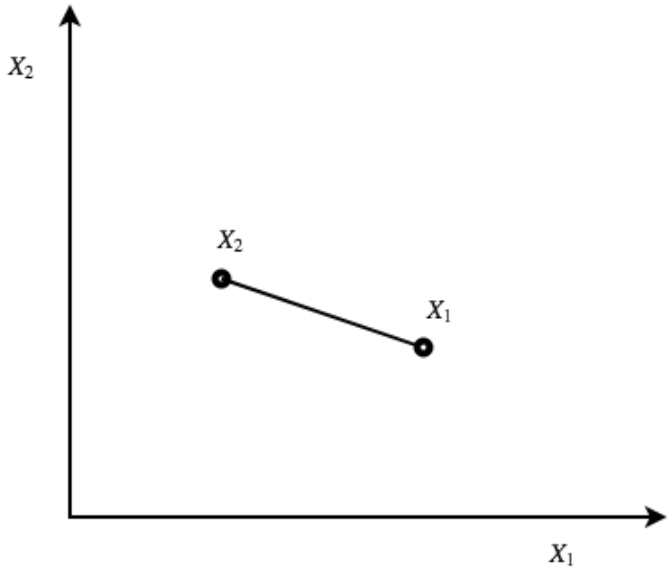


Figure 2.10: Example of Euclidean Distance

2.4.3 Word Embeddings

A type of language modelling technique used in NLP where a word in the vocabulary is mapped on to a real-valued vector, in other terms, they are vector representation of a particular word. It has a capability to capture context, semantic and syntactic similarity and the relationship between other words. These techniques when used is proved to boost the NLP tasks such as semantic analysis.

2.4.3.1 Word2vec

Word2vec is a representative group of models which are used to produce word embeddings, word2vec produces a vector space by taking into account a large corpus of text, where each unique word in the corpus represents a vector in a sparse dimensional space. These are designed in such a way that words which occur to be similar, their vector representations are positioned close to each other in the space.

It utilizes two main models CBOW and Skip-gram to produces the vectors[42]. Here in our thesis, we have used a pre-trained word2vec model given by Google to achieve our tasks. This model is trained on more than billions of words and their vector representations are generated. We make use of this pre-trained model to achieve all our tasks.

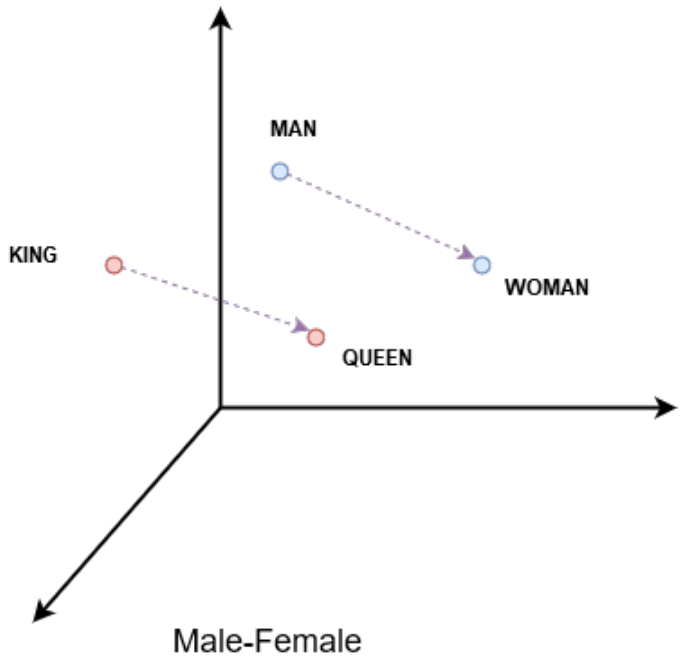


Figure 2.11: Word2vec Similarity,adapted from [42]

2.4.4 Word2vec and Cosine Similarity

We make use of the vectors from the pre-trained word2vec model and compute cosine similarity to identify meaningful closely related features in our work [42]. The main idea of using word2vec vectors with cosine similarity is to identify how similar two features are with respect to their semantic relationships. This helps us to group features which are semantically similar.This is achieved using a function called n_similarity of gensim libraries in our work[52]. The formula is given as,

$$n\_similarity = Cosine\left(\sum_{i=1}^{n} v(f) * \sum_{i=1}^{n} v(f)\right) \tag{2.3}$$

f : represents a feature which can be made up of more than one word.
v(f) : vectors obtained from the pretrained word2vec model.

2.5 TF-IDF Weighting

TF-IDF is a measure often used as a weighting factor in text mining processes. The value TF-IDF produced for a word in the corpus increases proportionally to the number of times the word has appeared in the document and is normalized by the number of documents contained in the whole corpus. In our context documents refer to sentences and the corpus refer to whole dataset. By this, we can get a visualized representation that which words appear frequently in the document. This method is widely used in many of the NLP tasks[1].

2.5.1 Term-Frequency

The normalized Term Frequency (TF) is a measure which tells us how many times (frequency) the word has appeared in the document or sentences in our context. Every single sentence we have might have a varied length, and the word might appear frequently in some sentences and not so frequent in others, henceforth this has to be normalized, hence the term normalized term frequency[1]. This is represented in the equation as follows,

$$TF(T) = \frac{Freq(T)}{q(T)} \tag{2.4}$$

Freq(T) : No of times term T appears in a sentence
q(T):Total number of terms in the sentence
The Equation 2.4 gives us how frequent the term is in the document.

2.5.2 Inverse Document Frequency

The Inverse Document Frequency (IDF) measures how important an actual term is for the whole document. When we compute TF, all terms are considered equally important. But however, we have to take into account that terms such as “is”, “a”, “the” often occur many times in the document and hence we should not give much importance to these terms which might not provide much meaning to the tasks we want to achieve [1]. Henceforth we need to weigh down the frequent terms such as listed above and give more importance to the rare ones, this is done by computing,

$$IDF(T) = log_e(\frac{N(D)}{N(T)}) \tag{2.5}$$

N(D): Total number of documents.
N(T): Number of documents with term T.
The Equation 2.5 gives us the IDF of all the terms in the document.

2.5.3 TF-IDF weight

The TF-IDF weight for the word is given by the product of its Term Frequency by its Inverse Document Frequency[1] and is given by,

$$TF - IDF(T) = TF(T) \times IDF(T) \tag{2.6}$$

The Equation 2.6 gives a normalized TF-IDF weight for the specific word in the corpus.

Natural Language Toolkit (NLTK):

NLTK[1] is a suite of libraries that help in processing and finding patterns in natural language. It helps in performing various NLP tasks such as tokenizing, removing stop words, parsing, named-entity recognition and also can perfom various semantic analysis tasks. It is one of the powerful libraries used extensively for NLP related tasks. [36].

spaCy:

Needing minimal to no tuning spaCy[2] is also a suite of libraries that helps in performing NLP tasks. The tasks performed by NLTK can also be performed by spaCy, but few tasks have relative advantage over performance when compared with NLTK. Both of the above mentioned tools have been used to achieve our tasks.

[1] https://www.nltk.org/
[2] https://spacy.io/

3. Related Work

In this chapter, we discuss other significant approaches that are more similar and related to our research work. In particular, we highlight works that are submitted by other authors on feature extraction and variability extraction approaches.

3.1 Feature Model Generation

Based on *vector space models (VSM)*and *latent semantic analysis (LSA)* author Alves et al. [4] conducted a brief study to determine the similarity between the requirements, but there has not been much work done on the variability information with this paper. Based on the work done by Alves et al., Weston et al. [58] then proposed an extension of the approach. The author developed a tool which generates feature models from natural language requirements. The specifications are first fragmented and clustering approaches are applied to find features. The sizes of the fragments are predefined by the user. With the help of *grammatical pattern-based approaches* the author then finds the variability information. Author Chen et al. in [16] proposed an approach to build feature models from application specifications using *classification of relationship approach*. For each application, the author Chen et al. makes use of the set of functional requirements and model the relationship between them. By clustering the functional requirements the author will then extract features from them. At last, the author merges all of the resulting feature models as one. Several other techniques proposed in Niu and Easterbrook [47],Chen et al.[16],Alves et al.[4] provide helpful insights into the approaches but assume that the existing requirement specifications provide a deeper and rather complete description of the products.

The author Acher et al. [2] proposed a semi-automatic approach to creating a feature model. Using the concept of *formal concept analysis* author Ryssel et al. [53] proposed an approach. The common assumption both the authors make in their respective work is the fact that the descriptions are complete and all the descriptions belonging to a product are available in the requirement specifications.

The concept of *probabilistic feature model* generation was introduced by Czarnecki et al.[19]. Two concepts are introduced in his paper. Features which are Formally described and which strongly indicate *conditional probabilities* between presence of features in configurations are termed soft constraints, whereas the hard constraints which express configuration rules that must be obeyed by all the other configurations or also termed features. The author further extended the probabilistic approach by incorporating a different extraction procedure which solely relies on association rule mining [18]. The work used *association rules* with the confidence of 100% and confidence below 100% but above a certain threshold to add extra information thereby building the hierarchy and creating a feature model.

Based on a few data mining techniques to find common features the author Hariri et al. in his paper [27]proposed a technique to extract common features from products and also relationships among them. A *Incremental Diffusive Clustering (IDC) approach* was incorporated to extract features. The author also used a few association mining techniques to extract relationships among features and give a recommendation. Another approach which used Latent Dirichlet Allocation (LDA) and also an extension of HAC was proposed by Yu et al. in [60] to identify features and their relationships. The relationships which are hidden in natural language texts were extracted and a recommender system based approach was incorporated, the findings from this paper had reported reasonable precision with a very low recall. Additionally, LDA was also used by the authors Guzman and Maalej in [24] to group features that tend to co-occur in the same user reviews of various mobile apps. After having systematically studied different feature extraction approaches the author Bakar et al.[9] made a brief review about the whole approaches presented in [4, 11, 12, 16] . Arora et al.[6] using the help of glossary terms from requirements specifications designed a semi-automatic approach for extracting candidate glossary terms which he terms as features. In [5] the author Arora et al. evaluates various different approaches done by other research workers in the field. Dumitru et al.[20] and Hamza and Walker[26] also proposed a recommender systems type model which can recommend product features. Dumitru et al. uses text mining and probabilistic feature models with cross-tree constraints approach whereas Hamza and Walker uses various NLP based approaches to identify feature terms.

3.2 Variability Extraction

Not many studies have examined how to extract variants from features using NLP requirements, which makes it hard to classify the approaches used. In [34] the author Kumaki et al. used VSM to determine the common features and manually determine the rest of the features. Another approach proposed by authors Ferrari, Spagnolo, and Dell'Orletta [21] identified conceptually independent expressions using Parts Of Speech (POS) Tags, *Lingusitic Filters* and incorporating term weighting approaches to find out the value and frequency of a term in a document to further analyze variabilities. Few of the studies indicated to have used EA-Miner tool[58] to detect and flag words with variability constraints. The work presented by author Yi et al. in [59] deals mainly with variability extraction. The author Yi et al. has not focused on extraction of the features or creation of a feature tree but has taken a well-defined feature model as an input and has extracted variability information from

them. He has treated the task as a supervised problem by using an Support Vector Machine (SVM) classifier and by tweaking its properties accordingly to their task he has extracted two main constraints require and exclude. His work was mainly dependent on the work done by the authors Chang in [15]. The optimization done on the classifier is explained in [28]. In another work done by the authors och Dag, Regnell, Carlshamre, Andersson, and Karlsson in [49] they conclude that *similarity based approaches* to find constraints or variability information prove to be very useful and also generate good results. In the research conducted by the authors Niu and Easterbrook in [46] and also in [48] functional requirements were extracted using lexical affinities and verb direct object relations. They are mainly designed keeping certain case theory in mind, one such being if a verb followed by an object is identified in a sentence it is extracted as a functional requirement. Another study was proposed where the authors Mu, Wang, and Guo in [44] improved the previous work by adopting orthogonal variability models and also by proposing ten semantic cases instead of the original six proposed by the previous work thereby expressing the variability better. The authors Bagheri, Ensan, and Gasevic in [8] used named entity models to label features and further analyze and extract variability by extracting structural relations between features[7, 17, 38]. Authors Niu and Easterbrook,Mu et al. in [47] and [43] respectively have used orthogonal Variability modelling to show the variant features. They have created a model based extraction approach where they define a set of rules on natural language texts to convert a long group of texts into described patterns and thereby analyze them and extract features from them.

3.3 Evaluation Methods

Out of 10 studies, five studies reported to have evaluations done within industrial settings [2, 4, 21, 27, 58], where as the rest [12, 16, 34, 48, 60] of these were done in academia. Out of these 10, there were 5 studies reported to have actual practitioners involvement during the evaluation. There were two other studies reported to have used students to evaluate.

Majority of the studies employed the state of the art quantitative methods to evaluate their proposed approach and additionally use expert opinion to further validate the experiments and methods. The feature extraction method mentioned by Boutkova and Houdek in [12] was tested in a automotive industry. In our research apart from having experiments and studies to evaluate the performance of our proposed approach, it is equally helpful to add metrics to help evaluate the quality. However not all approaches have been reported using metrics to evaluate their model, but few of the metric employed to evaluate the approaches have been,

Purity which is comparing the clusters generated by the clustering algorithm with the ground truth. Each cluster labels generated is matched with the cluster labels of ground truth[27]. Purity takes the values between 0 and 1, where a purity value close to 1 signifies perfect clustering and close to 0 otherwise. *Entropy* is also used which is a measure of average information content, one is missing when one does not know the value of the variable. Precision, recall and f-measure is used in most of the studies, precision is the probability value that all the retrieved features are relevant, whereas recall is the probability of a randomly searched feature is retrieved

in a search.F-measure is a combination of both precision and recall. All of these metrics are used to further evaluate the approaches.

Few of the studies mentioned in [2, 4, 21, 34, 44] are also tested in various other application domains.

4. Methodology

In this chapter, we highlight our approach in detail. We firstly, begin by explaining in detail all the concepts required to understand our approach. Secondly, we give our perspective in the detail as to why these concepts were chosen and incorporated.

4.1 Concept

Our main goal in this thesis is to extract good set of features and find relationships between them which can further help the product line business to make business decisions in a positive way. Our feature extraction approach is mainly designed using python language, few of the third party libraries and the concepts of clustering. In the next sections, we discuss briefly the concepts incorporated in our work, how these concepts work in general and how we used them and optimized a few of them to help design our workflow.

4.1.1 Text Clustering

In general, clustering is a process which mainly focuses on grouping similar texts(in our case sentences within a collection of documents). It is mainly useful in mining text-based resources to find patterns. One of the first steps to cluster any text is to convert them from being text units to representational vectors and form a vector space model. However, there are other ways to actually cluster data too but here in our work, we vectorize the texts since vectors tend to be more efficient to process and allow to benefit from existing clustering algorithms. These vectors can be processed and fed efficiently into clustering algorithm such as for example, K-means clustering algorithm. The result of the clustering process mainly depends on how the vectorization process is carried out(conversion of texts to vectors),where different similarity measures and clustering process often tend to drive the further process.

In our work, we tend to cluster based on sentences, since it fits our use case better and also sentence clustering tends to be more fine-grained and well performed. As

stated the clustering process tends to be dependent mainly on preprocessing of text, the similarity measures used and the clustering algorithms incorporated. Here in our work, we have gone through a few similarity measures and clustering algorithms which we will explain in the below sections.

4.1.1.1 Cosine Similarity using TF-IDF

We incorporated cosine similarity with TF-IDF as one of the similarity measures to find similar sentences. The TF-IDF weight calculation is explained in Section 2.5. This weight reflects the importance of that specific word in the corpus. The cosine distance of these TF-IDF vectors have proven to produce a better set of similarly grouped sentence clusters. As long as the corpus size is not too large, this process tends to be very fast.

4.1.1.2 Word Embeddings

These are low dimensional vectors obtained by training a neural network on a large corpus and then used to predict a word given a context and vice-versa, word embeddings adopt the semantic relationship between the words and they produce semantically similar vector representation for a given text. Here in our work, we have used pre-trained word2vec model released by Google to get vectors of each sentence and later group them according to the similarity. The main advantage of these word embeddings has been accuracy. The results from these word2vec models have proven to be very consistent and very close to human judgment[42].

Few of these methods listed above are helpful to vectorize a given text. There have been few other ways we have incorporated to vectorize a given text thereby attempting to increase the accuracy which will be discussed later in the following chapters.

4.1.2 K-means Clustering

The k-means algorithm is often used to partition a given set of observations into a predefined set of clusters. This clustering algorithm being widely used,tends to be faster comparatively to the rest. Computation of the distance between points, group centres and process to find better cluster centres also tends to be faster. Due to very few computations this algorithm has a complexity of O(n).

The working of the algorithm is fairly simple, to begin with, we select the number of clusters, and the algorithm starts by randomly initializing the cluster centre points. Each data point is classified by computing its distance between that point and every group centre and classifies the point to be placed in the group whose centre is closest to it. Based on this classification the algorithm recomputes the group centre by taking the mean of all the vectors present in that group. The above process repeats until the group centres do not change between iterations. If there are multiple centres, a random one would be chosen.

The main disadvantage of using this algorithm is the lack of consistency. Since the state of initialized cluster centres at the beginning change every time we run the

algorithm, the results are not the same for every iteration. The other main disadvantage of using this algorithm is its dependency on the initially chosen centroids. These centroids could actually end up dividing common data points. These separated points get grouped together if some centroids are more attracted by outliers. This algorithm also favours clusters of the same size and density and it is very sensitive to the outliers.

```
Input: Set the data points in the feature space;
Select the number of clusters
Select K initial centroids
Repeat:
  Assign each point to its closest centroid
  Recompute the positions of k centroids
Until the positions of centroids do not change anymore
Output: K clusters
```

Listing 4.1: K-means algorithm workflow

The Listing 4.1 shows the workflow of K-means clustering process.

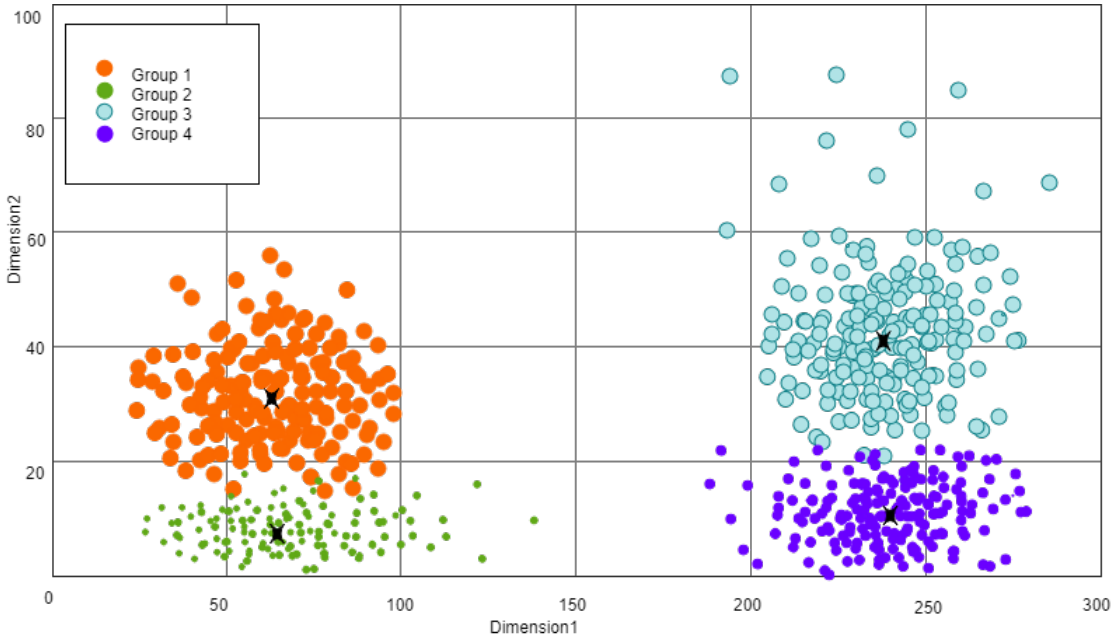


Figure 4.1: Example K-means clustering process

Figure 4.1 shows an example visualization of the output of Kmeans clustering. We can see four groups of clusters formed in two dimension space, having their centroids in the middle. two clusters are closely related while the other two clusters tend to be far apart, which indicates diversity in the data.

4.1.3 Hierarchical Agglomerative Clustering

HAC is one of the bottom-up clustering approaches where clusters are further broken down into sub-clusters and those sub-clusters are in-turn broken down again into sub-clusters untill a threshold breaking point is not set. The resultant hierarchy within the cluster has few good properties, clusters generated in early stages are merged in later stages. In general, the merge and splits are often determined in a greedy manner. The resultant of this algorithm is represented in a dendrogram. The Listing 4.2 code block represents the workflow of the HAC algorithm.

```
Assign each object to a separate cluster.
Compute all of its pair–wise distances
Construct a distance matrix
Repeat:
    Look for clusters with the shortest distance.
    Merge the pairs with the shortest distance.
    Evaluate:
    all distances from this new cluster to all other clusters
    update the matrix
    until the distance matrix is reduced to a single element.
End
```

Listing 4.2: HAC algorithm workflow

At the beginning, each object is assigned to a separate cluster and pair-wise distances are calculated. The distance matrix is built which keeps on updating when clusters with shortest pairwise distances are merged. This process iterates until the distance matrix reach threshold point. The advantages of using this clustering process is that very small clusters are generated which can be very helpful for respective tasks. An ordering approach is followed, which can also help in displaying and visualizing the data better. The disadvantages of this approach is, sometimes the results of clustering may not make much sense due to the objects been incorrectly grouped. There isn't much provision made on the relocation of objects. The use of different distance metrics often gives different results [55]. This clustering process can be visualized in the Figure 4.2

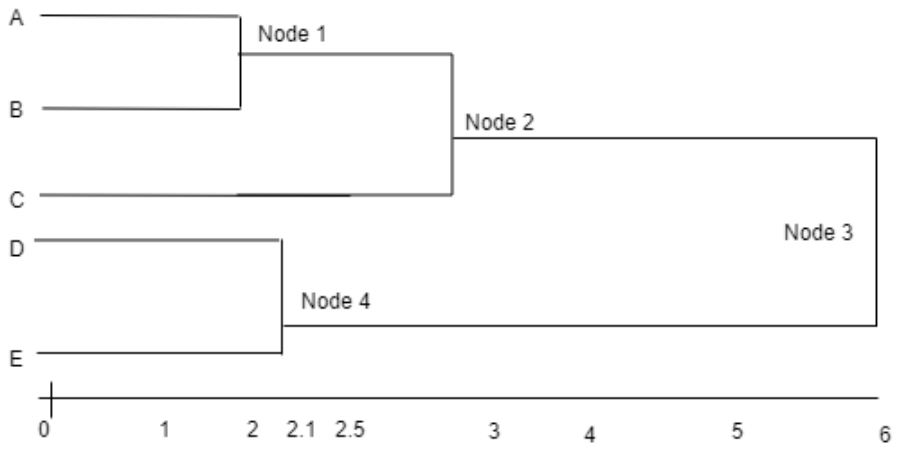


Figure 4.2: Example of HAC workflow process

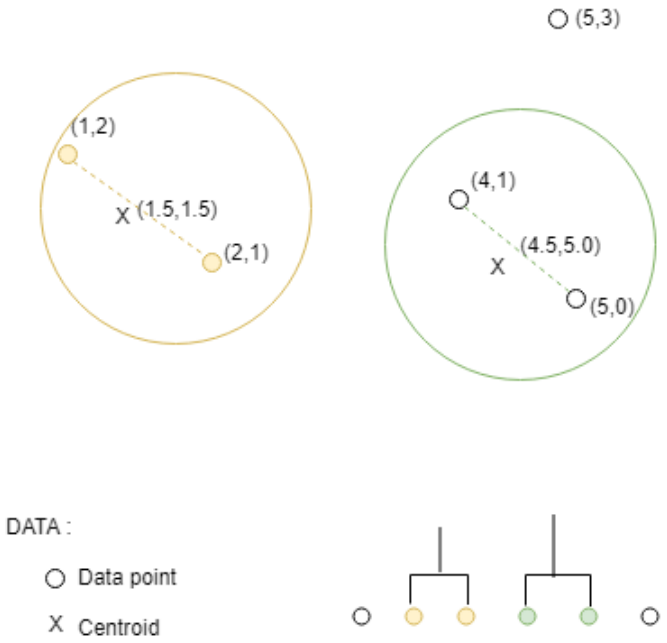


Figure 4.3: Example of HAC clustering Process

4.1.4 Spherical K-means Clustering

A variant of K-means which uses cosine similarity is one of the popular clustering process used o cluster high dimensional data. In SPK each sentence and cluster means are treated as a high dimensional unit vector and the cluster mean is updated only when all of the sentences are clustered[61]. The main idea of spherical K-means is to set the centre of each cluster such that it makes minimal angle and is uniform between components. The points should be consistent with each other. If the vectors in high dimensional space are distributed and the angles are dispersed we cannot compare the angle between them, but if we move all the data so that their tail starts from the same point we can compare those data vectors by their angle and group the similar once accordingly[14]. This approach allows us to contrive vectors which otherwise would have had no geometric dimensions. The main importance of spherical k-means is the project to a unit sphere to account for differences in document length.

4.1.5 K-medoids Clustering

The K-medoids algorithm came into existence as an improvisation of the actual K-means algorithm. Instead of choosing a mean value as a centroid in the clustering process K-medoids initially chooses a new data point inside a cluster as a centroid. The new data point also called a medoid is chosen such that all the other data points in that cluster from the selected data points have a minimal sum of squared error. In this, the clustering efficiency is maintained. The K-medoids algorithm working can be explained as follows, at first initial K medoids are chosen randomly as a cluster centroid, re-assignment of the medoids starts to happen through iterative process, and initially randomly chosen medoids are swapped with other objects if it clustering quality is improved during iterations. This step is repeated until

convergence is achieved Figure 4.4. The cluster quality is derived by calculating the sum of squared error.

```
Select initial K medoids randomly
Repeat
   Re-assignment of objects
   Swap medoid m with another object
   if it improves the clustering quality
Until Convergence criterion is satisfied
```

Listing 4.3: K-medoids algorithm workflow

The Listing 4.3 explains the working of K-medoids algorithm.

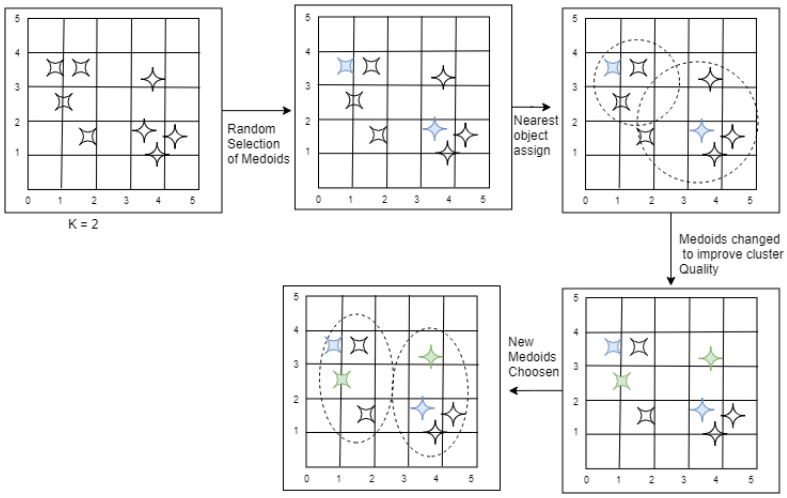


Figure 4.4: K-medoids clustering process,adapted from [50]

One disadvantage of this approach is that the efficiency and computational complexity becomes expensive when this clustering algorithm is applied to larger datasets. Hence sampling based method proposed by the author Hamilton in [25] and also in [45] can be incorporated to deal with the efficiency of this algorithm when it is applied to larger datasets.

4.1.6 Architecture

We have designed a three-layer architecture of our approach as seen in the Figure 4.5 which explains how the data flows and how it is broken down to find useful patterns and features. The three-layer architecture can be explained as,

- Data layer: The data layer mainly resides at the bottom and is the entry point of our application. It constitutes of two parts, the first part is where the data is collected from the source and the second part is where it is preprocessed to make it clean and processable.
- Processing layer: The processing layer resides just above the data layer where the preprocessed data arrives into this layer and the processing of the clean

data begins. The data is further preprocessed according to some techniques incorporated and the results are further passed. This is the core layer of our application where all of the main work happens. The data is clearly branched out to find the patterns and the results are forwarded. It takes input processed data and produces features to the management layer.

- Relationship management layer: The relationship management layer is responsible to find a relationship between features which are given by the processing layer. It takes input in the form of features and mines for the relationship between them. This mining phase can be based on similarity or can be based on certain patterns found in the data.

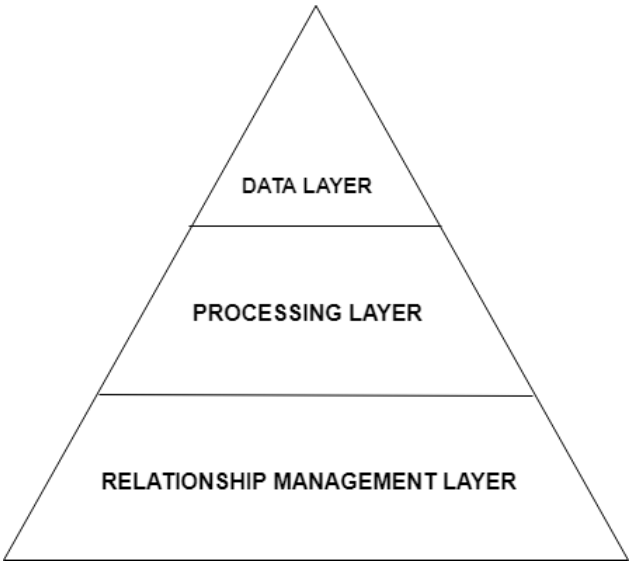


Figure 4.5: Architecture of feature model generation

4.1.7 Steps to achieve our goal

We have followed a five step process to achieve our goal. As seen from Figure 4.6 the five step process are explained as follows,

- Data preprocessing: The Data is acquired from the source and preprocessed properly. Preprocessing involves is transforming the raw data into an understandable format. Data acquired from source often lacks in certain behaviours and trends that if taken might affect the results. They often are incomplete and inconsistent to perform operations on, henceforth we need to preprocess them to remove all sorts of noise
- Vectorization: We make use of certain techniques and implement vectorization on our dataset, where we take each sentence and create a vector out of it. These vectors can be further efficiently applied to various techniques.
- Clustering: We apply various clustering algorithm and analyze the results.

- Analysis and Extraction: The resultant of the clustering algorithm will be a set of similarly grouped sentence clusters. We analyze each cluster from the set of clusters and extract features from them.
- Final Features and relationship extraction: Once the features are extracted we establish the relationships from all of the features and create a final feature model from them.

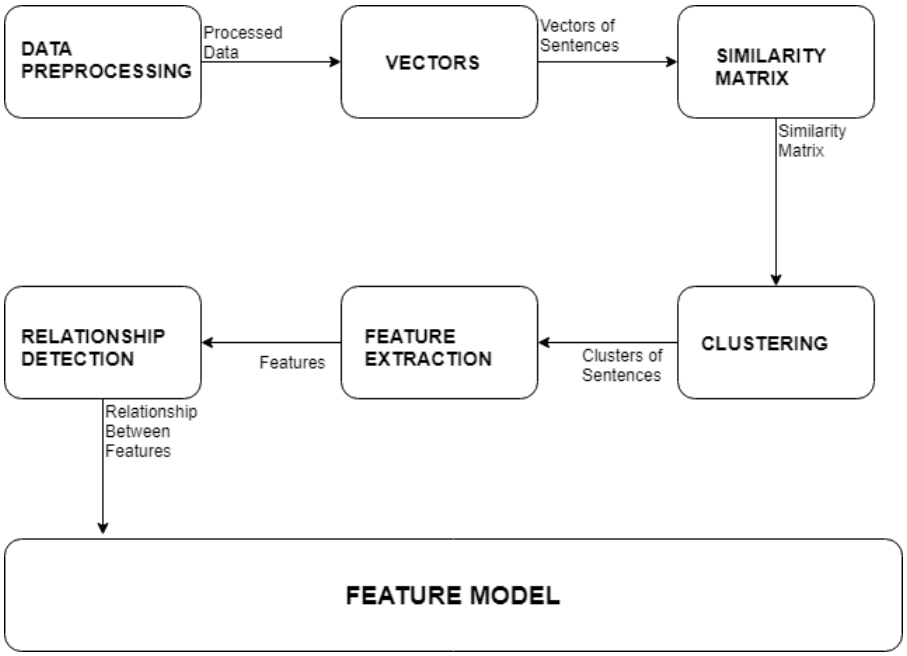


Figure 4.6: Workflow of feature extraction process

5. Implementation

In this chapter, we highlight our implementation details of our feature model extraction approach. We firstly, begin with the architecture and briefly take you through the workflow. Secondly, we look into the practical implementation aspects where we technically and also from a development perspective explain in detail our proposed approach.

5.1 Implementation

Now that we have briefly looked into the major concepts which are essential to understand our work, we head over to the general idea of our practical implementation. Our feature model extraction is mainly programmed in Python Programming Language. NLTK and Spacy libraries have been majorly used to design our approach and achieve several tasks. This section talks about the design and development of our application, which later on we have used for performing evaluations. It talks in detail the architecture and details of all the algorithms which are listed in our application.

5.1.1 Data Preprocessing

In the initial stage, we extract raw unstructured data from our source, in our case this data consists of the functionalities of a car, each and every row is a sentence which explains the functionality of a car. This data is inconsistent and contains noise and needs to be cleaned and hence we apply various types of NLP strategies to clean the data so that we can further apply various techniques on it to find good patterns from them. These strategies used for data preprocessing is mentioned in Section 2.2.1,Section 2.2.1.1,Section 2.2.1.2,Section 2.2.2. Once the data is cleaned and free from the noise, we further process it. The output of this process with regards to our work is the set of sentences which are cleaned and free from all noise and can be further processed.

5.1.2 Vectorization

Now that we have cleaned our data, we further process them with our clustering approach. We convert each and every sentence from our dataset into representational vectors to make our clustering algorithm understand and process our sentences, this method is called vectorization. The main advantage of this process are vectors are very easily processed and complex matrix operations can performed on them with ease. Vectorized implementation is faster than the naive operations and all the similarity and pair-wise distance calculations are done using vectors. Henceforth, we convert our sentences into representational vectors and feed them to our clustering algorithms.

Since there are many different ways in which you can convert a text data into representational vectors, We have experimented with few different vectorization techniques. For a clustering algorithm to give efficient results it needs a good similarity measure to compare vectors and a good choice of clustering algorithm, but for a good similarity measure to retrive good results representational vectors play a huge role. If we give a good set of the accurate vector representation of these sentences we can assume that we get a good similarity value which can greatly improve our clustering process later on. Henceforth, we plan used many different approaches to vectorise sentences of our dataset and later on record the best result from them. Few of the vectorization techniques are explained in the coming sections below,

5.1.2.1 TF-IDF Vectors

This TF-IDF vectors are one such process where we convert our raw preprocessed data into a matrix of TF-IDF features using TF-IDF. We make use of the sklearn libraries of python to achieve this task.

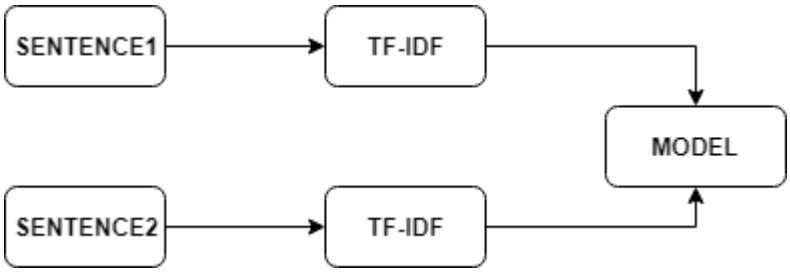


Figure 5.1: TF-IDF Vectorization

The Figure 5.1 shows an example as to how a sentence is converted into representational vectors and then further pushed to a clustering model.

5.1.2.2 Count Vectors

Bag of Words (BOW) vectors, In this approach we make use of frequency of each word occurring in a sentence to actually make a matrix of token counts. We then convert these token counts into representational vectors. We also make use of sklearn libraries to achieve this task.

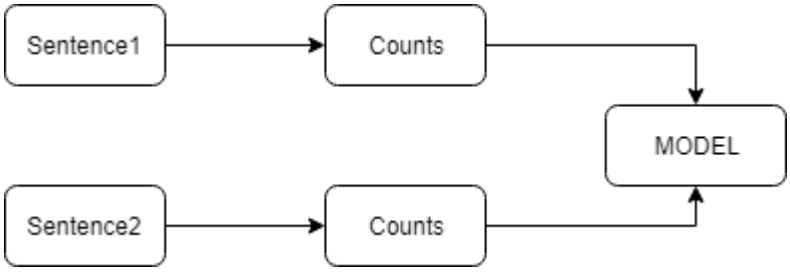


Figure 5.2: Count Vectorization

The Figure 5.2 shows also an example vectors which are created based on BOW model.

5.1.2.3 Hash Vectors

Has vectors,A representtaional vectors derived from a matrix of token occurrences. This approach uses hashing to convert a text into vectors. These vectors are often fast to processes, low in memory and scalable to a very large dataset. But this approach being advantageous also has many disadvantages since it contains no IDF weighting of the word, there are chances that sometimes all the words are termed important.

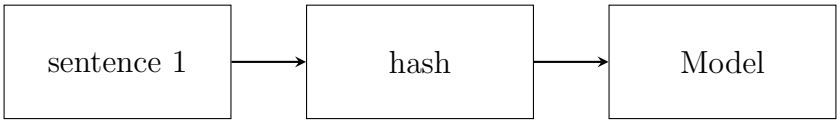


Figure 5.3: Example of Hash vectors

5.1.2.4 Averaging Word2vec Vectors

By our research, we understood that word2vec vectors tend perform better than the normal vectors mentioned[42]. Word2vec is a google embedding model which produces vector representation for a word in a n-dimensional space (see:Section 2.4.3). Since these vector representations were only available for pre-trained words, there was a need to take the advantages provided by these word2vec and create a representational vectors for our sentences. Henceforth, we accessed these vectors generated from word2vec for words and implemented an method which averages all the word vectors of a sentence and gives one vector representation of that sentence. There maybe instances where the words represented might not have a valid vector representation in such cases we term that word as noise and ignore it. We have used a pre-trained google news model consisting of several billion words with its vector representations to achieve this task.The average word2vec formula can be derived as,

$$Vector(A) = \frac{\sum_{w=1}^{n} Word2vec[w]}{\sum_{w=1}^{n} W} \tag{5.1}$$

5.1.2.5 Word2vec and TextRank score

We also came across TR as an important keyword extraction method which ranks keywords based on how important it will be to the corpus, the score from TR will give us an idea as to which terms are important. Having this information combined with the semantic meaning as a representational vectors might influence clustering process positively. Based on this assumption we tend to take dot product of TR scores of each word of our sentence with its word2vec vectors inorder to achieve a representational vectors.

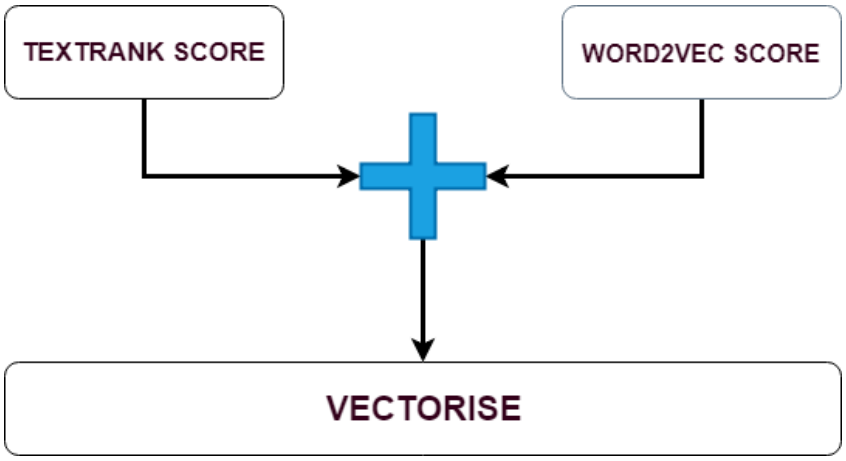


Figure 5.4: TextRank with word2vec

The Figure 5.4 shows TR scores and word2vec representation of wordvectors combining to form a representational vectors of each sentence.

5.1.3 Clustering

Now that we have our vectors from the vectorization process, it is time to further process these vectors to build a similarity matrix and feed them to the clustering process. There are many clustering algorithms which can achieve our tasks but to choose one algorithm needs careful observation of the result obtained and performance issues incurred during the process. We have carefully considered few criteria(see:Section 5.1.3.1) to choose an appropriate clustering algorithm which achieves our task in a desired amount of time, but before the algorithm is made to process our data we mainly focus on building the similarity matrix between all pairs of rows in our data. The similarity matrix we incorporate makes use of cosine distance between the representational vectors achieved from the vectorization process and build a similarity matrix from it. The similarity values will range from [0,1]. 0 being least similar and 1 being most similar in our dataset[9]. After we build our similarity matrix we finally run the clustering process.

5.1.3.1 Criteria for Selecting the Best Clustering Algorithm

Before selecting an algorithm which can be used in our work as a clustering procedure to extract features and relationships, we have take certain criteria into consideration. These criteria, if fulfilled should be able to achieve our desired task within the desired time-frame. A few criteria which we have chosen in our work is explained,

- Time complexity: Different clustering algorithm have different time complexities, accuracy being one main criteria that we intend to achieve, we give equal importance to time complexity in our work. Considering the day to day data growing in huge numbers, if our approach can be applied to a dataset consisting of thousands of records, to deal with them our choosen algorithm has to be efficient and also derive the results in a much lesser time-frame.

- Outliers handler: The algorithm has to handle the outliers in a reasonable way. Having outliers greatly impacts one's result. For example, if we have outliers which are extracted as features there lies a high probability that these features extracted might not make good sense. Dealing with noisy data is also an important criterion and we focus on a algorithm which can handle it in the clustering process, which can set to achieve good clustering results. Which then would impact feature and relationship extraction in a positive way.

These are few of the main constraints which have taken into account before approaching the clustering process, there are other factors too which we have been cautious about before starting the clustering process, these might include the initialization of the clusters for K-means algorithms, centroid assignments, how data is disperesed between clusters, inter and intra cluster distances. We have considered these constraints to select a clustering algorithm which we further use to extract good set of features and relationships and record our results.

5.1.4 Feature Extraction

Once the clustering process ends, the result of the process obtained is analyzed to retrieve a set of features. The result of the clustering process are a set of similarly grouped *descriptor*(each descriptor is nothing but a sentence) also termed clusters. The information inside a cluster describe a specific topic. This topic may act as an abstract feature, and set of these topics can become absatrct features. For example, if we have requirements of huge rows regarding bank payment data we can fairly assume that the clusters formed might be a group of descriptors talking about a specific mode of payment, One such cluster maybe a "credit card" cluster. This cluster which is taking about credit-card is one of the functionality of a requirement specification which can be further mined for features. Having this idea in mind, we create an approach which can mine for features in these clusters thereby extracting good set of features.We achieve this tasks with the help of NLTK libraries and POStags.
In the beginning, we take each formed cluster and breakdown one after another each descriptor present in the cluster. As we know a descriptor is nothing but a sentence containing a set of words and relations between words from our data, we take this descriptors and we classify the words of them as to which parts of speech they belong to. Likewise, we do it for all the words of a sentence, all the sentences of a cluster and for all the clusters formed from our clustering process. We make use of the following Parts of speech tags shown in the Table 5.1 below to achieve this task:

This approach is formulated keeping in mind how a human might extract certain features from a set of sentences when he reads it. If a paragraph is given to an

Numbers	Dataset
[NN]	Noun
[NNP]	Proper Noun
[JJ]	Adjective
[VB]	Verb
[CC]	Coordinating conjunction RB Adverb
[CD]	Cardinal number NNS Noun, plural
[DT]	Determiner PDT Predeterminer
[IN]	Preposition or conjunction NNP Proper noun, singular
[JJ]	Adjective VB Verb, base form
[MD]	Modal verb VBG Verb, gerund, participle
[NN]	Noun, singular or mass VBN Verb, past participle.

Table 5.1: Environments

individual to read and formulate a one line summary, the indicidual often carries his task by collecting information regarding each and ever sentence as he reads through. The brain collectively organizes the important keywords and key phrases which it reads on the flow, and once after completion of the whole paragraphs, the brain tends to mix those important words and formulate a sentence to give a one line summary. Keeping this approach in mind we tend to treat each clusters as a paragraph and thereby focus on extracting important words from these clusters which defines the characteristics or properties of this cluster and also defines certain additional features present in them.
The one keyword which can define the whole cluster can act as a root node also termed as a abstract feature and the other extracted keywords which are termed important and also add value to the cluster can act as their child nodes also termed as features. Using the parts of speech tag and named entity approach we achieve all of these tasks. Few of the POStags used in our work is defined as follows,

- Noun pairs[NN-NN]
- Adjective-Noun pairs[JJ-NN]
- Nouns[NN]
- Proper-nouns[NNP]
- Combination of noun and adjective pairs [NN-JJ]

From each cluster we parse all the descriptors presents and analyze them further to assign appropriate root node or child node characteristics to them. We incorporate a frequency based method to actually further derive the root and child node properties better.

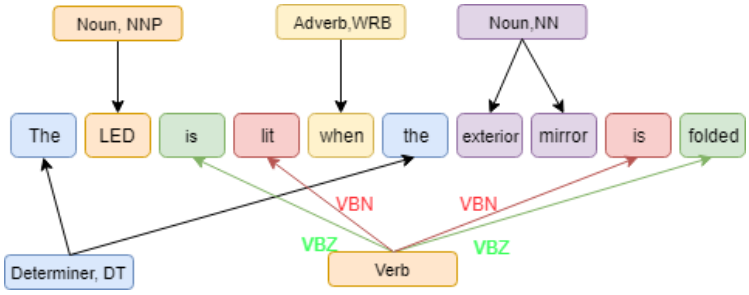


Figure 5.5: Feature extraction process

The Figure 5.5 shows an example of how features are categorized using parts of speech tags. The whole idea behind this is, if there is a keyword which occurs frequently inside the cluster, we assume that the keyword is used often either used to describe that cluster or its characteristics. Henceforth, we can extract those as feature and later decide on whether the extracted keyword can be termed as a abstract or root feature which is that one feature which can be used to descirbe the cluster or child feature which is termed as an actual feature.

We have used Proper Noun (NNP) from the POS tags to extact as keyword from a set of sentences and term them as a root feature if the keyword frequency crosses the threshold set. One of the criteria to select NNP as the root node is that NNP gives us a keyword which is descibes a specific topic, extracting that gives us an impression that the similar grouped sentences of a cluster are of a specific topic and if the cluster has no NNP in them we go with the frequency method to analyse the other extracted features and select a root node. The rest nodes which are not NNP are automatically termed as child nodes for the cluster adn further analysed to select a root node. From the above example presented in the Figure 5.5 we can deduce that for this sentence belonging to a cluster “LED” is extracted as a NNP and can be a root node if the frequency threshold is passed and “exterior mirror” is extracted is a child feature also if the frequency threshold is passed.The frequency threshold is usually set depending on the use case, but in our context we take the maximum frequent keyword and if any keyword is even half as frequent as the maximum frequent keyword in that cluter, they are extracted as a potential child features.

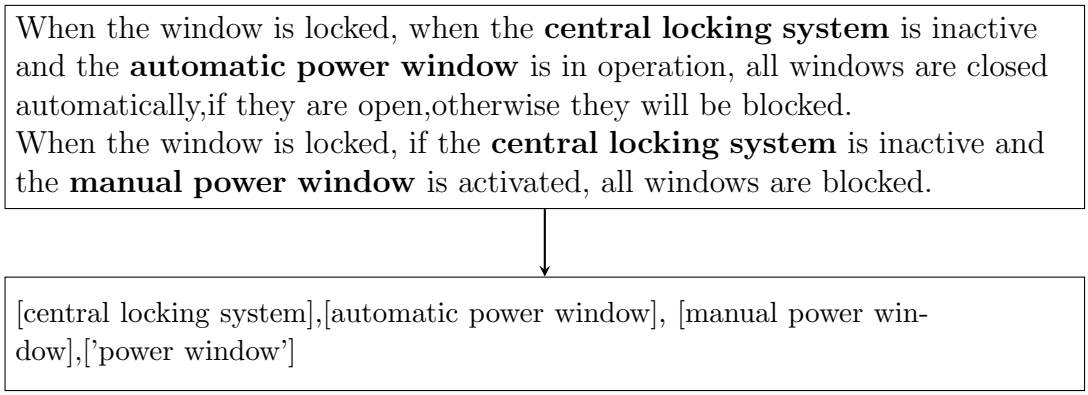


Figure 5.6: An example of features extraction process

The above Figure 5.6 shows a feature extraction process where an clustered set of descriptors are parsed and relevant features are extracted from it. From the figure, we can see that each descriptor is parsed and rules such as "noun" followed by another "noun" (NN-NN), "adjective" followed by a "noun" (JJ-NN) pair and an "adjective" followed by a "noun" and also followed by a "noun" (JJ-NN-NN) pairs are extracted as potential child features. The frequency of these pairs in the cluster will tells us how important these pairs are for the cluster. A frequency threshold is set and only those pairs, which are above the threshold parameter are selected as a feature.

Selection of root feature: To select a feature as a root node we first check if there exists an NNP for a cluster,Since we know now that NNP from a cluster holds a specific value in a cluster, if we find a NNP we term it as a root feature for that cluster and if there are more than one NNP in a cluster, the one which is more frequent is chosen as a root feature else we check set of all extracted feature and see whether there exists a feature which can be used as a representation for this cluster and can be excluded as a feature. For this, we check whether there exists a feature which is a sub-string in another feature. For Example, from Figure 5.6 we can see that the extracted feature "power window" is contained in "automatic power window" and "manual power window" and also passes the frequency threshold set, which tells us that power window can be represented as a root or abstract feature but can be discarded as a child feature since representing it as a child feature does not hold much of a value. Hence, we make this as a root feature for this cluster and make the rest of the features as its child features. If we still are unable find a root feature we simply take the highest frequently occuring feature from the set of extracted feature and make it as a root feature.

5.1.5 Optionality and Group constraints

To detect relationships between the features we have incorporated a mining based strategy where we mine constraints in that specific group of clusters of descriptors.

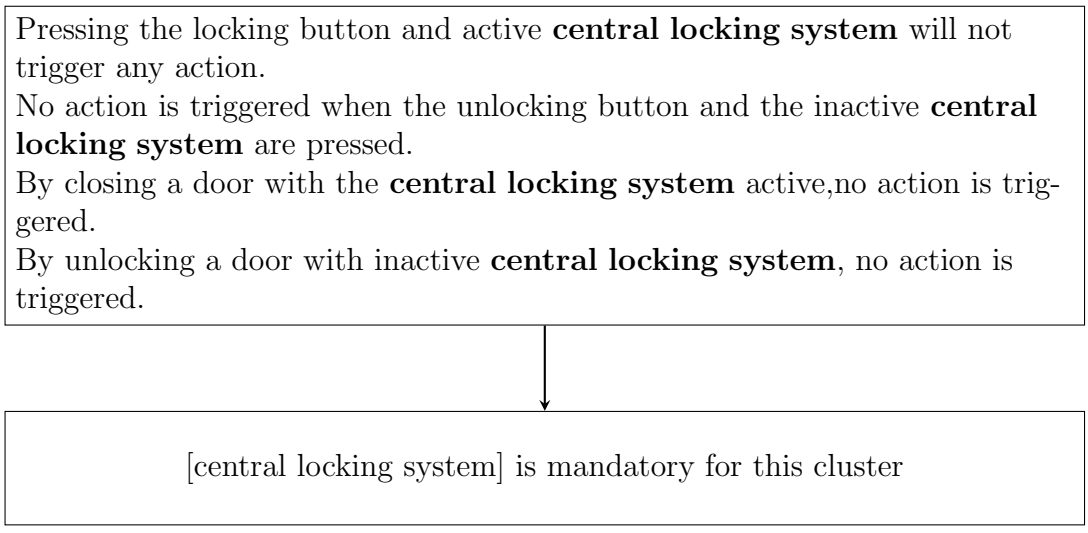


Figure 5.7: Variability Detection-Mandatory

For “mandatory” and “optional” relationship, we traverse through our clusters of descriptors and check whether the feature extracted is appearing in most of the descriptors of the cluster, if true we then make that feature as mandatory, which means that since the feature is present in most of the descriptors this feature is important to the cluster, if this criteria is not met we term the feature as a optional feature. For example in the Figure 5.7 we can see that an obtained cluster contains a set of descriptors which are grouped together based on similarity. From these set of descriptors, feature extraction step is carried out and after we have obtained certain features, if we find a feature or features appearing in maximum descriptors of the cluster, we can say that this feature or these features are important to the cluster and henceforth can term them as mandatory feature for this cluster, because it represents maximum descriptors of the cluster else term them as optional.

For “alternative” and “OR” relationship detection we take a similarity-based approach, we have set a threshold similarity value of around 0.75 and have computed the similarity between obtained features of a single cluster, which in-term will show how similar the two features are, if they are similar we represent it with an “alternative” relation and if they are dissimilar with respect to the threshold value set we represent it with an “OR” relationship between them.

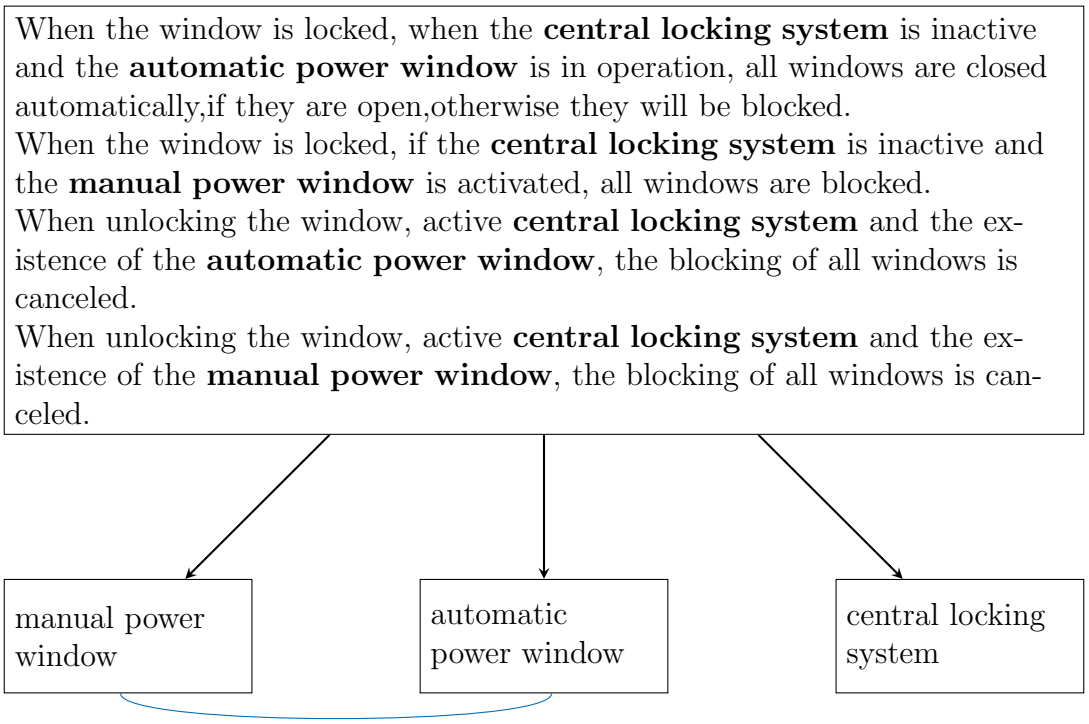


Figure 5.8: Variability Detection-Alternative-Or

The Figure 5.8 represents the “alternative” and “OR” relationship. From the figure, we can see that after the clustering results are obtained, a set of similarly grouped descriptors are further analyzed to extract features from it. After the process of root node selection and child node selection we further analyse them and assign relationships. The “alternative” and “or” relationship assigning process is done by

first grouping all the features of a single cluter and estimating the similarity between them. We set a threshold parameter of around 0.75 and term then two features of a cluster as similar if they pass the set threshold. If they pass the threshold parameter we assign them an "alternative" relationship between them, else we place an "or" relationship between them. When two clusters having the same "NNP" occurs we merge the two clusters based on the proper nouns and give an "OR" relationship between all its features.

Algorithm 1: Feature Extraction

InputRaw unstructured Documents.
OutputSet of features and their relationships
for *all sentence* **do**
    Sentences ← Preprocess()
    Sentences ← Vectorize()
    SimilarityMatrix() ← Build a similarity matrix
    Clustering() ← Run Clustering process
    for *Each Cluster* **do**
        PatternMatching() - Apply set of rules defined
        Term Weights() - Apply Term weights
        Extract Features
        RelationshipDetection() - Detect Relationships between features
    end
    WriteResultsFile(results);
end
Result: Set of high Probable features and their relationships

The Algorithm 1 explains how we select features from a set of raw unstructured texts. From the Algorithm 1 we can incur that the data is first preprocessed, vectorized to perform clustering and then the using ceratin patter matching,term weights and similarity approach will extract all features and relationships.

```
INPUT: Set of all Clusters
For i in AllCluster:
   If AllCluster[i] is root and
        contains features greater than one:
        MandatoryOptional(feature)
        AndOR(feature)
   else:
        return None
        break
endfor
```

Listing 5.1: Relationship detection

The Listing 5.1 describes how relationships are extracted from a cluster. The root node is not parsed for any relationships and all its child nodes are analysed and all the relationships are extracted.

5.1.5.1 Cross Tree Variants

To detect cross tree variants such as exclude and include relationships, we make use of three different approach but mainly use the concepts proposed by these authors [48] and [44]. These concepts explained in our work in Table 5.2 and Table 5.3 are built on Fillmore's case theory [22]. By taking the help of these concepts and orthogonal variability modelling we extract cross tree variants from the set of features.

Approach 1: At first, we extract the generated features from our clustering approach and feed them into a word2vec similarity model as in Figure 5.9, using the word2vec similarity and also by setting up a threshold value, we group similar features according to their semantic similarity.To group the features according to their semantic similarity we use an average similarity method(see: Section 2.4.4). Now having obtained semantically similar features grouped. From there on, we treat each feature extracted as a direct object, this assumption is made taking into account that the feature we extracted is made up of words which describe an action or the object on which it performs its actions on.

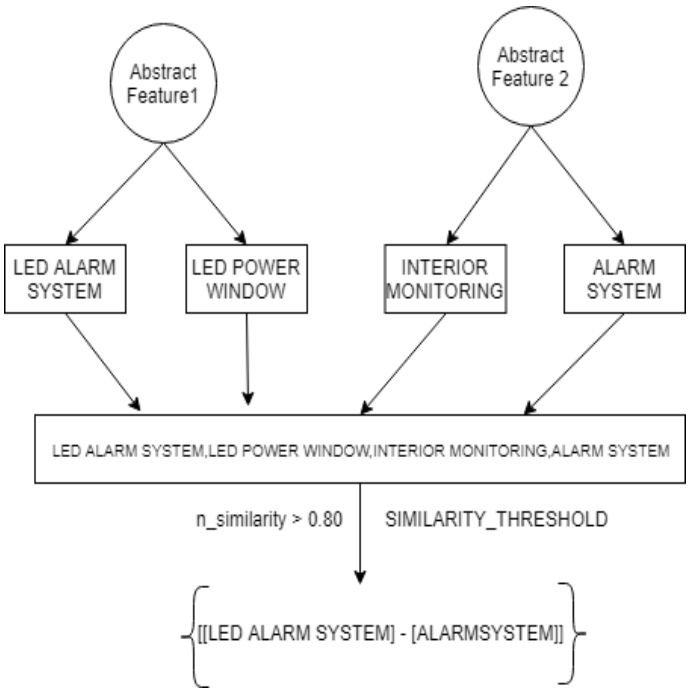


Figure 5.9: Possible Cross-tree variants

From Figure 5.9 we can see the grouped similarity of features, we iterate every group to find a feature(termed as a direct object) to be completely present as a sub-string in any other feature(also termed as a direct object) within the group, if we find any feature as a complete direct object in another feature we then place a require relationship between the two features.

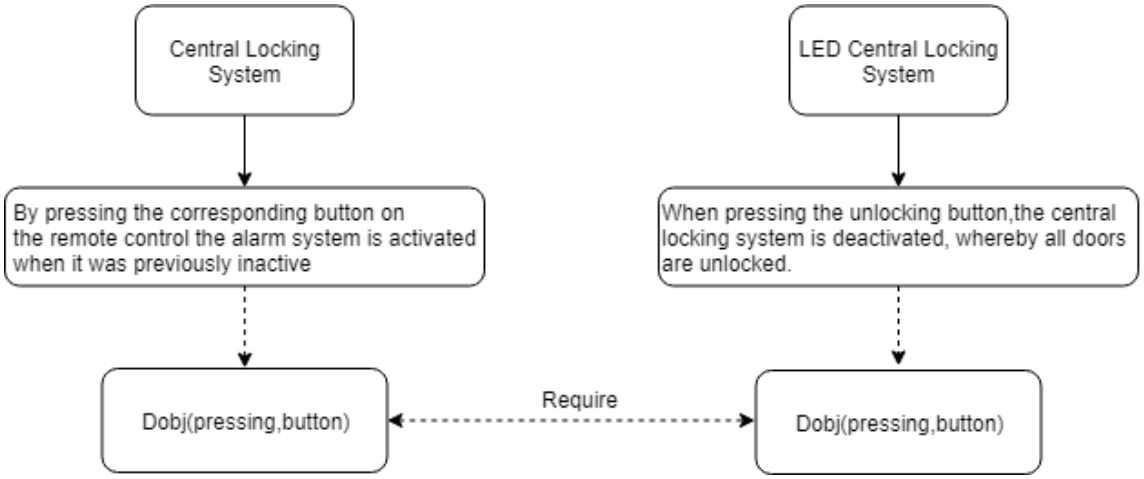


Figure 5.10: Direct Objects Mining

Approach 2: If we do not find any similar direct objects inside the category of similarly grouped features, we still keep the group intact but now we parse each descriptors of the two features and compare their functional direct object pairs(we extract diret-objects from each sentence and comapre them), an example of the direct-object mining can be seen from Figure 5.10, where sentences belonging to two separate features are mined for direct-objects, one is from a feature "central locking system" and another "LED central locking system" if their direct-objects tend to be similar like shown in the Figure 5.10 we assume that the two clusters are interdependent on certain level and we assign them a "require" variability relationship between the two feature pairs. After all the sentences belonging to two features are parsed and compared and their exists no direct-objects between them then we term these two feature as not dependent and assign them an "exclude" relationship. We make use of stanford dependency parser, spaCy dependency parser, certain dependency relations/rules and build a dependency tree to achieve this task[48].

Approach 3: There may be instances where the above direct-Object mining approach might not simply be sufficient enough to extract variability results, in such cases we need to parse sentences and extract more specific attributes from them and then compare against each other in order to mine suitable relationships. Henceforth, we incorporated another relationship mining process wherein we create rules over the extracted sentences. In this approach, we still keep the similarly grouped features and extract sentences from them. Now from each sentence instead of extracting direct-objects and comparing them, we make use of certain dependency relations between the words of a sentence in-order to compare and further extract relations and variability constraints[48]. The table Table 5.2 shows few of the dependency relations used in our work,

Case	Active voice	Passive voice
Agentive	nsubj(*, ×)	agent(*, ×)
Action	nsubj(×, ∗)	agent(×, ∗)
Objective	dobj(*Action*, ×)&*conj*(*objective*, ×)	nsubjpass(*,×)&*conj*(*objective*, ×)
Goal	advcl(Action, ×)&?(*Goal*, ×)	advcl(Action, ×)&?(*Goal*, ×)

Table 5.2: Rules for dependency parsing

Few of the abbreviations used in the Table 5.2 are as follows,

- "nsubj" represents the nominal subject, noun phrase which is the syntactic subject of a clause.
- "dobj" represents direct objects of a sentence.
- "advcl" represents adverbial clause modifier which modifies a verb or other predicate.
- "nsubjpass" represents a passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.
- "agent" represents complement of a passive verb.
- "conj" represents the relation between two elements connected by a coordinating conjunction.
- "rcmod" represents a relative clause modifying the noun phrase.
- "?" represents any syntactic relation with the extracted case .
- "*" represents any number of words.
- "×" represents result which needs to be extracted.
- "&" represents matching conditions.

We have used these cases which contains dependency relations mentioned in the table Table 5.2 to extract relations from the words of sentences in-order to compare them and check for any possible relationships. If their exist a relationship between two sentences belonging to a different feature we term those two features as "require" else we term them as "exclude", all of the sentences belonging to two separate features are compared before any relationships are extracted and if any one sentence belonging to one feature exhibits a dependency over another sentence belonging to a different feature we term these two seperate features as dependent.

As from the table Table 5.2 there are four extracted cases which are applied for every sentence. An agentive case is defined by the one who performs the action.For example, a sentence when parsed into a dependency parser gives a subject name in-terms if nsubj (marks,person1), here the agentive role is typically considered to be the second word typically considered as who is performing the action, hence person1 is given the agentive role and is extracted, likewise the first word of a subject of a sentence is termed as an action word and from example mentioned mark is extracted as an action case.

Few of the cases mentioned in the Table 5.2 can be abbreviated as,

- "Agentive" represents a semantic role or a case of a pronoun which indicates a primary causer.
- "Action" represents a value applied to a situation.
- "Objective" represents an unbiased representation of a fact.
- "Goal" represents an aim or a target to be performed or achieved.

For every sentence we classify its voice depending on the dependency it contains. A sentence is considered passive voice when the dependency parser returns a nsubjpass relation. However, passive voice can occur in different instances henceforth we consider a statement as passive if and only if a sentence contains a nsubjpass and rcmod followed with it. From the Table 5.2 we can incur that case such as agentive incur results when a rule is matched. Relation names such as nsubj represent dependent relations between two words in a sentence given by the parser.

A dependency parser returns the result in the form of dependency relations, in the Table 5.2 we can see the relations such as nsubj, advcl they are some of the relations which are extracted from the dependency parser, we look for these relations from the output of the parser and apply relevant conditions and combinations of these relations in order to give it a definitive case.

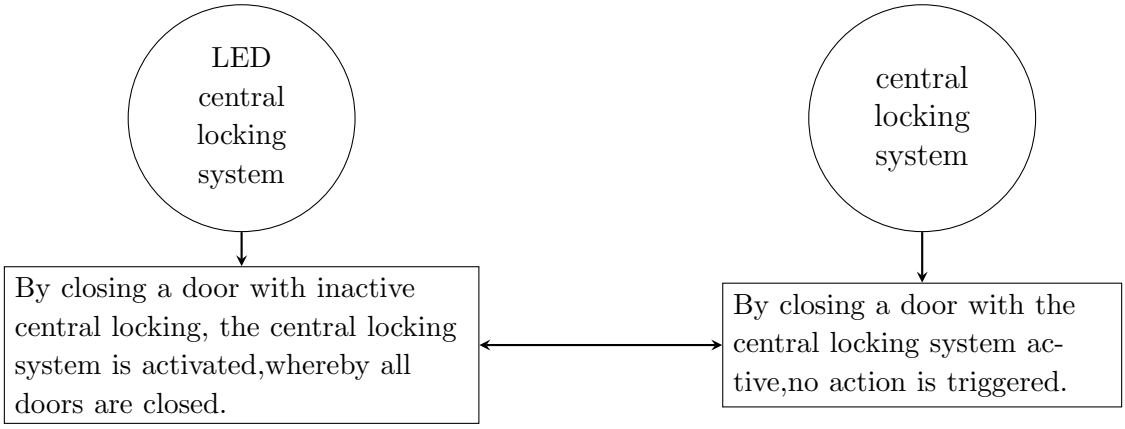


Figure 5.11: Case Mining

For example, an "agentive" case is assigned to a set of words in a sentence if and only if a sentence contains a relation called "nsubj" which is returned by the parser and there exists a second word in the tuple "nsubj". Not much of importance is given to the first word of the tuple returned by the relation "nsubj" for an "agentive" case. For a "goal" case the relation representing "advcl" is first retrieved and the word in the tuple is extracted. From that extracted word we see if there exists any relation containing the extracted word.

If there exists we retrieve all the relations involving the extracted word as words representing the goal case. Likewise, all the other rules are applied to a sentence extracting useful information and there by comparing them with other cases of different sentences and concluding a result.

Case	Sentence1	Sentence2
Agentive	(door)	(door)
Action	(closed)	null
Objective	(closing, door)	null
Goal	(activated)	(triggered)

Table 5.3: Cases for dependency parsing

From Table 5.3 extracted from Figure 5.10 we can see the results of all the cases from the dependency parser. The results are compared on with cases and a decision is taken based on the cases matched. The case extraction process has been recorded to have reasonable accuracy with all the cases traversed over a set of unstructured documents. The author has recorded relatively high accuracy for the "goal" case extraction process, the precision and recall values can be found in [48].

We take an example sentences mentioned in Figure 5.11 as a base of our example inorder to explain this approach. We can see that the two sentences belong to a two different cluster such as "LED central locking system" and "central locking system". We extract certain word relations from sentences of these two features and compare them, these relations are extracted from the Standford Parser and the obtained results are merged to see whether any of the cases mentioned in the table Table 5.2 match. If the cases mentioned in Table 5.2 match for any of these sentences we then give those features representing those sentences a require relationship between them or else we give them exclude relation. This is repeated for all the sentences of a fetaure and the process breaks once it finds atleast one "require" relationship between the two features. This idea presented is formulated on the work done by the authors in [48], we have incorporated the ideas presented by the authors.

Algorithm 2: Cross Tree Variants

Input: Set of features
Output: Cross Tree Variants between Features
for *all featrures* **do**
    Semantic Similarity() ← Features
    ExtractProbableFeatures() ← Set a threshold value and delete other features
    DirectObjects() ← Direct Object Mining between features
    CrossTreeVariants() ← Extracted cross tree variants are applied
end
Result: Set of high Probable features and their Cross variants relationships

The above mentioned Algorithm 2 gives a brief idea as to how cross tree variants are extracted.

ExtractProbableFeatures(): Extracts the probable features where cross tree variants can be applied. It selects few features which are similar and are dependent for further analysis.

DirectObjects(): The direct objects are extracted for features and are compared.

CrossTreeVariants() : After the analysis of all the direct objects, a cross tree variant relationship is applied.After this process we get features and their cross tree variants.

```
INPUT: Set of all Similar features grouped
For all SimilarFeatures:
   if DirectObjects match:
      return True:
   else:
      ExtractSentences()
      CompareDirectObjects()
      if DirectObjects match:
         return True
    return None
    endif
endfor
```

Listing 5.2: CrossTree Variants

The Listing 5.2 shows a workflow of how the direct objects are matched and the results are obtained. It explains a overview of our approach.

6. Evaluation

We briefly highlight the results of evaluating our proposed approach regarding and its performance. We kickoff with our experimental setup in the first section of this chapter, then in the following section we discuss briefly the eligibility criteria,research questions,methodology that was used for evaluating our approach. We record our results from the experiments in the experiment results section. Finally,in the last section of the chapter, we briefly summarize and conclude the results of our evaluation.

6.1 Experimental Setup

In this section, we discuss few of the setups involved in the setup and installations. This section also explains in detail how we performed experiments and evaluated the results and what criteria we have chosen to evaluate them.

6.1.1 Environments

All the experiments have been run on a 16GB RAM having 4 core and a storage space of more than 100GB, there have been two datasets used. Below table Table 6.1 shows the specifications used.

Features	Specifications
RAM	16GB(6GB available for running applications)
CORES	4
STORAGE	100GB

Table 6.1: Environments

6.1.2 Dataset

In our approach We have used three different datasets to validate our approach and have recorded our results. These three datasets belong to a functionality of a car,a e-commerce and a antivirus software systems respectively.

Numbers	Dataset
1	Body Comfort Systems
2	Estore Dataset

Table 6.2: Datasets Used

The dataset of Body Comfort System consists of 91 rows of Natural Language texts of specifications and functionalities of a components of a car. The Estore dataset consists of 100 rows of natural language texts describing the functionalities of a E-Commerce site.

6.1.3 Eligibility criteria

6.1.3.1 Domain Knowledge Evaluation

We manually evaluate our extracted features from a domain engineer perspective, by comparing the extracted features with our ground truth. We manually compare features presented in our ground truth with the extracted features from our approach. We manually check the relationships between the features and draw a rough conclusions as to how good our approach is with the provided ground truth. This perspective is very important because of the fact that from a domain engineer perspective the generated features and relationships have to make good sense.

6.1.3.2 Quantitative Evaluation

In this section we discuss various evaluation approaches used in the field of data mining to measure the qualities of our approach generated. There are several measure which can provide us a perspective of how good our approach has performed. We tend to use "purity" to evaluate cluster results and we tend to use precision, recall and f-measure to evaluate the extracted features and their relationships.

These both combined can give us a broader perspective of how well our approach is generated.

Precision: Precision is one of the three most important evaluation approach we have incorporated[51]. Precision will give us a clear idea as to how good our approach has extracted the features. If the number of features extracted by our model compared with the ground truth is higher we have achieved higher precision. Higher precision often leads to all relevant features successfully extracted. This evaluation metric is often combined with recall. This is given by the formula,

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

Where in our case TP denotes true positive, meaning number of features extracted which are relevant to the ground truth, FP denotes false positive which means number of features extracted by our model which are not relevant with respect to ground truth

Recall: This evaluation measure is used to present the proportion of features extracted by our model considering the features present in the truth set[51]. Higher the value of recall will give an indication that less irrelevant features are extracted with respect to ground truth.

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

Where in our case TP denotes true positive, meaning number of features extracted which are relevant to the ground truth, FP denotes false positive which means number of features extracted by our model which are not relevant with respect to ground truth,FN denotes False negatives that is number of features which are not present in the feature model generated by own process.

Fmeasure: A harmonic mean of precision and recall is one of the ways to find a good trade-off between precision and recall[51]. This is given by the formula,

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

6.2 Research Questions

Our evaluation approach mainly investigates the performance of our proposed feature extraction and relationship detection approach. In particular, we have used some real-world open-source libraries such as NLTK and spaCy which are mainly used to perform NLP tasks in pursuit to explore the following research questions.

- **RQ1.** *How can we design an approach which can produce reasonably accurate features?* Our approach very competently extracts maximum number of correct features. We thus verify this via comparing the results we extract from our approach with the ground truth which has been generated.
- **RQ2.** *How can we achieve variability completeness?* Our approach takes various measures into consideration by not only analyzing the semantic relations between features but also breaking them down into different cases from natural language to help find variability relationships.
- **RQ3.** *How can we evaluate the obtained features and results?* There are many methods which can evaluate a system and its performance, but choosing a rite evaluation method can benefit the decision making process of a system in a positive way, considering and comparing various evaluation methods we have created a workflow of three evaluation metrics which has been extensively used to evaluate our approach.

6.3 Methodology

As a baseline for our evaluation approach we have taken the ground truth of two dataset, one of which was manually created by us. We process our unstructured text by using various NLP tasks we extract features and relationships from them and evaluate them against the ground truth.

Ground Truth Created for E-Store dataset This ground truth Figure 6.1 has been manually created by us by carefully visualising the requirements. The dataset contains 100 rows of estore data where in a feature model of close to 28 features are formed.

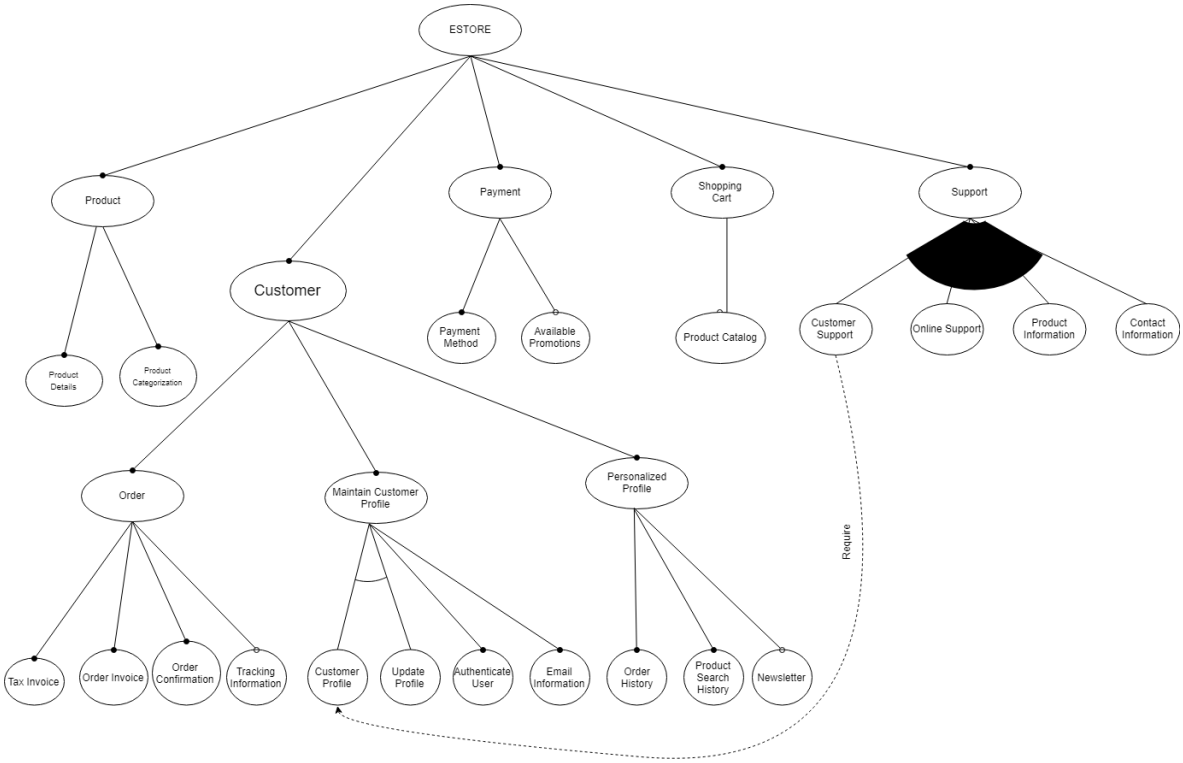


Figure 6.1: Estore Dataset GroundTruth

6.4 Experimental Results

In this section we briefly explain all the evaluation results obtained when we run our approach on the two datasets. We visualize the results and draw conclusions on the results generated, which can further conclude our approach. We first run various different clustering algorithms along with different vectorization techniques on the dataset and record our results. The extracted features obtained from the clustering process are analyzed to further extract relationships. All of the results are recorded.

6.4.1 K-means Clustering

We run Kmeans with various different vectorization techniques and record our results. The selection of number of clusters is done through Silhouette score and the cluster evaluation is done through purity values.

6.4.1.1 Body Comfort Systems Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Means with TF+IDF	0.96	0.63	0.76
K-Means with CountVectorization	0.9	0.5625	0.692
K-Means with HashVectorization	0.833	0.6667	0.74
K-Means with Word2vec	0.77	0.58	0.666
K-Means with textrank+word2vec	0.77	0.4375	0.56

Table 6.4: BCS Data and Kmeans Clustering

The Table 6.4 shows the evaluation results of the K-Means Clustering process. The visual representation of the results obtained from K-means algorithm can be seen in the Figure 6.2 where all the evaluation metrics are presented.

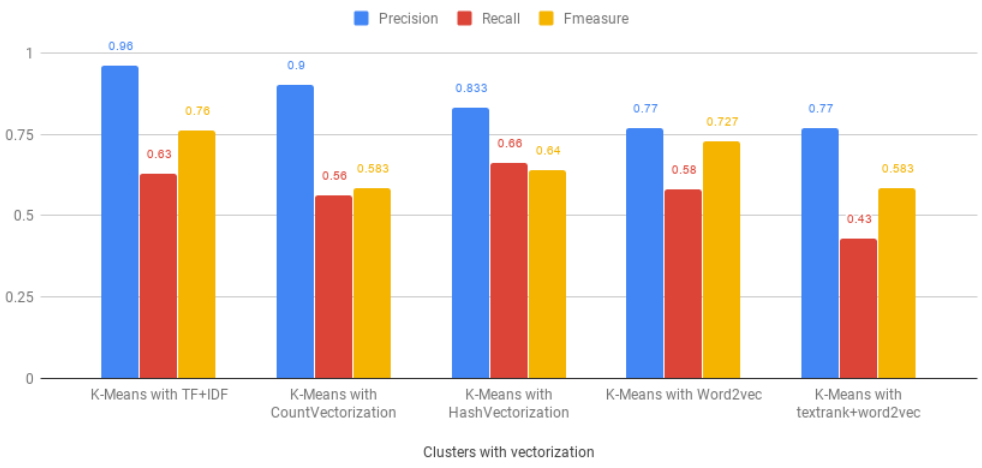


Figure 6.2: K-Means Feature Evaluation

Summary:
The purity values ranged from 0.87-0.71 where TF-IDF had the highest purity values and result of the evaluation for K-Means from Table 6.4 incurs that TF-IDF has performed better than all of the other methods incorporated.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Means with TF+IDF	0.75	0.40	0.529
K-Means with CountVectorization	0.66	0.40	0.50
K-Means with HashVectorization	0.66	0.42	0.516
K-Means with Word2vec	0.58	0.41	0.482
K-Means with textrank+word2vec	0.50	0.37	0.42

Table 6.5: BCS Data and Relationship Extraction

The Table 6.5 shows the evaluation results of relationship extraction process. The visual representation of this process is seen in Figure 6.7

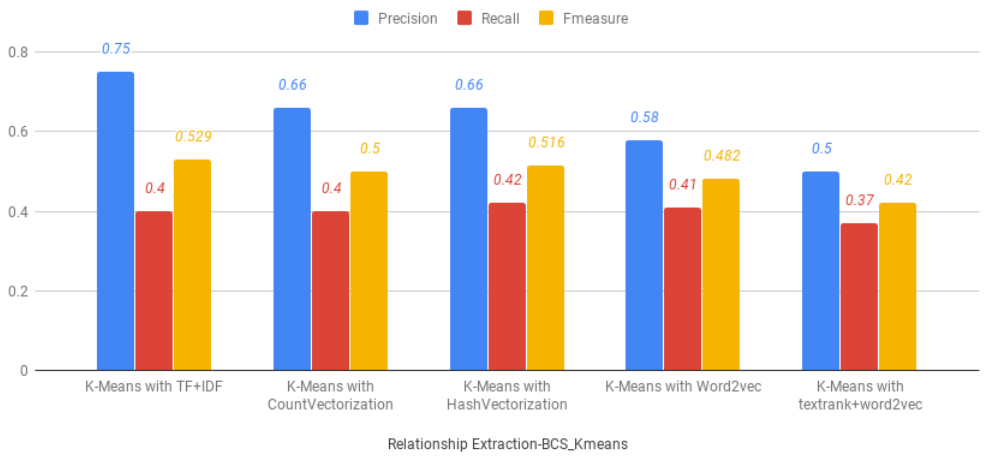


Figure 6.3: K-Means and BCS Relationship Evaluation

Summary:
The result of the evaluation of relationship extraction process for bcs data shows us that TF-IDF vectors produced better performance than the other vectorization techniques. Although hash vectorization and also count vectorization have provided similar or close results.

6.4.1.2 Estore Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Means with TF+IDF	0.80	0.59	0.68
K-Means with CountVectorization	0.75	0.53	0.62
K-Means with HashVectorization	0.75	0.468	0.57
K-Means with Word2vec	0.70	0.50	0.58
K-Means with textrank+word2vec	0.60	0.42	0.50

Table 6.6: Estore Data and Kmeans Clustering

The Table 6.6 shows the evaluation results of the K-Means Clustering process. The visual representation of the results obtained from K-means algorithm run on Estore data can be seen in the Figure 6.4 where all the evaluation metrics are presented.

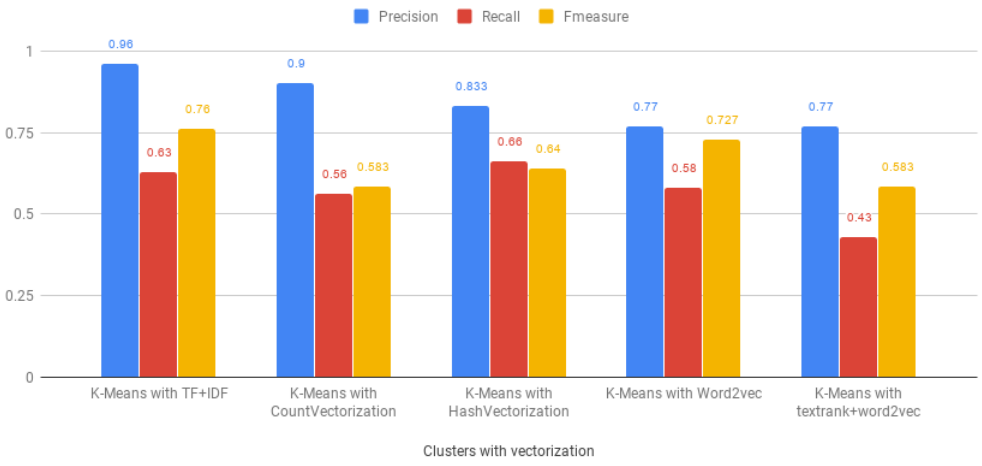


Figure 6.4: Estore K-Means Evaluation

Summary:
The purity values ranged from 0.89-0.73 where TF-IDF vectors had the highest purity value and the result obtained of the feature extraction process for K-Means from Table 6.6 for Eshop dataset concludes that TF-IDF has performed better than all of the other approaches incorporated. We also noticed a significant good performances from count vectorization approach comapred to the TF-IDF vectorization approach.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Means with TF+IDF	0.83	0.45	0.588
K-Means with CountVectorization	0.83	0.416	0.55
K-Means with HashVectorization	0.75	0.36	0.486
K-Means with Word2vec	0.58	0.38	0.466
K-Means with textrank+word2vec	0.58	0.411	0.482

Table 6.7: Estore Data and Relationship Extraction

The Table 6.7 shows the relationship evaluation results of process. The visual representation of this process is seen in Figure 6.9

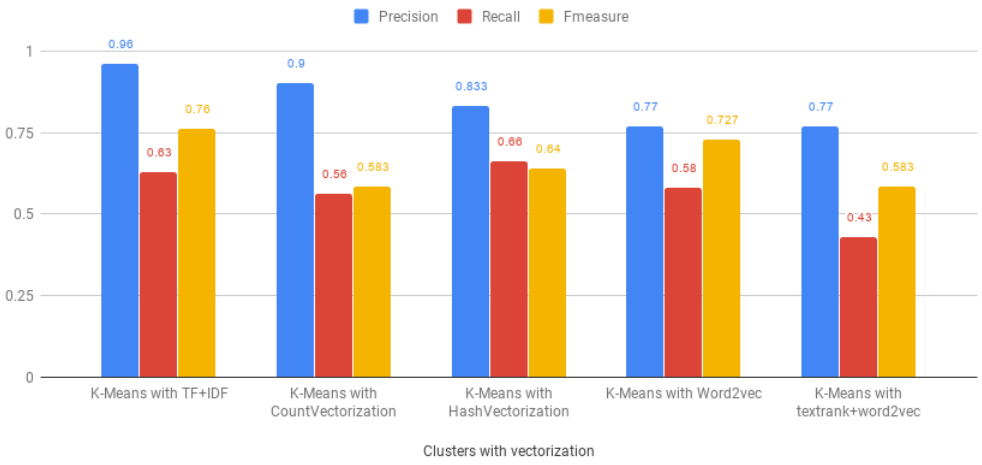


Figure 6.5: Estore data and Relationship Evaluation

Summary:
The result of the evaluation of relationship extraction process seen in Table 6.7 for Estore data shows us that TF-IDF vectors produced better performance than the other vectorization techniques with good precision and reasonable recall values.

6.4.2 K-mediods Clustering

We run Kmedoids with various different vectorization techniques implemented and record our results. The selection of number of clusters is done through Sillhouttee score.

6.4.2.1 Body Comfort Systems Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Mediods with TF+IDF	0.77	0.50	0.60
K-Mediods with CountVectorization	0.88	0.57	0.69
K-Mediods with HashVectorization	0.66	0.50	0.57
K-Mediods with Word2vec	0.77	0.583	0.66
K-Mediods with textrank+word2vec	0.66	0.4	0.50

Table 6.8: BCS data and Kmedoids Clustering

The Table 6.8 shows the feature extraction evaluation results from clustering process.The visual representation of the results obtained from K-medoids algorithm can be seen in the Figure 6.6 where all the evaluation metrics are presented.

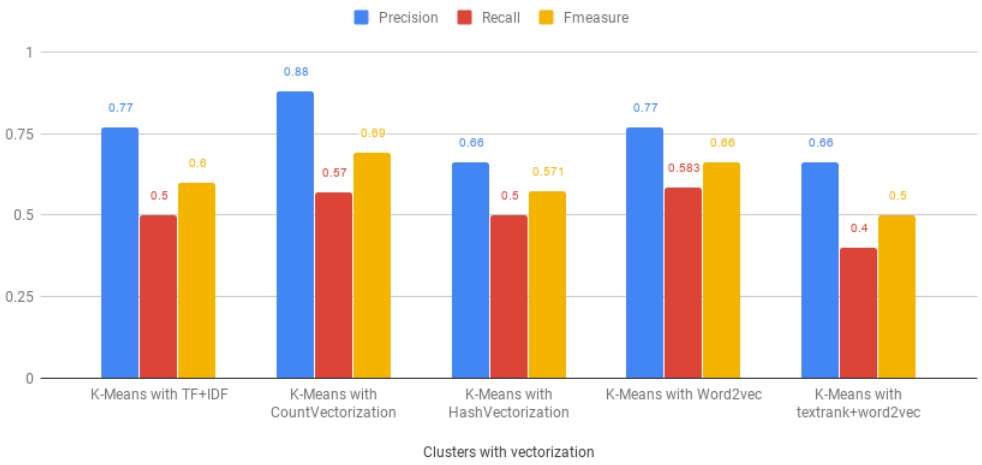


Figure 6.6: kMedoids Feature Evaluation

Summary:
As we can depict from the Table 6.4 that although TF-IDF has performed better but comparatively count vectorization process has also shown good results.The purity values for this dataset ranged from 0.79-0.69.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Mediods with TF+IDF	0.66	0.42	0.516
K-Mediods with CountVectorization	0.66	0.47	0.551
K-Mediods with HashVectorization	0.58	0.36	0.451
K-Mediods with Word2vec	0.75	0.42	0.545
K-Mediods with textrank+word2vec	0.58	0.38	0.466

Table 6.9: BCS Data and Relationship Extraction

The Table 6.9 shows the evaluation results of relationship extraction process.The visual representation of the results obtained from K-medoids algorithm can be seen in the Figure 6.7 where all the evaluation metrics are presented.

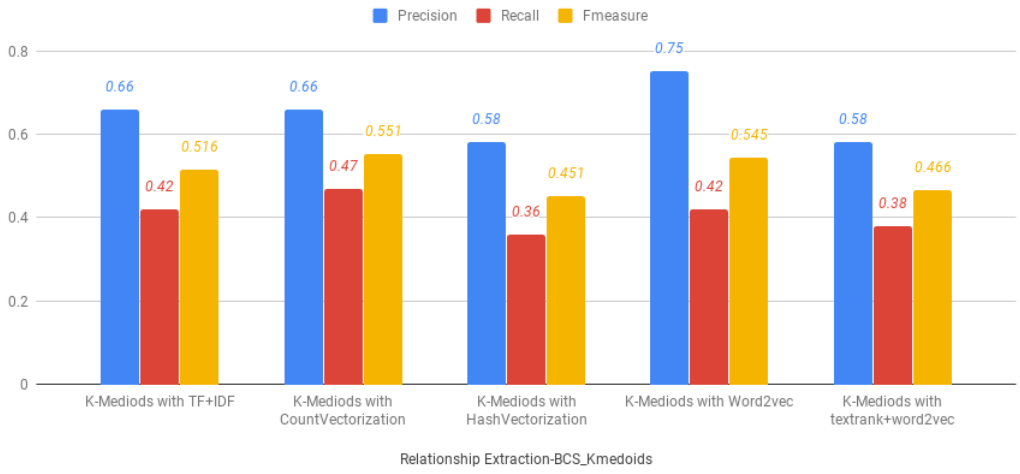


Figure 6.7: Kmedoid Relationship Evaluation

Summary:
The result of the evaluation of relationship extraction process seen from Table 6.5 shows us that we were able to extract many features that were present in the ground truth but our approach also extracted few irrelevant features which might not have been needed by a domain expert, hence there has been a drop in the recall values whereas the precision values remain good. With this clustering process, TF-IDF has performed reasonable well with count and avergae word2vec vector approach performing better.

6.4.2.2 Estore Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Mediods with TF+IDF	0.85	0.70	0.772
K-Mediods with CountVectorization	0.80	0.69	0.741
K-Mediods with HashVectorization	0.75	0.652	0.697
K-Mediods with Word2vec	0.75	0.55	0.638
K-Mediods with textrank+word2vec	0.70	0.50	0.583

Table 6.10: Estore Dataset and Kmedoids Clustering

The Table 6.10 shows the feature extraction evaluation results from clustering process.The visual representation of the results obtained from K-medoids algorithm can be seen in the Figure 6.8 where all the evaluation metrics are presented

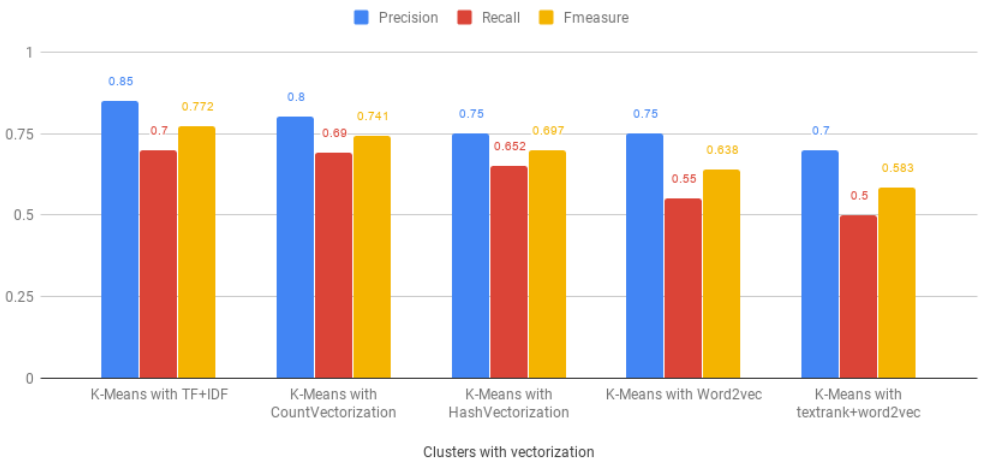


Figure 6.8: Kmedoids Estore Feature Evaluation

Summary:
Result of the feature extraction process for K-Medoid clustering process for EStore dataset states that TF-IDF with count vectorization approaches have performed better than all of the other methods incorporated. The purity values recorded for this dataset ranged from 0.83-0.76.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
K-Mediods with TF+IDF	0.75	0.50	0.60
K-Mediods with CountVectorization	0.75	0.47	0.58
K-Mediods with HashVectorization	0.75	0.42	0.54
K-Mediods with Word2vec	0.66	0.47	0.55
K-Mediods with textrank+word2vec	0.66	0.40	0.533

Table 6.11: Estore Data and Relationship Extraction

The Table 6.11 shows the evaluation results of clustering process.The visual representation of the results obtained from K-medoids Evaluation can be seen in the Figure 6.9 where all the evaluation metrics are presented.

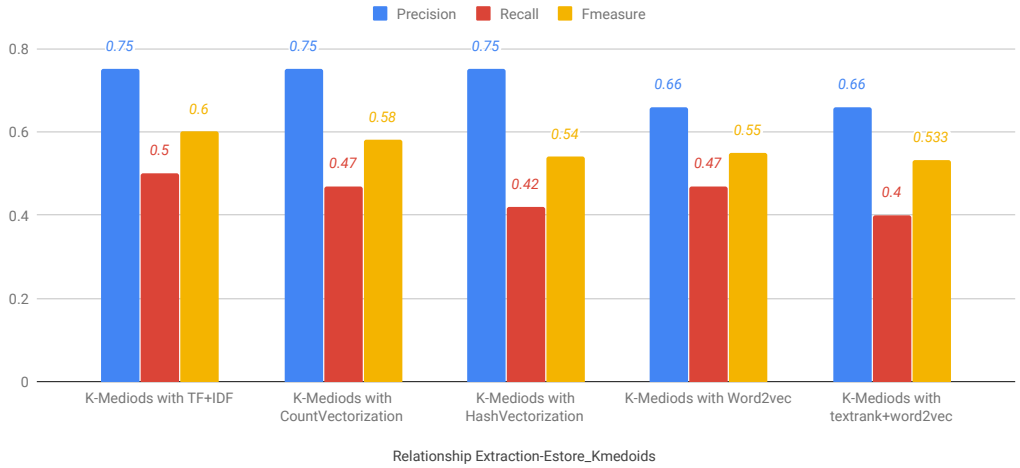


Figure 6.9: Kmedoid Estore Relationship Evaluation

Summary:
Result of the relationship extraction process seen from Table 6.11for K-Medoid clustering process for EStore data states that TF-IDF with count vectorization approaches have performed better than all of the other methods incorporated.

6.4.3 Spherical K-means Clustering

We run SPK with various different vectorization techniques implemented and record our results. The number of clusters are dervied through Sillhoutte score.

6.4.3.1 Body Comfort Systems Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
SPK-Means with TF+IDF	0.888	0.50	0.637
SPK-Means with CountVectorization	0.88	0.562	0.684
SPK-Means with HashVectorization	0.88	0.571	0.695
SPK-Means with Word2vec	0.88	0.615	0.72
SPK-Means with textrank+word2vec	0.77	0.53	0.636

Table 6.12: BCS Data and SPK

The Table 6.12 shows the evaluation results of clustering process.The visual representation of the results obtained from Spherical-Kmeans algorithm run on BCS dataset can be seen in the Figure 6.10 where all the evaluation metrics are presented.

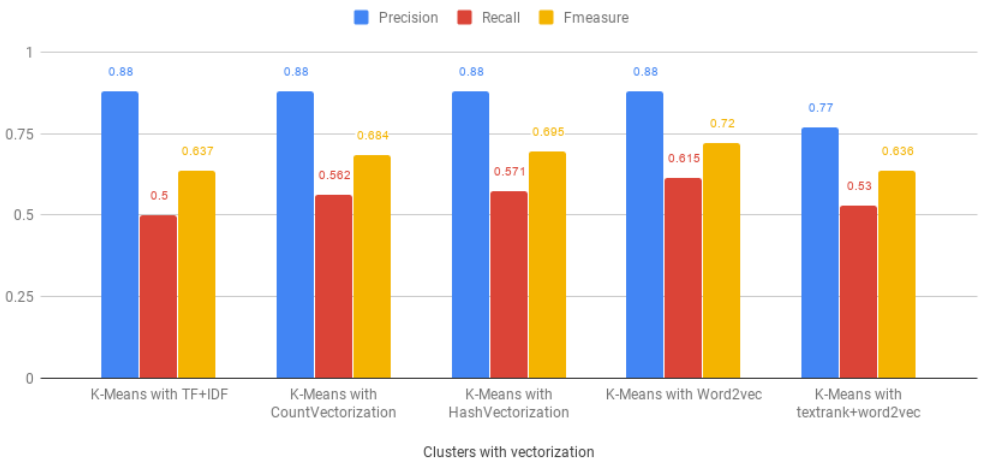


Figure 6.10: SPK and BCS Feature Evaluation

Summary:
Result of the feature extraction evaluation seen from Table 6.12 states that TF-IDF vectors and word2vec vectors have played a significant role in the process.The purity values recorded for this process ranges from 0.88-0.70.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
SPK-Means with TF+IDF	0.66	0.47	0.551
SPK-Means with CountVectorization	0.66	0.50	0.571
SPK-Means with HashVectorization	0.75	0.56	0.64
SPK-Means with Word2vec	0.66	0.50	0.571
SPK-Means with textrank+word2vec	0.58	0.41	0.482

Table 6.13: BCS data and Relationship Extraction

The Table 6.13 shows the evaluation results of clustering process.The visual representation of the results obtained from SPK algorithm run on BCS dataset can be seen in Figure 6.11where all the evaluation metrics are presented.

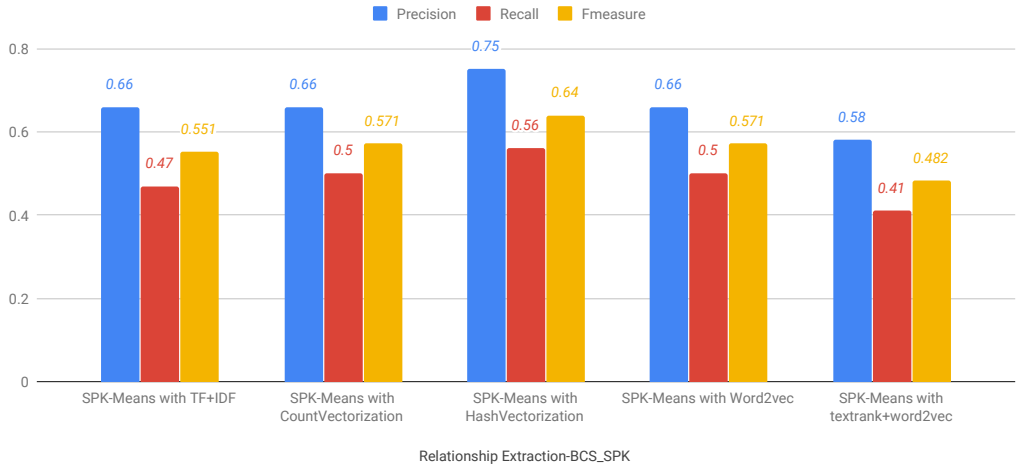


Figure 6.11: SPK BCS Relationship Evaluation

Summary:
The result of the relationship extraction process seen from Table 6.13 incurs that our approach was able to extract many features that were present in the ground truth but our approach also extracted few irrelevant features which might not have been needed by a domain expert.

6.4.3.2 EStore Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
SPK-Means with TF+IDF	0.80	0.65	0.73
SPK-Means with CountVectorization	0.80	0.571	0.666
SPK-Means with HashVectorization	0.75	0.576	0.652
SPK-Means with Word2vec	0.8	0.592	0.6808
SPK-Means with textrank+word2vec	0.65	0.44	0.5306

Table 6.14: Estore Data and SPK

The Table 6.14 shows the evaluation results of clustering process.The visual representation of the results obtained from Spherical-Kmeans algorithm run on Estore dataset can be seen in the Figure 6.12 where all the evaluation metrics are presented.

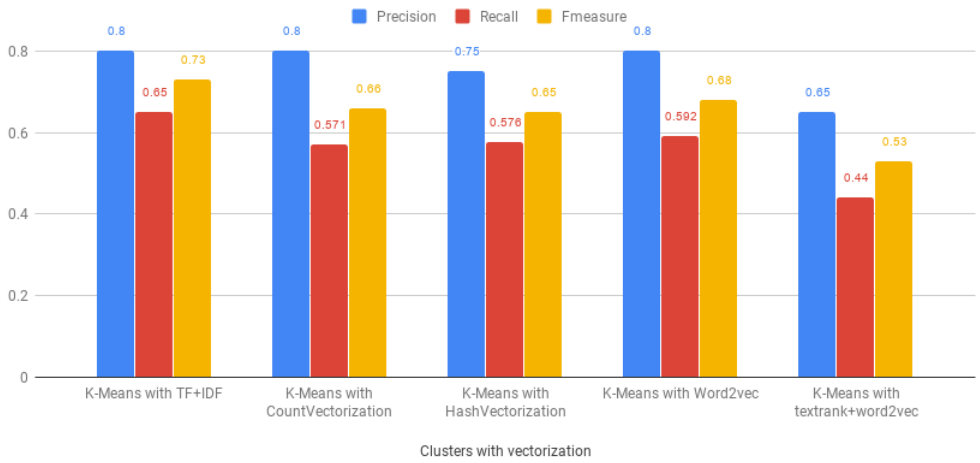


Figure 6.12: Estore Data and Feature Evaluation

Summary:
Result of the evaluation states that TF-IDF vectors have played a significant role in the process. The purity values recorded for this process ranges from 0.88-0.65.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
SPK-Means with TF+IDF	0.75	0.52	0.620
SPK-Means with CountVectorization	0.66	0.470	0.551
SPK-Means with HashVectorization	0.66	0.44	0.53
SPK-Means with Word2vec	0.75	0.50	0.60
SPK-Means with textrank+word2vec	0.66	0.421	0.516

Table 6.15: Estore data and Relationship Extraction

The Table 6.15 shows the evaluation results of clustering process.The visual representation of the results obtained from Spherical-Kmeans algorithm run on Estore dataset can be seen in the Figure 6.13 where all the evaluation metrics are presented.

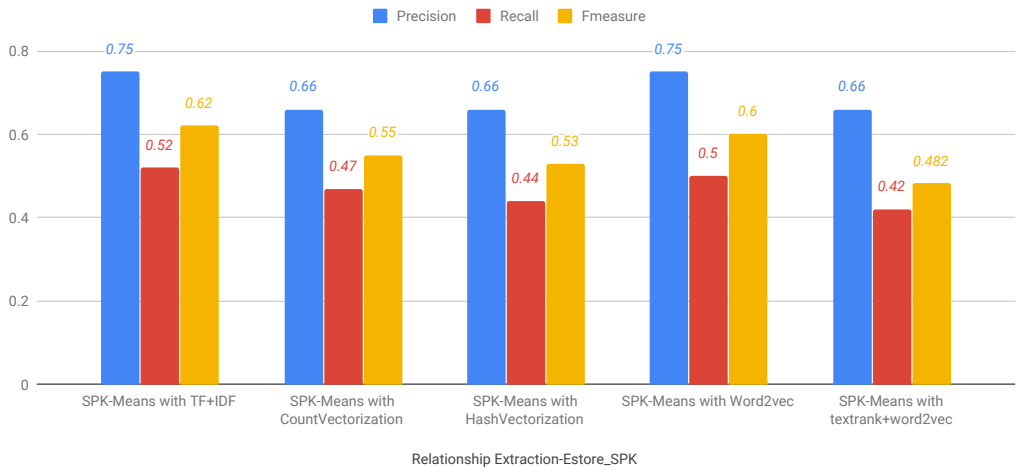


Figure 6.13: SPKmeans Estore Relationship Evaluation

Summary:
Result of the evaluation from Table 6.15 states that our approach was able to extract many features and played a significant role in obtaining good performance where as the word2vec vectors have also been useful for thjis approach.

6.4.4 Hierarchical Agglomerative Clustering

We run HAC with various different vectorization techniques implemented and record our results.

6.4.4.1 Body Comfort Systems Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
HAC with TF+IDF	0.88	0.66	0.761
HAC with CountVectorization	0.77	0.46	0.583
HAC with HashVectorization	0.88	0.50	0.64
HAC with Word2vec	0.88	0.615	0.727
HAC with textrank+word2vec	0.77	0.46	0.583

Table 6.16: HAC Clustering and BCS Data

The Table 6.16 shows the evaluation results of clustering process.The visual representation of the results obtained from HAC algorithm run on BCS data can be seen in the Figure 6.14 where all the evaluation metrics are presented.

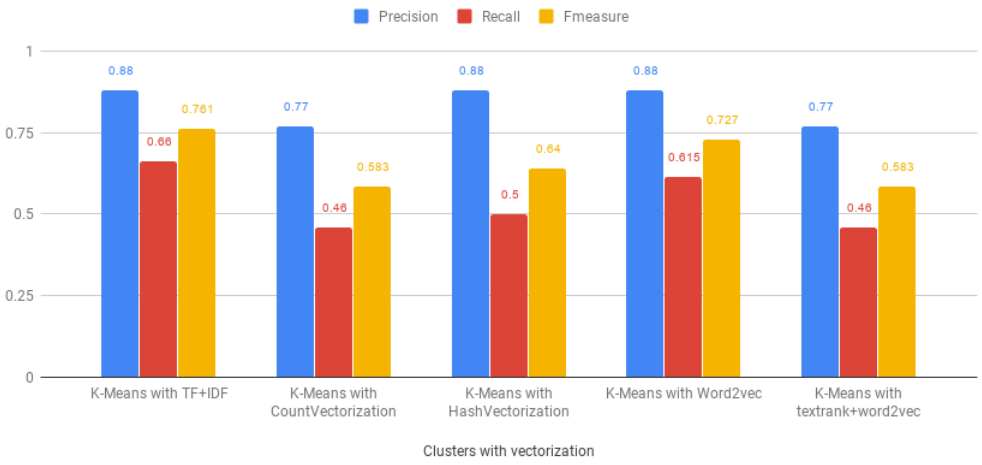


Figure 6.14: HAC and Feature Evaluation

Summary:
The purity values recorded for this clustering process ranges from 0.87-0.69 and from the table Table 6.16 we conclude that TF-IDF and also word2vec vectors have given good results compared to the rest of the approaches.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
HAC with TF+IDF	0.75	0.56	0.64
HAC with CountVectorization	0.58	0.43	0.50
HAC with HashVectorization	0.66	0.50	0.57
HAC with Word2vec	0.66	0.53	0.59
HAC with textrank+word2vec	0.58	0.43	0.50

Table 6.17: BCS data and Relationship Extraction

The table above Table 6.17 shows the evaluation results of clustering process.The visual representation of the results obtained from HAC algorithm run on BCS data can be seen in the Figure 6.15 where all the evaluation metrics are presented.

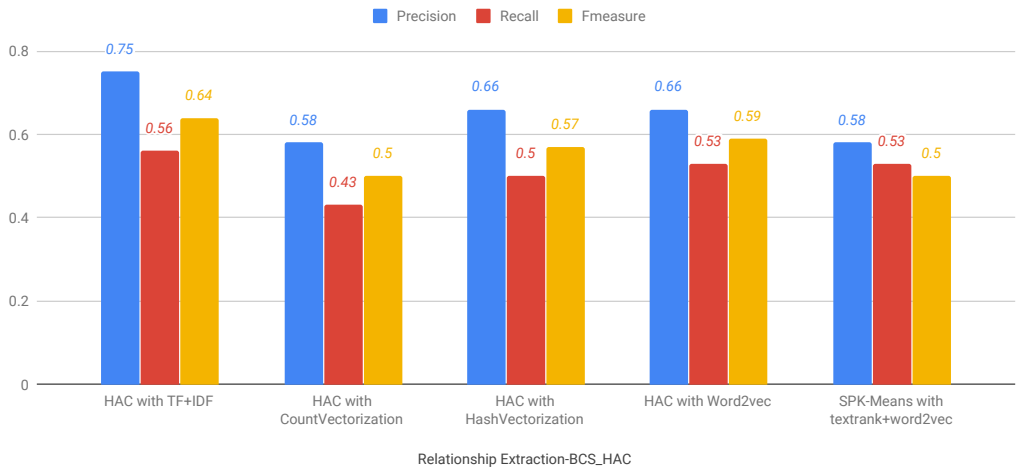


Figure 6.15: BCS data and Relationship Evaluation

Summary:
From the Table 6.17 we conclude that TF-IDF has provided better results than the rest of the approaches, but also to notice that word2vec approach has also comparatively given better results.

6.4.4.2 Estore Dataset

- **Feature Extraction**

Extracted Features	Precision	Recall	F-Measure
HAC with TF+IDF	0.90	0.78	0.837
HAC with CountVectorization	0.80	0.615	0.695
HAC with HashVectorization	0.77	0.666	0.6825
HAC with Word2vec	0.80	0.6695	0.744
HAC with textrank+word2vec	0.6	0.41	0.489

Table 6.18: Estore Data and Feature Extraction

The table above Table 6.18 shows the evaluation results of clustering process.The visual representation of the results obtained from HAC algorithm run on Estore data can be seen in the Figure 6.16 where all the evaluation metrics are presented.

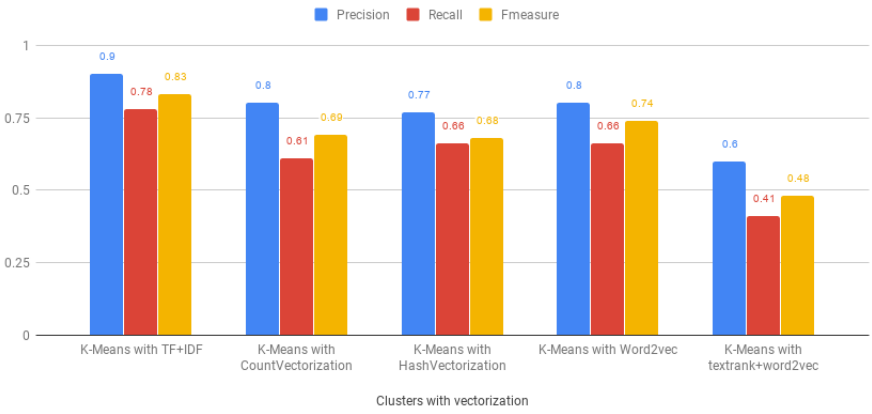


Figure 6.16: HAC Data and Estore Feature Evaluation

Summary:
From the table Table 6.18 we can see that the evaluation results generated produced good results for TF-IDF and also for word2vec vectors. The purity values recorded for this approach is in the range or 0.87-0.62.

- **Relationship Extraction**

Extracted Features	Precision	Recall	F-Measure
HAC with TF+IDF	0.75	0.56	0.64
HAC with CountVectorization	0.66	0.47	0.55
HAC with HashVectorization	0.66	0.44	0.53
HAC with Word2vec	0.75	0.50	0.60
HAC with textrank+word2vec	0.66	0.47	0.55

Table 6.19: Estore data and HAC relationship

The table above Table 6.19 shows the evaluation results of clustering process.The visual representation of the results obtained from HAC algorithm run on Estore data can be seen in the Figure 6.17 where all the evaluation metrics are presented.

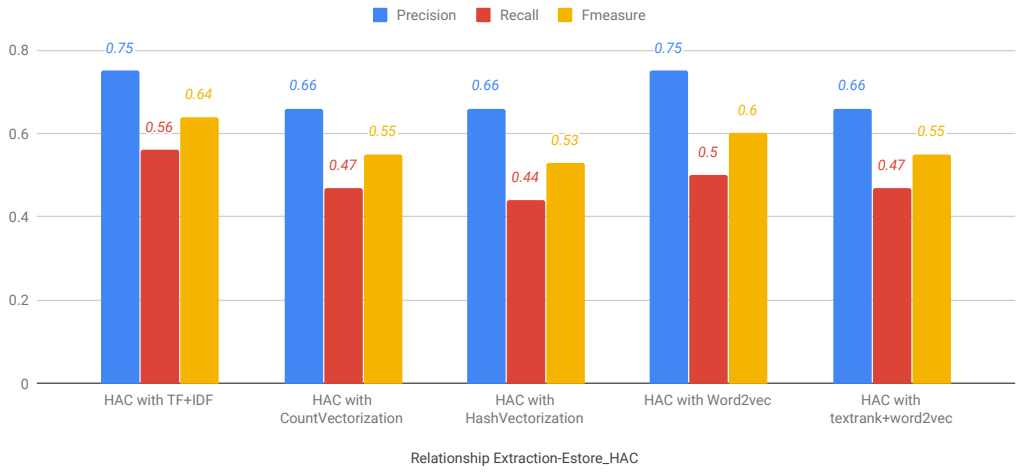


Figure 6.17: HAC Estore Relationship Evaluation

Summary:
From the Table 6.19 we can see that once again TF-IDF has outperformed the rest of the vectorization approaches, interms of the precision, recall and f-measure values.

6.4.5 Threats to validity

- We have concentrated on extraction of good set of features, from the evaluation we can see that the recall values for some clustering algorithms are low, this can also mean that our approach has extracted few features even though they are not present in the ground truth, but can make actual sense for the process.
- We have introduced the Orthogonal Variability Models to detect cross-tree relationship between features, as of now due to limited time span few rules have been applied, henceforth these models wont be handling all of the all cases, and can also be use case specific. More rules can be derived from sentences and relations can be created to better the detection process.
- The truth set used for evaluation of clusters generated were manually created. Henceforth there maybe a human bias which might be included in the label classes which might affect the external cluster evaluation process. And for few of the datasets used other than the body comfort system, there was no truth set available for this process. The feature model generated and the truth sets created are manually done and we have focused on making the data as close to the expert domain as possible.

6.4.6 Arguing RQ's

We evaluated our approach against five different vectorization techniques,four different clustering algorithms and two datasets by running three different evaluation metric on each of them to evaluate the features and relationships, all of which were carefully choosen taking our use case, research constraints and the advantages in mind.

- **RQ1:**The final results we have obtained from our approach can be seen in column charts(see:Table 6.4,Table 6.8,Table 6.12,Table 6.16) for the body comfort system dataset and also in(see:Table 6.6,Table 6.10,Table 6.14,Table 6.18) for the estore dataset which clearly concludes that our approach has been successful in extracting reasonable accurate features. Although we evaluate our approach with the ground truth we obtained, we have seen that there has been a reasonable trade off between the precision and the recall values to a certain degree,this is due to the fact that we have applied minimal rules over natural language texts, applying more semantic rules could impact these values in a much more positive way. We have used four different clustering algorithms although all four of them have their advantages and disadvantages,HAC gives an advantage where we do not have to worry about the selection of cluster numbers prior to the algorithm run. The different vectorization techinques incorporated have proven to provide good results where TF-IDF vectorization stands consistent with good precision and recall values throughout.
- **RQ2:** It is quite evident from the column charts(see:Table 6.5,Table 6.9,Table 6.13)for "bcs" dataset and (see:Table 6.7,Table 6.11,Table 6.15,Table 6.19) for estore dataset, our approach can achieve variability completeness, where from the set of features extracted we are able to extract variability relationships from them by semantically and syntactically understanding the features and their descriptions.
- **RQ3:** We have evaluated all of our approach on different evaluation metrics, these metrics are formulated after several research in the related field. The cluster evaluation has been done through two main evaluation metrics "Purity" and "silhouttee coefficient" but methods such as "elbow method" "rand index","cohesion" were also considered. The extracted features and their relationships where evaluated based on the "precision","recall" and "f-measure" values which has proven to be the baseline evaluation metrics to evaluate features and their relationships (see:Section 3.3).

6.4.7 Summary

In particular, we have performed our evaluation on five vectorization techniques, four clustering algorithms,two datasets i.e. body comfort system dataset and estore dataset respectively and can conclude that our approach can extract good set of features and relationship between them. All of the comparisons of different techniques we have made during our evaluation process,we can conclude that generally performs consistently well in extracting features and their relationships, although word2vec has also extracted reasonable or sometimes higher results but we found to be consistent with good set of precision and recall values.

7. Conclusion and Future Work

To finalize this thesis, we summarize our research, findings and propose ideas for future work in this chapter.

In this work, we successfully extracted useful features from a set of raw unstructured document by applying various different approaches of NLP and Clustering algorithms. Choosing clustering as one of the core processes allows us to choose between several different implementations, and approaches.

Furthermore, we have introduced different types of vectorization methods there by directly affecting the clustering results. They differ in methods of generating vectors. We have used TF-IDF, count and hash based methods but have also implemented the concepts of TR and averaging word2vec methods to further generate vectors which can be processed and clustered accordingly.

We then focused on the feature extraction and relationship extraction methods there by adopting several rule based and model based approach to extract good set of features and relationships between them. We proposed a way in-terms of rule based approaches where these rules are applied over natural language texts and using term weights we have selected features which we processes it further into orthogonal variability models to get relationships between features.

Our evaluation approaches showed that the initial clustering process can impact results by extracting good set of features from requirements. Clustering results also varied due to the choice of clustering algorithm and vectorization methods, these approaches impacted the results. The rule based approaches for feature extraction showed that we could achieve higher accuracy by applying these approaches, although our experimental cases showed that when there was a significant rise in the precision values there was correspondingly a drop with the values of recall, this can be further balanced by adapting different term weighting strategies. Adoption the orthogonal variability modelling based approaches to extract cross tree variants also showed reasonable accuracy.

In summary, we have showed few solutions to extract accuracte features and their relationships from natural language texts.

7.1 Future Work

In this section we present ideas for further research.

Clustering in the present state has its limitations, vector space models or latent semantic based approaches can be incorporated to balance those limitations caused by the clustering algorithms. Although we have used a relative frequency based term weighting approaches to weigh terms, we can use C-NC value term weighting approaches to further filter out features and extract good set of features from them. In relationship extraction methods we have used similarity and orthogonal variability model based approaches where we make use of the relation between words and extract dependency between sentences, we have defined few rules in our work this can be further explored and more rules can be generated thereby pushing the accuracy of the model.

Bibliography

[1] P. Achananuparp, X. Hu, and X. Shen. The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery*, pages 305–316. Springer, 2008. (cited on Page 13 and 16)

[2] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pages 45–54. ACM, 2012. (cited on Page 9, 19, 21, and 22)

[3] A. Alexandrescu and K. Kirchhoff. Data-driven graph construction for semi-supervised graph-based learning in nlp. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 204–211, 2007. (cited on Page 14)

[4] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler. An exploratory study of information retrieval techniques in domain analysis. In *Software Product Line Conference, 2008. SPLC'08. 12th International*, pages 67–76. IEEE, 2008. (cited on Page 19, 20, 21, and 22)

[5] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Extracting domain models from natural-language requirements: approach and industrial evaluation. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pages 250–260. ACM, 2016. (cited on Page 20)

[6] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated extraction and clustering of requirements glossary terms. *IEEE Transactions on Software Engineering*, (10):918–945, 2017. (cited on Page 20)

[7] M. A. Babar, L. Chen, and F. Shull. Managing variability in software product lines. *IEEE software*, 27(3):89–91, 2010. (cited on Page 21)

[8] E. Bagheri, F. Ensan, and D. Gasevic. Decision support for the software product line domain engineering lifecycle. *Automated Software Engineering*, 19(3):335–377, 2012. (cited on Page 21)

[9] N. H. Bakar, Z. M. Kasirun, and N. Salleh. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software*, 106:132–149, 2015. (cited on Page 20 and 34)

[10] S. Beliga, A. Meštrović, and S. Martinčić-Ipšić. An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*, 39(1):1–20, 2015. (cited on Page 11)

[11] D. Bogdanova. Extraction of high-level semantically rich features from natural language text. In *ADBIS (2)*, pages 262–271, 2011. (cited on Page 20)

[12] E. Boutkova and F. Houdek. Semi-automatic identification of features in requirement specifications. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 313–318. IEEE, 2011. (cited on Page 20 and 21)

[13] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. (cited on Page 12)

[14] C. Buchta, M. Kober, I. Feinerer, and K. Hornik. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012. (cited on Page 27)

[15] C.-C. Chang. " libsvm: a library for support vector machines," acm transactions on intelligent systems and technology, 2: 27: 1–27: 27, 2011. *http://www. csie. ntu. edu. tw/˜ cjlin/libsvm*, 2, 2011. (cited on Page 21)

[16] K. Chen, W. Zhang, H. Zhao, and H. Mei. An approach to constructing feature models based on requirements clustering. In *null*, pages 31–40. IEEE, 2005. (cited on Page 19, 20, and 21)

[17] L. Chen, M. Ali Babar, and N. Ali. Variability management in software product lines: a systematic review. In *Proceedings of the 13th International Software Product Line Conference*, pages 81–90. Carnegie Mellon University, 2009. (cited on Page 21)

[18] K. Czarnecki and A. Wasowski. Feature diagrams and logics: There and back again. In *Software Product Line Conference, 2007. SPLC 2007. 11th International*, pages 23–34. IEEE, 2007. (cited on Page 20)

[19] K. Czarnecki, S. She, and A. Wasowski. Sample spaces and feature models: There and back again. In *12th International Software Product Line Conference*, pages 22–31. IEEE, 2008. (cited on Page 20)

[20] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 181–190. ACM, 2011. (cited on Page 20)

[21] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta. Mining commonalities and variabilities from natural language documents. In *Proceedings of the 17th International Software Product Line Conference*, pages 116–120. ACM, 2013. (cited on Page 20, 21, and 22)

[22] C. J. Fillmore. The case for case. 1967. (cited on Page 41)

[23] A. Geitgey. Machine learning is fun part 5: Language translation with deep learning and the magic of sequences. *Erişim adresi*, 2016. (cited on Page 9)

[24] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162. IEEE, 2014. (cited on Page 20)

[25] L. Hamilton. Clustering of cumulative grainsize distribution curves for shallow-marine samples with software program clara. *Australian Journal of Earth Sciences*, 54(4):503–519, 2007. (cited on Page 28)

[26] M. Hamza and R. J. Walker. Recommending features and feature relationships from requirements documents for software product lines. In *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2015 IEEE/ACM 4th International Workshop on*, pages 25–31. IEEE, 2015. (cited on Page 20)

[27] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering*, 39(12):1736–1752, 2013. (cited on Page 20 and 21)

[28] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003. (cited on Page 21)

[29] N. Itzik and I. Reinhartz-Berger. Generating feature models from requirements: Structural vs. functional perspectives. In *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2*, pages 44–51. ACM, 2014. (cited on Page 10)

[30] N. Itzik and I. Reinhartz-Berger. Sova-a tool for semantic and ontological variability analysis. In *CAiSE (Forum/Doctoral Consortium)*, pages 177–184, 2014. (cited on Page 10)

[31] E. Jung. Three essays on enhancing clinical trial subject recruitment using natural language processing and text mining. 2015. (cited on Page xiii and 9)

[32] R. V. Kadison. The pythagorean theorem: I. the finite case. *Proceedings of the National Academy of Sciences*, 99(7):4178–4184, 2002. (cited on Page 14)

[33] S. Kübler, R. McDonald, and J. Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009. (cited on Page 11)

[34] K. Kumaki, R. Tsuchiya, H. Washizaki, and Y. Fukazawa. Supporting commonality and variability analysis of requirements and structural models. In *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pages 115–118. ACM, 2012. (cited on Page 20, 21, and 22)

[35] K. Lee, K. C. Kang, and J. Lee. Concepts and guidelines of feature modeling for product line software engineering. In *International Conference on Software Reuse*, pages 62–77. Springer, 2002. (cited on Page 1)

[36] E. Loper and S. Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002. (cited on Page 17)

[37] D. Maynard, Y. Li, and W. Peters. Nlp techniques for term extraction and ontology population., 2008. (cited on Page 11)

[38] A. Metzger and K. Pohl. Variability management in software product line engineering. In *Companion to the proceedings of the 29th International Conference on Software Engineering*, pages 186–187. IEEE Computer Society, 2007. (cited on Page 21)

[39] A. Metzger and K. Pohl. Software product line engineering and variability management: achievements and challenges. In *Proceedings of the on Future of Software Engineering*, pages 70–84. ACM, 2014. (cited on Page xiii, 5, and 6)

[40] R. Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, 2004. (cited on Page 12)

[41] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004. (cited on Page xiii, 11, and 12)

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. (cited on Page xiii, 15, 24, and 33)

[43] Y. Mu, Y. Wang, and J. Guo. Extracting software functional requirements from free text documents. In *Information and Multimedia Technology, 2009. ICIMT'09. International Conference on*, pages 194–198. IEEE, 2009. (cited on Page 21)

[44] Y. Mu, Y. Wang, and J. Guo. Extracting software functional requirements from free text documents. In *Information and Multimedia Technology, 2009. ICIMT'09. International Conference on*, pages 194–198. IEEE, 2009. (cited on Page 21, 22, and 41)

[45] R. T. Ng and J. Han. E cient and e ective clustering methods for spatial data mining. In *Proceedings of VLDB*, pages 144–155, 1994. (cited on Page 28)

[46] N. Niu and S. Easterbrook. Extracting and modeling product line functional requirements. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 155–164. IEEE, 2008. (cited on Page 21)

[47] N. Niu and S. Easterbrook. Concept analysis for product line requirements. In *Proceedings of the 8th ACM international conference on Aspect-oriented software development*, pages 137–148. ACM, 2009. (cited on Page 19 and 21)

[48] N. Niu, J. Savolainen, Z. Niu, M. Jin, and J.-R. C. Cheng. A systems approach to product line requirements reuse. *IEEE Systems Journal*, 8(3):827–836, 2014. (cited on Page 21, 41, 42, and 45)

[49] J. N. och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002. (cited on Page 1, 5, and 21)

[50] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009. (cited on Page xiii and 28)

[51] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011. (cited on Page 48 and 49)

[52] R. Řehřek and P. Sojka. Gensim—statistical semantics in python. *statistical semantics; gensim; Python; LDA; SVD*, 2011. (cited on Page 15)

[53] U. Ryssel, J. Ploennigs, and K. Kabitzsch. Extraction of feature models from formal contexts. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, page 4. ACM, 2011. (cited on Page 19)

[54] N. Sager, J. S. Siegel, and E. A. Larmon. *Natural Language Information Processing: a computer grammar of English and its applications*. Addison-Wesley Reading, MA, 1981. (cited on Page 9)

[55] M. Steinbach, G. Karypis, V. Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000. (cited on Page 26)

[56] M. Vijaymeena and K. Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28, 2016. (cited on Page 13)

[57] J. J. Webster and C. Kit. Tokenization as the initial phase in nlp. In *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*, volume 4, 1992. (cited on Page 9)

[58] N. Weston, R. Chitchyan, and A. Rashid. A framework for constructing semantically composable feature models from natural language requirements. In *Proceedings of the 13th International Software Product Line Conference*, pages 211–220. Carnegie Mellon University, 2009. (cited on Page 19, 20, and 21)

[59] L. Yi, W. Zhang, H. Zhao, Z. Jin, and H. Mei. Mining binary constraints in the construction of feature models. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 141–150. IEEE, 2012. (cited on Page 20)

[60] Y. Yu, H. Wang, G. Yin, and B. Liu. Mining and recommending software features across multiple web repositories. In *Proceedings of the 5th Asia-Pacific Symposium on Internetware*, page 9. ACM, 2013. (cited on Page 20 and 21)

[61] S. Zhong. Efficient online spherical k-means clustering. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 5, pages 3180–3185. IEEE, 2005. (cited on Page 27)