Text Summarization Using Intuitionistic Fuzzy Hypergraphs

Ph. D Thesis submitted to

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

under the Faculty of Technology

by

Dhanya P. M

(Reg. No: 4131)

Under the guidance of

Dr. A. Sreekumar

&

Dr. M. Jathavedan

Department of Computer Applications
Cochin University of Science and Technology
Kochi - 682 022, Kerala, India
December 2018

Text Summarization Using Intuitionistic Fuzzy Hypergraphs

Ph. D. thesis

Author:
Dhanya P. M
Department of Computer Applications
Cochin University of Science and Technology
Kochi - 682 022, Kerala, India
Email: dhanya.rajeshks@gmail.com

Supervising Guides:
Dr. A. Sreekumar
Professor
Department of Computer Applications
Kochi.
Email: askcusat@gmail.com
&
Dr. M. Jathavedan
Professor Emeritus
Department of Computer Applications
Cochin University of Science and Technology
Kochi - 682 022.
Email: mjvedan@gmail.com

Cochin University of Science and Technology
Kochi - 682 022, Kerala, India.
December 2018

Dr. A. Sreekumar	Dr. M. Jathavedan
Professor	Professor Emeritus
Department of Computer Applications	Department of Computer Applications
CUSAT	CUSAT
Kochi - 682 022	Kochi - 682 022

December 2018

Certificate

Certified that the work presented in this thesis entitled ***Text Summarization Using Intuitionistic Fuzzy Hypergraphs*** is based on the authentic record of research carried out by **Dhanya P.M** under our guidance in the Department of Computer Applications, Cochin University of Science and Technology, Kochi- 682 022 and has not been included in any other thesis submitted for the award of any degree.

Dr. A. Sreekumar	Dr. M. Jathavedan

Phone : +91 484 2576253

Dr. A. Sreekumar
Professor
Department of Computer Applications
Kochi- 682 022
CUSAT

Dr. M. Jathavedan
Professor Emeritus
Department of Computer Applications
Kochi- 682 022
CUSAT

December, 2018

Certificate

Certified that the work presented in this thesis entitled ***Text Summarization Using Intuitionistic Fuzzy Hypergraphs*** submitted to Cochin University of Science and Technology by **Dhanya P.M** for the award of degree of Doctor of Philosophy under the faculty of technology, contains all the relevant corrections and modifications suggested by the audience during the pre-synopsis seminar and recommended by the Doctoral Committee.

Dr. A. Sreekumar

Dr. M. Jathavedan

Phone : +91 484 2576253

Declaration of authorship

I, **DHANYA P. M**, declare that this thesis titled, **'Text Summarization Using Intuitionistic Fuzzy Hypergraphs'** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- 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.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

“The roots of education are bitter, but the fruit is sweet”

Aristotle

Acknowledgements

First and foremost, I would like to thank GOD Almighty for giving me a chance to do research in the Department of Computer Applications, Cochin University of Science and Technology. My sincere thanks to my research guides Prof. Dr. A. Sreekumar, for guiding me in the right direction whenever I needed it and Dr. M. Jathavedan for the technical inputs he has given me throughout the work. I take this opportunity to also thank Dr. Kannan B, Head of the Department, Department of Computer Applications, for the technical, motivational and moral support he extended to me. Let me also take this opportunity to thank Dr. K. V. Pramod for giving me the inspiration to work in the department. Let me also thank my research colleagues for the support they have given me, and the technical and administrative staffs of the department for giving me the ample support.

My sincere thanks to Dr. Ramkumar P. B, Associate Professor, Department of Basic Sciences and Humanities, Rajagiri School of Engineering and Technology, for his help in the mathematical modeling of this research work. Let me also thank the human summarizers who were involved in the testing phase of this work. Last but not the least, let me thank the management and authorities of Rajagiri School of Engineering and Technology for the generous facilities and the optimum environment they have provided for completion of this research work.

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Abstract

FACULTY OF TECHNOLOGY
DEPARTMENT OF COMPUTER APPLICATIONS

Doctor of Philosophy

Text Summarization Using Intuitionistic Fuzzy Hypergraphs

by
Dhanya P.M

Text Summarization is an area of research in Natural Language Processing. Extractive summarization is the creation of a concise text of a lengthy document by selecting the most important sentences. The proposed work creates extractive summary by modeling text as Intuitionistic Fuzzy Hypergraphs (IFHG). The IFHG is subjected to morphological operations like dilation and erosion. Summary is created by designing a morphological filter operator using these dilation and erosion. Two systems are developed, one which works for English documents and another for Malayalam documents.

The input text passes through a preprocessing phase, namely stemming, where the tokens are converted to their root form. A Malayalam lemmatizer is developed using tree-based method, where the suffix forms a path in the tree and the replacement forms the leaf. The preprocessed text is subjected to the clustering phase, where spectral partitioning of the hypergraph is applied to form clusters. For each cluster, IFHG is formed, upon which the summary filter is designed. Two types of IFHG modeling is done in the proposed work. In one method, sentences are modeled as hyperedges and words are modeled as nodes. In the second method, documents are modeled as hyperedges and keywords are modeled as nodes. This is a premier work which models text as IFHG and creates summary using a morphological filter.

Contents

List of Figures

List of Tables

Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
TF-IDF	**T**erm **Frequency** - **I**nverse **D**ocument **F**requency
IFHG	**I**ntuitionistic **F**uzzy **H**ypergraph
ASF	**A**lternate **S**equential **F**ilter
ROUGE	**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
GRU	**G**ated **R**ecurrent **U**nit
API	**A**pplication **P**rogramming **I**nterface
ATS	**A**utomated **T**ext **S**coring
RNN	**R**ecurrent **Neural** **N**etwork
LSA	**L**atent **S**emantic **A**nalysis
SVD	**S**ingular **V**alue **D**ecomposition
CWS	**C**lue **W**ord **S**ummarization
LDA	**L**inear **D**iscriminant **A**nalysis
RDF	**R**esource **D**escription **F**ramework
FSGM	**F**rameNet Based **S**emantic **G**raph **M**odel
MMR	**M**aximal **M**arginal **R**elevance
POS	**P**art **O**f **S**peech Tagging
ANSIF	**A**daptive **N**etwork **F**uzzy **I**nference **S**ystem

Symbols

τ	Text
H_τ	Hypergraph of text τ
ξ	Edge set in H_τ
ν	Node set in H_τ
L	Laplacian matrix
D_v	Diagonal matrix of node degree
D_e	Diagonal matrix of edge degree
W	Weight matrix
A	Adjacency matrix
$H_{w\tau}$	Weighted hypergraph
H_{IF}	Intuitionistic fuzzy hypergraph
μ_n	Membership degree of node in H_{IF}
γ_n	Non-membership degree of node in H_{IF}
μ_e	Membership degree of the edge in H_{IF}
γ_e	Non-membership degree of the edge in H_{IF}
H^n	Node set of H_{IF}
H^e	Edge set of H_{IF}
(α, β) cut	Alpha, beta cut
X_{IF}	Sub IFHG of H_{IF}
X^e	Edge set of X
X^n	Node set of X
$X^{e'}$	Edge complement of X
$X^{n'}$	Node complement of X

$X^{e'n}$	Nodes in $X^{e'}$
δ	Morphological dilation operator
δ^c	Conditional morphological dilation operator
ε	Morphological erosion operator
γ	Opening filter
ϕ	Closing filter
$\gamma_{1/2}$	Half opening filter
$\phi_{1/2}$	Half closing filter
λ	Number of edges/nodes in filter
γ_λ	Metric induced opening filter
ϕ_λ	Metric induced closing filter

Dedicated to my parents, my husband Rajesh, my son Adityan and my daughter Jyothika…

Chapter 1

Introduction

1.1 Motivation and scope of the work

Today, most of the documents available online are very lengthy and they require much time to fully read and comprehend it. In a world of busy schedule, even reading newspapers of 20-40 pages, consumes much of the morning time. Here comes the importance of automatic text summarization in daily life. Text Summarization is the process of creating a short summary of a lengthy text without losing core information. When it is automated, it is done by a software by taking into consideration, the important information in it, its length and many such aspects. Summarization has been getting extended to several other areas like image summarization, video summarization etc.

Summarization gives the required and important information in a nutshell. Moreover summaries save storage space. A cluster is a group of similar items. Documents when clustered, results in placing similar documents together in one cluster. Information retrieval from a clustered collection is more efficient than from a scattered collection of articles. Summarization is useful in the field of news summarization, summarization of technical reports, weather forecasting, medical report summarization etc.

1.2 Problem statement

Given a collection of articles, the proposed system groups them in to clusters based on various domains. For each cluster an extractive summary is created. Two systems are developed, one which summarizes English documents, the other which summarizes Malayalam documents. For processing Malayalam documents, a Malayalam lemmatizer is also developed.

The proposed system accepts text written in English and Malayalam languages. The text is subjected to preprocessing tasks like removal of punctuation marks, removal of special characters and removal of stop words. Stop words are words which do not contribute to the theme of the text. After stop word removal, words in the text are subjected to lemmatization, where the words are converted to their root form. A Malayalam lemmatizer is developed which identifies the suffix in the word and substitutes it with the replacement. This lemmatizer is developed using two methods namely:

1. Dictionary-based method.
2. Tree-based method.

Once the words in the text are lemmatized, clustering is done in order to put similar texts in one group. This clustering should be done such that there is more intra-cluster similarity. Clustering is done in four methods such as the following:

1. Modeling text as a simple graph.
2. Modeling text as a weighted graph.
3. Modeling text as a hypergraph.
4. Modeling text as a weighted hypergraph.

In hypergraph modeling, sentences are modeled as hyperedges and words in the sentences are modeled as nodes. Weights are assigned to the nodes by taking the term frequency of the words. Weights of the hyperedges/sentences are calculated by taking the sum of all node weights in it. The graph/hypergraph is subjected to spectral partitioning method, by considering the eigen values and eigen vectors of

the laplacian matrix $L = D_v - A$, where D_v is a diagonal matrix of node degrees and A is the adjacency matrix of the graph. The weighted hypergraph method gives a better result in clustering.
An IFHG, $H_{IF} = (H^n, H^e, (\mu_n, \gamma_n), (\mu_e, \gamma_e))$ is formed for each cluster formed from the spectral partitioning of the text, where H^n are the nodes, H^e are the hyperedges, μ_n is the membership degree of the node, γ_n is the non-membership degree of the node, μ_e is the membership degree of the hyperedge and γ_e is the non-membership degree of the hyperedge. Here nodes are words and hyperedges are sentences in the text. The pair of degrees (μ_n, γ_n) are assigned based on whether the words belong to the priority set or non-priority set. The pair of degrees (μ_e, γ_e) of the hyperedges/sentences depends on the degree of the nodes/words in them. Several morphological operations are done on such a text IFHG. Dilation is a morphological operation applied on IFHG. It is of two types:

1. Dilation w.r.to hyperedges-$\delta^e(X^n)$.
2. Dilation w.r.to nodes-$\delta^n(X^e)$.

Union/intersection of these dilations are done and tested for retrieving the priority words/sentences in the text. Morphological operations are done by creating a sub-IFHG X of the parent IFHG H_{IF}. De Morgan's law with sub-IFHGs is also tested. Erosion is another morphological operation which is again of two types namely:

1. Erosion w.r.to hyperedges-$\varepsilon^e(X^n)$.
2. Erosion w.r.to nodes-$\varepsilon^n(X^e)$.

Union/intersection of these erosions are tested which retrieves priority words and sentences from the text. A disjoint graph partitioning algorithm for IFHG is designed with the help of these dilations and erosions. A composition operator is applied on these dilations and erosions which results in a filter design. Such a summary filter applied on a text produces text summary.

We come across different types of text summaries [1] like headlines, outlines, minutes, previews, synopses, reviews, digests, bulletins, histories etc in our daily life. So far, researchers all over the world in this field have developed different methods to create a good concise version of the vast documents. They are briefed in the following sections.

1.3 Tensor flow model

The Google Brain Team developed an algorithm which creates a summary by extracting interesting parts of the text and rephrasing them to create an abstractive summarization. This tensor flow model is a sequence-to-sequence model [2] which generates headlines from the input sentences. Many candidate headlines are formed from which the most suitable one is selected. The method uses a bidirectional Gated Recurrent Unit(GRU) encoder and a GRU decoder. Implementation proceeds through several phases like bucketing, attention mechanism and beam search.

1.4 Term frequency - inverse document frequency (TF-IDF) method

In this model, documents are represented using the TF-IDF scores of words in the document. TF is the average number of occurrences of a word per sentence in a document. IDF value is computed based on the whole document. The importance of TF-IDF is well understood, and it is referred as a sentence weighing method [3]. The method of optimizing summarization [4] by first categorizing the documents on the basis of TF-IDF and then ranking the sentences using the existing scores to avoid redundant information, has outperformed other summarization tools. Each document is represented as a three-dimensional vector where each weight corresponds to one of the three categories of the document. Each weight is calculated as a combination of TF-IDF. Sentences are ranked using a score called SumBasic score. SumBasic score of a sentence calculates the importance of a sentence based on the word frequency. But in this summarization technique semantic relatedness among the consecutive sentences is not calculated.

1.5 Clustering and classification

Different documents usually address different topics. They are normally broken up explicitly or implicitly into sections. If the document collection for which the summary is to be produced is of totally different topics, then in order to create a meaningful summary, document clustering is required. A hybrid clustering method [5] (partitioning and hierarchical) is proposed to group Arabic documents into several clusters. This method uses a cluster-based summarization approach to gather similar texts, as well as a key phrase extraction approach by an unsupervised machine learning algorithm to identify sentences that include key phrases and to summarize original text documents. The preprocessing stage includes four steps namely, tokenization, stop word elimination, stemming text representation, and term weighting (TF-IDF). Document clustering involves unsupervised document organization, topic extraction and fast information retrieval. Clustering is divided into two major types: Hierarchical and K-means. The first clustering involves two main clustering algorithms, namely, single and complete-link clustering. The yield of single or complete-link clustering is K clusters. All noun phrases are extracted from the Arabic text as candidate key phrases. For each noun phrase, some set of features are extracted for ranking the candidate key phrase namely, term frequency, first occurrence in text, last occurrence in text and sentence count. The next step is sentence scoring to extract important sentences. It is assumed that only those sentences that contain key phrases are important. Cosine similarity and Jaccard coefficient are used to find similar sentences in each cluster and finally ROUGE is used to evaluate the results. The model is seen to have achieved an accuracy of 80% for single document and 62% for multi-document summarization. In another method, clustering is first applied to get document clusters. Later Linear Discriminant Analysis (LDA) method [6] is applied to get the cluster topics, which are given as input to the map reduce framework. Semantically similar terms are found with the help of the WordNet Application Programming Interface (API). Later the performance of the system without considering clustering and semantic similarity of sentences are found out. A time stamp based approach [7] with Naive Bayesian Classification is used where the time stamp provides an ordered look of sentences. Here, a seven stage process is used which includes preprocessing, selecting related documents, splitting into sentences and words, score calculation of words and sentences using Bayesian classifier and finally selecting sentences.

1.6 Neural networks

The neural networks are initially trained to learn the types of sentences that should be included in the summary. The network is trained with sentences in test paragraphs which are in the summary as well as not in the summary. This is done by a human reader. The neural network [8] learns the patterns inherent in sentences that should be included in the summary and those that should not be included. In Automated Text Scoring(ATS) using neural networks [9] proposed for Wikipedia articles, the method has several phases like tokenization, stop word removal, stemming and synonym checking to assign the same weight to words having the same meaning. In feature extraction phase, sentences are scored based on ten features namely relative position of sentence, named entities, similarities with other sentences, similarity with rest of the document, title relevance, relative length of sentence, frequency of words, citation and numerical data. These scores are then fed to a neural network, giving a single output score. Both the input feature scores and the output score have a range from zero to one. The neural network produces a single value, signifying the importance of the sentence in the summary. Results of the system are evaluated using the F1 score which is the harmonic mean of precision and recall. Summaries created from Microsoft Word 2007 have been used as model summaries. Evaluation indicates that the systems with only one feature have extreme results while the other systems with all the features, or all the features except one, have almost the same results. The best system is the one with only feature which considers citations. The system with only the feature being the relative length of sentence has the worst results. Systems with only features being relative position of sentence, title relevance and frequency of words have good results. A standard feed forward neural network language model [10] is applied which takes x as input and outputs shortened sentences y of length $N < M$, where M is the total number of sentences. Using the model, it estimates the contextual probability of the next word. Encoders like Bag-of-word encoder, Attention based encoder and Convolution encoders are used. In an encoder-decoder Recurrent Neural Network (RNN) [11], it creates an abstractive summary, traversing through several phases like capturing keywords using feature-rich encoder, modeling rare/unseen words using switching-generator pointer and capturing hierarchical document structure with hierarchical attention. In pointer-generator network [12], the included models are:

1. Baseline sequence-to-sequence model which attend to relevant words to generate novel words.

2. Pointer-generator model where for each decoder time step a generation probability is calculated which weighs the probability of generating words from vocabulary versus copying words from source text.

3. Coverage mechanism to be used to solve the problem of repetition.

1.7 Latent semantic analysis

Singular Value Decomposition (SVD) is a very powerful mathematical tool that can find principal orthogonal dimensions of multidimensional data. It is known as Latent Semantic Analysis (LSA) in text processing because SVD when applied to document-word matrices will group semantically related documents, even if they do not share common words. In another method, ATS using text features and SVD has been discussed. The document is tokenized, stop words are removed and stemming is performed. The most important sentences are determined using their TF-IDF weight. The terms-by-sentences matrix obtained is passed on to SVD matrix decomposition technique [13] where the matrix is decomposed into three matrices U, S and V. Here U is the matrix of the left singular vectors, S is the matrix of the singular values and V is the matrix of the right singular vectors. The k^{th} singular vector in V is taken and the sentence associated with the singular vector is taken into the summary. The system is evaluated for three levels of summary. As a result, it was found that the technique was able to produce machine summarization with significant level of precision and recall particularly when the summarization level is high. Automatic document compression which is done using LDA [14], word weighing and clustering algorithm proceeds through phases like preprocessing, feature extraction, K-means clustering, representation of documents as random mixtures over latent topics, TF-IDF and calculating similarity measure. The system is tested on many Indonesian blog articles in various domains and has shown good results. Email Summarization [15] using LDA is done by selecting the content for summarization, choosing topic distribution, generating word probability, Clue Word Summarization(CWS) of document, using CWS document for distribution matching, generating LDA document and using it for distribution matching. A project to automate topic modeling from Trademark and Domain Agreement [16] between two parties extracts text from pdf copy of the document using python pdf-miner, clean text, model topics using scikit-learn module countvectorizer and creates a final visual summary. The document term matrix is inputted to LDA algorithm for topic modeling. As a result five distinct topic contexts are isolated. In entity-summarization-LDA [17], an entity e is a document which is a collection of triples $< s, p, o >$, where s is the subject, p is the predicate and o is the object and they are used for constructing Resource Description Framework(RDF) graph. $Sum(e, k)$ selects the top-k subset of all predicates and corresponding objects that are most relevant to that entity. Given w as word, z as the topics and d as the document, the word-topic distribution $p(w/z)$, and the topic-document distribution $p(z/d)$ are learned in an unsupervised manner.

1.8 Graph based approach

In a method for Arabic text summarization [18] based on graph theory and semantic similarity, semantic similarity between sentences is used to calculate importance of each sentence in the document and the most important sentences are extracted to generate document summary. The words sharing a single root are related semantically. Feature selection techniques are applied to improve the semantic similarity between sentences. A graphical structure of the text will be helpful to understand the connection between different parts of the text. Graph-based algorithms use a ranking algorithm to rate different sections of a text where each section is considered as a node. Edges will represent the lexical or semantic relations between two nodes. In a graph ranking algorithm, all nodes are scored and sorted. Values are devoted to each vertex for selection decisions. Then, sentences with the highest score are selected for the final summary. In this work, after preprocessing (tokenizing, stop word removal and stemming) feature extraction is done based on the TF, IDF, sentence position and indicative expressions. In graph construction phase, nodes represent sentences and an edge is formed between similar sentences. Weight of the edge represents the degree of similarity. Cosine-similarity based on TF-IDF is used to calculate the similarity between sentences. Two sentences are linked if their similarity is greater than a predefined threshold. The result of this step is a highly connected undirected weighted graph. This is the input to the next step which calculates a salient score for each sentence. Then sentences are ranked through a random walk on the graph. A salient score for each node is calculated using PageRank algorithm. Selecting top-ranked sentences may cause redundancy and may also lead to the ignoring of important sentences. In semantic graph model [19], semantic information of the sentence is extracted using weighted FrameNet Based Semantic Graph Model(FSGM). Sentences are treated as weighted vertices, while the semantic relationships are treated as weighted edges. Similarity of the sentences are calculated with the help of FrameNet. In triangle-graph based method [20], it involves steps like preprocessing, graph construction, centrality calculation, sentence ranking, summary generation and finally evaluation of the results. In a graph based multi-document summarization [21], for every sentence (node), a degree centrality score is calculated which is equal to the number of edges incident on the node. Sentences are ranked using a combined score of degree centrality score and positional score. The centroid of a word is the TF-IDF of the word. Centroid of a sentence is the sum of individual word centroids. Now the summary is generated by using the Maximal Marginal Relevance (MMR) method that initially selects a sentence with top rank and selects the next ranked sentence only if it is dissimilar with any of the previously selected sentences. This process is continued until the required length is reached. This system with a graph based hybrid similarity measure was evaluated using ROUGE-1, F-score and achieved better result than in systems with cosine similarity measure and centroid based measure.

In text summarization using concept graph [22], the BabelNet knowledge base is presented. It is an abstractive summarization technique for multiple documents. The proposed system is based on identifying concepts of the BabelNet knowledge base. By using the concepts and relationships which are identified from documents, a graph is produced and then similar concepts are extracted from this graph, so that they are placed in related communities. Sentences with respect to these concepts and their communities are rated and final summary is produced. The proposed method consists of five main steps: preprocessing, identifying and weighing concepts, producing graph, community detection and selecting sentences. Preprocessing includes the removal of frequent words that carry little information such as prepositions and pronouns. In the next step, the extraction and weighting of documents concepts is done. For this the BabelNet knowledge base is used. Firstly words are mapped to BabelNet concepts and then by applying the existing disambiguation system in BabelNet, the best concepts for each word are selected. After extracting the concepts, TF-IDF is used for weighting concepts in document. After this step, the set of documents are changed to a weighted non-repetitive set of concepts. The semantic relations such as: derive, is-a, part-of, related and gloss-related are identified. An undirected graph is constructed where nodes are weighted concepts and edges are relation between the concepts. Further, similar communities in the graph are identified using Girvan Newman algorithm which is based on the elimination of edges between communities. At the end of this stage, a group of communities are created which includes a variable number of unique, similar and weighted concepts. For rating sentences, each community is weighted according to the weights of its concepts in that community. The weight of each community is the sum of the total weight of the concepts in the community. Then a linear function is used to rate sentences. Weight of each concept in a sentence is considered when calculating its rate. In this approach, after extracting each summary sentence, previously selected concepts are ignored and sentence weight is recalculated. ROUGE is used for evaluation of the results and it shows that the method outperforms Microsoft, LexRank etc. The use of concepts instead of words caused better understanding and produced summaries that are closer to human summaries.

In multi-graph based text summarizer [23], the number of edges in the graph between two sentences (two nodes) is equal to the number of same words in both sentences. A word may occur in a sentence more than once. Such occurrence may be added in a symmetric matrix. The total number of edges is stored in a symmetric matrix that represents the text being summarized. The row values of the matrix are summed up to generate a sum vector, which is used to rank the sentences. This value replaces the TF-IDF value used by researchers over many years. Sum vector is used to rank the sentences. A cut-off mechanism using the required threshold is applied to produce the summary. This method is different from the other methods in that it does not use the cosine equation to find the similarity between sentences. Preprocessing reduces the size of the matrix by a considerable amount, which increases the performance and accuracy of the algorithm. Preprocessing mainly includes removal of articles, prepositions and meaningless words (like a sentence starting with a bracket or any special character). Results show that the method is efficient and produced excellent results as compared to other online summarizers.

1.9 Genetic algorithm(GA)

In a hybrid-based Arabic single document text summarizer approach [24], GA that depends on semantic relationship between sentences is presented. The final summary is generated by selecting the optimal vector of sentences that is produced by the GA from the graph representation of the document. The cosine measure will be used as a similarity function between sentences and the graph-based approach is combined with the statistical method. In this paper, the preprocessing stage includes four main steps: segmentation and tokenization, eliminating stop-words, applying Part of Speech Tagging (POS) and finding n-gram for each noun. The sentences are scored using topic similarity score, location score and length score, which are together known as informative score. Text is represented as a graph, with the nodes of the graph representing text elements (i.e., normally words or sentences), while edges representing the links between those text elements. The cosine similarity measure is selected on the basis of a term weighting scheme which is the TF-IDF. The GA search space for the summarization problem is the set of permutations of the nodes that represent the sentences of the source document within the compression ratio (summary length). The most natural way to represent a solution is through a path representation. The fitness function is formulated for evaluating a given path by combining informative measure and semantic measure. ROUGE is used to evaluate the summary. The proposed approach has investigated the effects of using a scoring technique that combines the informative and the semantic scoring techniques. This solves the problem of a lack of attention of semantic relationships between the sentences that the statistical approaches suffer from. The proposed scoring technique also aims to solve the problem in ignoring the sentence's structural features such as its length, position etc., in graph-based approaches. In GA with map-reduce approach [25], text processing is done to clean the text from stop words and HTML tags. Text readability and text cohesion features are extracted and they are used to evaluate the individuals in GA. Every GA iteration works as a single map reduce job. The final winner sentences are sorted to create the summary.

1.10 Fuzzy logic based methods

In a fuzzy based summary system [26], sentence grades are calculated from syntactic parameters and semantic parameters. The degree of importance of a sentence and correlation is used to find the final summary. The Adaptive Network Fuzzy Inference System(ANSIF) model [27] takes five feature values of the document and convert those crisp values to fuzzy values which become strength of a rule. The output of the rule combined with the input variables are transferred to the consequent layers. In summarization of legal documents [28], the membership function in the fuzzifier section translates the inputs to linguistic variables. Fuzzy rule base derive the linguistic values and defuzzifier converts it to linguistic values. The output membership function divides sentences in to unimportant, average and important categories. Membership functions are based on fuzzy centroid method which uses a triangular membership function.

1.11 Recent text summarization in Indian languages

A Malayalam summary system with semantic graph creation [29] and its reduction has shown a precision of 0.466, recall of 0.667 and f-measure of 0.400. The Malayalam summarizer [30], passes through various phases like preprocessing, sentence scoring, sentence ranking and finally generates summary by selecting top k sentences. Another Malayalam summarizer [31] that was developed, has shown a ROUGE-1 of 0.57 and ROUGE-2 of 0.53 which was applied on Malayalam news from Mathrubhumi daily. In a Hindi Text summarizer [32], linguistic rules are used. Sentences are scored based on a number of parameters and the system is tested for Hindi documents in various domains. The system showed a recall of 0.69. A Tamil summarizer [33], using centroid approach and applied on Tamil newspaper, finds group of words which are statistically important for a document and extracts sentences nearest to the centroid. In a Gujarati summarizer [34], mainly two features like stem weight and similarity score are calculated. It uses DHIYA - stemmer, stem weightage module, LexRank module and anaphora resolver.

1.12 Conclusion

In this survey, an effort has been made to review the variety of approaches for automatic text summarization. The significance and the utility of various kinds of summarization techniques are presented. The study shows that both statistical methods and semantic-based graph methods are available, and recently graph-based approaches that consider the relatedness between the sentences are trending more. The survey shows that research in the field of text summarization has been an interesting area throughout the past years and that there is scope for further improvements and extended works. Text summarization can also be combined with other systems of Human-machine interactions and can lead to systems which are truly useful for mankind.

Chapter 2

Lemmatization

Since artificial intelligence mainly deals with inducing intelligence in computers so that they behave more like human beings, programming them to understand natural languages like English, Hindi, Malayalam etc. is gaining importance. It is obviously due to the need for artificial intelligence, that sub-domains like Natural Language Processing(NLP), expert systems etc. are developing. Interaction of the users with the systems using natural language can be either through natural typed text or natural speech. Thus, text mining has been an important area of research under NLP. Text document processing has many sub-domains where their accuracy increases only if the words are reduced to their root form which we call as the process of lemmatization. This text mining in any language requires the word to be converted to the root form. Malayalam is a language which is spoken by over 33 million people in the state of Kerala, India. Malayalam words are subjected to morphology to a large extent. A particular word in Malayalam can take more than 100 affixes and same is discussed in section 2.2. Due to the complexity of the words, lemmatization is very much required and very difficult in the case of a language like Malayalam. Stemmers are already available in languages like English, Arabic, Persian etc. Many stemming algorithms have been developed in various Indian languages also. A stemmer [35] has been developed in Malayalam which uses a three pass method.

Morphology is a term in NLP which refers to the different forms a particular word can take. The process of removing the affixes from the word and extracting the stem is called stemming. The lemmatizer generates the root word from the given word rather than reducing it to the stem. Lemmatization is very important in any NLP project since only the root word can contribute to the word count. This exploratory work presents two methods for extracting the root word from a Malayalam word. One is clustered indexed dictionary based, which uses a suffix replacement method by considering around 1135 rules which are identified by several test cases. The second method creates a tree from the set of rules. The permanently stored tree is then searched for a path which matches with the suffix and the corresponding leaf node, which is the replacement, is retrieved. Testing with online Malayalam documents helped in finding out and adding several rules to the dictionary. This is a pioneering lemmatizer which uses a tree based method.

2.1 Literature review

A Spanish version of porter stemmer [36] is used as preprocessing tool for correctly extracting the text in each document image. This helps in forming the bag-of-words vector. The performance of various page classifiers are being compared as for all of which stemming is an important module which affects the accuracy of the system. A work in Mongolian language [37] has considered stem and suffixes for word segmentation. Application of predefined pattern to derive the stem words are also discussed [38]. The presence of affixes results in a large vocabulary in any language. They have discussed rule based, direct, interlingua, transfer, statistical, example based, knowledge based and hybrid methods of translation where stem and affixes play a vital role. The document similarity calculations like tree based, time series, vector based and different text clustering methods like hierarchical, partitioning, combination and multilevel XML document clustering [39] require the word to be stemmed. Stemming, the removal of affixes, is used as the preprocessing task for the calculation of term frequency and is used in map reduce framework [40] for text summarization. Factorizing words into the morphological components [41] requires the stem to be extracted.

There are some stemmers [42] developed in Asian languages like Arabic, which removed both the prefix and suffix of the given word. Since prefix words are very rare in Malayalam, the algorithm we developed has considered suffixes only. The

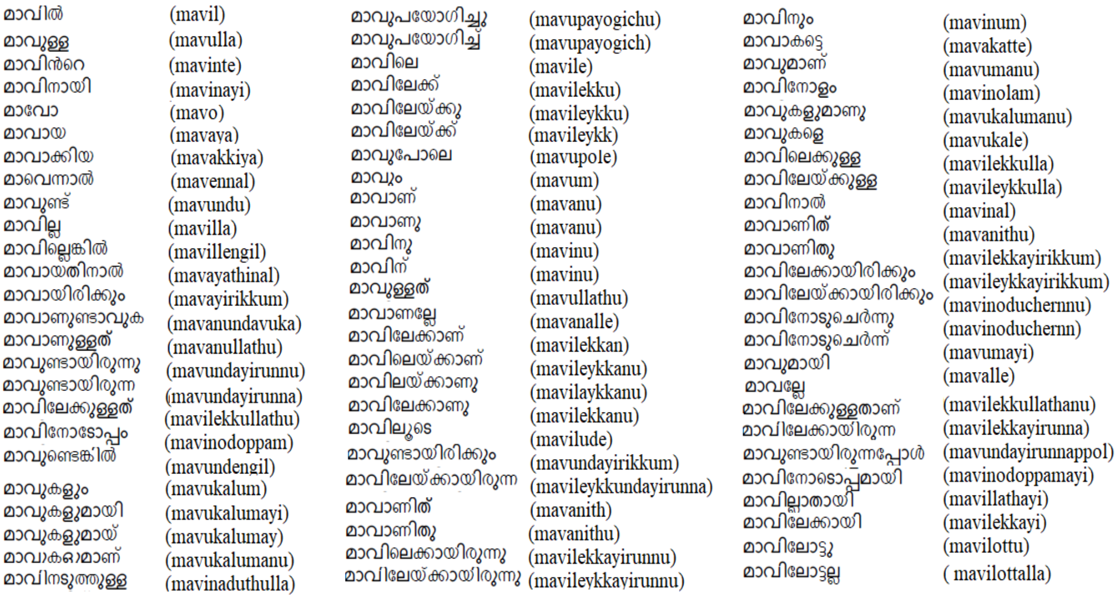
Indonesian stemmer [43], is based on finite state automata. A best word candidate is selected from a candidate list. A detailed survey on stemming [44] is done referring to many languages around the world like English, Czech, Bulgarian, Turkish, Greek, German, Dutch, Portuguese, French etc. Some methods discuss about integrating stemming rules [45] and has considered both suffixes and prefixes.

The stemmers developed in Indian languages have been surveyed [46]. The Gujarati stemmer [47] and the one developed in Bengali [48] use a rule based approach. Stemming has reduced the number of unique words and both under stemming and over stemming have been considered. The use of the minimum stem set model of stemming [49] gives an accuracy of 80% - 88%. The system has been tested for languages like Hindi, Marathi, Malayalam and English. The method has an input word list and an input suffix list. There are methods which uses a probabilistic approach [50] using a suffix list. Urdu stemmer [51] and Hindi stemmer [52] have also been developed. The stemmer developed in Punjabi [53], stems only nouns and proper names with the help of a dictionary. One among 19 conditions is applied for an input Punjabi word and the result is checked against a Punjabi Name list. Word sense disambiguation [54] is very much benefited from stemming where testing is done with the help of a sense list created from Hindi WordNet. Word stemming implemented with the help of hashing [55], has considered nominal inflections, prenominal inflections and verb inflections. The rule based stemmer in Hindi [56] makes use of a Hindi WordNet. The Tamil stemmer [57] is clustering stemmed words using K-means to improve the Information Retrieval(IR) system and another method deals with Tamil suffixes[58] only.

In a Malayalam stemmer [59], stemming is done using Finite State Automata(FSA). FSA is modelled using all possible suffixes and have considered singular nouns, plural nouns and verbs. A method of recursive suffix stripping [60] is also developed with an accuracy of 83.67%. If a word has three suffixes then it is processed thrice. The system is a sub-module of a sentence parser and the accuracy relies on the dictionaries being used. In morphological analyzer [61], rather than root extraction, morphological features like noun/verb, number(plural/singular) and tense(past/present/future) are also extracted. In the light weight stemmer [62], the authors developed a suffix dictionary and used the suffix stripping algorithm to get the stem. Here the word after stripping is taken as the output which is not the actual root word. In another method [63], a rule list is provided separately for nouns and verbs, give an accuracy of only 60% and uses the suffix stripping algorithm to get the stem.

Table 2.1: Comparison of stemmers in Indian languages

AUTHOR, YEAR	LANGUAGE	METHOD	TESTING, ACCURACY
Prajitha U et. al, 2013	Malayalam	One pass. Suffix stripping. Used Suffix Dictionary. Scanning from right to left for the longest match.	Dictionary of 1000 words with their inflected forms 86%
Prajisha K 2013	Malayalam	Three pass. Suffix Stripping. Rule based system.	Testing done with news articles in the web
Vinod P. M et. al, 2012	Malayalam	Morphological analyzer. Recursive suffix stripping. Uses lexical dictionary, monolingual dictionary and bilingual dictionary.	Testing with words taken from Malayalam dictionary
Jisha P. J et. al, 2011		Malayalam Morphological analyzer. Uses bilingual dictionary and root dictionary	Input and Output test cases in Unicode format
Vijay S. R et. al, 2010	Malayalam	Finite state automata Next state is determined using morphotactic rules	Testing done with Online news papers
Jikitsha S Bankim P 2014	Gujarathi	Substitution rules	Evaluation based on EMILE corpus 92.41%
Das S, Mitra. P 2010	Bengali	Using hash table containing suffixes of nouns verbs. Considers derivational inflectional words.	Tested using FIRE 2010 data set Recall of 96.27%
Vasudevan N, Pushpak B, 2012	Hindi	Semi supervised. Stemming by weighted minimum set.	84%
Ramachandran V. A, Illango K 2012	Tamil	Iterative suffix stripping.	84.79%
Kasthuri M, Britto R. K 2014	Tamil	Prefix, question, conjunction, case plural and imperative tense suffixes are stripped.	Testing with docs from net



മാവിൽ	(mavil)	മാവുപയോഗിച്ചു	(mavupayogichu)	മാവിനും	(mavinum)
മാവുള്ള	(mavulla)	മാവുപയോഗിച്ച്	(mavupayogich)	മാവാകട്ടെ	(mavakatte)
മാവിൻറെ	(mavinte)	മാവിലെ	(mavile)	മാവുമാണ്	(mavumanu)
മാവിനായി	(mavinayi)	മാവിലേക്ക്	(mavilekku)	മാവിനോളം	(mavinolam)
മാവോ	(mavo)	മാവിലേയ്ക്കു	(mavileykku)	മാവുകളുമാണു	(mavukalumanu)
മാവായ	(mavaya)	മാവിലേയ്ക്ക്	(mavileykk)	മാവുകളെ	(mavukale)
മാവാക്കിയ	(mavakkiya)	മാവുപോലെ	(mavupole)	മാവിലെക്കുള്ള	(mavilekkulla)
മാവെന്നാൽ	(mavennal)	മാവും	(mavum)	മാവിലേയ്ക്കുള്ള	(mavileykkulla)
മാവുണ്ട്	(mavundu)	മാവാണ്	(mavanu)	മാവിനാൽ	(mavinal)
മാവില്ല	(mavilla)	മാവാണു	(mavanu)	മാവാണിത്	(mavanithu)
മാവില്ലെങ്കിൽ	(mavillengil)	മാവിനു	(mavinu)	മാവാണിതു	
മാവായതിനാൽ	(mavayathinal)	മാവിന്	(mavinu)	മാവിലേക്കായിരിക്കും	(mavilekkayirikkum)
മാവായിരിക്കും	(mavayirikkum)	മാവുള്ളത്	(mavullathu)	മാവിലേയ്ക്കായിരിക്കും	(mavileykkayirikkum)
മാവാണുണ്ടാവുക	(mavanundavuka)	മാവാണല്ലേ	(mavanalle)	മാവിനോടുചെർന്നു	(mavinoduchernnu)
മാവാണുള്ളത്	(mavanullathu)	മാവിലേക്കാണ്	(mavilekkan)	മാവിനോടുചെർന്ന്	(mavinoduchernn)
മാവുണ്ടായിരുന്നു	(mavundayirunnu)	മാവിലെയ്ക്കാണ്	(mavileykkanu)	മാവുമായി	(mavumayi)
മാവുണ്ടായിരുന്ന	(mavundayirunna)	മാവിലയ്ക്കാണു	(mavilaykkanu)	മാവല്ലേ	(mavalle)
മാവിലേക്കുള്ളത്	(mavilekkullathu)	മാവിലേക്കാണു	(mavilekkanu)	മാവിലേക്കുള്ളതാണ്	(mavilekkullathanu)
മാവിനോടോപ്പം	(mavinodoppam)	മാവിലൂടെ	(mavilude)	മാവിലേക്കായിരുന്ന	(mavilekkayirunna)
മാവുണ്ടെങ്കിൽ	(mavundengil)	മാവുണ്ടായിരിക്കും	(mavundayirikkum)	മാവുണ്ടായിരുന്നപ്പോൾ	(mavundayirunnappol)
മാവുകളും	(mavukalum)	മാവിലേയ്ക്കായിരുന്ന	(mavileykkundayirunna)	മാവിനോടൊപ്പമായി	(mavinodoppamayi)
മാവുകളുമായി	(mavukalumayi)	മാവാണിത്	(mavanith)	മാവില്ലാതായി	(mavillathayi)
മാവുകളുമായ്	(mavukalumay)	മാവാണിതു	(mavanithu)	മാവിലേക്കായി	(mavilekkayi)
മാവുകളുമാണ്	(mavukalumanu)	മാവിലെക്കായിരുന്നു	(mavilekkayirunnu)	മാവിലോട്ടു	(mavilottu)
മാവിനടുത്തുള്ള	(mavinaduthulla)	മാവിലേയ്ക്കായിരുന്നു	(mavileykkayirunnu)	മാവിലോട്ടല്ല	(mavilottalla)

FIGURE 2.1: Morphology of word *'mavu'*

Here the word after stripping is taken as the output which is not the actual root word. Rather than using a three pass algorithm [35] for the lemmatizer, the proposed method uses a single pass algorithm. The system is used as a sub-module of a question answering system. A comparison of stemming methods developed in Indian languages is shown in Table 2.1.

2.2 Morphology in Malayalam

Since Malayalam language is rich in morphology, the lemmatizer developed here employs a total of 1135 suffix replacement rules. Morphology in Malayalam language is so complex that a single word can take many different forms. Difficulty in developing a lemmatizer is due to this language complexity. We can show this complexity by taking as example a simple word *'mavu'* which means mango tree in English. Some of the morphological forms of the mentioned word are shown in Fig. 2.1.

The word *'mavu'* which is a noun can be attached with one preposition as *'mavil'*, *'mavulla'*, *'mavinte'*, *'mavo'*, *'mavaya'*, *'mavakkiya'*, *'mavennal'*, *'mavundu'*, *'mavilla'*, *'mavum'*, *'mavanu'*, *'mavinu'*, *'mavalle'*, *'maville'* etc. The same word can come with two suffixes as in the words *'mavilninnu'*, *'mavillengil'*, *'mavayathinal'*, *'mavayirikkum'*, *'mavanullathu'*, *'mavupayogichu'*, *'mavileykkanu', 'mavundengil', 'mavanithu', 'mavilekkayi', 'mavillathayi', 'mavilottu', 'mavukalum', 'mavinoduchernnu'* etc. The word *'mavu'* can come with three affixes as the cases *'mavilninnukondu', 'mavukalumayi', 'mavukalumanu'* etc and also with four affixes as in the cases *'mavundayirunnappol'* and *'mavilekkayirikkum'*. Similar inflections happens in the case of verbs also. For example the verb *'varuka'* can take one affix as in *'vannal', 'vannilla', 'vannittu'* etc. The verb can take two affixes as in the case *'vannennal', 'vannittundu', 'vannittanu', 'vannittilla'* etc.

Suffixes in Malayalam

The suffixes used in Malayalam language can be divided in to singular suffixes, binary suffixes, triple suffixes and suffixes with more than three parts. They can be detailed as the following:

1. **Singular suffixes**

   - Suffixes which end with *'chillu'* - *'ththil', 'kal', 'ral', 'mbol', 'ngalil', 'mengil', 'ppol', 'uppol'* etc are examples which belong to this category.
   - Suffixes which end with *'chandrakkala'* - Few examples of such category include *'athu', 'manu', 'ru', 'rodu', 'rkku', 'rkkai', 'arundu', 'rnnu', 'mennu', 'nnathu', 'ttathu'*.
   - Suffixes which end with *'u', 'e'* - *'rathu', 'chchu', 'ththe', 'ththinte', 'kale', 'ththode', 'loode', 'katte', 've', 'nude', 'yalle'* etc belong to this category.
   - Suffixes which end with *'i'* - *'kumayi', 'koodi', 'dakki', 'kkayi', 'ippoyi', 'raadi'* etc are few in this category.
   - Suffixes which end with *'anunasika'* - *'kum', 'makaam', 'ththinum', 'yolam', 'kalum', 'injnjum'*.
   - Suffixes which end with *'a', 'o'* - Some of the cases are *'maya', 'malla', 'killa', 'ninna', 'unna', 'kulla', 'mo', 'yallo'*.

2. **Binary suffixes**

   These suffixes consist of two singular suffixes joined together. They can be further classified as the following:

- Suffixes which end with *'chillu'* - Some cases include *'yennal'*, *'unnappol'*, *'rumbol'* etc.
- Suffixes which end with *'chandrakkala'* - *'ththileykku'*, *'mbaththekku'*, *'yumanu'*, *'mbozhanu'*, *'ngalkku'*, *'yittullathu'*, *'ninnanu'*, *'kondathinu'*, *'mayundu'*, *'nullathu'*, *'dunnathu'*, *'lekkulathu'*, *'mayittu'*, *'nathinu'*, *'nilekkanu'*, *'ilekkullathu'*, *'yilninnu'*, *'lanithu'*, *'ththilumundu'*.
- Suffixes which end with *'u'*, *'e'* - *'rnnathinu'*, *'kkappedunnu'*, *'kalpole'*, *'lanivide'*, *'njnjathalle'*.
- Suffixes which end with *'i'* - *'ththodukoodi'*, *'lumundayi'*, *'marundaayi'*, *'ththilekkayi'*, *'yallathaayi'*.
- Suffixes which end with *'a'* - This include *'rumaayirunna'*, *'mundaayirunna'*, *'ndndakkiya'*, *'ndndavunna'*, *'ndndakavunna'*, *'yanundavuka'*, *'naduththulla'*, *'kalkkalla'*, *'kallulla'*, *'mallaththa'*.

3. **Triple suffixes**

   These suffixes consist of three suffixes joined together. They can be further classified in to the following:

   - Suffixes which end with *'chillu'* - They include *'ndndakkiyennal'*, *'ndndavukayennal'*, *'ngngittillengil'*, *'markadiyil'*, *'markidayil'*, *'ndndayirunnappol'*.
   - Suffixes which end with *'i'* - They include *'mundayirunnathayi'*, *'inodoppamayi'*.
   - Suffixes which end with *'chandrakkala'* - *'kalilonnanu'*, *'kalolottallaththathu'*, *'kalilottallaththathanu'*, *'iyappozhanu'*.
   - Suffixes which end with *'u'* - They include *'kkumaayirunnu'*, *'umundaayirunnu'*, *'ththilekkayirunnu'*.

4. **More than three suffixes**

   Here more than three suffixes are joined together to form a single suffix. They include *'lokkeyundakarundengil'*, *'kalilottallaththathanennanu'*, *'ilekkayirikkum'*, *'millaththathukondulla'* etc.

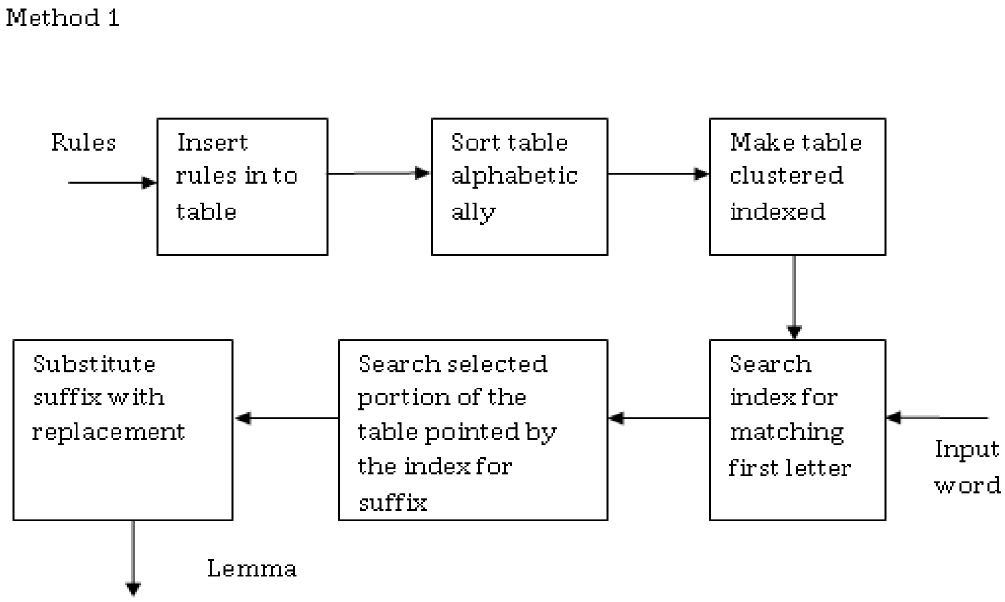

Method 1
Rules
Insert rules in to table
Sort table alphabetic ally
Make table clustered indexed
Search index for matching first letter
Input word
Search selected portion of the table pointed by the index for suffix
Substitute suffix with replacement
Lemma

Figure 2.2: Architecture of dictionary based method

2.3 System architecture

Here, two methods are proposed, Method-1, which is a dictionary based and Method-2, which is a suffix tree based method. The suffix-replacement dictionary which is developed in Method-1 is used to create the suffix tree. The two methods are illustrated in the Fig. 2.2 and Fig. 2.3. The concept of suffix tree has been slightly modified. In actual definition of suffix tree, except for the root, every internal node has at least two children and each edge is labeled with a non-empty substring of the search word S. The tree implemented here does not pose any such restrictions. Here a node can have empty substring of S. The tree is being pickled in order to make it a permanent storage. Tree pickling is a concept in python programming whereby permanent data structures can be created.

2.4 Methodology

- Dictionary based method

  The method proposed here uses a suffix replacement methodology where it creates a Malayalam dictionary of suffixes and replacements. This method follows a lookup table approach as in English stemmers, but the difference

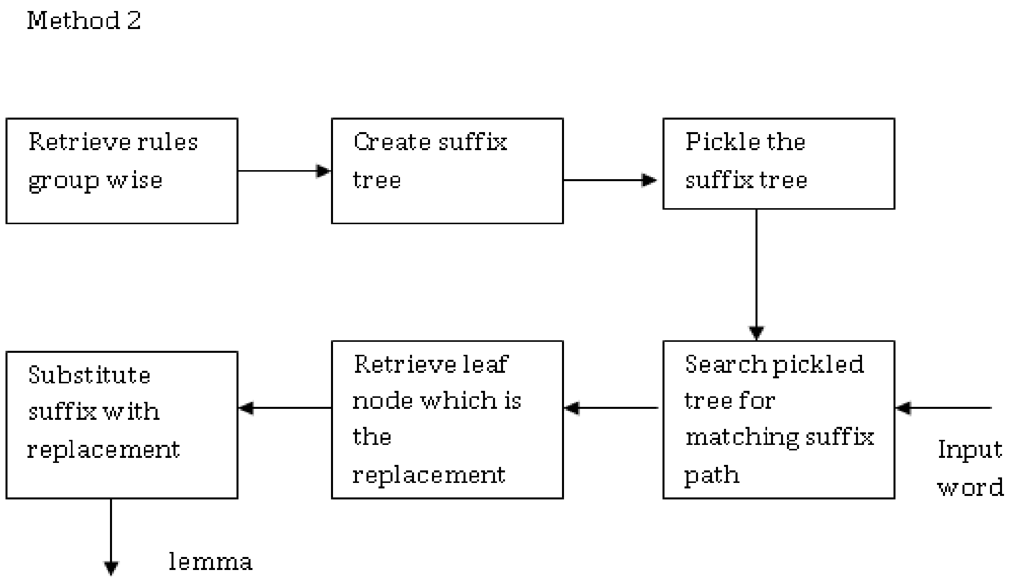

Method 2
Retrieve rules group wise
Create suffix tree
Pickle the suffix tree
Search pickled tree for matching suffix path
Input word
Retrieve leaf node which is the replacement
Substitute suffix with replacement
lemma

Figure 2.3: Architecture of tree based method

is that, in latter the lookup table is maintained only for some exceptional cases. Here the suffix is not stripped, but the suffix itself gets replaced with another.
Eg:- *'avalude'* - *'aval'*, *'vannupoyi'* - *'varuka'*, *'kazhukum'* - *'kazhukuka'*, *'pokanayirikkum'* - *'pokuka'*, *'thengilninnanu'* - *'thengu'*. The Malayalam suffix-replacement dictionary is being developed using MySQL. Since the table is a large one, in order to reduce the searching time, it is clustered indexed as shown in Fig. 2.4. The table is permanently sorted based on the alphabetical order of the suffix as we can see in Fig. 2.5. The clustered index is created based on the first letter of suffixes. The search is first directed to the index table which consists of the unique first letter of the suffixes and pointers to the first occurrence of the suffix starting with that unique first letter in the dictionary. The largest suffix from the word is found out and is substituted with the replacement in the dictionary.

For a given input word(token), the identification of the suffix starts from the first letter itself and only that portion of the table consisting of the suffixes starting with that letter is being searched. This limitation is imposed due to the presence of the clustered index. If a suffix could not be found in that area, the algorithm takes the next letter of the token and searches the table in the limited area containing suffixes starting with that letter. Once a proper suffix is found, it is replaced with the corresponding replacement. More suffix replacement pairs are given in Appendix A.

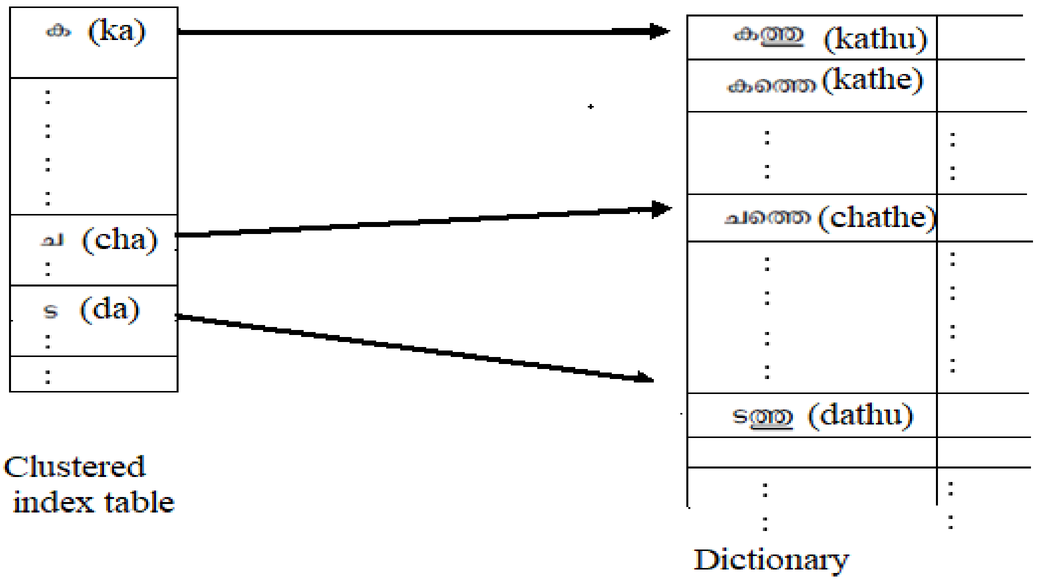
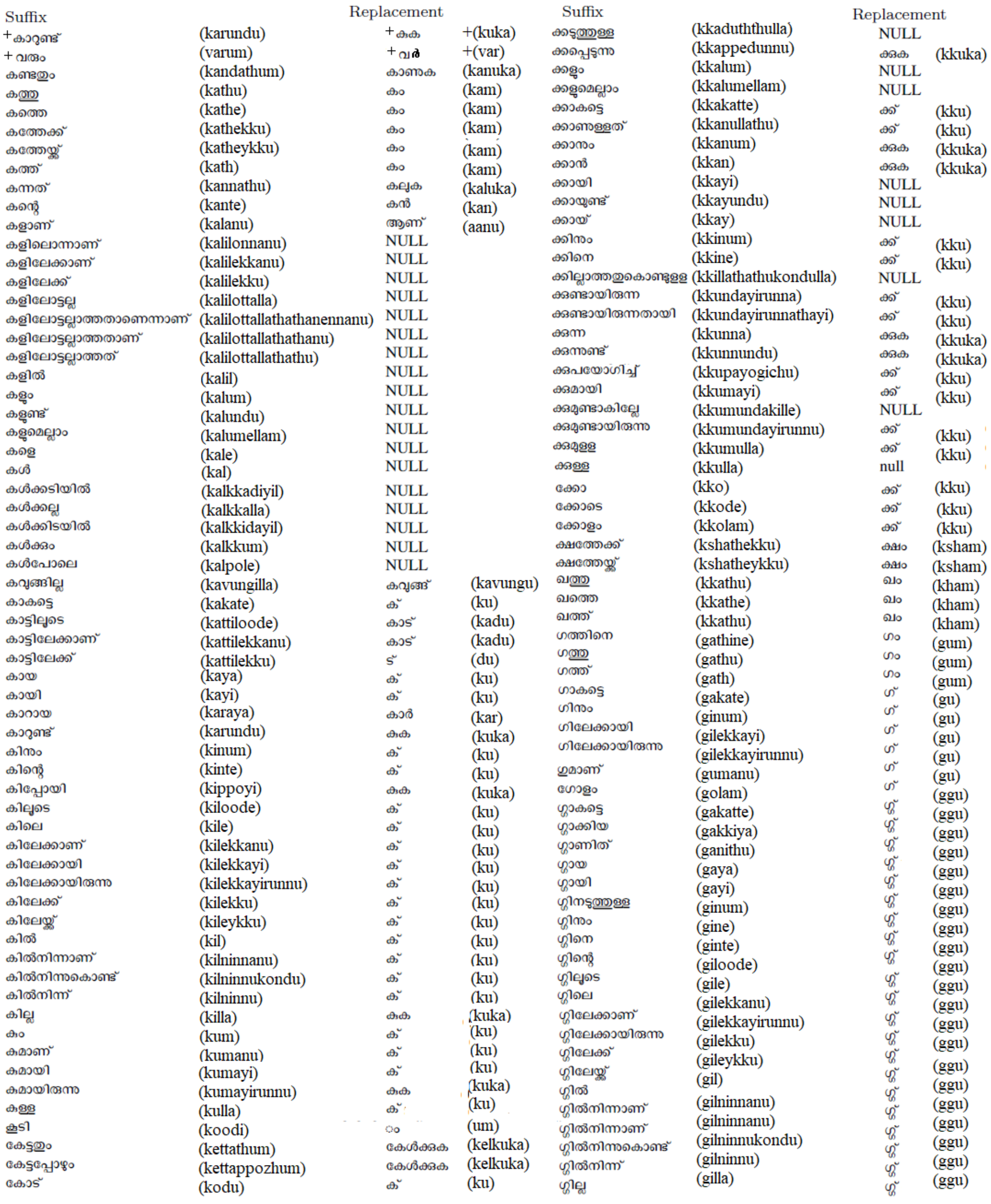


FIGURE 2.4: Indexed table

- Tree based method

In this method, a tree is constructed from the set of rules available in the database. Here, the rules are retrieved group wise, where one group consists of rules which start with the same first letter. The tree is being pickled using the pickling technique of python which makes the tree a permanent structure. This permanent tree is being queried in search of the suffix. The root node of the tree is the head node without storing any identifier. The first level children are unique first letter of the suffixes. The second level nodes are formed by taking the unique prefixes of the suffixes in the dictionary after removing the first letter. The third level nodes are constructed by taking the rest of the suffix after removing the prefix from it. The leaf nodes of the tree are the replacements. The method passes through two phases where the first phase is dealing with tree creation which is shown in Figs. 2.6 to 2.8 and tree pickling. The second phase is dealing with searching the tree for a matching suffix path and replacing the matched string in the word with the leaf node of that path. The search starts with the first letter of the word. The letter is compared with the identifier of the nodes in the first level children. Once a match is found, the letter is removed from the word and the unique prefixes of the resultant string is taken. The prefixes are now compared with the second level children of the identified path. Once a match is found up to the third level,



Suffix		Replacement		Suffix		Replacement	
+കാറുണ്ട്	(karundu)	+കുക	+(kuka)	ക്കടുത്തുള്ള	(kkaduththulla)	NULL	
+ വരും	(varum)	+ വർ	+(var)	ക്കപ്പെടുന്നു	(kkappedunnu)	ക്കുക	(kkuka)
കണ്ടതും	(kandathum)	കാണുക	(kanuka)	ക്കളും	(kkalum)	NULL	
കത്തു	(kathu)	കം	(kam)	ക്കളുമെല്ലാം	(kkalumellam)	NULL	
കത്തെ	(kathe)	കം	(kam)	ക്കാകട്ടെ	(kkakatte)	ക്ക്	(kku)
കത്തേക്ക്	(kathekku)	കം	(kam)	ക്കാണുള്ളത്	(kkanullathu)	ക്ക്	(kku)
കത്തേയ്ക്ക്	(katheykku)	കം	(kam)	ക്കാനും	(kkanum)	ക്കുക	(kkuka)
കത്ത്	(kath)	കം	(kam)	ക്കാൻ	(kkan)	ക്കുക	(kkuka)
കന്നത്	(kannathu)	കലുക	(kaluka)	ക്കായി	(kkayi)	NULL	
കന്റെ	(kante)	കൻ	(kan)	ക്കായുണ്ട്	(kkayundu)	NULL	
കളാണ്	(kalanu)	ആണ്	(aanu)	ക്കായ്	(kkay)	NULL	
കളിലൊന്നാണ്	(kalilonnanu)	NULL		ക്കിനും	(kkinum)	ക്ക്	(kku)
കളിലേക്കാണ്	(kalilekkanu)	NULL		ക്കിനെ	(kkine)	ക്ക്	(kku)
കളിലേക്ക്	(kalilekku)	NULL		ക്കില്ലാത്തതുകൊണ്ടുള്ള	(kkillathathukondulla)	NULL	
കളിലോട്ടല്ല	(kalilottalla)	NULL		ക്കുണ്ടായിരുന്ന	(kkundayirunna)	ക്ക്	(kku)
കളിലോട്ടല്ലാത്തതാണെന്നാണ്	(kalilottallathathanennanu)	NULL		ക്കുണ്ടായിരുന്നതായി	(kkundayirunnathayi)	ക്ക്	(kku)
കളിലോട്ടല്ലാത്തതാണ്	(kalilottallathathanu)	NULL		ക്കുന്ന	(kkunna)	ക്കുക	(kkuka)
കളിലോട്ടല്ലാത്തത്	(kalilottallathathu)	NULL		ക്കുന്നുണ്ട്	(kkunnundu)	ക്കുക	(kkuka)
കളിൽ	(kalil)	NULL		ക്കുപയോഗിച്ച്	(kkupayogichu)	ക്ക്	(kku)
കളും	(kalum)	NULL		ക്കുമായി	(kkumayi)	ക്ക്	(kku)
കളുണ്ട്	(kalundu)	NULL		ക്കുമുണ്ടാകില്ലേ	(kkumundakille)	NULL	
കളുമെല്ലാം	(kalumellam)	NULL		ക്കുമുണ്ടായിരുന്നു	(kkumundayirunnu)	ക്ക്	(kku)
കളെ	(kale)	NULL		ക്കുമുള്ള	(kkumulla)	ക്ക്	(kku)
കൾ	(kal)	NULL		ക്കുള്ള	(kkulla)	null	
കൾക്കടിയിൽ	(kalkkadiyil)	NULL		ക്കോ	(kko)	ക്ക്	(kku)
കൾക്കല്ല	(kalkkalla)	NULL		ക്കോടെ	(kkode)	ക്ക്	(kku)
കൾക്കിടയിൽ	(kalkkidayil)	NULL		ക്കോളം	(kkolam)	ക്ക്	(kku)
കൾക്കും	(kalkkum)	NULL		ക്ഷത്തേക്ക്	(kshathekku)	ക്ഷം	(ksham)
കൾപോലെ	(kalpole)	NULL		ക്ഷത്തേയ്ക്ക്	(kshatheykku)	ക്ഷം	(ksham)
കവുങ്ങില്ല	(kavungilla)	കവുങ്ങ്	(kavungu)	ഖത്തു	(kkathu)	ഖം	(kham)
കാകട്ടെ	(kakate)	ക്	(ku)	ഖത്തെ	(kkathe)	ഖം	(kham)
കാട്ടിലൂടെ	(kattiloode)	കാട്	(kadu)	ഖത്ത്	(kkathu)	ഖം	(kham)
കാട്ടിലേക്കാണ്	(kattilekkanu)	കാട്	(kadu)	ഗത്തിനെ	(gathine)	ഗം	(gum)
കാട്ടിലേക്ക്	(kattilekku)	ട്	(du)	ഗത്തു	(gathu)	ഗം	(gum)
കായ	(kaya)	ക്	(ku)	ഗത്ത്	(gath)	ഗം	(gum)
കായി	(kayi)	ക്	(ku)	ഗാകട്ടെ	(gakate)	ഗ്	(gu)
കാറായ	(karaya)	കാർ	(kar)	ഗിനും	(ginum)	ഗ്	(gu)
കാറുണ്ട്	(karundu)	കുക	(kuka)	ഗിലേക്കായി	(gilekkayi)	ഗ്	(gu)
കിനും	(kinum)	ക്	(ku)	ഗിലേക്കായിരുന്നു	(gilekkayirunnu)	ഗ്	(gu)
കിന്റെ	(kinte)	ക്	(ku)	ഗുമാണ്	(gumanu)	ഗ്	(gu)
കിപ്പോയി	(kippoyi)	കുക	(kuka)	ഗോളം	(golam)	ഗ്	(ggu)
കിലൂടെ	(kiloode)	ക്	(ku)	ഗ്ഗാകട്ടെ	(gakatte)	ഗ്ഗ്	(ggu)
കിലെ	(kile)	ക്	(ku)	ഗ്ഗാക്കിയ	(gakkiya)	ഗ്ഗ്	(ggu)
കിലേക്കാണ്	(kilekkanu)	ക്	(ku)	ഗ്ഗാണിത്	(ganithu)	ഗ്ഗ്	(ggu)
കിലേക്കായി	(kilekkayi)	ക്	(ku)	ഗ്ഗായ	(gaya)	ഗ്ഗ്	(ggu)
കിലേക്കായിരുന്നു	(kilekkayirunnu)	ക്	(ku)	ഗ്ഗായി	(gayi)	ഗ്ഗ്	(ggu)
കിലേക്ക്	(kilekku)	ക്	(ku)	ഗ്ഗിനടുത്തുള്ള	(ginum)	ഗ്ഗ്	(ggu)
കിലേയ്ക്ക്	(kileykku)	ക്	(ku)	ഗ്ഗിനും	(gine)	ഗ്ഗ്	(ggu)
കിൽ	(kil)	ക്	(ku)	ഗ്ഗിനെ	(ginte)	ഗ്ഗ്	(ggu)
കിൽനിന്നാണ്	(kilninnanu)	ക്	(ku)	ഗ്ഗിന്റെ	(giloode)	ഗ്ഗ്	(ggu)
കിൽനിന്നുകൊണ്ട്	(kilninnukondu)	ക്	(ku)	ഗ്ഗിലൂടെ	(gile)	ഗ്ഗ്	(ggu)
കിൽനിന്ന്	(kilninnu)	ക്	(ku)	ഗ്ഗിലെ	(gilekkanu)	ഗ്ഗ്	(ggu)
കില്ല	(killa)	കുക	(kuka)	ഗ്ഗിലേക്കാണ്	(gilekkayirunnu)	ഗ്ഗ്	(ggu)
കും	(kum)	ക്	(ku)	ഗ്ഗിലേക്കായിരുന്നു	(gilekku)	ഗ്ഗ്	(ggu)
കുമാണ്	(kumanu)	ക്	(ku)	ഗ്ഗിലേക്ക്	(gileykku)	ഗ്ഗ്	(ggu)
കുമായി	(kumayi)	ക്	(ku)	ഗ്ഗിലേയ്ക്ക്	(gil)	ഗ്ഗ്	(ggu)
കുമായിരുന്നു	(kumayirunnu)	കുക	(kuka)	ഗ്ഗിൽ	(gilninnanu)	ഗ്ഗ്	(ggu)
കുള്ള	(kulla)	ക്	(ku)	ഗ്ഗിൽനിന്നാണ്	(gilninnanu)	ഗ്ഗ്	(ggu)
കൂടി	(koodi)	◌ം	(um)	ഗ്ഗിൽനിന്നാണ്	(gilninnukondu)	ഗ്ഗ്	(ggu)
കേട്ടതും	(kettathum)	കേൾക്കുക	(kelkuka)	ഗ്ഗിൽനിന്നുകൊണ്ട്	(gilninnu)	ഗ്ഗ്	(ggu)
കേട്ടപ്പോഴും	(kettappozhum)	കേൾക്കുക	(kelkuka)	ഗ്ഗിൽനിന്ന്	(gilla)	ഗ്ഗ്	(ggu)
കോട്	(kodu)	ക്	(ku)	ഗ്ഗില്ല		ഗ്ഗ്	

FIGURE 2.5: Dictionary

Algorithm 1: Dictionary based method

```
Input: Input document and suffix-replacement dictionary which is clustered
       indexed.
Output: lemma.
read the text and tokenize the sentences;
remove stop words;
$i = 0$;
for each token tk do
    ch = $tk[i]$;
    repeat
        search the clustered index for the entry ch;
        if found then
            search the portion of the table pointed by the pointer for the suffix
              starting with ch;
            if found then
                retrieve the replacement from the table corresponding to that
                  suffix;
                replace the suffix with the replacement;
                display the lemma and exit from loop;
            else
                $i = i + 1$;
            end
        else
            continue;
        end
    until $i < n$;
    if $i == n$ then
        display the token unchanged
end
```

the leaf node is taken as the replacement of the suffix. If at any level there is a mismatch, we do not backtrack to the previous level, but advance the pointer in the word to the second letter and do the above process. The backtracking is avoided by the careful selection of prefixes in the second level node formation. All the prefixes selected are unique and this reduce the chance of backtracking to nil. Execution times of both tree based method and dictionary based method is calculated for 100 Malayalam words from online documents and is shown in the Fig. 2.17. Tree based method which has a time complexity of $O(n^2)$ shows an average execution time of 0.00073s and the dictionary based method shows an average execution time of 0.0084s. Tree has reduced the searching time very much since we need to search only a portion of the tree.

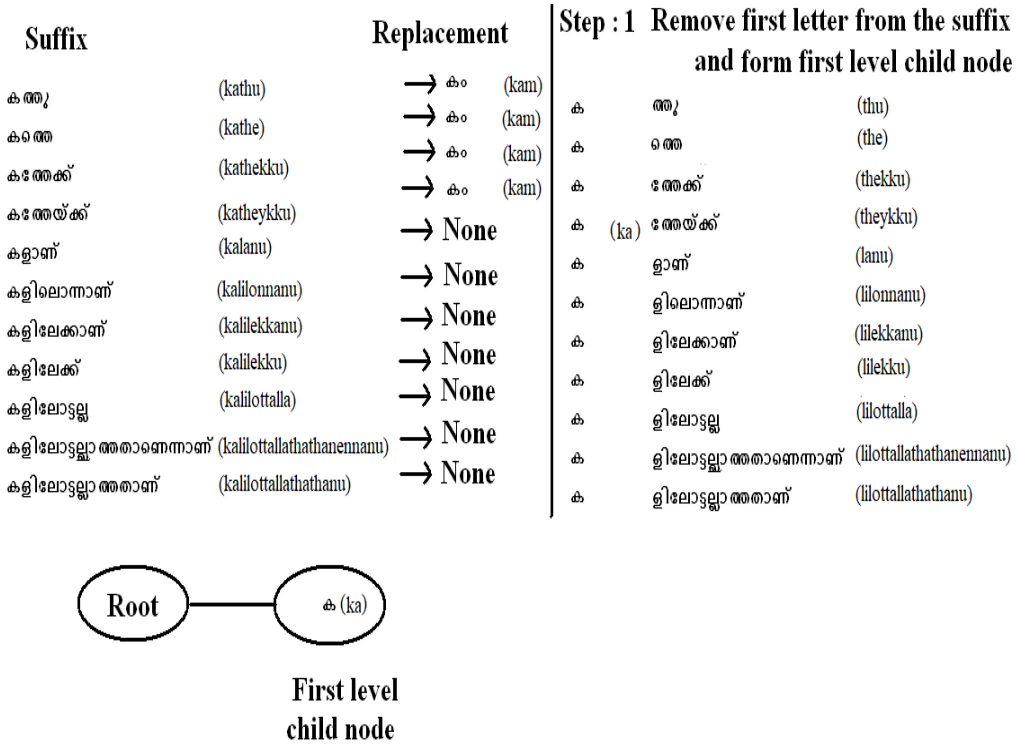

Suffix
Replacement
Step : 1 Remove first letter from the suffix and form first level child node
കത്തു (kathu) → കം (kam)
കത്തെ (kathe) → കം (kam)
കത്തേക്ക് (kathekku) → കം (kam)
കത്തേയ്ക്ക് (katheykku) → കം (kam)
കളാണ് (kalanu) → None
കളിലൊന്നാണ് (kalilonnanu) → None
കളിലേക്കാണ് (kalilekkanu) → None
കളിലേക്ക് (kalilekku) → None
കളിലോട്ടല്ല (kalilottalla) → None
കളിലോട്ടല്ലാത്തതാണെന്നാണ് (kalilottallathathanennanu) → None
കളിലോട്ടല്ലാത്തതാണ് (kalilottallathathanu) → None
ക (ka)
ത്തു (thu)
ത്തെ (the)
ത്തേക്ക് (thekku)
ത്തേയ്ക്ക് (theykku)
ളാണ് (lanu)
ളിലൊന്നാണ് (lilonnanu)
ളിലേക്കാണ് (lilekkanu)
ളിലേക്ക് (lilekku)
ളിലോട്ടല്ല (lilottalla)
ളിലോട്ടല്ലാത്തതാണെന്നാണ് (lilottallathathanennanu)
ളിലോട്ടല്ലാത്തതാണ് (lilottallathathanu)
Root
ക (ka)
First level child node

FIGURE 2.6: Step1: Lemmatization

2.5 N-gram rules

N-gram rules are formed by considering more number of letters to the left of actual suffix.

- Bi-gram rules

  As in Fig. 2.1, we can see that, *'thinu'* can't be taken as a suffix, as it can come with many other affixes and can create specific rules. It can be converted to a bi-gram rule by considering one more letter on the left side and the same is shown in the entries one and two in Fig. 2.11. *'thinu'* along with the letter *'na'* and *'nja'* form two different suffixes *'nathinu'* and *'njathinu'* and get replaced with two different replacements *'zhuka'* and *'yuka'*. Similarly the *'ttathinu'*, *'yathinu'*, *'thathinu'* are also bi-gram suffixes which are formed by considering one more letter*('tta', 'ya', 'tha'* respectively) on the left side of *'thinu'*. They get converted to the replacements like *'kkuka'*, *'vuka'*, *'um'* respectively.

- Tri-gram rules

  Tri-gram rules are formed by considering two more letters to the left of the

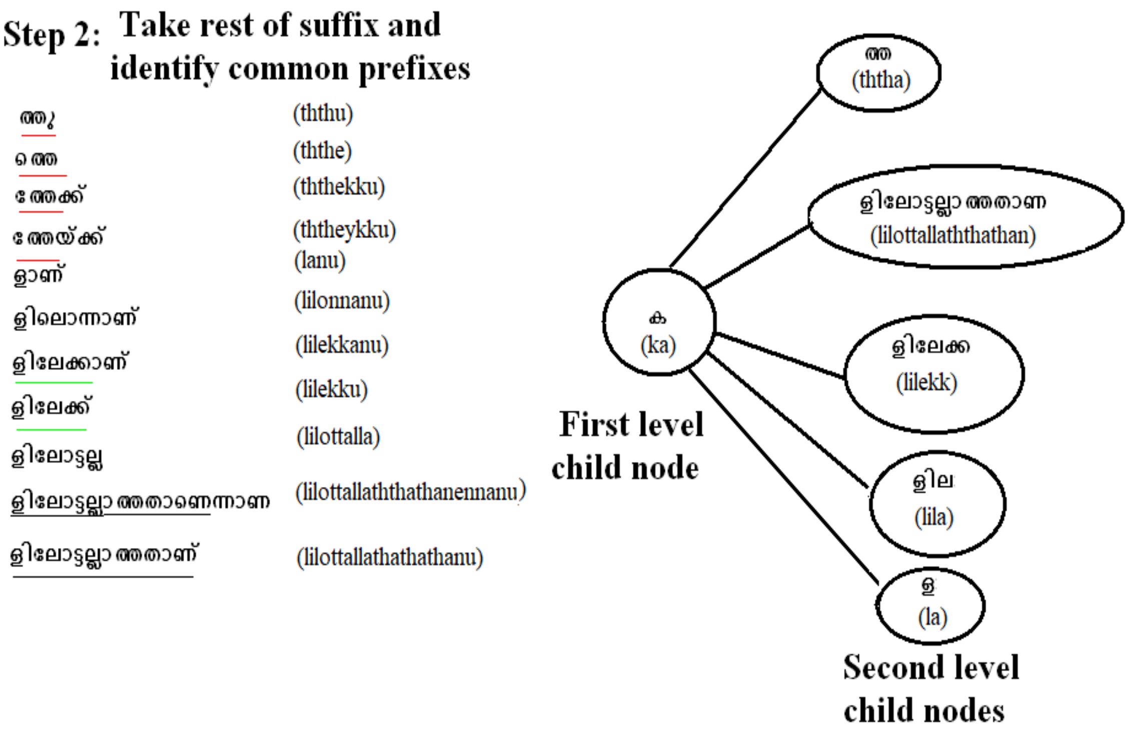


Figure 2.7: Step2: Lemmatization

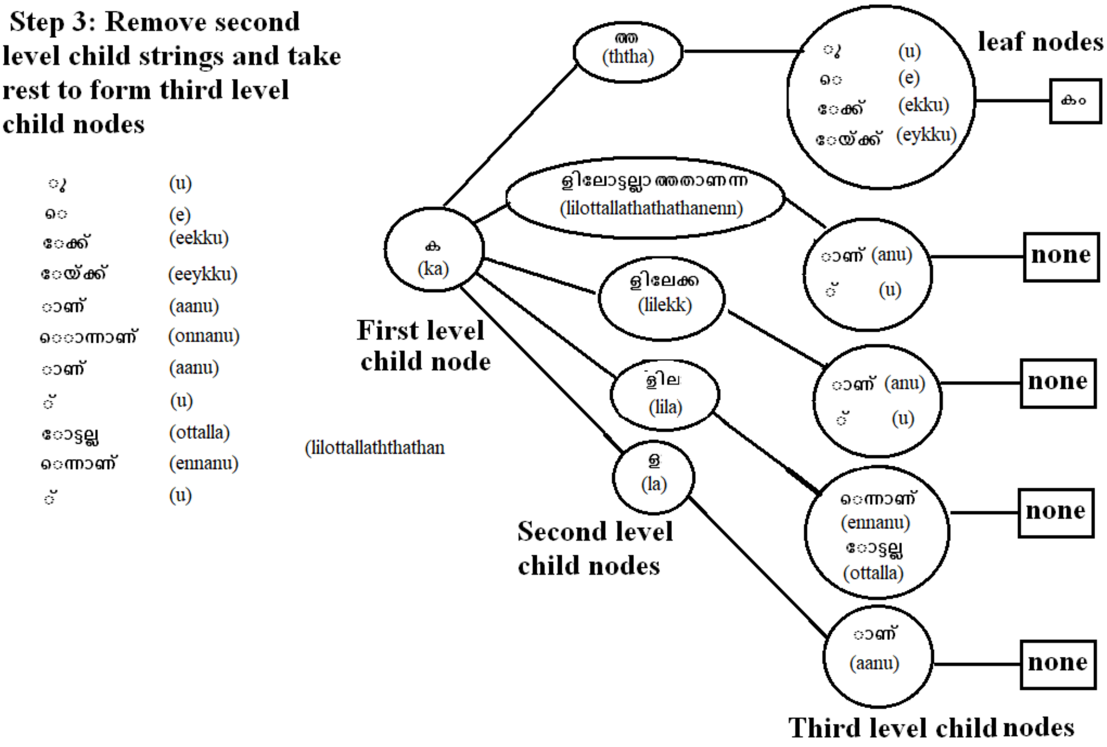
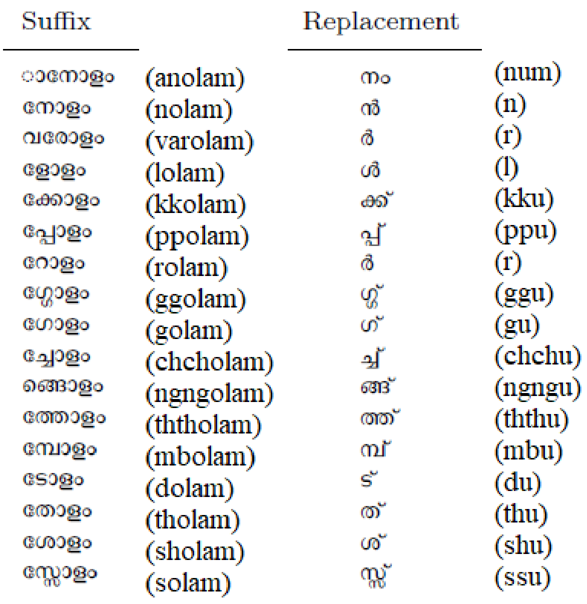


Figure 2.8: Step3: Lemmatization

Algorithm 2: Tree creation

```
Input: Rules from the database.
Output: Tree.
read the rules of the form $S - R$ group wise from the suffix replacement
  dictionary;
$SF_i$ = unique first letter of suffixes;
where $i$ ranges from 1 to $n$ and $n$ is the number of unique first letter;
create root node $R_n$;
create first level child nodes of the tree with $SF_i$;
for each group $i$ do take the suffixes $S_j$ where $j$ ranges from 1 to $m$;
and $m$ is the no of suffixes in a group $(S - R)_i$;
$RS_j$ = $S_j$ - $SF_i$;
$PR_k$ = Unique prefixes of $RS_j$;
where $k$ ranges from 1 to $p$ and $p$ is the no of unique prefixes in that group ;
create second level child nodes with $PR_k$;
$RPR$ = $RS_j$ - $PR_k$;
create third level child nodes with $RPR$;
create leaf nodes $R_l$ which corresponds to the replacement in $S - R$ rule, where $l$
  ranges from 1 to $r$ and $r$ is the no of replacements ;
```



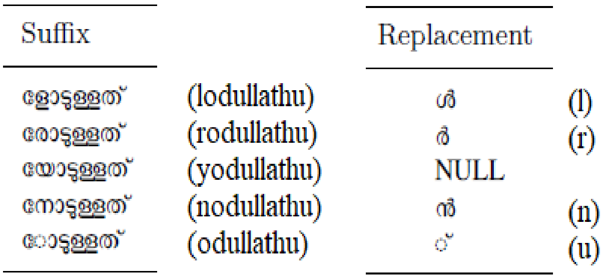
Suffix		Replacement	
ാനോളം	(anolam)	നം	(num)
നോളം	(nolam)	ൻ	(n)
വരോളം	(varolam)	ർ	(r)
ളോളം	(lolam)	ൾ	(l)
ക്കോളം	(kkolam)	ക്ക്	(kku)
പ്പോളം	(ppolam)	പ്പ്	(ppu)
റോളം	(rolam)	ർ	(r)
ഗ്ഗോളം	(ggolam)	ഗ്ഗ്	(ggu)
ഗോളം	(golam)	ഗ്	(gu)
ച്ചോളം	(chcholam)	ച്ച്	(chchu)
ങ്ങോളം	(ngngolam)	ങ്ങ്	(ngngu)
ത്തോളം	(ththolam)	ത്ത്	(ththu)
മ്പോളം	(mbolam)	മ്പ്	(mbu)
ടോളം	(dolam)	ട്	(du)
തോളം	(tholam)	ത്	(thu)
ശോളം	(sholam)	ശ്	(shu)
സ്സോളം	(solam)	സ്സ്	(ssu)

FIGURE 2.9: Suffixes which end with *'anunasika'*

Algorithm 3: Tree search(tree,word)

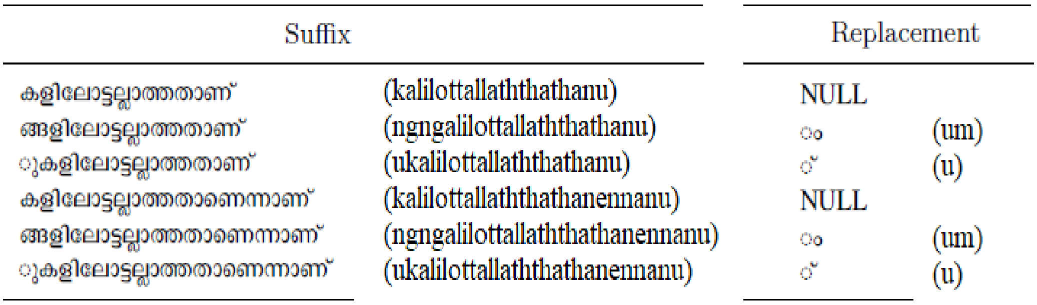
```
Input: Word to be searched.
Output: lemma.
$c$ = first letter of $W_i$ where $W_i$ is the $i^{th}$ word of the document;
children = level1 nodes of the tree;
for each child in children do
    compare child.identifier with $c$;
    if not match then
        break;
    else
        nextnode = child;
        $CW_i = W_i$ - $c$;
        children = childnodes of nextnode;
        find prefixes of $CW_i$ ;
        for child in children do
            compare prefixes with child.identifier;
            if not match then
                break;
            else
                nextnode = child;
                $RW_i = CW_i$ - child.identifier;
                children = childnodes of nextnode;
                for child in children do
                    compare $RW_i$ with child.identifier;
                    if not match then
                        call Tree search(tree, $CW_i$)
                    else
                        return leaf node
                    end
                end
            end
        end
    end
end
```

actual suffix word. The entries 3, 4, 6, 8 and 11 in Fig. 2.11 are examples of such tri-gram rules. As sample rule we can take *'ttiyathinu'* - *'ttuka'* where two letters *'tti'* and *'ya'* to the left of *'thinu'* is also considered for creating the rule. The third entry in Fig. 2.9 *'varolam'* - *'r'* is also an example of tri-gram rule where the suffix is formed by adding two letters *'va'* and *'r'* along with the actual suffix *'olam'*. All the entries in Fig. 2.10 and Fig. 2.11 are examples of n-gram rules where more than 3 letters to the left of actual suffix is considered for the creation of the L.H.S of the rules.



Suffix		Replacement	
ളോടുള്ളത്	(lodullathu)	ൾ	(l)
രോടുള്ളത്	(rodullathu)	ർ	(r)
യോടുള്ളത്	(yodullathu)	NULL	
നോടുള്ളത്	(nodullathu)	ൻ	(n)
ോടുള്ളത്	(odullathu)	്	(u)

FIGURE 2.10: Suffixes which end with *'chandrakkala'*

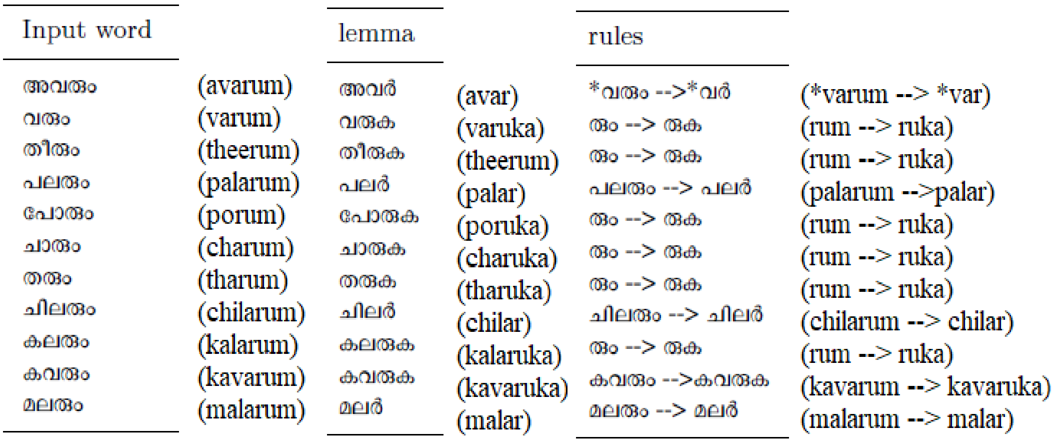


Suffix		Replacement	
കളിലോട്ടല്ലാത്തതാണ്	(kalilottallaththathanu)	NULL	
ങ്ങളിലോട്ടല്ലാത്തതാണ്	(ngngalilottallaththathanu)	ം	(um)
ുകളിലോട്ടല്ലാത്തതാണ്	(ukalilottallaththathanu)	്	(u)
കളിലോട്ടല്ലാത്തതാണെന്നാണ്	(kalilottallaththathanennanu)	NULL	
ങ്ങളിലോട്ടല്ലാത്തതാണെന്നാണ്	(ngngalilottallaththathanennanu)	ം	(um)
ുകളിലോട്ടല്ലാത്തതാണെന്നാണ്	(ukalilottallaththathanennanu)	്	(u)

FIGURE 2.11: Suffixes which ends with *'aanu'*

2.6 Comparison with existing methods

Suffix tree method is the one used in human genome project, where a tree of suffixes is created in which every node has a single letter of the substring to be searched, but here the nodes in level two and level three can have more than a single letter as the node identifier. The node in level one has only single letter as the node identifier, which in this case is the starting letter of the suffix. As mentioned earlier, the tree created in the proposed method avoids backtracking with a careful selection of prefixes. Initially common prefix with maximum length is found out and the words with these prefixes are removed from the list. From the remaining list, again common prefix with maximum length is found out. This process continues until there are no common prefixes. The stemmer lalitha [62] has mentioned about the suffix *'yolam'*, the present method has a related similar set which consists of the rules given in Fig. 2.9. The stemmer lalitha has given the suffix *'kalodullathu'* and few similar suffixes identified by the present lemmatizer are given in Fig. 2.10. While the lalitha stemmer has implemented the suffix, *'kalilottallathathanennanu'* the method here has identified some related rules as given in Fig. 2.11. Stemmer lalitha uses a suffix dictionary and the method here, uses a suffix-replacement dictionary which is permanently alphabetically sorted and clustered indexed, followed by a tree



Input word		lemma		rules	
അവരും	(avarum)	അവർ	(avar)	*വരും -->*വർ	(*varum --> *var)
വരും	(varum)	വരുക	(varuka)	രും --> രുക	(rum --> ruka)
തീരും	(theerum)	തീരുക	(theerum)	രും --> രുക	(rum --> ruka)
പലരും	(palarum)	പലർ	(palar)	പലരും --> പലർ	(palarum -->palar)
പോരും	(porum)	പോരുക	(poruka)	രും --> രുക	(rum --> ruka)
ചാരും	(charum)	ചാരുക	(charuka)	രും --> രുക	(rum --> ruka)
തരും	(tharum)	തരുക	(tharuka)	രും --> രുക	(rum --> ruka)
ചിലരും	(chilarum)	ചിലർ	(chilar)	ചിലരും --> ചിലർ	(chilarum --> chilar)
കലരും	(kalarum)	കലരുക	(kalaruka)	രും --> രുക	(rum --> ruka)
കവരും	(kavarum)	കവരുക	(kavaruka)	കവരും -->കവരുക	(kavarum --> kavaruka)
മലരും	(malarum)	മലർ	(malar)	മലരും --> മലർ	(malarum --> malar)

FIGURE 2.12: Rules

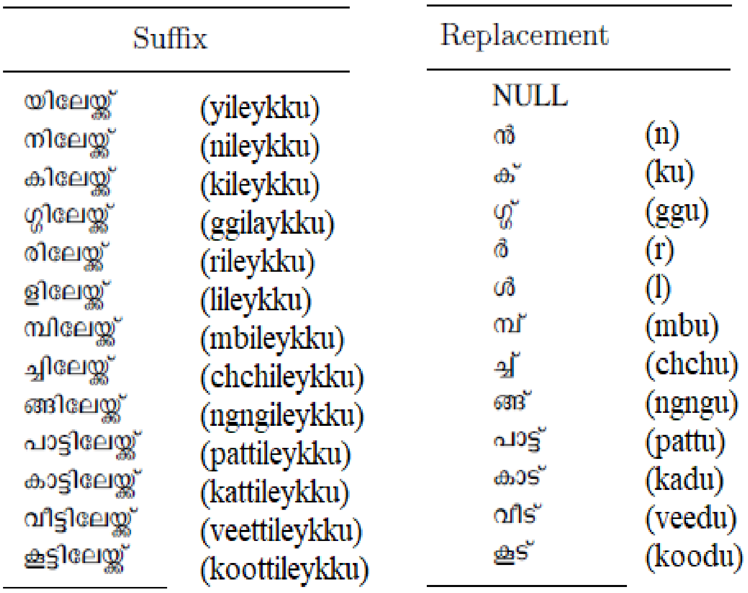
created out of it.

The Malayalam stemmer [62] searches for the suffix from right to left and considers only stemming where word *'angangal'* get converted to *'anga'*, but our method searches from left to right and generates the output *'anagam'* which is lemma, the real meaningful word. The suffixes identified in the words *'odunnu'*, *'odum'*, *'odumbhol'*, *'odarundu'*, *'odan'*, *'odiyappol'*, *'odippoyi'*, *'odimari'*, *'odivannu'*, *'odikkondu'*, *'odikkondirunnu'* etc get substituted with the replacement *'duka'* and result in the word *'oduka'* which is the correct lemma. But most of the Malayalam stemmers give the output *'odu'* which is only the stem.

The morphological analyzer and generator for Malayalam-Tamil translation [61] converts the input word *'varum'* to the output *'varu'*, but my method gives the correct output *'varuka'*. Another morphological analyzer which is based on a hybrid approach [60] gives the output *'odu'* for the input word *'odikkondirikkukayayirunnu'*, but our method gives the output *'oduka'* which is the correct meaningful word. More comparisons are given in Table 2.2. This table shows nine separate categories in which VRIKSH lemmatizer gives a better result when compared with Indic stemmer. The cases include examples from bi-gram and tri-gram rules.

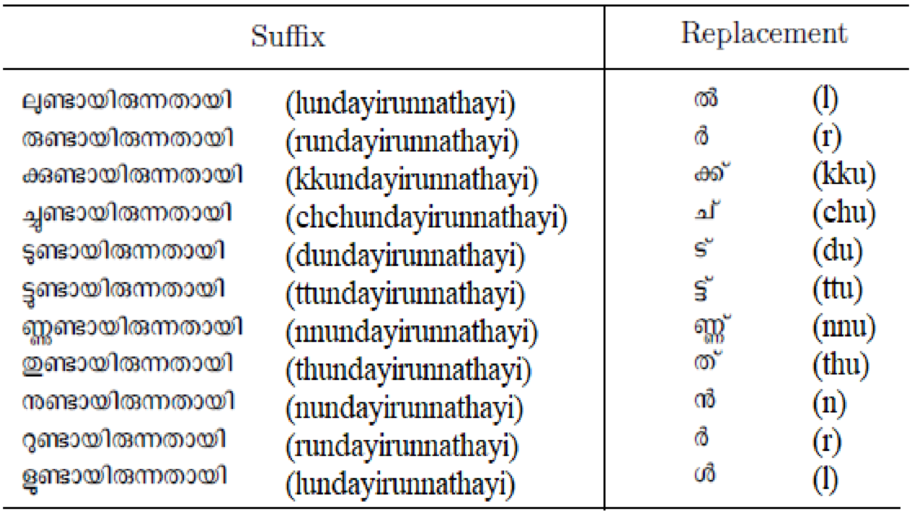
TABLE 2.2: Comparison of special cases with Indic stemmer

Cases	Results of VRIKSH lemmatizer
Case 1	The rule *’rum’* - *’r’* in online stemmer [63], stems the words *’avarum’* - *’avar’*, *’palarum’* - *’palar’* correctly but stems the words wrongly as *’varum’* - *’var’*, *’theerum’* - *’theer’*, *’chaarum’* - *’chaar’* , *’tharum’* - *’thar’*. The results correctly obtained in the new method and the addition of the extra rules is shown in Fig. 2.12. This will take the largest matching suffix and substitute with its replacement.
Case 2	Online stemmer has given only the rule *’thilekku’* - *’um’*, but the new lemmatizer has considered additional rules as in Fig. 2.13.
Case 3	While the Indic stemmer has considered only the rule *’mundayirunnathayi’* - *’um’+ ’undu’*, the new method has identified many additional rules as shown in Fig. 2.14.
Case 4	Many stemmers developed so far generalize the rules which lead to many stemming errors. For eg:- *’akaam’* *’um’*, but is true only for certain words like *’pakamakaam’* *’pakam’*. We can see that the rules fail in the case of the words like *’avaralakaam’*, *’mazhayakaam’*, *’avalakaam’*, *’vayasakaam’* etc. Some of the new suffixes identified by this method to sort out this issue are given in Fig. 2.15.
Case 5	Let us consider some words like *’veenathinu’*, *’marinjathinu’*, *’thottathinu’*, *’pettathinu’*, *’kattathinu’*, *’njettiyathinu’*, *’poyathinu’*, *’pottiyathinu’*, *’charithrathinu’*, *’ponnathinu’*. A generalized rule can’t be used here as they get converted to their root form in different ways. The solution is given in Fig. 2.16. The fourth, eighth, last two entries in Fig. 2.12 and last four entries in Fig. 2.13 are exceptions where the suffix and replacement is the entire word.
Case 6	The rules *’nanu’* = *’n’* + *’anu’*, *’nalla’* = *’n’* + *’alla’*, *’nilla’* = *’n’* + *’illa’* are some of the rules in Indic stemmer which again creates the stop word *’alla’*, which need to be removed again.The present method modified the rules as *’nanu’* - *’n’*, *’nalla’* - *’n’*, *’nilla’* *’n’*, where by it eliminates the creation of stop words. Similar is the case with the *’lanu’* = *’l’* + *’anu’*, *’lalla’* = *’l’* + *’alla’*, *’lilla’* = *’l’* + *’illa’*. Since the above rules create stop words *’anu’*, *’alla’*, *’illa’*, they are modified as *’lanu’* *’l’*, *’lalla’* - *’l’*, *’lilla’* - *’l’*.
Case 7	The rules *’ykanayi’* = *’ykan’* + *’avuka’* is actually creating a word which need to be stemmed again, so we have modified the replacement as *’ykuka’* and the rule *’kanayi’* = *’kan’* + *’avuka’* is also modified as the *’kanayi’* *’kuka’* which gives the real meaningful lemma.
Case 8	Unwanted stop words are again being created by the rules mentioned in online stemmer which can be seen in the following ones *’yaaniva’* = *’anu’* + *’iva’*, *’yullava’* = *’ulla’* + *’ava’*, *’yullathu’* = *’ulla’* + *’athu’*, *’yallo’* = *none* + *’allo’*. The proposed method has created new rules which replaces all these rules with null as R.H.S .
Case 9	The Indic stemmer has applied stemming rules separately for noun/verb. He has given the rule *’rilla’* = *’r’* + *’illa’*. This is true only for nouns i.e., *’avarilla’* = *’avar’* + *’illa’*. In case of verb *’kavarilla’*, the stemmed output should be *’kavaruka’* + *’illa’*. The new method takes in to consideration a lot of such stemming errors. Similarly the Indic stemmer has given the rules *’nil’* *’n’*, *’ril’* - *’r’*, *’yil’*- *null*, *’lil’* - *’l’*, but the new method has considered a lot of relevant rules like *’kil’* - *’k’*, *’ngil’* - *’ng’*, *’chil’* - *’ch’*, *’dil’* - *’d’*, *’nnil’* *’nn’*, *’thil’* - *’th’* and specifically *’mil’* - *’m’* Eg:- *’assamil’* - *’assam’* which is not addressed in Indic stemmer.



Suffix		Replacement	
യിലേയ്ക്ക്	(yileykku)	NULL	
നിലേയ്ക്ക്	(nileykku)	ൻ	(n)
കിലേയ്ക്ക്	(kileykku)	ക്	(ku)
ഗ്ഗിലേയ്ക്ക്	(ggilaykku)	ഗ്ഗ്	(ggu)
രിലേയ്ക്ക്	(rileykku)	ർ	(r)
ളിലേയ്ക്ക്	(lileykku)	ൾ	(l)
മ്പിലേയ്ക്ക്	(mbileykku)	മ്പ്	(mbu)
ച്ചിലേയ്ക്ക്	(chchileykku)	ച്ച്	(chchu)
ങ്ങിലേയ്ക്ക്	(ngngileykku)	ങ്ങ്	(ngngu)
പാട്ടിലേയ്ക്ക്	(pattileykku)	പാട്ട്	(pattu)
കാട്ടിലേയ്ക്ക്	(kattileykku)	കാട്	(kadu)
വീട്ടിലേയ്ക്ക്	(veettileykku)	വീട്	(veedu)
കൂട്ടിലേയ്ക്ക്	(koottileykku)	കൂട്	(koodu)

FIGURE 2.13: Suffixes which end with *'leykku'*

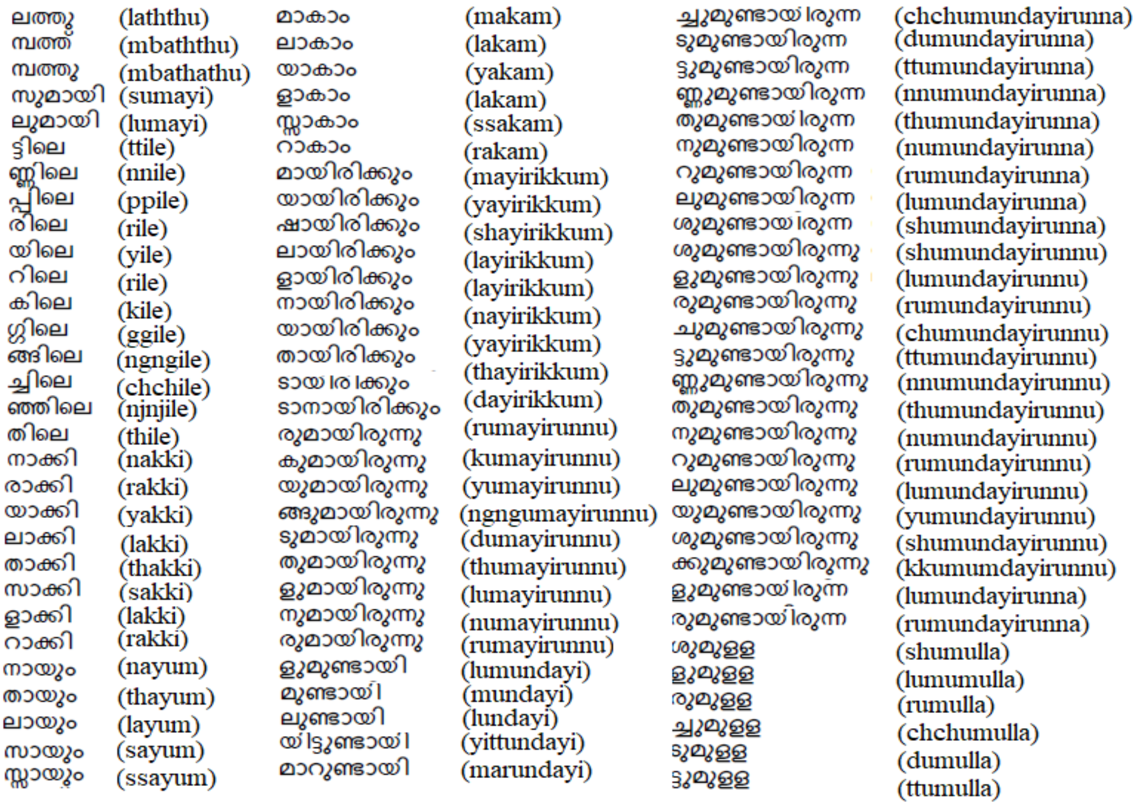


Suffix		Replacement	
ലുണ്ടായിരുന്നതായി	(lundayirunnathayi)	ൽ	(l)
രുണ്ടായിരുന്നതായി	(rundayirunnathayi)	ർ	(r)
ക്കുണ്ടായിരുന്നതായി	(kkundayirunnathayi)	ക്ക്	(kku)
ച്ചുണ്ടായിരുന്നതായി	(chchundayirunnathayi)	ച്	(chu)
ടുണ്ടായിരുന്നതായി	(dundayirunnathayi)	ട്	(du)
ട്ടുണ്ടായിരുന്നതായി	(ttundayirunnathayi)	ട്ട്	(ttu)
ണ്ണുണ്ടായിരുന്നതായി	(nnundayirunnathayi)	ണ്ണ്	(nnu)
തുണ്ടായിരുന്നതായി	(thundayirunnathayi)	ത്	(thu)
നുണ്ടായിരുന്നതായി	(nundayirunnathayi)	ൻ	(n)
റുണ്ടായിരുന്നതായി	(rundayirunnathayi)	ർ	(r)
ളുണ്ടായിരുന്നതായി	(lundayirunnathayi)	ൾ	(l)

FIGURE 2.14: Suffixes which end with *'ndaayirunnathaayi'*

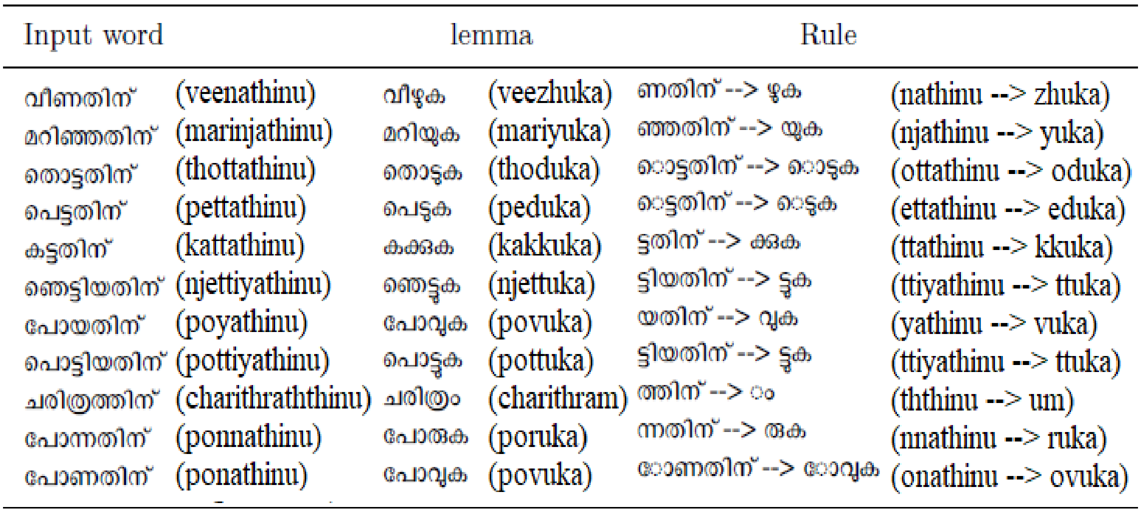
2.7 Data set

Online Malayalam documents from various domains are considered for testing the performance of the system developed. The documents selected are of varying sizes as shown in Table 2.4. The results obtained from the Vriksh lemmatizer and the target system is being manually checked by Malayalam speaking natives. Repeated study of the Malayalam documents revealed that, only 50% of the words come with suffixes and so need to be lemmatized. Out of that, majority of the words end with *'chandrakkala'*. The words which end with *'anunasika'* are less



ലത്തു	(laththu)	മാകാം	(makam)	ച്ചുമുണ്ടായ ിരുന്ന	(chchumundayirunna)
മ്പത്ത്	(mbaththu)	ലാകാം	(lakam)	ടുമുണ്ടായിരുന്ന	(dumundayirunna)
മ്പത്തു	(mbathathu)	യാകാം	(yakam)	ട്ടുമുണ്ടായിരുന്ന	(ttumundayirunna)
സുമായി	(sumayi)	ളാകാം	(lakam)	ണ്ണുമുണ്ടായിരുന്ന	(nnumundayirunna)
ലുമായി	(lumayi)	സ്സാകാം	(ssakam)	തുമുണ്ടായ ിരുന്ന	(thumundayirunna)
ട്ടിലെ	(ttile)	റാകാം	(rakam)	നുമുണ്ടായിരുന്ന	(numundayirunna)
ണ്ണിലെ	(nnile)	മായിരിക്കും	(mayirikkum)	റുമുണ്ടായിരുന്ന	(rumundayirunna)
പ്പിലെ	(ppile)	യായിരിക്കും	(yayirikkum)	ലുമുണ്ടായിരുന്ന	(lumundayirunna)
രിലെ	(rile)	ഷായിരിക്കും	(shayirikkum)	ശുമുണ്ടായ ിരുന്ന	(shumundayirunna)
യിലെ	(yile)	ലായിരിക്കും	(layirikkum)	ശുമുണ്ടായിരുന്നു	(shumundayirunnu)
റിലെ	(rile)	ളായിരിക്കും	(layirikkum)	ളുമുണ്ടായിരുന്നു	(lumundayirunnu)
കിലെ	(kile)	നായിരിക്കും	(nayirikkum)	രുമുണ്ടായിരുന്നു	(rumundayirunnu)
ഗ്ഗിലെ	(ggile)	യായിരിക്കും	(yayirikkum)	ചുമുണ്ടായിരുന്നു	(chumundayirunnu)
ങ്ങിലെ	(ngngile)	തായിരിക്കും	(thayirikkum)	ട്ടുമുണ്ടായിരുന്നു	(ttumundayirunnu)
ച്ചിലെ	(chchile)	ടായ ിര ിക്കും	(dayirikkum)	ണ്ണുമുണ്ടായിരുന്നു	(nnumundayirunnu)
ഞ്ഞിലെ	(njnjile)	ടാനായിരിക്കും		തുമുണ്ടായിരുന്നു	(thumundayirunnu)
തിലെ	(thile)	രുമായിരുന്നു	(rumayirunnu)	നുമുണ്ടായിരുന്നു	(numundayirunnu)
നാക്കി	(nakki)	കുമായിരുന്നു	(kumayirunnu)	റുമുണ്ടായിരുന്നു	(rumundayirunnu)
രാക്കി	(rakki)	യുമായിരുന്നു	(yumayirunnu)	ലുമുണ്ടായിരുന്നു	(lumundayirunnu)
യാക്കി	(yakki)	ങ്ങുമായിരുന്നു	(ngngumayirunnu)	യുമുണ്ടായിരുന്നു	(yumundayirunnu)
ലാക്കി	(lakki)	ടുമായിരുന്നു	(dumayirunnu)	ശുമുണ്ടായിരുന്നു	(shumundayirunnu)
താക്കി	(thakki)	തുമായിരുന്നു	(thumayirunnu)	ക്കുമുണ്ടായിരുന്നു	(kkumumdayirunnu)
സാക്കി	(sakki)	ളുമായിരുന്നു	(lumayirunnu)	ളുമുണ്ടായ ിരുന്ന	(lumundayirunna)
ളാക്കി	(lakki)	നുമായിരുന്നു	(numayirunnu)	രുമുണ്ടായ ിരുന്ന	(rumundayirunna)
റാക്കി	(rakki)	രുമായിരുന്നു	(rumayirunnu)	ശുമുള്ള	(shumulla)
നായും	(nayum)	ളുമുണ്ടായി	(lumundayi)	ളുമുള്ള	(lumumulla)
തായും	(thayum)	മുണ്ടായ ി	(mundayi)	രുമുള്ള	(rumulla)
ലായും	(layum)	ലുണ്ടായി	(lundayi)	ച്ചുമുള്ള	(chchumulla)
സായും	(sayum)	യ ിട്ടുണ്ടായ ി	(yittundayi)	ടുമുള്ള	(dumulla)
സ്സായും	(ssayum)	മാറുണ്ടായി	(marundayi)	ട്ടുമുള്ള	(ttumulla)

Figure 2.15: List of suffixes



Input word		lemma		Rule	
വീണതിന്	(veenathinu)	വീഴുക	(veezhuka)	ണതിന് --> ഴുക	(nathinu --> zhuka)
മറിഞ്ഞതിന്	(marinjathinu)	മറിയുക	(mariyuka)	ഞ്ഞതിന് --> യുക	(njathinu --> yuka)
തൊട്ടതിന്	(thottathinu)	തൊടുക	(thoduka)	ൊട്ടതിന് --> ൊടുക	(ottathinu --> oduka)
പെട്ടതിന്	(pettathinu)	പെടുക	(peduka)	െട്ടതിന് --> െടുക	(ettathinu --> eduka)
കട്ടതിന്	(kattathinu)	കക്കുക	(kakkuka)	ട്ടതിന് --> ക്കുക	(ttathinu --> kkuka)
ഞെട്ടിയതിന്	(njettiyathinu)	ഞെട്ടുക	(njettuka)	ട്ടിയതിന് --> ട്ടുക	(ttiyathinu --> ttuka)
പോയതിന്	(poyathinu)	പോവുക	(povuka)	യതിന് --> വുക	(yathinu --> vuka)
പൊട്ടിയതിന്	(pottiyathinu)	പൊട്ടുക	(pottuka)	ട്ടിയതിന് --> ട്ടുക	(ttiyathinu --> ttuka)
ചരിത്രത്തിന്	(charithraththinu)	ചരിത്രം	(charithram)	ത്തിന് --> ം	(ththinu --> um)
പോന്നതിന്	(ponnathinu)	പോരുക	(poruka)	ന്നതിന് --> രുക	(nnathinu --> ruka)
പോണതിന്	(ponathinu)	പോവുക	(povuka)	ോണതിന് --> ോവുക	(onathinu --> ovuka)

Figure 2.16: Lemma

TABLE 2.3: Distribution of words in Malayalam documents

Malayalam word categories	Average distribution of words
Words to be lemmatized	50% of N
Words ending with *'chandrakkala'*	30% of L
Words ending with *'aekaaram'*	20% of L
Words ending with *'akaaram'*	20% of L
Words ending with *'ikaaram'*	12% of L
Words ending with *'ukaaram'*	6% of L
Words ending with *'chillu'*	6% of L
Words ending with *'anunasika'*	6% of L

compared to the total number of words to be stemmed. Statistics of distribution of various category of words are shown in the Table 2.3. Here N is the total number of words in the document and L is the total number of words to be lemmatized.

2.8 Performance evaluation

Comparison with other Malayalam stemmers can be done by considering the measures like precision, recall and accuracy. Here true positives, false positives, true negatives and false negatives are defined as follows.

$$t_p = \text{no of words correctly lemmatized} \tag{2.1}$$

$$f_p = \text{no of words mistakenly lemmatized} \tag{2.2}$$

$$t_n = \text{no of words correctly not lemmatized} \tag{2.3}$$

$$f_n = \text{no of words mistakenly not lemmatized} \tag{2.4}$$

For comparison online Malayalam documents were taken as the data set. Altogether documents of varying lengths are considered. Here $t_p + t_n$ forms the correct result and $f_p + f_n$ forms the incorrect result. The results obtained can be tabulated as in Table 2.4. The documents are tested with dictionary based, tree based and Indic stemmer. The dictionary based and tree based gives the same result as the second is developed from the first. The difference lie in the execution time which is shown in Fig. 2.17. Testing was done on a system with memory - 1.9 Gb, Processor - Intel Core 2 Duo CPU T6570@2.10 Ghz x 2, OS type 32 bit Ubuntu 11.10. Accuracy, precision, recall and F1 score measures are

Table 2.4: Comparison of correct results with Indic stemmer

	Vriksh stemmer		Indic stemmer	
Word count in dataset	Correct words	%	Correct words	%
500	475	95	241	48
1000	821	82	563	56
2000	1706	85	782	39
2500	2025	81	1725	69
3000	2582	86	2040	68
5000	4102	82	3550	71
7500	6154	82	4651	63
10000	8608	86	4659	63
15000	14793	98	6301	68
20000	17609	88	12000	60

Table 2.5: Result analysis

	Vriksh stemmer				Indic stemmer			
Data set word count	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
500	0.95	0.96	0.98	0.97	0.48	0.68	0.61	0.64
1000	0.82	0.78	0.94	0.85	0.56	0.43	0.47	0.45
2000	0.85	0.90	0.77	0.83	0.39	0.48	0.26	0.34
2500	0.81	0.75	0.80	0.77	0.69	0.88	0.44	0.59
3000	0.86	0.88	0.79	0.83	0.68	0.80	0.42	0.55
5000	0.82	0.85	0.87	0.86	0.71	0.71	0.52	0.59
7500	0.82	0.93	0.88	0.90	0.63	0.87	0.58	0.70
10000	0.86	0.79	0.98	0.86	0.63	0.78	0.61	0.68
15000	0.98	0.98	0.98	0.98	0.68	0.77	0.61	0.68
20000	0.88	0.84	0.98	0.90	0.60	0.77	0.40	0.53

calculated and tabulated in Table 2.5. There are some words which gave incorrect results in the proposed method. The addition of rule for correcting those words may result in the incorrect result of other words. So such words can be considered as the exceptional cases and should be dealt separately. The only solution is to consider the entire word as the suffix and the expected lemma as the replacement. Some of the examples are *'doore' - 'door', 'mylilere' - 'myliler'*. Here the false positives and false negatives show the incorrect results which are given by both the methods.

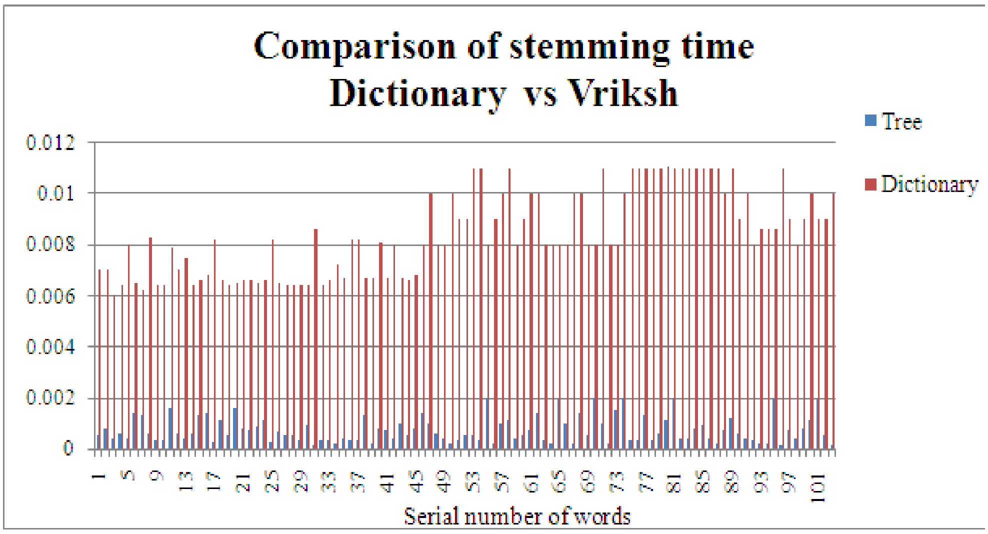


FIGURE 2.17: Comparison of execution time

2.9 Conclusion

There has been a lot of work in Indian languages for developing stemmers, taggers, translators etc. These types of applications help the people to communicate and work in their own mother tongue rather than depending on other languages. Nowadays, lot of software applications in native languages are available as mobile apps. There lies one importance of this work. The implementation of the tree based method makes the retrieval faster. Both the positive and negative aspects of the present method are discussed. There is still room for improvement and the accuracy can be increased by adding more and more rules. Moreover, the number of nodes can be reduced by making the method graph based. On an average our method has shown an accuracy of 87%. Since Indic stemmer is the only link available online, the performance could be directly compared with it alone.

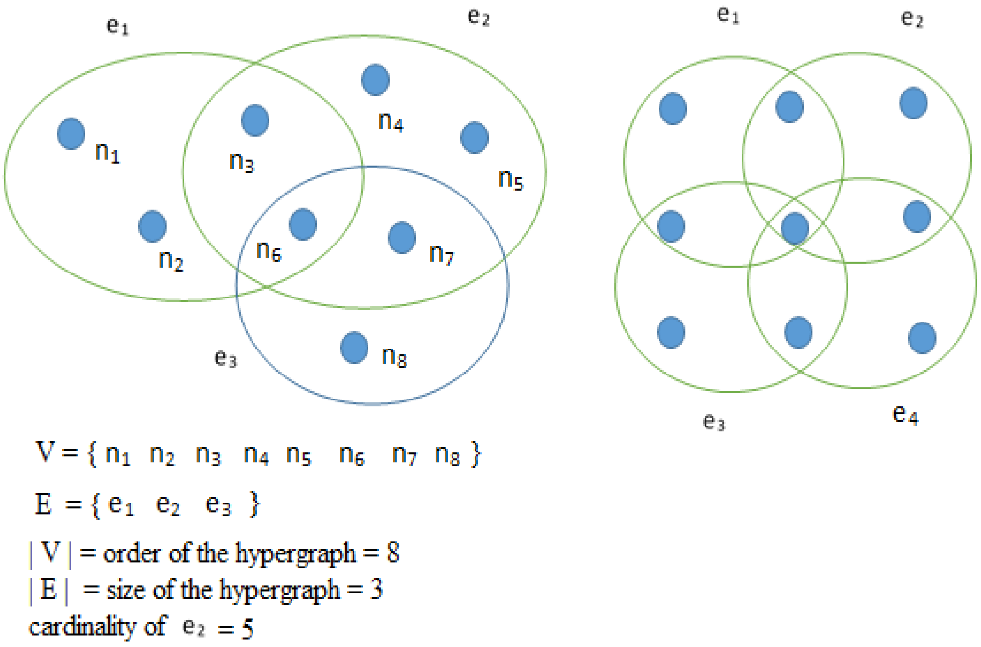
Chapter 3

Clustering

Documents have been represented using graphs for many applications like document clustering, document summarization etc. They have also been modeled using the vector space for various text processing activities. The purpose of this work is to model text using hypergraph and apply the morphological operators on hypergraph created from the underlying text to get text clusters. The document is considered as a graph and partitioning is applied which finally results in clusters. Here document is modeled as a hypergraph and two methods for text clustering are discussed. The first method uses simple hypergraph and the second method uses a weighted hypergraph. This work also discusses on how to model multiple documents as a single hypergraph. The method can be extended for multi-document clustering also.

3.1 Introduction

Hypergraph [88] $H = (V, E)$ as shown in Fig 3.1(a), is a graph in which every edge is associated with many nodes as opposed to exactly two nodes in the case of a normal graph. Because of this, edges in a hypergraph are called hyperedges. Here V is the set of nodes and E is the set of hyperedges. $|V|$ is the order of the hypergraph and $|E|$ is the size of the hypergraph. The number of nodes with which a hyperedge is associated is called the cardinality of the hyperedge. If the cardinality of all the hyperedges of a hypergraph is k, then it is called a $k-$uniform hypergraph. The same is shown in Fig. 3.1(b). The number of hyperedges with which a node is associated is called the degree of the node. If all


e_1
e_2
n_1
n_2
n_3
n_4
n_5
n_6
n_7
n_8
e_3
$V = \{ n_1 \ n_2 \ n_3 \ n_4 \ n_5 \ n_6 \ n_7 \ n_8 \}$
$E = \{ e_1 \ e_2 \ e_3 \}$
$| V |$ = order of the hypergraph = 8
$| E |$ = size of the hypergraph = 3
cardinality of e_2 = 5
e_1
e_2
e_3
e_4

(a) Hypergraph $H = (V, E)$

(b) k-uniform hypergraph where k = 4

FIGURE 3.1: Hypergraph

the nodes in a hypergraph are having the degree d, then such a hypergraph is called $d-$regular hypergraph. Document clustering has been widely used in many information retrieval systems [64]. Clustering helps in finding the nearest neighbour of a document. They are used in search engines in response to a user's query. They also help in creating a hierarchical cluster of documents. In a hypergraph modelling [65] for documents, the nodes are the documents and hyperedges are the authors. Based on the authors, the documents are being grouped. But in the present method, while converting to the area of text, a hyperedge is a sentence and nodes are the unique words in that sentence. The number of hyperedges in this graph will be the number of sentences considered for clustering. Just as a sentence can have many words in it, a hyperedge is having many nodes in it. While modeling multiple documents, a hyperedge is a document itself and the nodes in it are the unique words in that document. The number of hyperedges will be same as the number of documents considered for clustering. This is the pioneer work which uses the concept of hypergraph in text clustering as well as document clustering. Let us consider the input text given below:

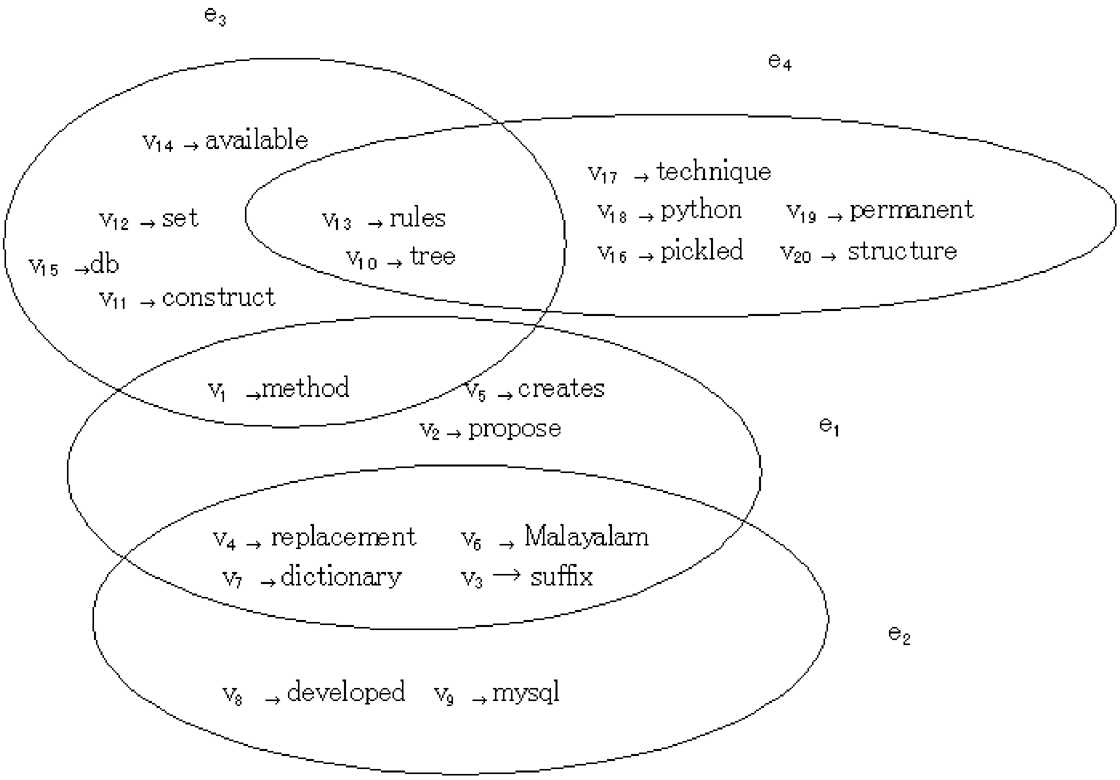


FIGURE 3.2: Text modeled as hypergraph

"The method proposed here uses a suffix replacement methodology where it creates a Malayalam dictionary of suffixes and replacements. The Malayalam suffix replacement dictionary is being developed using MySQL. In this method tree is being constructed from a set of rules available in the database. The tree of rules is being pickled using the pickling technique of python which makes tree a permanent structure."

Modeling of the above text as hypergraph is shown in Fig. 3.2. The main aim here is to model text using hypergraphs, partition the hypergraph and create text clusters. Outline of the remaining sections of this chapter is as follows: Section 3.2 presents a detailed literature review on the graphical methods, non graphical methods and parallel algorithms in text clustering. Section 3.3 creates a mathematical framework for the text hypergraph and the text weighted hypergraph. Section 3.4 introduces various mathematical operations on the hypergraph and the weighted hypergraph. Hypergraph morphology is discussed in section 3.5, the proposed methodology is discussed in section 3.6, its implementation, data set and the result analysis in section 3.7. Section 3.8 offers some idea on the future work.

3.2 Literature survey

This section gives an overview of various methods currently used in the field of text clustering. The text clustering algorithms are divided in to graphical, non-graphical methods and their extensions in parallelism. The graphical methods used are dependency graph based, document graph based, K-NN graph based, co-occurrence graph based, semantic graph based and graphical network of documents. The non graphical methods include K-means, K-medoid, density based, hierarchical methods etc.

- Non graphical methods

  In a method using association [66], the authors find correlation between the terms using internal correlation measures like association between the terms, normalized association between the terms, co-variance and pearson correlation coefficients. Finally these are applied in K-means algorithm to improve the performance. Word sense disambiguation is done using WordNet in [67] and the theme of the text is represented using lexical chains. Only core semantic terms are considered to reduce dimensionality of feature set. In another paper [68], the raw input text is subjected to many operations like wide one dimensional convolution, folding, dynamic K-max pooling etc. The main problem with many algorithms are that, they cluster together only if there are common terms. They won't cluster based on concepts. In order to solve this problem, along with document term matrix, document concept matrix [69] is also created with the help of Wikipedia and then clustering is done. A hierarchy of clusters [70] is created using frequent item sets. The method spans through several phases like tree construction, pruning, sibling merging etc. Sibling merging is done by finding inter cluster similarity. A hierarchical clustering method [71] is also discussed. Birch [72] is a bottom up method of clustering. When applied to the document, the clustering feature(CF) is created from the vector representation of the document and the CF tree is created by storing the features incrementally. The branching factor is decided and tree nodes are split accordingly. Density based methods like DBSCAN, CHAMELEON [73] are being discussed, which mainly looks for the density of the key words occurring in the short text. They accumulate neighbours in a dense region and form clusters.

- Graph oriented methods

  The importance of fidel vector in graph partitioning is well demonstrated and is used to decompose it in to non-overlapping neighbourhoods [74]. Uniform hypergraph partitioning is applied in geometric grouping [75]. A detailed study on eigen values, eigen vectors and eigen spaces [76] are also made. Text can be modelled using a bipartite graph [77] and it can be subjected to clustering. A detailed survey [78] on document representations, text classification, clustering and implementations is available. The method which represents sentences and documents as dependency graph [79] shows how words in it are related to each other. Every word in a sentence will be related to the sentence head. After construction of the graph several operations like merging, union etc. of edges are performed which would improve the result of clustering. Dependency graph for the entire text can be created. Similarity of the graphs are found out and K-means clustering is applied. A document graph is constructed using WordNet noun taxonomy [80] where graphs with similar subgraphs are clustered together. Always similar subgraphs reflect similar sense. They are mined using apriori algorithm. Firstly, the method finds 1-edge subgraphs, then 2-edge subgraphs, K-edge subgraphs and so on. Finally hierarchical clustering algorithm is applied to find the similar subgraphs. In another method, weighted K-NN graph [81] is constructed by assigning each node with K random neighbours with the help of similarity matrix created. In the reduce phase, top-K similar nodes are selected. Edges between the vertices are weighted, where the weight shows the similarity. Edge pruning is done where those below a threshold weight are deleted. In co-occurrence graph based method, the feature terms in the document will be modeled as vertices and the edges represents the co-occurrence of the terms [82] in the same sentence/ paragraph /window of size n. Graphs are subjected to various operations like graph union, subgraph calculation etc. Similar subgraphs are found out and clustering is done. A network of documents [83] are created with documents as vertices, edges represent similarity between the documents. Further cluster centroids are calculated with the centrality values where the K-means clustering is later applied.

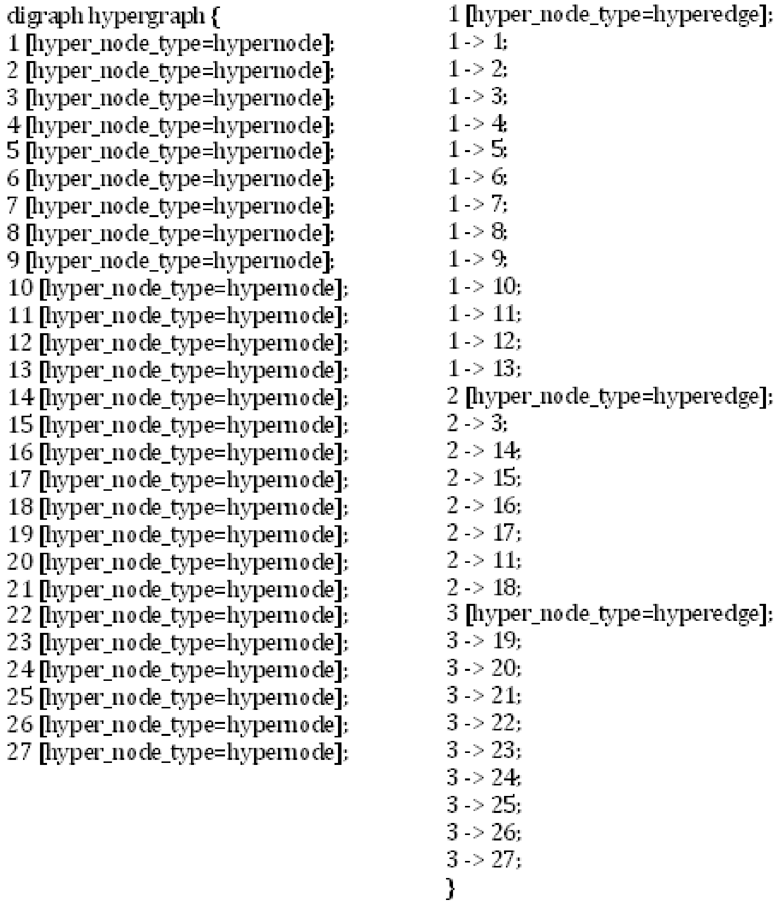
- Parallel algorithms

  A parallel k-means algorithm [84] has been proposed where neighbour matrix is created and it includes parallelism by including more processors in computing. The initial centroids are updated. The method finally shows how speed in clustering is improved by increasing the number of processors used. A parallel method with map-reduce [85] is implemented where it shows a good scalability and works well on large data sets. In hierarchical agglomerative clustering [86] made parallel, there is increase in the speed and quality of clustering. For parallel document processing [87], anchors, pivots, hierarchy of anchors and sorting are defined on documents.

3.3 Mathematical framework for text hypergraph

Here a text is modeled using a hypergraph and a weighted hypergraph. A sample hypergraph created is shown in the Fig. 3.3, where there are 27 nodes and 3 hyperedges. The second hyperedge is overlapping on the first hyperedge as the nodes 3 and 11 are repeated in second hyperedge also. The third hyperedge is not overlapping on the other hyperedges. While partitioning using the spectral partition method, we should get the first two edges in one partition and the third edge in the second partition. So let us describe a mathematical frame work for this hypergraph.

Let τ denote the text to be clustered. Let $(H_\tau, S, \xi, \upsilon)$ denote the hypergraph structure corresponding to the text τ where
$H_\tau \Rightarrow$ the hypergraph corresponding to the text τ,
$\xi \Rightarrow$ the edge set in H_τ which represents the sentences S in the text τ and
$\upsilon \Rightarrow$ the node set in the H_τ which represents the unique words in the text τ.
The Edge set ξ can be partitioned in to disjoint equivalence classes. Equivalence classes generate unique partitions of the given text τ.



```
digraph hypergraph {
1 [hyper_node_type=hypernode];
2 [hyper_node_type=hypernode];
3 [hyper_node_type=hypernode];
4 [hyper_node_type=hypernode];
5 [hyper_node_type=hypernode];
6 [hyper_node_type=hypernode];
7 [hyper_node_type=hypernode];
8 [hyper_node_type=hypernode];
9 [hyper_node_type=hypernode];
10 [hyper_node_type=hypernode];
11 [hyper_node_type=hypernode];
12 [hyper_node_type=hypernode];
13 [hyper_node_type=hypernode];
14 [hyper_node_type=hypernode];
15 [hyper_node_type=hypernode];
16 [hyper_node_type=hypernode];
17 [hyper_node_type=hypernode];
18 [hyper_node_type=hypernode];
19 [hyper_node_type=hypernode];
20 [hyper_node_type=hypernode];
21 [hyper_node_type=hypernode];
22 [hyper_node_type=hypernode];
23 [hyper_node_type=hypernode];
24 [hyper_node_type=hypernode];
25 [hyper_node_type=hypernode];
26 [hyper_node_type=hypernode];
27 [hyper_node_type=hypernode];
1 [hyper_node_type=hyperedge];
1 -> 1;
1 -> 2;
1 -> 3;
1 -> 4;
1 -> 5;
1 -> 6;
1 -> 7;
1 -> 8;
1 -> 9;
1 -> 10;
1 -> 11;
1 -> 12;
1 -> 13;
2 [hyper_node_type=hyperedge];
2 -> 3;
2 -> 14;
2 -> 15;
2 -> 16;
2 -> 17;
2 -> 11;
2 -> 18;
3 [hyper_node_type=hyperedge];
3 -> 19;
3 -> 20;
3 -> 21;
3 -> 22;
3 -> 23;
3 -> 24;
3 -> 25;
3 -> 26;
3 -> 27;
}
```

Figure 3.3: A sample hypergraph created

The same is shown in Fig. 3.3 where the hyperedge set = [[1], [2], [3]] and the node set consists of nodes [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] being spanned across three hyperedges. Let $(H_w\tau, S, W, \xi, \upsilon)$ denote a weighted hypergraph structure corresponding to the text τ where $H_w\tau \Rightarrow$ the weighted hypergraph corresponding to the text τ,
$\xi \Rightarrow$ the edge set in $H_w\tau$ which represents the sentences S in the text τ,
$\upsilon \Rightarrow$ the node set in $H_w\tau$ which represents the unique words in the text τ and
$W \Rightarrow$ the term frequency of the words in the document. The Edge set ξ can be partitioned in to disjoint equivalence classes. These equivalence classes generate unique partitions of the given text τ.

Lemma 3.1

The intersections of the partitions P_i of the hypergraph H_τ gives an empty set. i.e., $\forall i \quad \bigcap P_i = \phi$

Proof

Let τ denote the text under consideration containing different topics which are to be partitioned. Let H_τ be the hypergraph created from the text with vertex

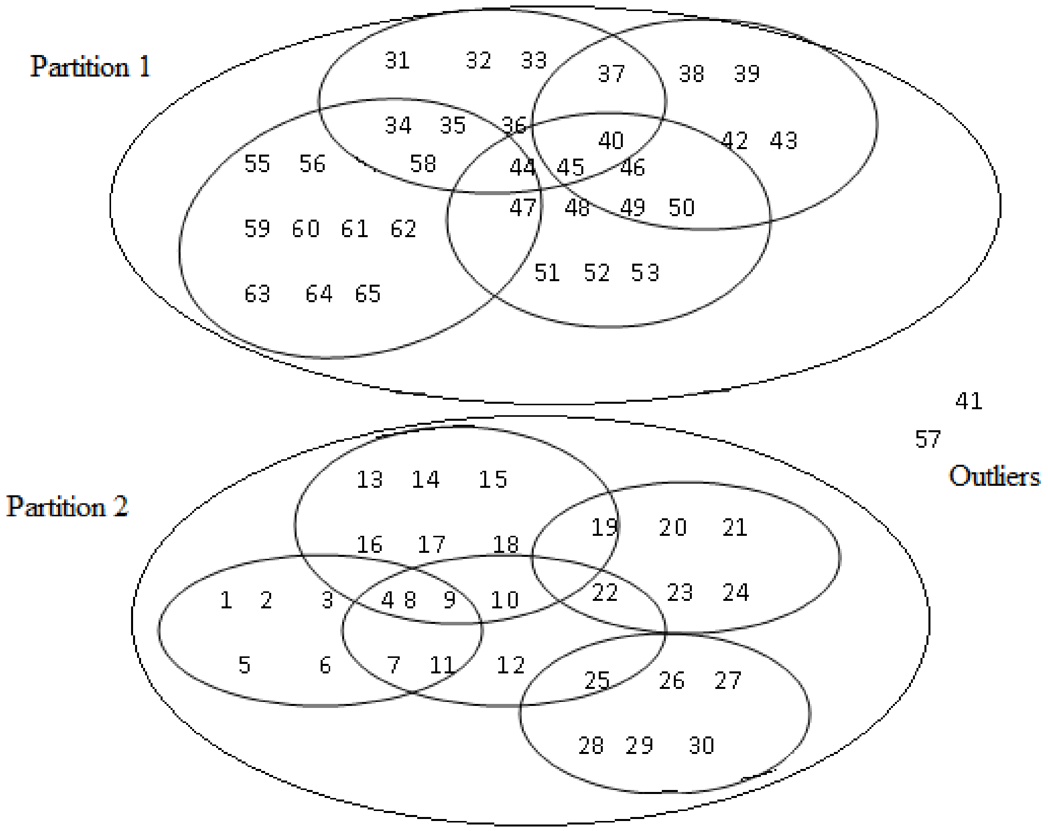


Figure 3.4: Partitions in hypergraph

set υ which represents the set of words and edge set ξ which represents the set of sentences. Let $P_{i;1\leq i\leq n}$ be the partition created corresponding to the edge set $\xi_{i;1\leq i\leq n}$. With respect to text τ, an edge set ξ_i is taken as the set of all sentences related to a particular topic $t_{p_i;1\leq i\leq n}$.

Therefore

$$\xi_i \cap \xi_j = \phi \tag{3.1}$$

So also the corresponding partition P_i represents the category of text related to a particular topic t_{p_i}. Hence P_i contains t_{p_i} which is distinct from P_j which contains t_{p_j}. i.e., P_i and P_j does not have any common sentences. Since each P_i is disjoint, it implies

$$P_i \cap P_j = \phi \tag{3.2}$$

Therefore

$$\forall_{i;1\leq i\leq n} \cap P_i = \phi \tag{3.3}$$

E.q(3.3) is implied from E.q(3.1) and E.q(3.2).

Illustration

Consider the hypergraph H_τ in the Fig. 3.4. Here two partitions P_1 and P_2 are created. These two partitions are not having a common edge. So When we take the intersection of both we get an empty set ϕ.

Lemma 3.2

The union of all the partitions P_i of the hypergraph H_τ gives the original hypergraph H_τ.

i.e., $\forall i \quad \bigcup P_i = H_\tau$ (Equality holds only when outliers are also considered)

Proof

In lemma 3.1, different edge sets $\xi_{i;1\leq i\leq n}$ of H_τ represent different text categories belonging to t_{p_i}. While partitioning, some outlier sentences are also produced which do not belong to any of these edge sets. The original hypergraph H_τ is obtained by combining all the edge sets i.e.,

$$\forall_{i_1\leq i\leq n} \cup \xi_i = H_\tau \tag{3.4}$$

As mentioned in lemma 3.1, since an edge set ξ_i in the hypergraph H_τ represents a partition P_i of text τ it implies

$$\forall_{i_1\leq i\leq n} \cup P_i = H_\tau \tag{3.5}$$

Illustration

According to Fig. 3.4, while partitioning, two partitions P_1 and P_2 are created. Some nodes(nodes 41 and 57) are seen outside the partitions which do not belong to any of the edge sets. These nodes are called outliers. The union of these outliers and the partitions P_1 and P_2 gives the original hypergraph H_τ.

Theorem 3.1

Edge set ξ in the hypergraph H_τ corresponding to the text τ generates unique partitions P_i of the text τ.

Proof

The proof of theorem follows from proof of i), ii) and iii).

i) $t_{p_i} \in P_i; \forall_{1\leq i\leq n}$

Let t_{p_i} and t_{p_j} be two different topics in text τ. Then $t_{p_i} \in P_i$ and $t_{p_j} \in P_j$ for $i \neq j$.

ii) By lemma 3.1, either $P_i = P_j$ or $P_i \cap P_j = \phi, i \neq j$

iii)By lemma 3.2, $\cup P_i = H_\tau; \forall_{1 \leq i \leq n}$
An edge in H_τ represents a sentence in τ. So edge set of the hypergraph H_τ represents sentences in the text τ. When spectral partitioning is applied to H_τ, edge set is getting divided in to two subsets. These two sets represent the first level clustering of the text τ. When this is iterated, it ultimately leads to partitions(clusters) which are unique and the intersection of these partitions will be a null set as said in lemma 3.1.

3.4 Mathematical operations

The hypergraph H_τ and the weighted hypergraph $H_w\tau$ created for the text τ undergo various mathematical operations.

- Adjacency matrix of H_τ

  The adjacency matrix A of the hypergraph H_τ is the square matrix of the nodes υ in the hypergraph H_τ. Here we can see that $A_{ij} = 1$ and $A_{ji} = 1$, if the node υ_i and node υ_j are part of the same hyperedge ξ_k. In turn it means both the nodes υ_i and υ_j are words co-occurring in many sentences. A row in the matrix A shows the neighbouring words of a particular word, taking in to consideration all the sentences in the text τ. Referring to Fig. 3.3, we can see words 1 to 13 and 14 to 18 are neighbours of word 3.

- Diagonal matrix of H_τ

  Diagonal matrix D_v of the hypergraph H_τ is a square matrix of node degree, where the diagonal entries D_{vii} = Number of hyperedges in which that node υ_i is present. It actually shows the number of sentences in which a word occurs. While term frequency shows the total count of a word in a text, this shows the number of sentences in which that word occurs. The term frequency can be higher than this since it also counts the word repetitions in a single sentence.

- Laplacian matrix L of H_τ Laplacian matrix L of the hypergraph H_τ is a square matrix, and can be written as

$$L = D_v - A \tag{3.6}$$

- Diagonal matrix of edge degree

  Diagonal matrix of edge degree D_e of hypergraph H_τ is a square matrix, where the entries are the degrees of the hyperedges. The degree of a hyperedge ξ_i is equal to the sum of the degrees of all the nodes υ_i in that hyperedge. This is the sum of number of sentences to which each word in a sentence belongs. Suppose a sentence has n words. Let c_{wi} be the count of sentences of the text τ in which that word occurs. The edge degree is equal to the sum of c_{wi} of all words in that sentence S.

- Matrix H of weighted hypergraph $H_w\tau$

  The matrix H is the one where the rows represent the nodes and the columns represent the hyperedges. An entry $H(\upsilon_i, \xi_j) = 1$ iff υ_i is a part of the edge ξ_j. In turn it means that a word w_i is a part of the sentence S_j.

- Weight matrix W of weighted hypergraph $H_w\tau$

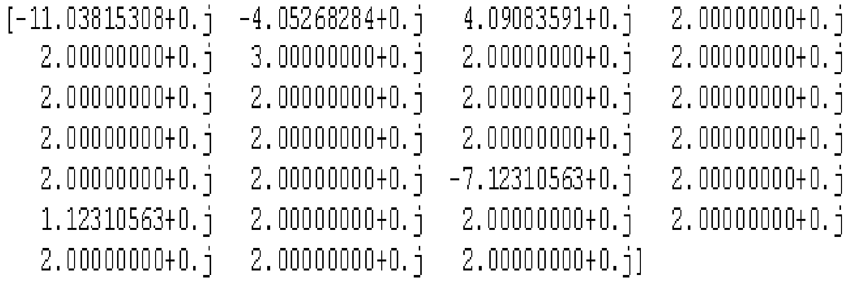
  The weight matrix W of a hypergraph $H_w\tau$ is a diagonal matrix of weights w of hyperedges ξ_i. The weight w_i of a hyperedge ξ_i is equal to the sum of the weights of all nodes υ_j in that hyperedge ξ_i. With respect to the text τ and the hypergraph H_τ, the weight of a node is the term frequency of the word in the entire document.

- Adjacency matrix of weighted hypergraph $H_w\tau$

  Adjacency matrix

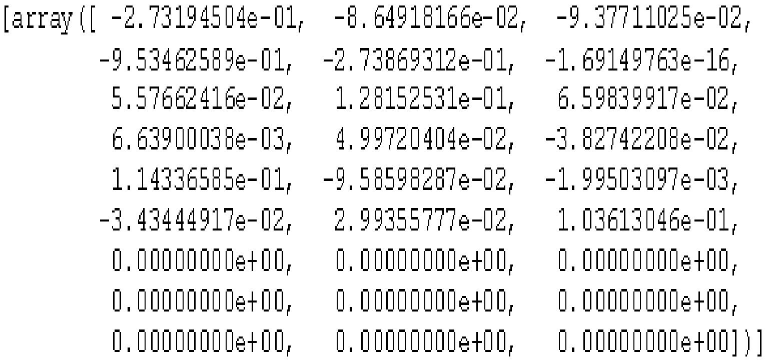
$$A = H * W * H^T - D_v \tag{3.7}$$

where D_v is the diagonal matrix of node degree of weighted hypergraph.



```
[-11.03815308+0.j  -4.05268284+0.j   4.09083591+0.j   2.00000000+0.j
   2.00000000+0.j   3.00000000+0.j   2.00000000+0.j   2.00000000+0.j
   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j
   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j
   2.00000000+0.j   2.00000000+0.j  -7.12310563+0.j   2.00000000+0.j
   1.12310563+0.j   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j
   2.00000000+0.j   2.00000000+0.j   2.00000000+0.j]
```

Figure 3.5: Eigen values of L



```
[array([ -2.73194504e-01,  -8.64918166e-02,  -9.37711025e-02,
        -9.53462589e-01,  -2.73869312e-01,  -1.69149763e-16,
         5.57662416e-02,   1.28152531e-01,   6.59839917e-02,
         6.63900038e-03,   4.99720404e-02,  -3.82742208e-02,
         1.14336585e-01,  -9.58598287e-02,  -1.99503097e-03,
        -3.43444917e-02,   2.99355777e-02,   1.03613046e-01,
         0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
         0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
         0.00000000e+00,   0.00000000e+00,   0.00000000e+00])]
```

Figure 3.6: Eigen vector of the selected eigen value

- Laplacian matrix L_w of weighted hypergraph $H_w\tau$

  The laplacian matrix L_w of weighted hypergraph $H_w\tau$ can be calculated as

$$L_w = D_v - A \tag{3.8}$$

3.5 Hypergraph morphology - hypergraph contraction

Since spectral partitioning is applied to the fidel vector of the laplacian matrix L of the hypergraph H_τ of the text τ in Method-I and weighted hypergraph $H_w\tau$ in Method-II, the vector is being divided in to two. Correspondingly two sets of text τ_i and τ_j are created for the text τ. Two hypergraphs H_τ^+ and H_τ^- are again created with the new set of nodes formed as part of the partitioning. Thus the initial hypergraph is being contracted in each iterative phase.

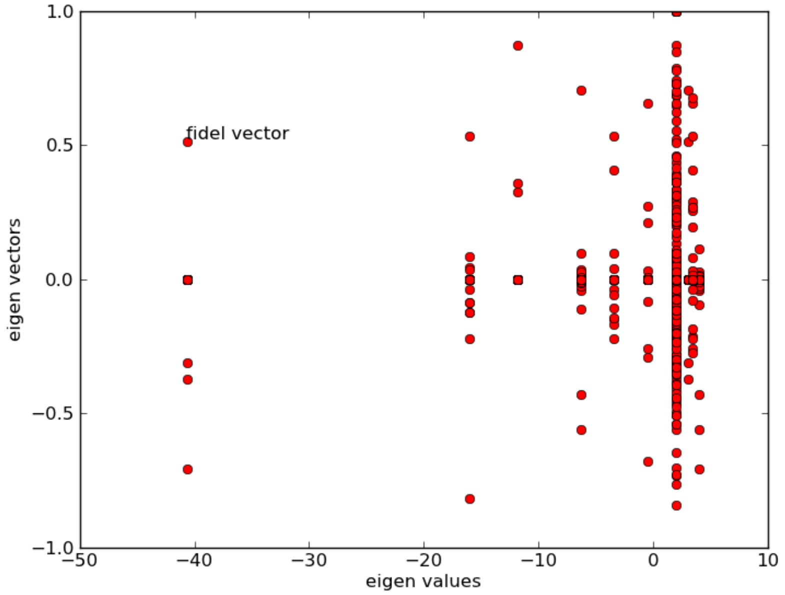


FIGURE 3.7: Iteration 1

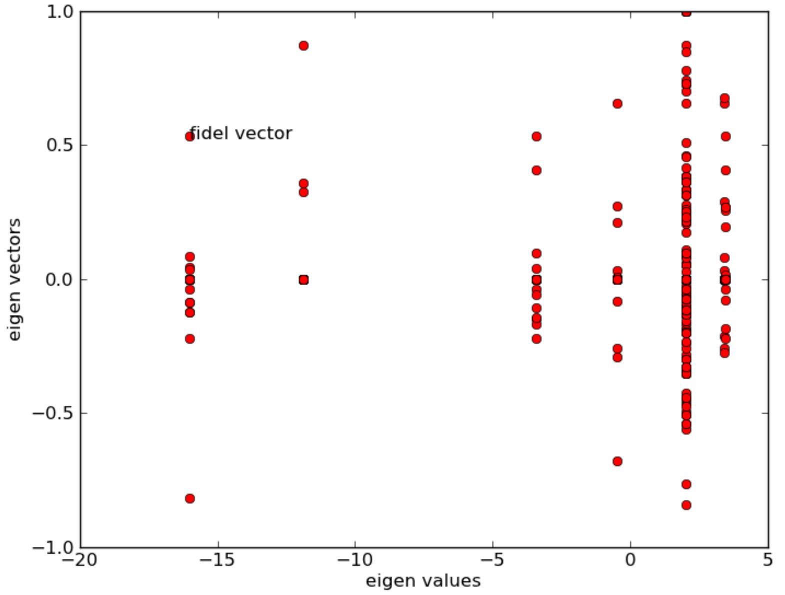


FIGURE 3.8: Iteration 2

```
First split

[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13], [3, 14, 15, 16, 17, 11,18]]

Second split

[[19, 20, 21, 22, 23, 24, 25, 26, 27]]
```

FIGURE 3.9: Negative and positive splits

3.6 Proposed method for text clustering

As mentioned above, this work has implemented two methods for text clustering. In this method, text clustering refers to clustering text(sentences) into groups, while in document clustering multiple documents are involved. The eigen vector corresponding to the maximum absolute eigen value is selected as the fidel vector. The same is shown in Fig. 3.5 and Fig. 3.6. The first method in text clustering uses a non-weighted hypergraph and spectral partitioning is applied to it based on the sign of fidel vector. The method has many iterations as seen in Fig. 3.7 and Fig. 3.8, until there are no change in signs. Fig. 3.9 shows the positive and negative split made in the selected vector. The figures shown in this chapter do not pertain to a single test case, but are outputs of various test cases. The second method uses a weighted hypergraph where the weight of the hyperedge of the graph is sum of the weights of the nodes in that edge. In turn the weight of the node will be equal to the term frequency of the word in all the sentences to be clustered. The detailed algorithm can be seen in section 3.6.1.

3.6.1 Algorithm: Method I

Algorithm 4: Clustering text τ by spectral partitioning hypergraph H_τ

```
Data: Hypergraph
Result: Partitions
repeat
    create a hypergraph $H_\tau$ of the text $\tau$;
    create adjacency matrix $A$ of the hypergraph $H_\tau$;
    find $D_v$ of the hypergraph $H_\tau$;
    find laplacian matrix $L$ of $H_\tau$;
    find the eigen values of $L$;
    search for the eigen value $\lambda_i$ which has a maximum absolute value;
    select fidel vector corresponding to this eigen value $\lambda_i$;
    partition the vector based on the +/- values;
    divide the sentences such that all those with negative value fall in one group
      and those with positive value fall in the second group;
until there is no change in sign or there are < two elements in a group;
```

3.6.2 Algorithm: Method II

Algorithm 5: Clustering text τ by spectral partitioning weighted hypergraph $H_w\tau$

```
Data: Weighted hypergraph
Result: Partitions
repeat
    create a weighted hypergraph $H_w\tau$ of the text $\tau$;
    calculate weight matrix $W$ of $H_w\tau$;
    calculate matrix $H$ of $H_w\tau$;
    find the matrix $L_w$ of $H_w\tau$;
    find the eigen values $\lambda_i$ of $L_w$;
    select the eigen value with maximum absolute value;
    find the fidel vector;
    partition the elements in the fidel vector to + and - ;
    divide the sentences such that all those with negative value fall in one group
      and those with positive value fall in the second group;
until there is no change in sign or $<$ two elements in a group;
```

3.7 Implementation

The implementation of text clustering using hypergraph is done in python. The input document contains Malayalam/English text. Malayalam is a regional language spoken in Kerala state and Lakshwadeep islands in India. A Malayalam lemmatizer is developed using a tree based method which reduces the words to its root lemma form, so that the term frequency can be calculated. Porter stemmer is used for stemming English text. The stemming also help in identifying the connection between the sentences for the graph creation and the hypergraph creation.

3.7.1 Data set

The data set consists of Malayalam articles taken from the Malayalam news sites and English articles taken from English news sites. These documents after preprocessing like punctuation removal, white space removal and stop word

Table 3.1: Data set statistics

Categories	word-count	stemmed word-count	stop word-count
sports, medicine, film,	125	40	10
sports travel, politics,	500	160	52
film medicine	2000	700	160
travel film politics	5000	1900	320
cricket football tennis films	10000	4050	990
medicine travel football	20000	6400	1400

removal are subjected to stemming. The text containing lemmatized words are used for creating graph and hypergraph. Both the graphs are subjected to spectral partitioning and results are compared. The clustering applied here help in grouping together news articles related to particular topic. The data set consists of news related to various topics like politics, travel, sports, medical news, film news etc. The work has successfully clustered the articles in to groups which finally resulted in an automatic topic identification. An overview of the Malayalam data set is shown in Table 3.1.

3.7.2 Performance comparison

Hypergraph modeling of a text is compared with graph modeling of a text which has been followed so far in many languages. A document is modeled as a graph as in Fig. 3.10, by considering the sentences as vertices and an edge existing between two vertices if there is a common word in both sentences. The connectivity of the

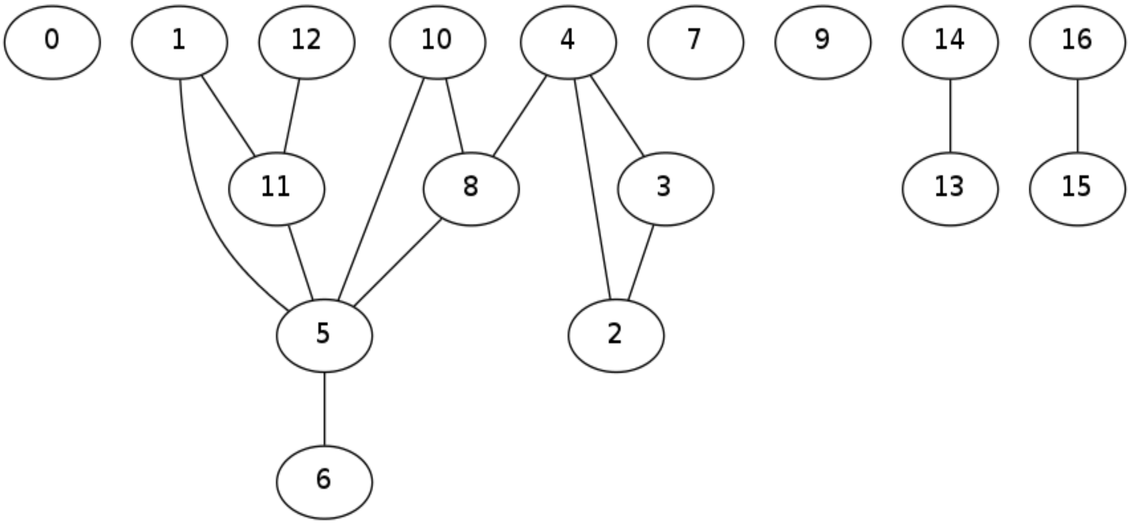


FIGURE 3.10: Text modeled as graph

graph shows which all sentences are being connected and the disconnectivity of the graph shows isolated sentences. This type of modeling has many disadvantages as the graph which is created does not show, how the document is really organized. While the graph method does not convey the idea about which all words make the sentences connected, the hypergraph shown in Fig. 3.2, gives a clear idea about the actual organization of the document, sentences and the words in it. From Fig. 3.2, the distribution of the words in the document is evident. The performance of text clustering using graph and hypergraph is done based on the parameters like speed, accuracy and complexity of the methods. Among the two methods discussed above, the weighted method shows more accuracy than the non weighted method, so that it is used for comparison with simple graph method. The weight of the edge will be the sum of the weights of the vertices in that edge. In the following sections, the two new methods like hypergraph and weighted hypergraph are compared with contemporary methods like graph and weighted graph.

- **Iteration**s

  The hypergraph method takes more iterations till it identifies all the outliers in the data set correctly. Outliers are the text which do not belong to any of the clusters. Outlier detection in the graph method is less. Since the graph modeling using hypergraph is more meaningful with respect to the organization of the document, outliers can be eliminated more correctly. But since the document modeling using the graph convey less knowledge about the document, the outlier detection and elimination is affected and not according to the expectations of the reader. The anonymous sentences eliminated from the hypergraph method always satisfy the reader.

- **Space complexity**

  The number of edges in the hypergraph representing the text are equal to the number of sentences considered. If there are 100 sentences to be clustered, whether they are connected or not, the number of edges will be fixed and is equal to 100. But for the graph of such a text, considering connections among all the vertices, the number of edges will be 99 + 98 +......+ 1, which is very large. Generally we can say that for text with n sentences, the number of edges in a hypergraph is n, while the number of edges in a graph is $n-1+n-2.....+1$ which can be written as $(n-1)n/2$ and is of order $O(n^2)$. The number of nodes in the hypergraph method is equal to the number of unique terms considered for clustering, while the number of vertices in the graph method is equal to the number of sentences considered for clustering. The number of unique terms will be less as we remove the stop words from the text in the preprocessing phase. Moreover, there will be many repeating words in the text. Word repetition can happen in a single sentence and also across sentences. So also the space complexity of the hypergraph is less when compared to the graph.

- Time complexity

  Time complexity of both graph and hypergraph methods are evaluated based on the above two parameters like number of iterations and the number of edges and vertices that need to be constructed in each iteration of the algorithm. Even though there is a slight increase in the number of iterations in the hypergraph method, it is compensated by the reduction in the number of edges and nodes that need to be constructed in every iteration of the algorithm. Even though the number of iterations in the graph method is less, it takes more time because of the increased number of edges and vertices that need to be constructed in each iteration of the algorithm. The time comparison of both graph and hypergraph are shown in the Fig. 3.11. Both the plots show a peak value initially in the time of graph creation, because the algorithm creates the original graph of the

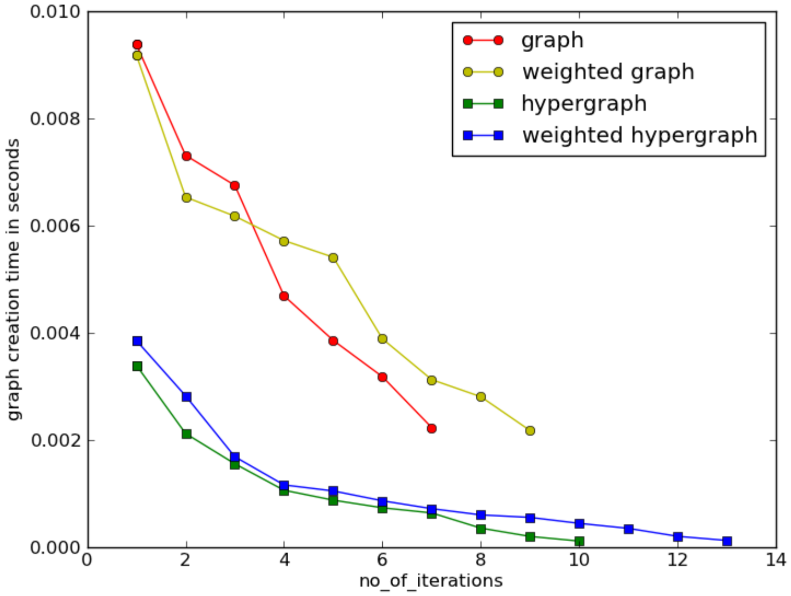


FIGURE 3.11: Execution time for graph creation

document in the first step. In each iteration, due to spectral partitioning, the size of the graph to be created decreases. That is why, the execution time for graph creation is minimum in the final iteration. Considering all the iterations, the total time taken for graph creation is 0.0663611889 seconds in the case of graph based method, while the total time taken for hypergraph creation is only 0.0194737911 seconds. This time is for the second smallest data set mentioned in Table 3.1. Since this algorithm iterates over the edges and the nodes in it, the time complexity is $O(n^2)$.

- Accuracy and entropy

The hypergraph method gives more accurate results in the case of cluster formation and in the case of outlier detection. The clusters which end up with single sentences are considered ouliers and others are actual clusters which speak about a particular topic. The accuracy is being tested manually with native Malayalam news readers. The clusters returned by the algorithm is compared with the clusters returned manually by the native people. The number of clusters returned by the weighted hypergraph method shows an accuracy of the 98% and the clusters returned by the graph method shows an accuracy near to 80%. The precision, recall, F-measure of the clustering is calculated based on the true positives, true negatives, false positives and false negatives. True positive is defined as the number of clusters correctly assigned by the method. True negatives are defined as the number of outliers correctly identified by the system, false positives are the number of outliers that the system marked as clusters and the false negatives are the number of clusters, the system marked as outliers. Once these measures are obtained, the following are calculated.

$$Precision = t_p/(t_p + f_p) \tag{3.9}$$

$$Recall = t_p/(t_p + f_n) \tag{3.10}$$

$$Accuracy = t_p + t_n/Total \tag{3.11}$$

and

$$F - measure = 2 * Precision * Recall/(Precision + Recall) \tag{3.12}$$

The results can be tabulated as in Table 3.2. The results are generated for different data sets of varying sizes. The recall is always 1.0 since the false negatives generated by the system is always nil. i.e., the number of clusters, the system identifies as outliers is nil. The main contributor of the good value of recall, is the efficient outlier detection by the proposed system. Similarly, result analysis is made for the spectral partitioning by the weighted hypergraph. The results obtained are consolidated in Table 3.3. Entropy is a metric that is a measure of the amount of disorder in a vector. Among the various versions of entropy, the one which is selected here is Shannon's entropy. Fig. 3.12 has four entropy plots representing graph, weighted graph, hypergraph and weighted hypergraph. Among the four methods weighted hypergraph has the lowest entropy. The graph shows a decrease in the disorder as the number of clusters increase.

3.8 Multi-document clustering

The two methods discussed above can be used for multi-document clustering. The hypergraph is constructed by modeling documents as hyperedges and unique words as nodes. The above mentioned two methods of spectral partitioning is applied where the nodes are partitioned based on the sign of the elements in the fidel vector.

Table 3.2: Result analysis of spectral partition of H_τ

Data set	t_p	t_n	f_p	f_n	Precision	Recall	Accuracy	F-measure
125	9	0	1	0	0.90	1.0	0.90	0.95
500	9	3	1	0	0.90	1.0	0.92	0.95
2000	10	2	0	0	1.0	1.0	1.0	1.0
5000	20	5	1	0	0.95	1.0	0.96	0.99
10000	50	10	2	0	0.96	1.0	0.97	0.98
20000	100	5	2	0	0.98	0.95	0.98	0.96

Table 3.3: Result analysis of spectral partition of $H_w\tau$

Data set	t_p	t_n	f_p	f_n	Precision	Recall	Accuracy	F-measure
125	10	0	0	0	1.0	1.0	1.0	1.0
500	9	3	0	0	1.0	1.0	1.0	1.0
2000	10	2	0	0	1.0	1.0	1.0	1.0
5000	20	5	1	0	0.95	1.0	0.96	0.98
10000	50	10	1	0	0.98	1.0	0.98	0.99
20000	100	5	1	0	0.99	1.0	0.99	0.99

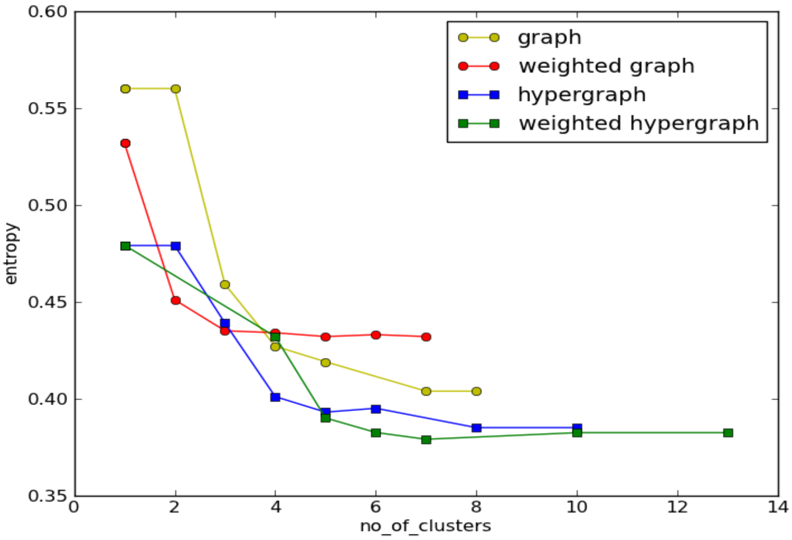


Figure 3.12: Entropy comparison

All nodes(words) which are associated with negative values form one cluster and those with positive values fall in the second cluster. All documents(edges) with those nodes with negative values fall in first cluster and those with positive values fall in the second one. This is iteratively done until there is no change in sign or the cluster has only one document in it.

3.9 Conclusion

In this chapter, we have presented a novel method of modeling text using hypergraphs and weighted hypergraphs. The sentences in the text forms the edge set and the words in the text forms the node set. Once the hypergraph is constructed various mathematical operations like finding the adjacency matrix, diagonal matrix of node degree, diagonal matrix of edge degree, laplacian matrix, matrix H, weight matrix are calculated. On applying spectral partitioning to the laplacian matrix, the edges are divided in to partitions where by, it results in a hypergraph morphology named hypergraph contraction. Hypergraph contraction leads to the formation of text clusters. Both hypergraph method and weighted hypergraph method are being compared with existing graph and weighted graph method. Hypergraph methods saves time in graph creation and clustering by 29%. Hypergraph method also saves space since only less space required in each iteration. Moreover the accuracy of the text clustering is more in the case of hypergraphs. While hypergraph method shows an accuracy of 95.5%, weighted hypergraph partitioning shows an accuracy of 98%.

Chapter 4

Intuitionistic fuzzy hypergraph modeling

Intuitionistic fuzzy hypergraphs (IFHG) are hypergraphs in which a second degree (non-membership) is also included with membership degree for every node in it. Likewise, every hyperedge is also having a membership and non-membership degree. If a system is modeled using IFHG, the membership degree actually shows the wantedness of the hyperedge/node with respect to the application and the non-membership degree shows the unwantedness of the node/hyperedge. In order to create a summary we need to model each text cluster as an IFHG. In this chapter we show the various morphological operations that can be applied on an IFHG and their results.

4.1 Recent works in hypergraphs/IFHG

Lattice structures on hypergraphs [88] has shown many properties like partial ordering, infimum, supremum, isomorphism etc. The authors have introduced complete lattice, dualities, discrete probability distribution on vertices and hypergraph similarity based on dilation. Mathematical morphology of hypergraphs are also used for classification or matching problems [89] on data represented by hypergraphs. As an example, the authors have applied it on a 2-D image and they proposed further applications of hypergraph in image analysis. New similarity measures and pseudo-metrics on lattices of hypergraphs [90] are detailed, which are incorporated in existing system for hypergraph-based feature selection, indexing, retrieval and matching. Morphological dilation, erosion, opening, closing, filtering on graphs [91] are illustrated with image processing, where the authors apply it in binary and grey scale image denoising and the method has outperformed many existing methods. Images are represented using set union [92] of hyperedges and are subjected to contra harmonic mean filter for salt and pepper noise removal. The method give better results in terms of visual quality, peak signal to noise ratio and mean absolute error.

Intuitionistic fuzzy sets and its operators [93] are applied to electoral example. Using modal operators and its extended version, the authors have represented people's changing opinion on voting day and afterwards. Fuzzy traversals of fuzzy hypergraphs, coloring of fuzzy hypergraphs and strongly interconnected hypergraphs [94] are also detailed. With suitable illustrations the concept of Intuitionistic Fuzzy Hypergraphs (IFHG) [95] and Dual IFHG are illustrated, where the authors also explain (α, β) cut on hypergraph, strength of an edge, incidence matrix etc. Also, the authors propose to use this concept in clustering problem. Operations like complement, join, union, intersection, ringsum, cartesian product and composition are defined for intuitionistic fuzzy hypergraphs [96], where the authors further propose to apply these operations in clustering techniques. Isomorphism between two IFHG and the cartesian product of two IFS [97] over the same universe are found out, where they also illustrate in-degree, out-degree of vertex v, weak isomorphism and co-weak isomorphism. A hyper-network [98] is created with processors as vertices and connections between the processors modeled as hyperedge. Radio coverage networks in a geographic region is modeled with radio receivers as vertices, where the membership values signify the quality of reception of a station/radio. The authors also propose further research in intuitionistic fuzzy soft hypergraphs and rough hypergraphs. An application with Intuitionistic Fuzzy sets for career choice [99] which is a decision making system is developed where the system represents the performance of students using membership μ, non-membership ν and hesitation margin π. The authors apply normalized Euclidean distance to determine the apt career choice. Operations on transversals of IFDHG [100], their union, intersection, addition, structural subtraction, multiplication and complement are defined and discussed. The authors also propose to work on application in coloring of IFHG. Generalized strong IFHG (GSIFHG), spanning IFHG and generalized strong spanning IFHG [101] are discussed, which can be used to analyze the structure of a system and to represent a partition, covering and clustering. Morphological dilation [102] is applied to different operations like union, intersection and complement of subgraphs of IFHG, which find role in text processing and image processing. The authors have also proved De Morgan's law applied to IFHG.

Applications in propositional logic, related databases with visits of directed hypergraphs and optimal paths are studied in detail[103]. It has many concepts like connectivity, path and cuts of hypergraphs. Many mathematical operators like dilation, erosion, dual adjunctions are defined on hypergraphs [104], [105] which are applied to image filtering by modeling image as hypergraphs.

4.2 Need for an intuitionistic fuzzy hypergraph

Let us consider a cricket team with altogether 15 players. Let there be n such teams. We can model each team as a hyperedge and the players in the team as nodes. A player can be part of more than one team. So the entire set of teams can be modeled as an IFHG, where each team member can be given a membership and non-membership degree. Assume that not all the players are good performers. Good performers can be given a higher membership degree (chance of selection to play) and low non-membership degree (chance of non selection to play). Like wise low performers can be given high non-membership degree (chance of non selection to play) and low membership degree (chance of selection to play). Always those with membership degree > 0.5 will be permanent members of the team. Consider a scenario where 11 out of 15 players are having membership degree > 0.5 and are permanent members. Rest four members are having high non-membership degrees since they are not so good players and they are considered as substitute players. Once a player is to be replaced, the one with lowest non-membership degree from the substitute players is selected for replacement. Membership degree of a team member should not be zero. Once it is zero he is not part of team, so also that node is not part of the IFHG. Likewise the non-membership degree should not be one. Here the membership and non-membership degree of the team is calculated from the membership and non-membership degrees of the team members. If all the team members are good players (high membership degree), then membership degree of the team (chance of selection to the tournament) is also high. If there is at least one member with very high non-membership degree, it may affect the teams performance and increase the non-membership degree of the team (chance of rejection from the tournament).

4.3 Preliminaries of IFHG

Let $[H_{IF}, (\mu_n, \gamma_n), (\mu_e, \gamma_e), H^n, H^e]$ be a finite intuitionistic fuzzy hypergraph with membership degree μ_n and non membership degree γ_n defined on the set of nodes H^n and membership degree μ_e and non-membership degree γ_e defined on a set of hyperedges H^e of H_{IF}. Depending on the membership degree μ_n, the node can be treated as high priority, medium priority and low priority. The non membership degree $\gamma_n <= 1 - \mu_n$. Similarly depending on the membership degree μ_e, the hyperedge can be treated as high priority, medium priority and low priority. The sum of the membership degree and non-membership degree of the node is less than or equal to 1 [95]. i.e., $\mu_n + \gamma_n <= 1$. So also the sum of the membership degree and non-membership degree of the hyperedge is less than or equal to 1 [95]. i.e., $\mu_e + \gamma_e <= 1$. If all the nodes in a hyperedge has $\mu_n > 0.5$, then μ_e is the supremum of all μ_n in that edge. In such a case $\gamma_n <= 1 - \mu_n$. If there is at least one node with $\gamma_n > 0.5$, then the γ_e of that edge is the supremum of all γ_n in that edge. In such a case $\mu_e <= 1 - \gamma_e$.

A sample IFHG and the priorities assigned are shown in Fig. 4.1.

Node priorities are assigned as follows:

1. Low priority node : $\mu_n < 0.5$.
2. Medium priority node :$\mu_n = 0.5$.
3. High priority node : $\mu_n > 0.5$.
4. Low priority edge : $\mu_e < 0.5$.
5. Medium priority edge : $\mu_e = 0.5$.
6. High priority edge : $\mu_e > 0.5$.

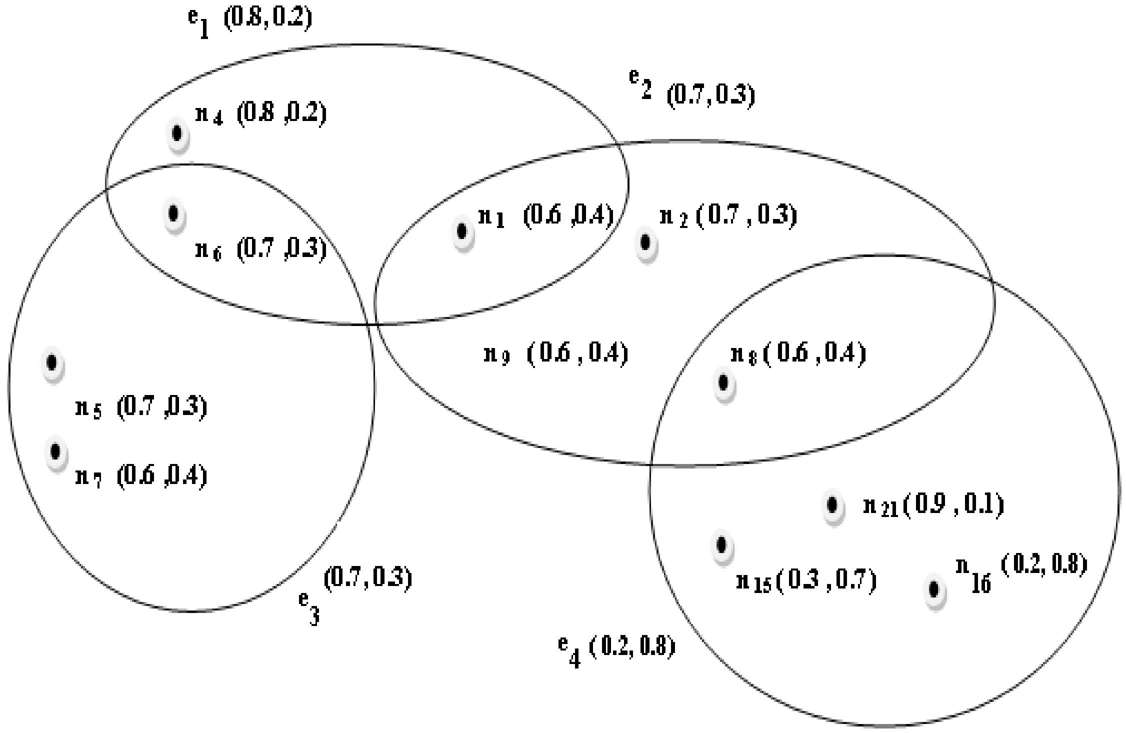


FIGURE 4.1: IFHG with degrees

4.4 (α, β) cut

Let $[X_{IF}, (\mu'_n, \gamma'_n), (\mu'_e, \gamma'_e), X^n, X^e]$ be the sub-hypergraph obtained by applying the (α, β) cut on H_{IF}, where α corresponds to the membership degree and β corresponds to the non-membership degree of nodes/edges. i.e., $H_{\alpha,\beta} = X_{IF}$. The (α, β) cut of H_{IF} can be written as follows:

$H_{\alpha,\beta} = [X_{IF}, (\mu'_n, \gamma'_n), (\mu'_e, \gamma'_e), X^n, X^e]$ $=\{(\mu_n'^{\alpha}, \gamma_n'^{\beta}), (\mu_e'^{\alpha}, \gamma_e'^{\beta})$ / $\mu_n'^{\alpha} = \{$ $\mu(n_i)/\mu(n_i) > \alpha$ $\} \cap \gamma_n'^{\beta} = \{\gamma(n_i)/\gamma(n_i) < \beta$ $\} \cap \mu_e'^{\alpha} = \{$ $\mu(e_i)/\mu(e_i) > \alpha$ $\} \cap \gamma_e'^{\beta}$ $= \{\gamma(e_i)/\gamma(e_i) < \beta$ $\}$ $\}$where $\mu(e_i)$ and $\gamma(e_i)$ are defined as in section 4.3.

Here $X_{IF} \subset H_{IF}$, such that X_{IF} consists of nodes with membership degree $>$ 0.5. The hyperedges in X_{IF} has at least one node with membership degree $>$ 0.5 and it should not contain any node with non-membership degree $>$ 0.5. i.e., the membership degree can be greater than 0.5, but the non-membership degree should be less than 0.5. Now X_{IF} is a collection of priority edges and priority nodes.

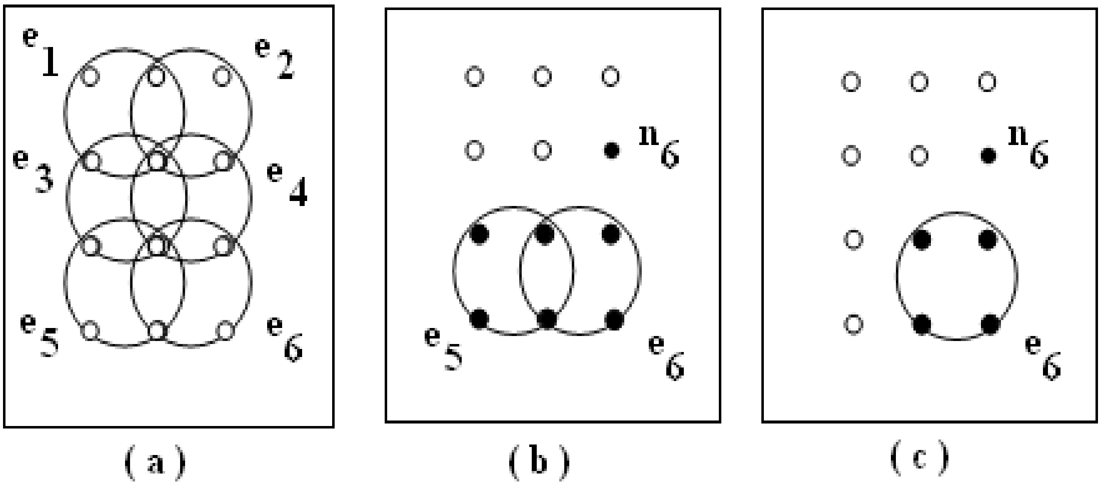


FIGURE 4.2: (α, β) cut and sub-IFHGs

4.5 Complement of a sub-IFHG

Given a parent IFHG H, sub-IFHG X, we can define (edge complement) of X as

$$X^{e'} = X' = H^e - X^e, \tag{4.1}$$

and nodes in $X^{e'}$ can be defined as

$$X^{e'n} = [H^n - X^n] \cup \{n_i/n_i \in X^n \cap n_i \in IX^n\}, \tag{4.2}$$

where IX^n are isolated nodes in X without an edge. Now the node complement of X can be written as $X^{n'}$ where

$$X^{n'} = H^n - X^n. \tag{4.3}$$

Both $X^{e'n}$ and $X^{n'}$ is shown in Fig. 4.3. We can see that node n_8 is a node in X^n. According to eq(4.2), it is also present in $X^{e'n}$ since it is an isolated node in X without a hyperedge. But we can see that $X^{n'}$ is not having node n_8. In this work X' should be considered as $X^{e'}$.

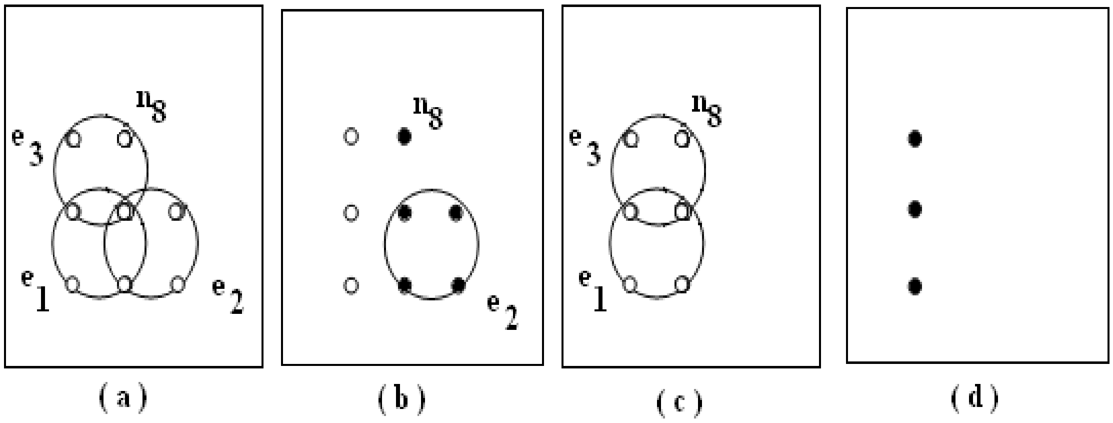


FIGURE 4.3: IFHGs (a): H, (b): X, (c): $X^{e'}$, (d): $X^{n'}$

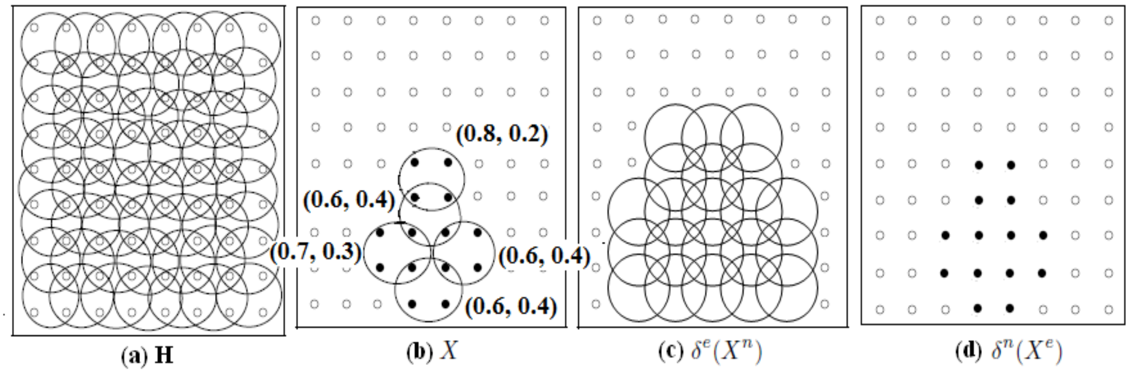


FIGURE 4.4: Results of morphological dilation

4.6 Morphological dilation

Morphological Dilation [88] [104] is of two types:

1. Dilation w.r.to hyperedge - $\delta^e(X^n)$ - Returns the set of edges which has at least one node in X.

2. Dilation w.r.to node - $\delta^n(X^e)$ - Returns the set of nodes within the hyperedges of X.

The parent and sub-IFHGs considered for dilation operation are given in Fig. 4.5 and Table 4.1.

Let us apply union operation on these dilations and verify the results
Proposition 4.1: Let H_{IF} be the parent IFHG, X and Y be the sub-IFHGs, then the following holds

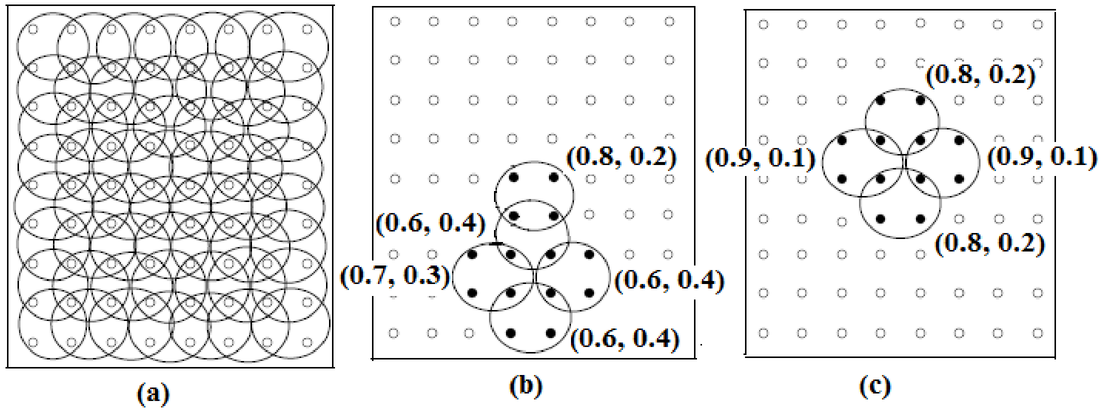


FIGURE 4.5: (a) H (b) X (c) Y

TABLE 4.1: Details of hypergraph H_{IF}

Hyperedges	nodes				Edge priority
e_1 (0.4, 0.6)	n_1 (0.5, 0.5)	n_2 (0.5, 0.5)	n_3 (0.4, 0.6)	n_4 (0.5, 0.5)	Low
e_2 (0.5, 0.5)	n_2 (0.5, 0.5)	n_4 (0.5, 0.5)	n_5 (0.5, 0.5)	n_6 (0.5, 0.5)	Medium
e_3 (0.4, 0.6)	n_3 (0.4, 0.6)	n_4 (0.5, 0.5)	n_7 (0.6, 0.4)	n_8 (0.8, 0.2)	Low
e_4 (0.9, 0.1)	n_4 (0.5, 0.5)	n_6 (0.5, 0.5)	n_8 (0.5, 0.5)	n_9 (0.9, 0.1)	Low
e_5 (0.8, 0.2)	n_7 (0.7, 0.3)	n_8 (0.5, 0.5)	n_{10} (0.5, 0.5)	n_{11} (0.6, 0.4)	High
e_6 (0.9, 0.1)	n_8 (0.5, 0.5)	n_9 (0.9, 0.1)	n_{11} (0.8, 0.2)	n_{12} (0.9, 0.1)	High

$$\delta^n(X \cup Y)^e = \delta^n(X^e) \cup \delta^n(Y^e), \tag{4.4}$$

where $\delta^n(X \cup Y)^e$ is a dilation w.r.to nodes. This dilation retrieves only priority nodes.

Proof: Consider the union operation of two sub-IFHGs X and Y. In L.H.S of eq(4.4), consider $X \cup Y = (Z^n, Z^e)$ where $Z^n = X^n \cup Y^n$ and $Z^e = X^e \cup Y^e$ and the union operation is same as the operation on graph. Therefore

$$\delta^n(X \cup Y)^e = Z^n \tag{4.5}$$

by definition of δ^n.

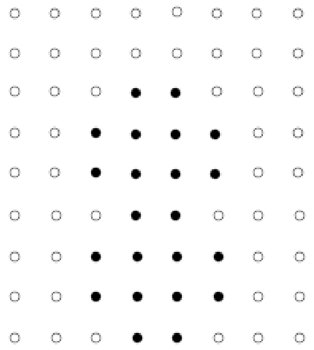


FIGURE 4.6: $\delta^n(X \cup Y)^e$

Also $\delta^n(X^e) = X^n$ and $\delta^n(Y^e) = Y^n$. Thus

$$\delta^n(X^e) \cup \delta^n(Y^e) = X^n \cup Y^n = Z^n. \tag{4.6}$$

Equations(4.5) and (4.6), imply $\delta^n(X \cup Y)^e = \delta^n(X^e) \cup \delta^n(Y^e)$. Fuzzy membership and non-membership degrees are invariant under this equation. The result of this operation is shown in the Fig. 4.6. As seen in this figure, this dilation operation will retrieve all nodes within the priority sub-IFHGs. Since the graphs are of higher priority, the nodes retrieved are also of high priority.

Example 4.1: Consider $H = (H^n, H^e)$ as an IFHG, where H^n are the nodes such that $H^n = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$ and H^e be the hyperedges such that $H^e = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Let X be the sub-IFHG, where $X = (X^n, X^e)$; where $X^n = \{n_7, n_8, n_{10}, n_{11}\}$ and $X^e = e_5$. Let Y be the sub-IFHG, where $Y = (Y^n, Y^e)$; where $Y^n = \{n_8, n_9, n_{11}, n_{12}\}$ and $Y^e = e_6$. The membership degree and the non-membership degree of the nodes and hyperedges of IFHG H are given in Table 4.1. Here we set $X = H_{\alpha,\beta}/\{0.5 < \alpha <= 0.9; \beta <= 1 - \alpha\}$;$Y = H_{\alpha,\beta}/\{\alpha >= 0.9; \beta <= 1 - \alpha\}$. The result of the operation $\delta^n(X \cup Y)^e$ is $Z^n = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Also the same results are obtained from $\delta^n(X^e) \cup \delta^n(Y^e)$; i.e., $\{n_7, n_8, n_{10}, n_{11}\} \cup \{n_8, n_9, n_{11}, n_{12}\} = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\} = Z^n$. Here the proof is substantiated with the above example.

Proposition 4.2: Let X and Y be sub-IFHGs of H_{IF} and δ be the dilation operator defined, then edge dilation of $X \cup Y$ is

$$\delta^e(X \cup Y)^n = \delta^e(X^n) \cup \delta^e(Y^n). \tag{4.7}$$

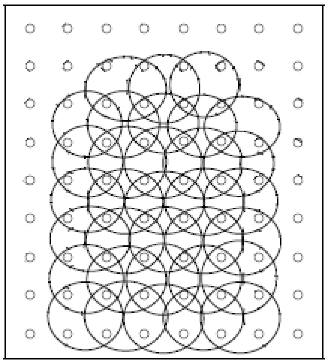


FIGURE 4.7: $\delta^e(X \cup Y)^n$

Proof: In L.H.S of eq(4.7), $\delta^e(X \cup Y)^n$ is the collection of all edges which consists of nodes in $X \cup Y$. As shown in Fig. 4.7, it consists of edges which are of priority and also of non priority. This is because, the nodes of sub-IFHGs X and Y are also part of other hyper edges which are of medium priority or low priority. All other edges are discarded in this operation. Let

$$\delta^e(X \cup Y)^n = Z^e. \tag{4.8}$$

In R.H.S of eq(4.7), $\delta^e(X^n) = X^e$ and $\delta^e(Y^n) = Y^e$.

Hence

$$\delta^e(X^n) \cup \delta^e(Y^n) = X^e \cup Y^e = Z^e. \tag{4.9}$$

From equations(4.8) and (4.9), it follows that $\delta^e(X \cup Y)^n = \delta^e(X^n) \cup \delta^e(Y^n)$.

Example 4.2: Consider the same problem defined in example 4.1. Applying it in L.H.S of eq(4.7), we get $(X \cup Y)^n = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Now $\delta^e(X \cup Y)^n = \{e_3, e_4, e_5, e_6\}$. Considering R.H.S, we get $\delta^e(X^n) = \{e_3, e_4, e_5, e_6\}$ and $\delta^e(Y^n) = \{e_3, e_4, e_5, e_6\}$. Now $\delta^e(X^n) \cup \delta^e(Y^n) = \{e_3, e_4, e_5, e_6\}$.

Proposition 4.3: Let H_{IF} be the parent IFHG, X and Y be the sub-IFHGs, δ be the dilation operator, then dilation w.r.to nodes of $(X' \cup Y')$ is

$$\delta^n(X' \cup Y')^e = \delta^n(X'^e) \cup \delta^n(Y'^e). \tag{4.10}$$

Proof: Let $X' = H - X, Y' = H - Y, X' = (X^{n'}, X^{e'})$. Let $(X' \cup Y')^e$ be the set of all hyperedges not in $X' \cup Y' = Z'$. Also $Z^{n'}$ and $Z^{e'}$ are the nodes and hyperedges of Z'. Hence

$$\delta^n(X' \cup Y')^e = Z^{n'}. \tag{4.11}$$

Also

$$\delta^n(X'^e) \cup \delta^n(Y'^e) = X^{n'} \cup Y^{n'}. \tag{4.12}$$

From eq(4.11) and eq(4.12), eq(4.10) is implied. Here $(X' \cup Y')^e$ retrieves all edges which are of high, medium and low priority. So also the dilation operation $\delta^n(X' \cup Y')^e$ retrieves all nodes within these high, medium and low priority hyperedges. The same is shown in Fig. 4.8(a).

Example 4.3: Considering L.H.S of eq(4.10), we obtain $(X' \cup Y')^e$ as the set $\{e_1, e_2, e_3, e_4, e_5, e_6\}$. From that we get, $\delta^n(X' \cup Y')^e$ as the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Consider R.H.S of eq(4.10), where we get $\delta^n(X'^e) = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{11}, n_{12}\}$. Also $\delta^n(Y'^e) = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. Now $\delta^n(X'^e) \cup \delta^n(Y'^e)$ is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$.

Proposition 4.4: Let H_{IF} be the parent IFHG, X and Y be the sub-IFHGs, δ be the dilation operator then dilation w.r.to edge is

$$\delta^e(X' \cup Y')^n = \delta^e(X'^n) \cup \delta^e(Y'^n). \tag{4.13}$$

Proof: Let $X' = H - X, Y' = H - Y, X' = (X^{n'}, X^{e'}), Y' = (Y^{n'}, Y^{e'})$. $(X' \cup Y')^n$ be the set of all nodes not in $X \cup Y$, where $X \cup Y = Z'$. Also $Z^{n'}$ and $Z^{e'}$ are the nodes and hyperedges of Z'. Hence

$$\delta^e(X' \cup Y')^n = Z^{e'}. \tag{4.14}$$

Also

$$\delta^e(X'^n) \cup \delta^e(Y'^n) = X^{e'} \cup Y^{e'}. \tag{4.15}$$

From eq(4.14) and eq(4.15), eq(4.13) is implied. This dilation operation will retrieve all edges which are of high, medium and low priority. The same is shown in Fig. 4.8(b).

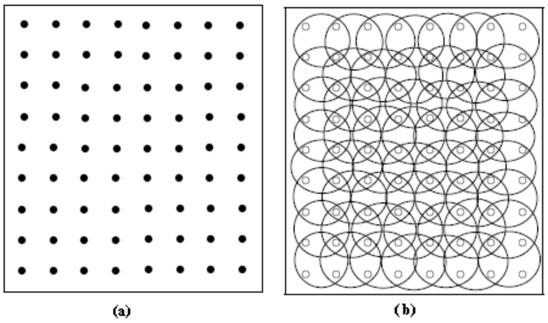


FIGURE 4.8: (a) $\delta^n(X' \cup Y')^e$ (b)$\delta^e(X' \cup Y')^n$

Example 4.4: Considering the L.H.S of eq(4.13), we obtain the result $(X' \cup Y')^n$ as the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Now $\delta^e(X' \cup Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Consider the R.H.S, where $\delta^e(X'^n) = \{e_1, e_2, e_3, e_4, e_6\}$. Now $\delta^e(Y'^n) = \{e_1, e_2, e_3, e_4, e_5\}$. From this we get $\delta^e(X'^n) \cup \delta^e(Y'^n) = \{e_1, e_2, e_3, e_4, e_5, e_6\}$.

Let us apply intersection operation on these dilations and verify the results.

Proposition 4.5: Let H_{IF} be the parent IFHG, X and Y be the sub-IFHGs, then the following holds:

$$\delta^n(X \cap Y)^e = \delta^n(X^e) \cap \delta^n(Y^e), \tag{4.16}$$

provided there are common edges in X and Y, where $\delta^n(X \cap Y)^e$ is a morphological dilation w.r.to nodes, which retrieves priority nodes which are present in common edges of both the subgraphs.

Proof: Consider the intersection of two sub-IFHGs X and Y. In L.H.S of eq(13), consider $X \cap Y = (Z^n, Z^e)$ where $Z^n = X^n \cap Y^n$ and $Z^e = X^e \cap Y^e$ and the intersection operation is same as the operation on graph. Therefore

$$\delta^n(X \cap Y)^e = Z^n \tag{4.17}$$

by definition of δ^n. Also $\delta^n(X^e) = X^n$ and $\delta^n(Y^e) = Y^n$. Hence

$$\delta^n(X^e) \cap \delta^n(Y^e) = X^n \cap Y^n = Z^n. \tag{4.18}$$

Eq(4.16) is implied by eq(4.17) and eq(4.18). Fuzzy membership and

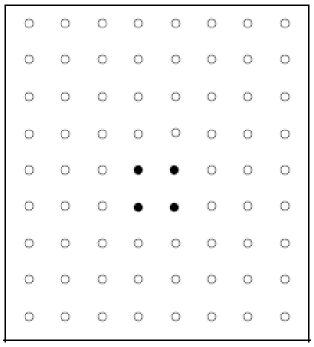


FIGURE 4.9: $\delta^n(X \cap Y)^e$

non-membership are invariant under this equation. The resultant graph is shown in Fig. 4.9. This dilation will retrieve only priority nodes which are found in both X and Y.

Example 4.5: Since the above result is true only if there are common edges in X and Y, let us modify X by including a common edge with Y so that now new $X = (X^n, X^e)$, where $X^e = \{e_5, e_6\}$ and $X^n = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Consider result of L.H.S of eq(4.16), where we get $(X \cap Y)^e = \{e_6\}$, so we get $\delta^n(X^e) = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Consider R.H.S of eq(4.16), where we get $\delta^n(X^e) = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$ and $\delta^n(Y^e)$ is the set $\{n_8, n_9, n_{11}, n_{12}\}$. Then Proposition 4.5 is proved with the result $\delta^n(X^e) \cap \delta^n(Y^e)$ as $\{n_8, n_9, n_{11}, n_{12}\}$.

Proposition 4.6: Let H_{IF} be the parent IFHG, X and Y be the sub-IFHGs, then the following holds:

$$\delta^e(X \cap Y)^n = \delta^e(X^n) \cap \delta^e(Y^n), \tag{4.19}$$

where $\delta^e(X \cap Y)^n$ is a dilation w.r.to edges which retrieves edges which contains at least one priority node common in both the subgraphs.

Proof: In L.H.S of eq(4.19), $(X \cap Y)^n$ is the collection of all nodes in $X \cap Y$. i.e., $\delta^e(X \cap Y)^n$ is the collection of all hyperedges which contains these nodes. Let

$$\delta^e(X \cap Y)^n = Z^e. \tag{4.20}$$

In R.H.S of eq(4.19), $\delta^e(X^n) = X^e$ and $\delta^e(Y^n) = Y^e$. Hence

$$\delta^e(X^n) \cap \delta^e(Y^n) = X^e \cap Y^e = Z^e. \tag{4.21}$$

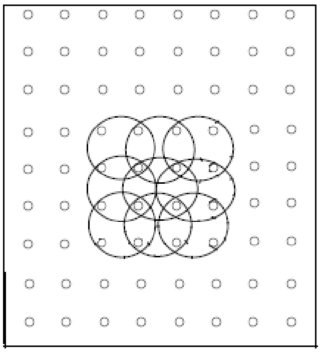


FIGURE 4.10: $\delta^e(X \cap Y)^n$

Eq(4.19) is implied by eq(4.20) and eq(4.21). The same result is shown in Fig. 4.10. As we see in the figure, not all edges are of high priority. It retrieves all kinds of edges. But it ensures that at least one node in that edge is of high priority.

Example 4.6: Consider X and Y mentioned in example 4.1. In L.H.S of eq(4.19), while we get $(X \cap Y)^n = \{n_8, n_9\}$. we get $\delta^e(X \cap Y)^n = \{e_3, e_4, e_5, e_6\}$. Considering R.H.S of eq(4.19), we obtain $\delta^e(X^n) = \{e_3, e_4, e_5, e_6\}$; $\delta^e(Y^n) = \{e_3, e_4, e_5, e_6\}$. Now $\delta^e(X^n) \cap \delta^n(Y^n) = \{e_3, e_4, e_5, e_6\}$. Now $\delta^e(X^n) \cap \delta^e(Y^n) = \{e_3, e_4, e_5, e_6\}$.

Proposition 4.7: Let H_{IF} be the parent IFHG, X, Y the sub-IFHGs, then

$$\delta^n(X' \cap Y')^e = \delta^n(X'^e) \cap \delta^n(Y'^e), \tag{4.22}$$

provided there are common edges in X and Y, where $\delta^n(X' \cap Y')^e$ is a dilation w.r.to nodes which retrieves all types of nodes in edges which are present in X' and Y'.

Proof: Let $X' = H - X = (X^{n'}, X^{e'})$ and $Y' = H - Y = (Y^{n'}, Y^{e'})$. Let $(X' \cap Y')^e$ be the set of all hyperedges not in $X \cap Y$, where $X' \cap Y' = Z'$. Also $Z^{n'}$ and $Z^{e'}$ are the nodes and hyperedges of Z'. Hence

$$\delta^n(X' \cap Y')^e = Z^{n'}. \tag{4.23}$$

Also

$$\delta^n(X')^e \cap \delta^n(Y')^e = X^{n'} \cap Y^{n'}. \tag{4.24}$$

Eq(4.22) is implied by eq(4.23) and eq(4.24). Resultant graph obtained is shown in Fig. 4.11(a).

Example 4.7: Let us consider the modified $X = (X^n, X^e)$, where $X^e = \{e_5, e_6\}$ and $X^n = \{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Now in L.H.S of eq(4.22), we get $(X' \cap Y')^e$ as $\{e_1, e_2, e_3, e_4\}$. Now $\delta^n(X' \cap Y')^e = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. Consider R.H.S of eq(4.22) where we get $\delta^n(X')^e = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. $\delta^n(Y')^e = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. Thus we get $\delta^n(X')^e \cap \delta^n(Y')^e = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$.

Proposition 4.8: Let H_{IF} be the parent IFHG, X, Y the sub-IFHGs, then

$$\delta^e(X' \cap Y')^n = \delta^e(X'^n) \cap \delta^e(Y'^n), \tag{4.25}$$

where $\delta^e(X' \cap Y')^n$ is a dilation w.r.to edges which retrieves all types of edges which contains at least one node common in X' and Y'.

Proof: Let $X' = H - X = (X^{n'}, X^{e'})$ and $Y' = H - Y = (Y^{n'}, Y^{e'})$. Let $(X' \cap Y')^n$ be the set of all nodes not in $X \cap Y$, where $X' \cap Y' = Z'$. Also $Z^{n'}$ and $Z^{e'}$ are the nodes and hyperedges of Z'. Hence

$$\delta^e(X' \cap Y')^n = Z^{e'}. \tag{4.26}$$

Also

$$\delta^e(X')^n \cap \delta^e(Y')^n = X^{e'} \cap Y^{e'}. \tag{4.27}$$

Eq(4.25) is implied by eq(4.26) and eq(4.27). The resultant graph is shown in Fig. 4.11(b).

Example 4.8: Take X and Y as defined in Example 4.1. Applying it in eq(4.25), we get $(X' \cap Y')^n = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. So $\delta^e(X' \cap Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. $\delta(X')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Also $\delta^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Thus $\delta^e(X')^n \cap \delta^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$.

4.7 Generalized associative law

- **Union**

  Proposition 4.9: Let $X_1, X_2, ..., X_n$ be the sub-IFHGs of H_{IF}. Let δ^n be the dilation w.r.to nodes, then

$$\delta^n(X_1 \cup X_2 \cup .. \cup X_n)^e = \delta^n(X_1)^e \cup \delta^n(X_2)^e .... \cup \delta^n(X_n)^e. \tag{4.28}$$

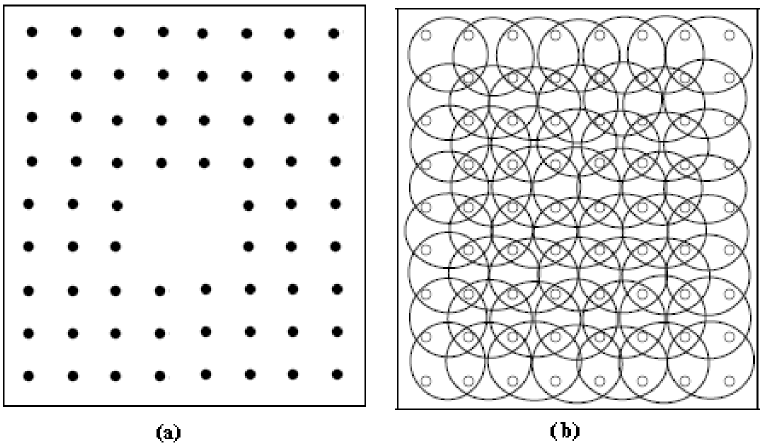


FIGURE 4.11: (a) $\delta^n(X' \cap Y')^e$ (b)$\delta^e(X' \cap Y')^n$

The proof follows from proposition 4.1. Similarly

$$\delta^e(X_1 \cup X_2 \cup ... \cup X_n)^n = \delta^e(X_1)^n \cup \delta^e(X_2)^n .... \cup \delta^e(X_n)^n, \tag{4.29}$$

where the proof follows from proposition 4.2.

- **Intersection**
  Proposition 4.10: Let $X_1, X_2, ..., X_n$ be sub-IFHGs of H_{IF}. Let δ^n be the dilation w.r.to nodes, then

$$\delta^n(X_1 \cap X_2 \cap ... \cap X_n)^e = \delta^n(X_1)^e \cap \delta^n(X_2)^e .... \cap \delta^n(X_n)^e. \tag{4.30}$$

The proof follows from proposition 4.5.
Similarly

$$\delta^e(X_1 \cap X_2 \cap ... \cap X_n)^n = \delta^e(X_1)^n \cap \delta^e(X_2)^n .... \cap \delta^e(X_n)^n, \tag{4.31}$$

where the proof follows from proposition 4.6.

- **Distributive law**
  Let X, Y and T be three sub-IFHGs of H_{IF}, then

$$\delta^n((X \cup Y) \cap T)^e = \delta^n(X \cap T)^e \cup \delta^n(Y \cap T)^e. \tag{4.32}$$

The proof follows from proposition 4.1 and proposition 4.5.

4.8 De Morgan's law applied to morphological dilation

Dilation with respect to nodes considering union of sub-IFHGs

Proposition 4.11: Let X, Y be the sub-IFHGs, δ be the dilation operator then

$$\delta^n(X \cup Y)^{'e} = \delta^n(X')^e \cap \delta^n(Y')^e. \tag{4.33}$$

Proof: Let $(X \cup Y)'$ be the sub-IFHG with edges which are not present in $X \cup Y$. Let $(X \cup Y)^{'e}$ be the edges in that hypergraph. Hence $\delta^n(X \cup Y)^{'e}$ is the set of all nodes in the sub-IFHG $(X \cup Y)'$. Let it be $Z^{n'}$. Also $\delta^n(X')^e$ is the set of all nodes in X' and $\delta^n(Y')^e$ is the set of all nodes in Y'. Let v be an arbitrary node in $\delta^n(X \cup Y)^{'e}$, which implies that v belongs to $\delta^n(X')^e$ and v belongs to $\delta^n(Y')^e$. Hence

$$\delta^n(X \cup Y)'^e \subseteq \delta^n(X')^e \cap \delta^n(Y')^e. \tag{4.34}$$

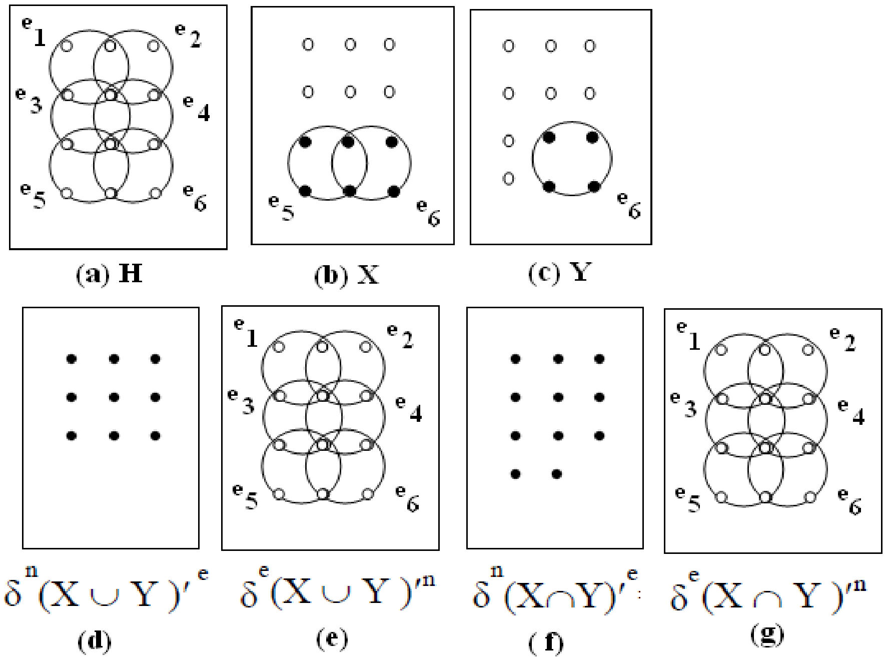
Let v belongs to $\delta^n(X')^e \cap \delta^n(Y')^e$ which implies that v belongs to X' and v belongs to Y'. Hence $v \notin X$ and $v \notin Y$.

Therefore $v \notin X \cup Y$. Hence $v \in (X \cup Y)'$. i.e.,

$$\delta^n(X')^e \cap \delta^n(Y')^e \subseteq \delta^n(X \cup Y)'^e. \tag{4.35}$$

Eq (4.33) is implied by eq(4.34) and eq(4.35).

Example 4.11: Consider the IFHG given in Table 4.1(also shown in Fig. 4.12(a)(b)(c)). Let X be a sub-IFHG $=(X^n, X^e)$ and $X^e = \{e_5, e_6\}$ and $X = (X^n, X^e)$, where X^e is the set $\{e_5, e_6\}$ and X^n is the set $\{n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$. Let Y be another sub-IFHG such that $Y = (Y^n, Y^e)$, where $Y^e = \{e_6\}$ and $Y^n = \{n_8, n_9, n_{11}, n_{12}\}$. Considering L.H.S of eq(4.33), we get $(X \cup Y)^{'e}$ as the set of hyperedges $\{e_1, e_2, e_3, e_4\}$. Now $\delta^n(X \cup Y)'^e$ is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. Considering R.H.S of eq(4.33), we get $\delta^n(X')^e$ as the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. Also $\delta^n(Y')^e$ is the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. Thus we get $\delta^n(X')^e \cap \delta^n(Y')^e$ as the set given by $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$ as shown in Fig. 4.12(d).



(a) H (b) X (c) Y

$\delta^n(X \cup Y)'^e$ (d) $\delta^e(X \cup Y)'^n$ (e) $\delta^n(X \cap Y)'^e$ (f) $\delta^e(X \cap Y)'^n$ (g)

FIGURE 4.12: De Morgan's law and morphological dilation

Dilation with respect to hyperedge considering union of sub-IFHGs

Proposition 4.12: Let X, Y be the sub-IFHGs and δ be the dilation operator, then

$$\delta^e(X \cup Y)'^n = \delta^e(X')^n \cap \delta^e(Y')^n. \tag{4.36}$$

Proof: Let $(X \cup Y)'$ be the sub-IFHG edges which are not present in $X \cup Y$. Let $(X \cup Y)'^e$ be the edges in that hypergraph. Hence $\delta^e(X \cup Y)'^n$ is the set of all hyperedges in the sub-IFHG $(X \cup Y)'$. Let it be $Z^{e'}$. Also $\delta^e(X')^n$ is the set of all hyperedges in X' and $\delta^e(Y')^n$ is the set of all hyperedges in Y'. Let v be an arbitrary node in $\delta^e(X \cup Y)'^n$, which implies that v belongs to both $\delta^e(X')^n$ and $\delta^e(Y')^n$. Hence

$$\delta^e(X \cup Y)'^n \subseteq \delta^e(X')^n \cap \delta^e(Y')^n. \tag{4.37}$$

Let v belongs to $\delta^e(X')^n \cap \delta^e(Y')^n$ which implies that v belongs to both X' and Y'. Hence $v \notin X$ and $v \notin Y$.

Therefore $v \notin X \cup Y$.

which implies $v \in (X \cup Y)'$. Hence

$$\delta^e(X')^n \cap \delta^e(Y')^n \subseteq \delta^e(X \cap Y)'^n. \tag{4.38}$$

Eq (4.36) is implied by eq(4.37) and eq(4.38).

Example 4.12: Consider the IFHG given in Table 4.1(also shown in Fig. 4.12(a)(b)(c)). Considering L.H.S of eq(4.36), we get $(X \cap Y)'^n$ as the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. Thus we get $\delta^e(X \cap Y)'^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Considering R.H.S of eq(4.36), we get $\delta^e(X')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Thus $\delta^e(X')^n \cap \delta^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ as shown in Fig. 4.12(e).

Dilation with respect to node considering intersection of sub-IFHGs

Proposition 4.13: Let X, Y be the sub-IFHGs, δ be the dilation operator, then

$$\delta^n(X \cap Y)'^e = \delta^n(X')^e \cup \delta^n(Y')^e, \tag{4.39}$$

provided there are edges in $X \cap Y$.

Proof: Let $(X \cap Y)'$ be the IFHG with edges which are not present in $X \cap Y$. Let $(X \cap Y)'^e$ be the edges in that IFHG. let $\delta^n(X \cap Y)'^e$ be the set of all nodes in the sub-IFHG $X \cap Y$ as in Fig. 4.12(f). Let v be a node in $\delta^n(X \cap Y)'^e$, then it is not a node of $X \cap Y$ ie $v \notin X \cap Y$.

$$\delta^n(X \cap Y)'^e \notin X \cap Y. \tag{4.40}$$

Also $\delta^n(X')^e$ is the set of nodes in X' and $\delta^n(Y')^e$ is the set of nodes in Y'. Let v belong to $\delta^n(X')^e \cup \delta^n(Y')^e$ which implies that v either belongs to any node in X' or Y'. i.e., $v \notin X \cap Y$. Therefore

$$\delta^n(X')^e \cup \delta^n(Y')^e \notin X \cap Y. \tag{4.41}$$

Eq(4.39) is implied by eq(4.40) and eq(4.41).

Example 4.13: Consider the IFHG given in Table 4.1(also shown in Fig. 4.12(a)(b)(c)). Consider L.H.S of eq(4.39), we get $(X \cap Y)'^e = \{e_1, e_2, e_3, e_4, e_5\}$. Now $\delta^n(X \cap Y)'^e$ is $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. Now from R.H.S of eq(4.39), we get $\delta^n(X')^e$ as the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$. Also $\delta^n(Y')^e$ gives the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. Therefore

$\delta^n(X')^e \cup \delta^n(Y')^e = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$ as shown in Fig. 4.12(f).

Dilation with respect to hyperedge considering intersection of sub-IFHGs

Proposition 4.14 Let X, Y be the sub-IFHGs, δ be the dilation operator, then

$$\delta^e(X \cap Y)'^n = \delta^e(X')^n \cup \delta^e(Y')^n, \tag{4.42}$$

provided there are edges in $X \cap Y$. Let $(X \cap Y)'$ be an IFHG with nodes which are not present in $X \cap Y$. Let $(X \cap Y)'^n$ be the nodes in that IFHG. Let $\delta^e(X \cap Y)'^n$ be the set of all edges in the sub-IFHG $(X \cap Y)'$. Let e be an edge in $\delta^e(X \cap Y)'^n$. i.e., it is not an edge of $X \cap Y$. Therefore $e \notin X \cap Y$ and hence

$$\delta^e(X \cap Y)'^n \notin X \cap Y. \tag{4.43}$$

Also $\delta^e(X')^n$ is the set of all edges in X'. Also $\delta^e(Y')^n$ is the set of all edges in Y'. Let e belongs to $\delta^e(X')^n \cup \delta^e(Y')^n$, which implies that e either belongs to X' or Y'. i.e., $e \notin X \cap Y$. Therefore

$$\delta^e(X')^n \cup \delta^e(Y')^n \notin X \cap Y. \tag{4.44}$$

Then Eq(4.42) is implied by eq(4.43) and eq(4.44).

Example 4.14: Consider the IFHG given in Table 4.1(also shown in Fig. 4.12(a)(b)(c)). Considering L.H.S of eq(4.42), we get $\delta^e(X \cap Y)'^n$ as the set of hyperedges $\{e_1, e_2, e_3, e_4, e_5, e_6\}$. Consider R.H.S of eq(4.42), where we get $\delta^e(X')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Now $\delta^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Thus $\delta^e(X')^n \cup \delta^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ as shown in Fig. 4.12(g).

4.9 Morphological erosion

Morphological erosion [88] [104] is of two types:

1. Erosion w.r.to hyperedge - $\varepsilon^e(X^n)$ - Returns the set of edges which has only nodes in X. Given a parent IFHG H and sub-IFHG X as in Fig. 4.14(a) and Fig. 4.14(b) respectively, the result of this erosion is shown in Fig. 4.14(c).

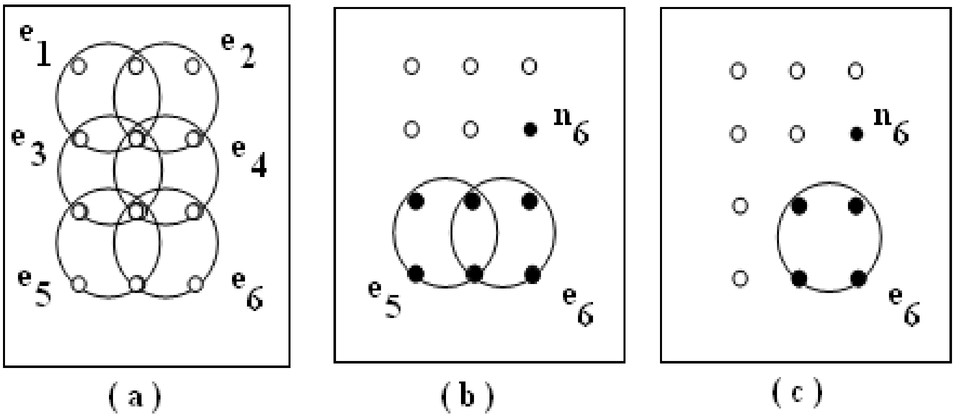


FIGURE 4.13: (a)Parent IFHG H, (b) sub-IFHG X, (c)sub-IFHG Y

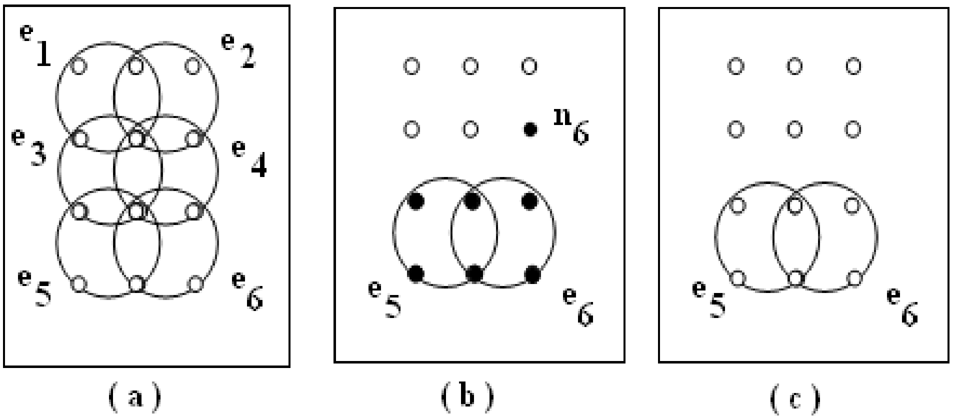


FIGURE 4.14: Result of erosion w.r.to edge

2. Erosion w.r.to node - $\varepsilon^n(X^e)$ - Returns the set of nodes which are not present in $X^{e'}$. Given a parent IFHG H and sub-IFHG X as in Fig. 4.15(a) and Fig. 4.15(b) respectively, the result of this erosion is shown in Fig. 4.15(c).

4.10 Mathematical modeling for morphological erosion

In this section, let us model a parent IFHG and introduce (α, β) cut of the parent IFHG in order to create priority sub-IFHGs. Let us define $H_{IF} = [H^n, H^e]$, where $H^n = [n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}]$ and $H^e = [e_1, e_2, e_3, e_4, e_5, e_6]$ as given in Fig. 4.13. Some nodes are of low priority ($\mu_n < 0.5$), some are of medium priority ($\mu_n = 0.5$) and few others are of high

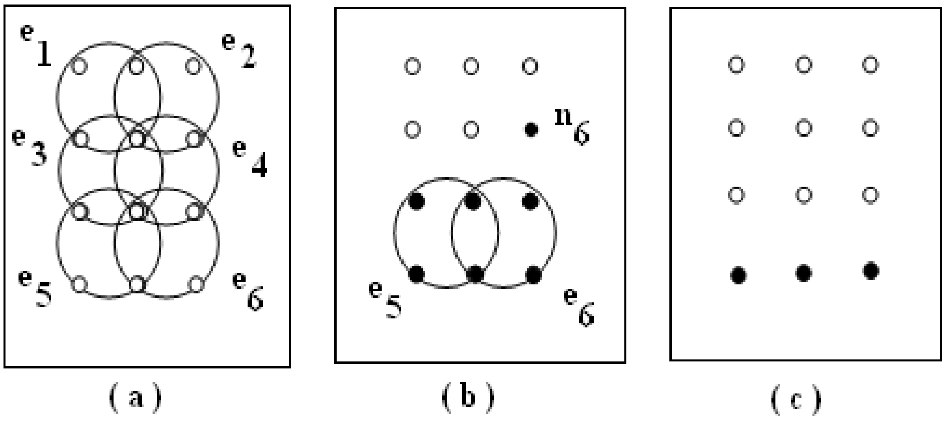


Figure 4.15: Result of erosion w.r.to node

Table 4.2: Details of hypergraph H_{IF}

Hyperedges	Nodes				Edge priority
e_1 (0.4, 0.6)	n_1 (0.5, 0.5)	n_2 (0.5, 0.5)	n_3 (0.4, 0.6)	n_4 (0.4, 0.5)	Low
e_2 (0.4, 0.6)	n_2 (0.5, 0.5)	n_4 (0.4, 0.6)	n_5 (0.4, 0.6)	n_6 (0.9, 0.1)	Low
e_3 (0.4, 0.6)	n_3 (0.4, 0.6)	n_4 (0.4, 0.6)	n_7 (0.6, 0.4)	n_8 (0.8, 0.2)	Low
e_4 (0.4, 0.6)	n_4 (0.4, 0.6)	n_6 (0.9, 0.1)	n_8 (0.8, 0.2)	n_9 (0.9, 0.1)	Low
e_5 (0.8, 0.2)	n_7 (0.6, 0.4)	n_8 (0.8, 0.2)	n_{10} (0.8, 0.2)	n_{11} (0.8, 0.2)	High
e_6 (0.9, 0.1)	n_8 (0.8, 0.2)	n_9 (0.9, 0.1)	n_{11} (0.8, 0.2)	n_{12} (0.9, 0.1)	High

priority ($\mu_n > 0.5$). Let X_{IF} be obtained by (α, β) cut on $H_{IF}/0.5 < \alpha \leq 0.9; \{\beta \leq 1 - \alpha\} \cap \{\beta \leq 0.1\}$. Here α corresponds to membership degree and β corresponds to non-membership degree. We can see that thresholds are applied on both α and β. Since α has a wider range, we get a sub-IFHG with more number of edges and nodes. Let Y_{IF} be obtained by (α, β) cut on $H_{IF}/\alpha \geq 0.9; \{\beta \leq 1 - \alpha\} \cap \{\beta \leq 0.1\}$. Since α has high limit, we get a sub-IFHG with relatively less number of hyperedges and nodes when compared with the first one. The details of the hypergraphs H_{IF}, X_{IF} and Y_{IF} are given in Table 4.2, Table 4.3 and Table 4.4 respectively. For simplicity let us refer them as H, X and Y.

Table 4.3: Details of hypergraph X_{IF}

Hyperedges	Nodes				Edge priority
e_5 (0.8, 0.2)	n_7 (0.6, 0.4)	n_8 (0.8, 0.2)	n_{10} (0.8, 0.2)	n_{11} (0.8, 0.2)	High
e_6 (0.9, 0.1)	n_8 (0.8, 0.2)	n_9 (0.9, 0.1)	n_{11} (0.8, 0.2)	n_{12} (0.9, 0.1)	High
Hyperedges	Nodes				Node priority
Nil	n_6 (0.9, 0.1)				High

Table 4.4: Details of hypergraph Y_{IF}

Hyperedges	nodes				Edge priority
e_6 (0.9, 0.1)	n_8 (0.8, 0.2)	n_9 (0.9, 0.1)	n_{11} (0.8, 0.2)	n_{12} (0.9, 0.1)	High
Hyperedges	nodes				Node priority
Nil	n_6 (0.9, 0.1)				High

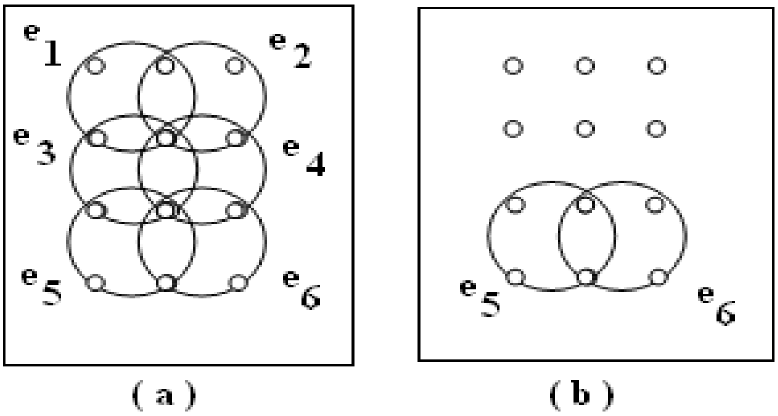


Figure 4.16: (a) H (b) Result of union of erosion w.r.to edge

Proposition 4.15: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^e(X \cup Y)^n = \varepsilon^e(X^n) \cup \varepsilon^e(Y^n), \tag{4.45}$$

where $\varepsilon^e(X \cup Y)^n$ is an erosion w.r.to edge, which retrieves edges consisting of only nodes either in X or in Y.

Proof: In L.H.S of eq(4.45), $\varepsilon^e(X \cup Y)^n$ = collection of all edges which consists of nodes in $(X \cup Y)^n$ only. Let $\{n_1, n_2, ..., n_k\}$ be the nodes in $(X \cup Y)$. Let $\{e_1, e_2, ..., e_k\}$ be the hyperedges which consists of these nodes only. Let $\varepsilon^e(X^n)$ be the set of hyperedges which contains nodes in X only and $\varepsilon^e(Y^n)$ be the set of hyperedges which consists of nodes in Y only. So these edges are present either in $\varepsilon^e(X^n)$ or in $\varepsilon^e(Y^n)$. We can explain this w.r.to an IFHG formed, where technical documents are hyperedges and authors are nodes. Let X be the sub-IFHG consisting of only documents and authors in the area of graphs. So X consists of hyperedges which represent documents and isolated nodes which represent authors. Let Y be the sub-IFHG which consists of documents and authors in the area of fuzzy graphs. Y also consists of hyperedges which represent documents and isolated nodes which represent authors. Let $Y \subset X$. Now $(X \cup Y)^n$ is the union of authors in graphs and fuzzy graphs. Also $\varepsilon^e(X \cup Y)^n$ is the set of documents which consists of these authors only. i.e.,

$$\varepsilon^e(X \cup Y)^n \subseteq \varepsilon^e(X^n) \cup \varepsilon^e(Y^n), \tag{4.46}$$

where $\varepsilon^e(X^n)$ can be considered as the documents in the area of graphs only and $\varepsilon^e(Y^n)$ can be documents in the area of fuzzy graph only.

Now considering R.H.S of eq(4.45), $\varepsilon^e(X^n)$ represents the collection of edges which contains nodes in X only. Also $\varepsilon^e(Y^n)$ represents the collection of edges which contains the nodes in Y only. Let $\{n_1, n_2, ..., n_p\}$ be the nodes in X and $\{e_1, e_2, ..., e_p\}$ be the edges in $\varepsilon^e(X^n)$. Let $\{n_{p+1}, n_{p+2}, ..., n_k\}$ be the nodes in Y and $\{e_{p+1}, e_{p+2}, ..., e_k\}$ be the edges in $\varepsilon^e(Y^n)$. Now an edge e in $\{e_1, e_2, ..., e_k\}$ can be present either in $\{e_1, e_2, ..., e_p\}$ or in $\{e_{p+1}, e_{p+2}, ..., e_k\}$ i.e.,

$$\varepsilon^e(X^n) \cup \varepsilon^e(Y^n) \subseteq \varepsilon^e(X \cup Y)^n. \tag{4.47}$$

Eq(4.45) is implied by eq(4.46) and eq(4.47).

Example 4.15: Consider Table 4.2, Table 4.3 and Table 4.4. In L.H.S of eq(4.45), we get $(X \cup Y)^n = \{n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}\}$ i.e., we are taking all nodes in $X \cup Y$. Now find edges from H which consists of these nodes only. This will give us $\varepsilon^e(X \cup Y)^n = \{e_5, e_6\}$. Now in R.H.S of (4.45), $\varepsilon^e(X^n)$ are the edges in H which consists of nodes of X only. i.e., $\varepsilon^e(X^n) = \{e_5, e_6\}$. Again $\varepsilon^e(Y^n)$

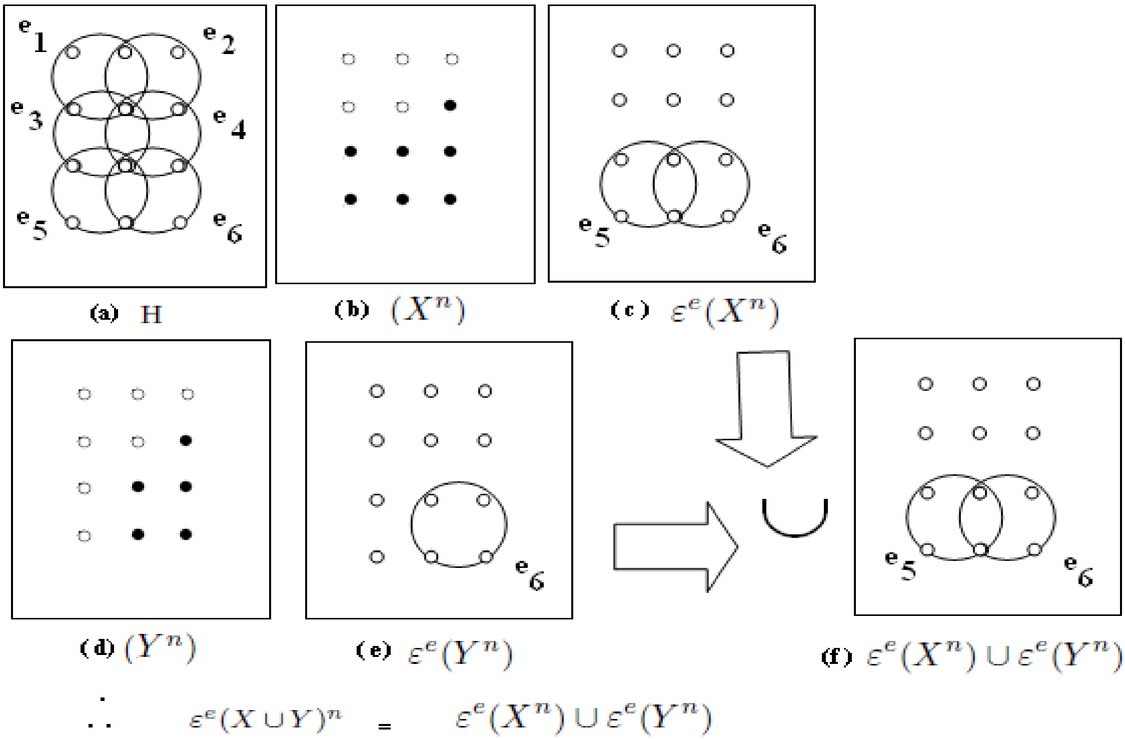

e1
e2
e3
e4
e5
e6
(a) H
(b) (X^n)
(c) $\varepsilon^e(X^n)$
e5
e6
(d) (Y^n)
(e) $\varepsilon^e(Y^n)$
e6
$\cup$
e5
e6
(f) $\varepsilon^e(X^n) \cup \varepsilon^e(Y^n)$
$\therefore$ $\varepsilon^e(X \cup Y)^n = \varepsilon^e(X^n) \cup \varepsilon^e(Y^n)$

FIGURE 4.17: $\varepsilon^e(X^n) \cup \varepsilon^e(Y^n)$

are the edges in H which consists of nodes on Y only. i.e., $\varepsilon^e(Y^n) = \{e_6\}$.Thus $\varepsilon^e(X^n) \cup \varepsilon^e(Y^n) = \{e_5, e_6\} \cup \{e_6\} = \{e_5, e_6\}$. The result is shown in Fig. 4.16(b) and Fig. 4.17.

Proposition 4.16: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^e(X \cap Y)^n = \varepsilon^e(X^n) \cap \varepsilon^e(Y^n), \tag{4.48}$$

where $\varepsilon^e(X \cap Y)^n$ is an erosion w.r.to edge which retrieves edges consisting of only nodes common in X and Y.

Also, If $X \cap Y = \phi$, then $\varepsilon^e(X \cap Y)^n = \varepsilon^e(X^n) \cap \varepsilon^e(Y^n) = \phi$.

Proof: In L.H.S of eq(4.48), $\varepsilon^e(X \cap Y)^n$ is the collection of all edges which consists of nodes in $(X \cap Y)$ only. Let $\{n_1, n_2, ..., n_k\}$ be the nodes in $(X \cap Y)$. Let $\{e_1, e_2, ...e_k\}$ be the hyperedges which consists of these nodes only. So these hyperedges are present both in $\varepsilon^e(X^n)$ and $\varepsilon^e(Y^n)$. i.e.,

$$\varepsilon^e(X \cap Y)^n \subseteq \varepsilon^e(X^n) \cap \varepsilon^e(Y^n). \tag{4.49}$$

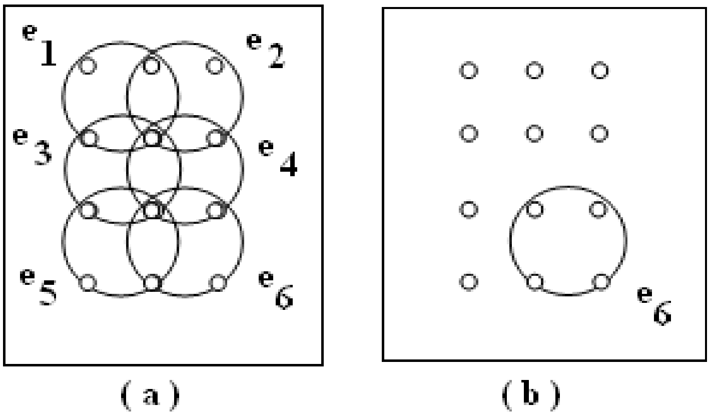


FIGURE 4.18: (a) H (b) Result of intersection of erosion w.r.to edge

Now considering R.H.S of eq(4.48), $\varepsilon^e(X^n)$ represents the collection of edges which contains nodes in X only. Also $\varepsilon^e(Y^n)$ represents the collection of edges which contains the nodes in Y only. Let $\{n_1, n_2, ..., n_p\}$ be the nodes in X and $\{e_1, e_2, ..., e_p\}$ be the edges in $\varepsilon^e(X^n)$. Let $\{n_{p+1}, n_{p+2}, ..., n_k\}$ be the nodes in Y and $\{e_{p+1}, e_{p+2}, ..., e_k\}$ be the edges in $\varepsilon^e(Y^n)$. Now an edge e in $\{e_1, e_2, ..., e_k\}$ must be present both in $\{e_1, e_2, ..., e_p\}$ and in $\{e_{p+1}, e_{p+2}, ..., e_k\}$ i.e.,

$$\varepsilon^e(X^n) \cap \varepsilon^e(Y^n) \subseteq \varepsilon^e(X \cap Y)^n. \tag{4.50}$$

Eq(4.48) is implied by eq(4.49) and eq(4.50).

Example 4.16: Consider Tables 4.2, 4.3 and Table 4.4. In L.H.S of (4.48), We get $(X \cap Y)^n = \{n_6, n_8, n_9, n_{11}, n_{12}\}$ i.e., we are taking all nodes in $X \cap Y$. Now find edges from H which consists of these nodes only. This will give us $\varepsilon^e(X \cap Y)^n = \{e_6\}$. Now in R.H.S of (4.48), $\varepsilon^e(X^n)$ are the edges in H which consists of nodes of X only. i.e., $\varepsilon^e(X^n) = \{e_5, e_6\}$. Again $\varepsilon^e(Y^n)$ are the edges in H which consists of nodes on Y only. i.e., $\varepsilon^e(Y^n) = \{e_6\}$. Thus $\varepsilon^e(X^n) \cap \varepsilon^e(Y^n) = \{e_5, e_6\} \cap \{e_6\} = \{e_6\}$. The result is shown in Fig. 4.18(b).

Proposition 4.17: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^n(X \cup Y)^e = \varepsilon^n(X^e) \cup \varepsilon^n(Y^e), \tag{4.51}$$

where $\varepsilon^n(X \cup Y)^e$ is an erosion w.r.to node which retrieves high priority nodes which are not present in $(X \cup Y)^{e'}$.

Proof: Consider L.H.S of eq(4.51). Let v be an arbitrary node in $\varepsilon^n(X \cup Y)^e$. i.e., v is a node in $(X \cup Y)^e$, but it is not a not of $(X \cup Y)^{e'}$.

i.e.,

$$v \in (X \cup Y)^e. \tag{4.52}$$

Also

$$v \notin (X \cup Y)^{e'}. \tag{4.53}$$

Consider R.H.S of eq(4.51), where v is a node of $\varepsilon^n(X^e) \cup \varepsilon^n(Y^e)$ i.e.,

$$v \in X^e. \tag{4.54}$$

Also

$$v \notin X^{e'}. \tag{4.55}$$

or

$$v \in Y^e. \tag{4.56}$$

Also

$$v \notin Y^{e'}. \tag{4.57}$$

Eq(4.52) is implied by eq(4.54) and eq(4.56). Also eq(4.53) is implied by eq(4.55) and eq(4.57). Also Eq(4.51) is implied by eq(4.52) - eq(4.57).

Example 4.17: Consider Table 4.2, Table 4.3 and Table 4.4. In L.H.S of eq(4.51), we get $(X \cup Y)^e = \{e_5, e_6\}$. i.e,, we are taking all edges in $X \cup Y$. Now find $(X \cup Y)^{e'}$ i.e., all edges which are not included in this. Now find all $(X \cup Y)^n$ which are not in any edge of $(X \cup Y)^{e'}$. i.e., $\varepsilon^n(X \cup Y)^e = \{n_{10}, n_{11}, n_{12}\}$. Consider R.H.S of eq(4.51). $X^e =$ all edges in X. Find $X^{e'} = H^e - X^e =$ all edges other than that in X. Now find all nodes in X which are not in any hyperedge of $X^{e'}$. i.e., $\varepsilon^n(Y^e) = \{n_{10}, n_{11}, n_{12}\}$. Now $\varepsilon^n(Y^e)$

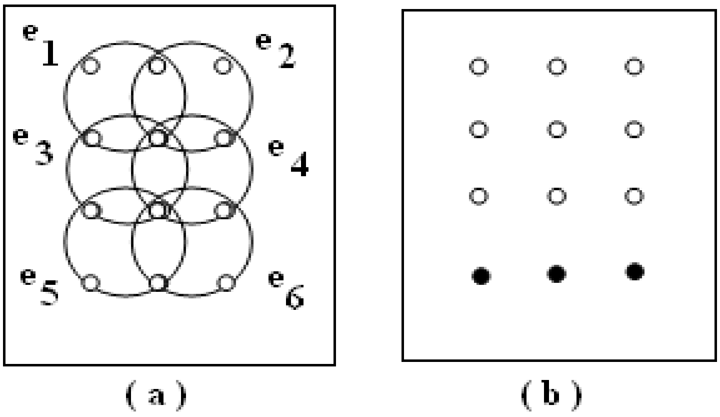


Figure 4.19: (a) H (b) Result of union of erosion w.r.to node

= all nodes in Y which are not in any edge of $Y^{e'}$, where $Y^{e'} = H^e - Y^e$. i.e., $\varepsilon^n(Y^e) = \{n_{12}\}$. Thus $\varepsilon^n(X^e) \cup \varepsilon^n(Y^e) = \{n_{10}, n_{11}, n_{12}\} \cup \{n_{12}\} = \{n_{10}, n_{11}, n_{12}\}$. The result is shown in Fig. 4.19(b).

Proposition 4.18: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^n(X \cap Y)^e = \varepsilon^n(X^e) \cap \varepsilon^n(Y^e), \tag{4.58}$$

where $\varepsilon^n(X \cap Y)^e$ is an erosion w.r.to node which retrieves high priority nodes which are not present in $(X \cap Y)^{e'}$.

Proof: Let v be an arbitrary node in $\varepsilon^n(X \cap Y)^e$. Consider L.H.S of eq(4.58), where v is a node in $(X \cap Y)^e$; but not a node of $(X \cap Y)^{e'}$. A node in $(X \cap Y)^{e'}$ is written as $(X \cap Y)^{e'n}$ and is defined as per eq(4.2).

i.e.,

$$v \in (X \cap Y)^e. \tag{4.59}$$

Also

$$v \notin (X \cap Y)^{e'}. \tag{4.60}$$

Consider R.H.S of eq(4.58), where v is a node of $\varepsilon^n(X^e) \cap \varepsilon^n(Y^e)$, then it is a node of (X^e), but not a node of $(X^{e'})$. A node of $(X^{e'})$ is written as $(X^{e'n})$ and is defined as per eq(4.2). Also v is a node of (Y^e), but not a node of $(Y^{e'})$.

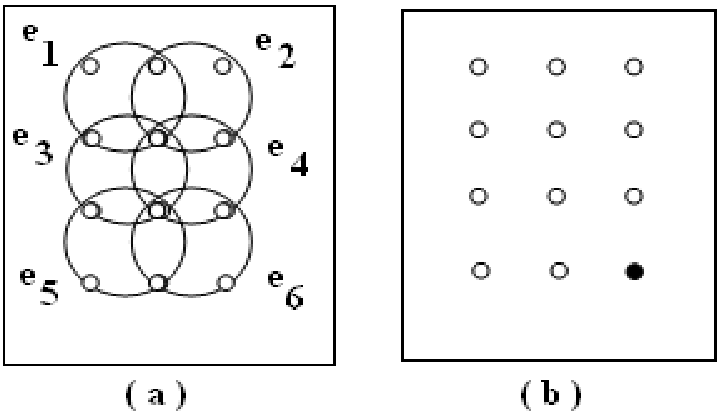


FIGURE 4.20: (a) H (b) Result of intersection of erosion w.r.to node

i.e.,

$$v \in (X^e); v \in (Y^e). \tag{4.61}$$

Also

$$v \notin (X^{e'}); v \notin (Y^{e'}). \tag{4.62}$$

Eq(4.59) is implied by eq(4.61); eq(4.60) is implied by eq(4.62). Eq(4.58) is implied by eq(4.59) - eq(4.62).

Example 4.18: In L.H.S of eq(4.58), $(X \cap Y)^e = \{e_6\}$. Now $(X \cap Y)^{e'} = \{e_1, e_2, e_3, e_4, e_5\}$. Now nodes in $(X \cap Y)$ not in any edge of $(X \cap Y)^{e'} = \varepsilon^n(X^e) \cap \varepsilon^n(Y^e) = \{n_{12}\}$. Let us take R.H.S of eq(4.58), $\varepsilon^n(X^e)$ is the nodes in X but not in any edge of $X^{e'}$ which is the set $\{\text{n}_9, n_{10}, n_{11}, n_{12}\}$; $\varepsilon^n(Y^e)$ is the nodes in Y but not in any edge of $Y^{e'} = \{n_{12}\}$; $\varepsilon^n(X^e) \cap \varepsilon^n(Y^e) = \{n_9, n_{10}, n_{11}, n_{12}\} \cap \{n_{12}\} = \{n_{12}\}$. The result is shown in Fig. 4.20(b).

Proposition 4.19: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^e(X' \cup Y')^n = \varepsilon^e(X')^n \cup \varepsilon^e(Y')^n, \tag{4.63}$$

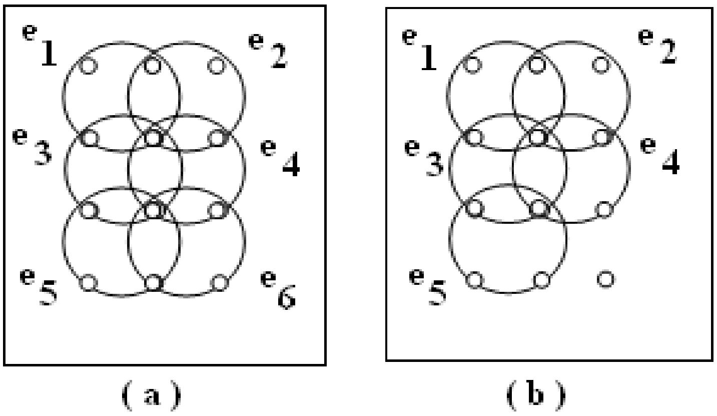


FIGURE 4.21: (a) H (b) Result of union of complement of erosion w.r.to edge

where $\varepsilon^e(X' \cup Y')^n$ is an erosion w.r.to edge which retrieves edges which consists of $(X' \cup Y')^n$ only.

Here X' and Y' are defined as in eq(4.1).

Proof: In L.H.S of eq(4.63), $\varepsilon^e(X' \cup Y')^n$ = collection of all edges which consists of nodes in $(X' \cup Y')^n$ only. Let $\{n_1, n_2, ..., n_k\}$ be the nodes in $(X' \cup Y')$. Let $\{e_1, e_2, ..., e_k\}$ be the hyperedges which consists of these nodes only. Therefore these nodes are present either in $\varepsilon^e(X')^n$ or in $\varepsilon^e(Y')^n$. i.e.,

$$\varepsilon^e(X' \cup Y')^n \subseteq \varepsilon^e(X')^n \cup \varepsilon^e(Y')^n. \tag{4.64}$$

Now considering R.H.S of eq(4.63), $\varepsilon^e(X')^n$ represents the collection of edges which contains nodes in X' only. Also $\varepsilon^e(Y')^n$ represents the collection of edges which contains the nodes in Y' only. Let $\{n_1, n_2, ..., n_p\}$ be the nodes in X' and $\{e_1, e_2, ..., e_p\}$ be the edges in $\varepsilon^e(X')^n$. Let $\{n_{p+1}, n_{p+2}, ..., n_k\}$ be the nodes in Y' and $\{e_{p+1}, e_{p+2}, ..., e_k\}$ be the edges in $\varepsilon^e(Y')^n$. Now an edge e in $\{e_1, e_2, ..., e_k\}$ can be present either in $\{e_1, e_2, ..., e_p\}$ or in $\{e_{p+1}, e_{p+2}, ..., e_k\}$ i.e.,

$$\varepsilon^e(X')^n \cup \varepsilon^e(Y')^n \subseteq \varepsilon^e(X' \cup Y')^n. \tag{4.65}$$

Eq(4.63) is implied by eq(4.64) and eq(4.65).

Example 4.19: Consider Table 4.2, Table 4.3 and Table 4.4. In L.H.S of eq(4.63), $(X' \cup Y')^n$ = all nodes in $(X' \cup Y')$. Now $\varepsilon^e(X' \cup Y')^n$ = all hyperedges from H which consists of $(X' \cup Y')^n$ only $= \{e_1, e_2, e_3, e_4, e_5\}$. In R.H.S of eq(4.63), we find $\varepsilon^e(X')^n$ as all edges in H which consists of nodes of X' only, which gives the set $\{e_1, e_2, e_3, e_4\}$. Now $\varepsilon^e(Y')^n$ is all edges in H which consists of nodes of Y' only, which gives the set $\{e_1, e_2, e_3, e_4, e_5\}$. Therefore $\varepsilon^e(X')^n \cup \varepsilon^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5\}$. The results are shown in Fig. 4.21(b).

Proposition 4.20: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^e(X' \cap Y')^n = \varepsilon^e(X')^n \cap \varepsilon^e(Y')^n, \tag{4.66}$$

where $\varepsilon^e(X' \cap Y')^n$ is an erosion w.r.to edge which retrieves edges which consists of $(X' \cap Y')^n$ only.

Here X' and Y' are defined as in eq(4.1).

If $X' \cap Y' = \phi$, then $\varepsilon^e(X' \cap Y')^n = \varepsilon^e(X')^n \cap \varepsilon^e(Y')^n = \phi$.

Proof: In L.H.S of eq(4.66) , $\varepsilon^e(X' \cap Y')^n$ = collection of all edges which consists of nodes in $(X' \cap Y')^n$ only. Let $\{n_1, n_2, ..., n_k\}$ be the nodes in $(X' \cap Y')$. Let $\{e_1, e_2, ..., e_k\}$ be the hyperedges which consists of these nodes only. Therefore these nodes are present both in $\varepsilon^e(X')^n$ and in $\varepsilon^e(Y')^n$. i.e.,

$$\varepsilon^e(X' \cap Y')^n \subseteq \varepsilon^e(X')^n \cap \varepsilon^e(Y')^n. \tag{4.67}$$

Now considering R.H.S of eq(4.66), $\varepsilon^e(X')^n$ represents the collection of edges which contains nodes in X' only. Also $\varepsilon^e(Y')^n$ represents the collection of edges which contains the nodes in Y' only. Let $\{n_1, n_2, ..., n_p\}$ be the nodes in X' and $\{e_1, e_2, ..., e_p\}$ be the edges in $\varepsilon^e(X')^n$. Let $\{n_{p+1}, n_{p+2}, ..., n_k\}$ be the nodes in Y' and $\{e_{p+1}, e_{p+2}, ..., e_k\}$ be the edges in $\varepsilon^e(Y')^n$. Now an edge e in $\{e_1, e_2, ..., e_k\}$ can be present both in $\{e_1, e_2, ..., e_p\}$ and in $e_{p+1}, e_{p+2}, ..., e_k$ i.e.,

$$\varepsilon^e(X')^n \cap \varepsilon^e(Y')^n \subseteq \varepsilon^e(X' \cap Y')^n. \tag{4.68}$$

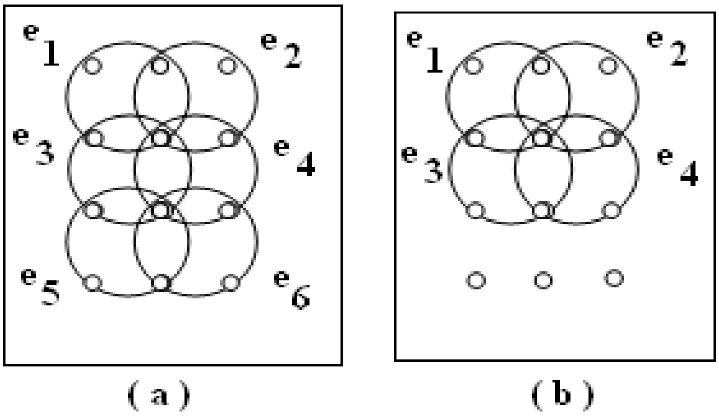


Figure 4.22: (a) H (b) Result of intersection of complement of erosion w.r.to edge

Eq(4.66) is implied by eq(4.67) and eq(4.68).

Example 4.20: Consider Table 4.2, Table 4.3 and Table 4.4. In L.H.S of eq(4.66), we get $\varepsilon^e(X' \cap Y')^n = \{e_1, e_2, e_3, e_4\}$ which are those edges from H, which consists of nodes in $(X' \cap Y')$ only. Consider R.H.S of eq(4.66), where $\varepsilon^e(X')^n = \{e_1, e_2, e_3, e_4\}$. $\varepsilon^e(Y')^n = \{e_1, e_2, e_3, e_4, e_5\}$. Now $\varepsilon^e(X')^n \cap \varepsilon^e(Y')^n = \{e_1, e_2, e_3, e_4\}$. The results are shown in Fig. 4.22(b).

Proposition 4.21: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^n(X' \cup Y')^e = \varepsilon^n(X')^e \cup \varepsilon^n(Y')^e, \tag{4.69}$$

where $\varepsilon^n(X' \cup Y')^e$ is an erosion w.r.to node which retrieves nodes not present in $(X' \cup Y')^{e'}$ and X', Y' are defined as in eq(4.1).

Proof: Consider L.H.S of eq(4.69). Let v be an arbitrary node in $\varepsilon^n(X' \cup Y')^e$. Now v is not a node of $(X' \cup Y')^{e'}$. i.e.,

$$v \in (X' \cup Y')^e. \tag{4.70}$$

and

$$v \notin (X' \cup Y')^{e'}. \tag{4.71}$$

Consider R.H.S of eq(4.69), where v is a node of $\varepsilon^n(X')^e \cup \varepsilon^n(Y')^e$. Then it is a node of $(X')^e$ or a node of $(Y')^e$, but not a node of X^e and also not a node of Y^e. i.e.,

$$v \in (X')^e. \tag{4.72}$$

i.e.,

$$v \notin (X)^e, \tag{4.73}$$

or

$$v \in (Y')^e. \tag{4.74}$$

i.e.,

$$v \notin (Y)^e. \tag{4.75}$$

Eq(4.70) is implied by eq(4.72) and eq(4.74). Eq(4.71) is implied by eq(4.73) and eq(4.75). Eq(4.69) is implied by eq(4.70) to eq(4.75).

Example 4.21: Consider Table 4.2, Table 4.3 and Table 4.4. Consider L.H.S of eq(4.69). $\varepsilon^n(X' \cup Y')^e$ is the set of all nodes in $X' \cup Y'$ which do not belong to any edge in $(X' \cup Y')^{e'}$. Here edges in $(X' \cup Y')^{e'} = \{e_6\}$. Now nodes in $X' \cup Y'$ which do not belong to $\{e_6\}$ is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{10}\}$. Let us consider R.H.S of eq(4.69), where $\varepsilon^n(X')^e$ is the set of all nodes which do not belong to any of the edge in $X'^{e'}$ which is the set $\{n_1, n_2, n_3, n_4, n_5, n_6\}$. Also $\varepsilon^n(Y')^e$ is the set of all nodes which do not belong to any of the edge in $Y'^{e'}$, which is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{10}\}$. Now $\varepsilon^n(X')^e \cup \varepsilon^n(Y')^e$ is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{10}\}$. The result is shown in Fig. 4.23(b).

Proposition 4.22: Let H_{IF} be the parent IFHG. Let X and Y be the sub-IFHGs and ε be the erosion operator, then

$$\varepsilon^n(X' \cap Y')^e = \varepsilon^n(X')^e \cap \varepsilon^n(Y')^e, \tag{4.76}$$

where $\varepsilon^n(X' \cap Y')^e$ is an erosion w.r.to node which retrieves nodes that are not present in $(X' \cap Y')^{e'}$.

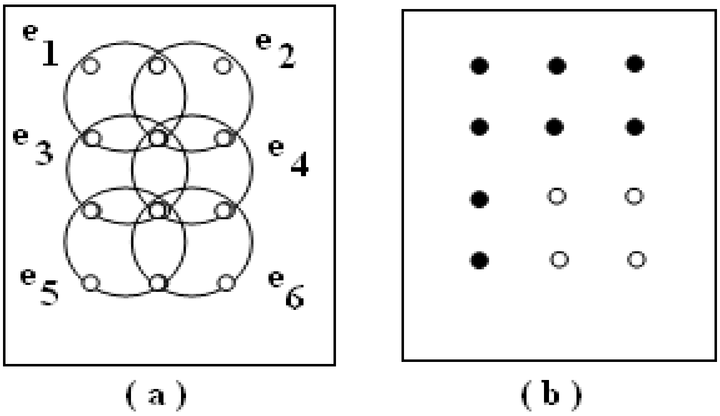


FIGURE 4.23: (a) H (b) Result of union of complement of erosion w.r.to node

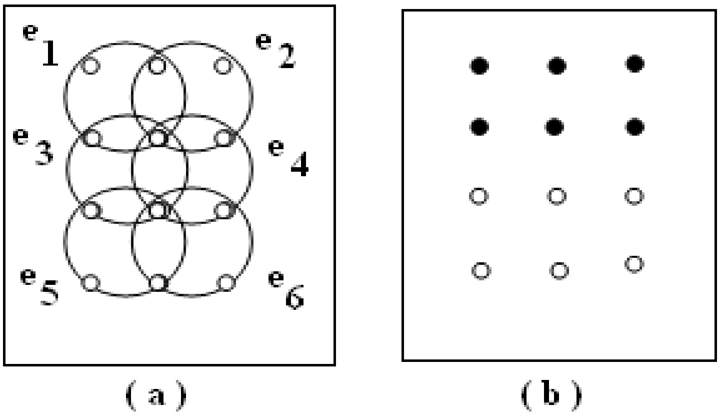


FIGURE 4.24: (a) H (b) Result of intersection of complement of erosion w.r.to node

Here X' and Y' are defined as per eq(4.1).

Also if $X' \cap Y' = \phi$, then $\varepsilon^n(X' \cap Y')^e = \varepsilon^n(X')^e \cap \varepsilon^n(Y')^e = \phi$.
Proof: Consider L.H.S of eq(4.76). Let v be an arbitrary node in $\varepsilon^n(X' \cap Y')^e$. Now v is not a node of $(X' \cap Y')^{e'}$. i.e.,

$$v \in (X' \cap Y')^e. \tag{4.77}$$

implies

$$v \notin (X' \cap Y')^{e'}. \tag{4.78}$$

Consider R.H.S of eq(4.76), where v is a node of $\varepsilon^n(X')^e \cap \varepsilon^n(Y')^e$. Then it is a node of $(X')^e$ and a node of $(Y')^e$, but not a node of X^e and also not a node of Y^e. i.e.,

$$v \in (X')^e; v \in (Y')^e, \tag{4.79}$$

which implies

$$v \notin (X)^e; v \notin (Y)^e. \tag{4.80}$$

Eq(4.77) is implied by eq(4.79). Eq(4.78) is implied by eq(4.80).

Example 4.22: Consider Table 4.2, Table 4.3 and Table 4.4. Consider L.H.S of eq(4.77). $\varepsilon^n(X' \cap Y')^e$ is the set of all nodes in $X' \cap Y'$ which are not in any of the edges in $(X' \cap Y')^{e'}$. Edges in $(X' \cap Y')^{e'} = \{e_5, e_6\}$. Now nodes which are not in $\{e_5, e_6\}$ is the set $\{n_1, n_2, n_3, n_4, n_5, n_6\}$. Let us consider R.H.S of eq(4.77), where $\varepsilon^n(X')^e$ = nodes which are not in any of the edges of $X'^{e'} = \{n_1, n_2, n_3, n_4, n_5, n_6\}$. Now $\varepsilon^n(Y')^e$ = Set of all nodes which are not in $Y'^{e'}$ which is the set $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_{10}\}$. Now we get $\varepsilon^n(X')^e \cap \varepsilon^n(Y')^e = \{n_1, n_2, n_3, n_4, n_5, n_6\}$. The result is shown in Fig. 4.24(b).

Proposition 4.23: Let H_{IF} be the parent IFHG, $X_1, X_2, ..., X_n$ be the sub-IFHGs of H and ε be the erosion operator, then the following are true for the union of subgraphs.

$$\varepsilon^e(X_1 \cup X_2 .... \cup X_n)^n = \varepsilon^e(X_1)^n \cup \varepsilon^e(X_2)^n ...... \cup \varepsilon^e(X_n)^n, \tag{4.81}$$

$$\varepsilon^n(X_1 \cup X_2 .... \cup X_n)^e = \varepsilon^n(X_1)^e \cup \varepsilon^n(X_2)^e ...... \cup \varepsilon^n(X_n)^e, \tag{4.82}$$

$$\varepsilon^e(X_1' \cup X_2' .... \cup X_n')^n = \varepsilon^e(X_1')^n \cup \varepsilon^e(X_2')^n ...... \cup \varepsilon^e(X_n')^n \tag{4.83}$$

and

$$\varepsilon^n(X_1' \cup X_2' .... \cup X_n')^e = \varepsilon^n(X_1')^e \cup \varepsilon^n(X_2')^e ...... \cup \varepsilon^n(X_n')^e, \tag{4.84}$$

where X_i' is defined as in eq(4.1).

Proposition 4.24: Let H_{IF} be the parent IFHG, $X_1, X_2, ..., X_n$ be the sub-IFHGs of H_{IF} and ε be the erosion operator, then the following are true for the intersection of subgraphs.

$$\varepsilon^e(X_1 \cap X_2 .... \cap X_n)^n = \varepsilon^e(X_1)^n \cap \varepsilon^e(X_2)^n ...... \cap \varepsilon^e(X_n)^n, \tag{4.85}$$

$$\varepsilon^n(X_1 \cap X_2 .... \cap X_n)^e = \varepsilon^n(X_1)^e \cap \varepsilon^n(X_2)^e ...... \cap \varepsilon^n(X_n)^e, \tag{4.86}$$

$$\varepsilon^e(X_1' \cap X_2' .... \cap X_n')^n = \varepsilon^e(X_1')^n \cap \varepsilon^e(X_2')^n ...... \cap \varepsilon^e(X_n')^n \tag{4.87}$$

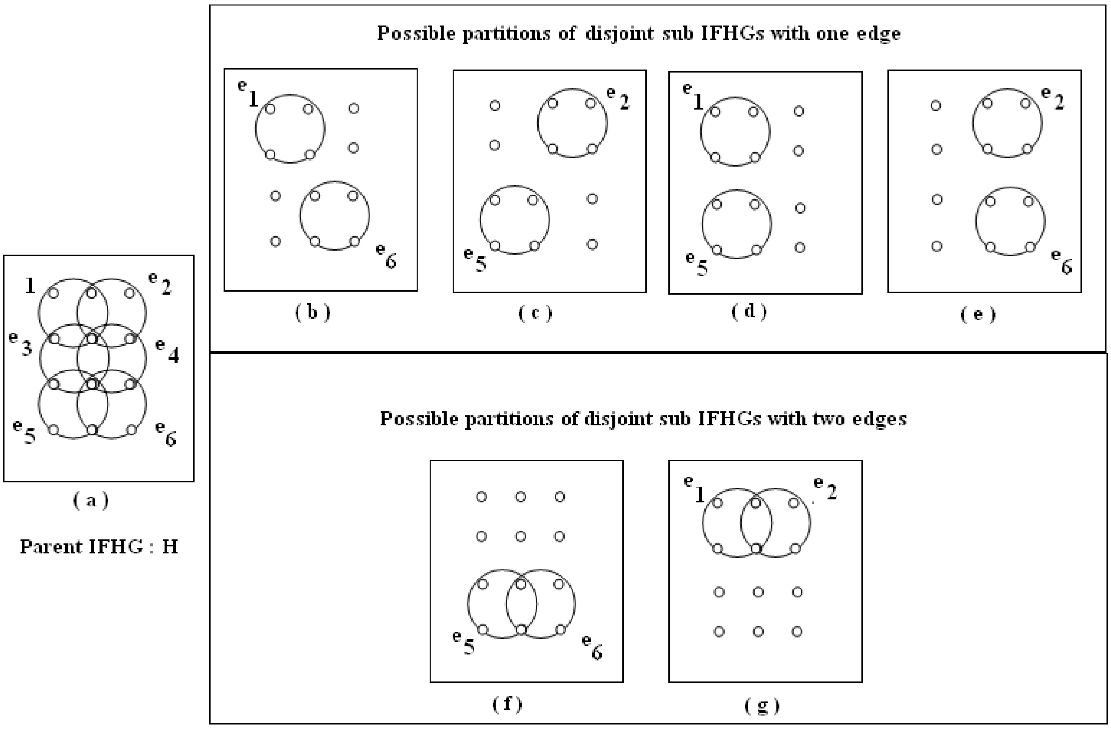
and

$$\varepsilon^n(X_1' \cap X_2' .... \cap X_n')^e = \varepsilon^n(X_1')^e \cap \varepsilon^n(X_2')^e ...... \cap \varepsilon^n(X_n')^e, \tag{4.88}$$

where $X_1 \cap X_2 .... \cap X_n \notin \phi$; $X_1' \cap X_2' .... \cap X_n' \notin \phi$ and X_i' is defined as in eq(4.1).

4.11 Partitioning of intuitionistic fuzzy hypergraph

The above erosion and dilation can be combined to perform partitioning of IFHG in to disjoint sub-IFHGs. So let H be an IFHG. Then H can be partitioned in to disjoint IFHGs $X_1, X_2, ..., X_n$, where $X_1 \cap X_2 \cap .... \cap X_n = \phi$. The method is shown in algorithm 6. The algorithm accepts an IFHG H and creates its disjoint partitions. Firstly create a sub-IFHG X_1. Create X_1' as $H - X_1$. From this X_1', we take a sub-IFHG X_2, such that, there are no common edges in the intersection of $\delta^e(X_1)^n$ and $\varepsilon^e(X_2)^n$. Now X_2' is created as $H - [X_1 \cup X_2]$. Now create $X_3 \subset X_2'$ such that there are no common edges in $\delta^e(X_1 \cup X_2)^n$ and $\varepsilon^e(X_3)^n$. This process is continued until no more partitions are possible from H. The results of this partitioning algorithm for IFHG with 4 nodes per hyperedge are shown in Table 4.5. Such a case can happen in a software firm, where 4 members are allotted to each project. A project can be modeled as a hyperedge and team members as nodes. A member may be working in several teams to have maximum projects done with less resource persons. As seen in Table 4.5, in case: 1, only up to disjoint partitions of size e = 2 are possible. In case: 2, a maximum of size


Possible partitions of disjoint sub IFHGs with one edge

Possible partitions of disjoint sub IFHGs with two edges

Parent IFHG : H

Figure 4.25: Case:1. Disjoint partitioning

e = 3 is possible. In case: 3, we can have two sub-IFHGs of size e= 4. In case: 4, sub-IFHGs of size e = 5 are possible. In case: 5, up to size e = 9 is possible. Case: 1 is shown in Fig. 4.25. With respect to document IFHG, the partitions in Fig. 4.25 shows disjoint technical documents without same authors. Disjoint two document sets are also shown.

Algorithm 6: Intuitionistic fuzzy hypergraph partitioning

```
Input : Parent IFHG.
Output : Partitions.
$H$ = Intuitionistic Fuzzy Hypergraph.
Create $X_1 \subset H$.
Create $X_1' = H - X_1$.
Create $X_2 \subset X_1'$ such that $\delta^e(X_1)^n \cap \varepsilon^e(X_2)^n = \phi$;where $\delta$ is the dilation
    operator and $\varepsilon$ is the erosion operator.
Create $X_2' = H - [X_1 \cup X_2]$.
Create $X_3 \subset X_2'$ such that $\delta^e(X_1 \cup X_2)^n \cap \varepsilon^e(X_3)^n = \phi$.
Create $X_n$ such that $\delta^e(X_1 \cup X_2 ..... \cup X_{n-1})^n \cap \varepsilon^e(X_n)^n = \phi$.
Partitions = $X_1, X_2, ......, X_n$.
```

Table 4.5: Analysis of IFHG partitioning

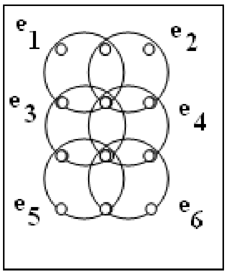
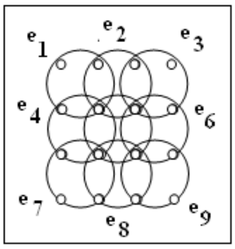
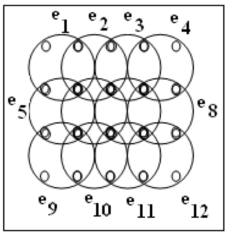
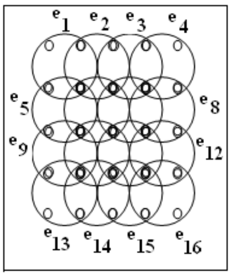
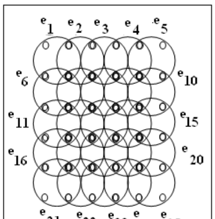
Figure	Details
	1. No of nodes = 12 2. No of hyperedges = 6 3. No of disjoint partitions(size e=1) = 2 4. No of disjoint partitions(size e=2) = 2 5. No of disjoint partitions (size e=3) = nil
 e_1 e_2 e_3 e_4 e_6 e_7 e_8 e_9	1. No of nodes = 16 2. No of hyperedges = 9 3. No of disjoint partitions (size e=1)= 4 4. No of disjoint partitions (size e=2)= 2 5. No of disjoint partitions(size e=3)= 2 6. No of disjoint partitions (size e=4)= nil
 e_1 e_2 e_3 e_4 e_5 e_8 e_9 e_{10} e_{11} e_{12}	1. No of nodes = 20, No of hyperedges = 12 2. No of disjoint partitions (size e=1)= 4, (size e=2)= 3 3. No of disjoint partitions(size e=3)= 2, (size e=4)= 2 4. No of disjoint partitions (size e=5)= nil
 e_1 e_2 e_3 e_4 e_5 e_8 e_9 e_{12} e_{13} e_{14} e_{15} e_{16}	1. No of nodes = 25, No of hyperedges = 16 2. No of disjoint partitions (size e=1)= 4, (size e=2)= 3 3. No of disjoint partitions(size e=3)= 2, (size e=4)= 2 4. No of disjoint partitions (size e=5)= 2, (size e=6)= nil
 e_1 e_2 e_3 e_4 e_5 e_6 e_{10} e_{11} e_{15} e_{16} e_{20} e_{21} e_{22} e_{23} e_{24} e_{25}	1. No of nodes = 30,No of hyperedges = 25 2. No of disjoint partitions (size e=1)= 9,(size e=2)= 5 3. No of disjoint partitions(size e=3)= 4,(size e=4)= 4 4. No of disjoint partitions (size e=5)= 3,(size e=6)= 2 5. No of disjoint partitions (size e=7)= 2,(size e=8)= 2 6. No of disjoint partitions (size e=9)= 2,(size e=10)= nil

Table 4.6: Applications of erosion of IFHG

Technical document processing using IFHG		
Notation	Operation	Result
$\varepsilon^n(X_1)^e$	Erosion w.r.to nodes	Retrieve all the high profile authors, who are not part of low quality documents.
$\varepsilon^e(X_1)^n$	Erosion w.r.to hyperedges	Retrieval of documents with only high profile authors.
$\varepsilon^n(X_1')^e$	Erosion w.r.to nodes	Retrieve all authors in low quality documents, who are not part of any high quality documents
$\varepsilon^e(X_1')^n$	Erosion w.r.to hyperedge	Retrieve all low quality documents

4.12 Metric-induced morphological operators on intuitionistic fuzzy hypergraphs

Preliminary Definitions

Let us define $H_{IF} = [H^n, H^e]$, where H^n is the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{16}\}$ and H^e is the set of hyperedges $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$ as given in Fig. 4.26. Here nodes with low priority are having $\mu_n < 0.5$, nodes of medium priority are having $\mu_n = 0.5$ and nodes of high priority are with $\mu_n > 0.5$. Let X_{IF} be obtained by (α, β) cut on $H_{IF}/0.5 < \alpha \leq 0.7; \{\beta \leq 1 - \alpha\} \cap \{\beta \leq 0.3\}$. Let Y_{IF} be obtained by (α, β) cut on $H_{IF}/\alpha \geq 0.7; \{\beta \leq 1 - \alpha\} \cap \{\beta \leq 0.3\}$. Here α corresponds to membership degree and β corresponds to non-membership degree. The details of the IFHGs H_{IF}, X_{IF} and Y_{IF} are given in Table 4.7, Table 4.8, Table 4.9 respectively.

4.13 Adjunction of IFHG

The adjunctions that we are going to state here are already defined on hypergraphs in [104][105]. We are extending these adjunctions to IFHG.

Proposition 4.25: Let H be an intuitionistic fuzzy hypergraph, let X, Y be the sub-IFHGs, ε be the erosion operator and δ be the dilation operator. We observe

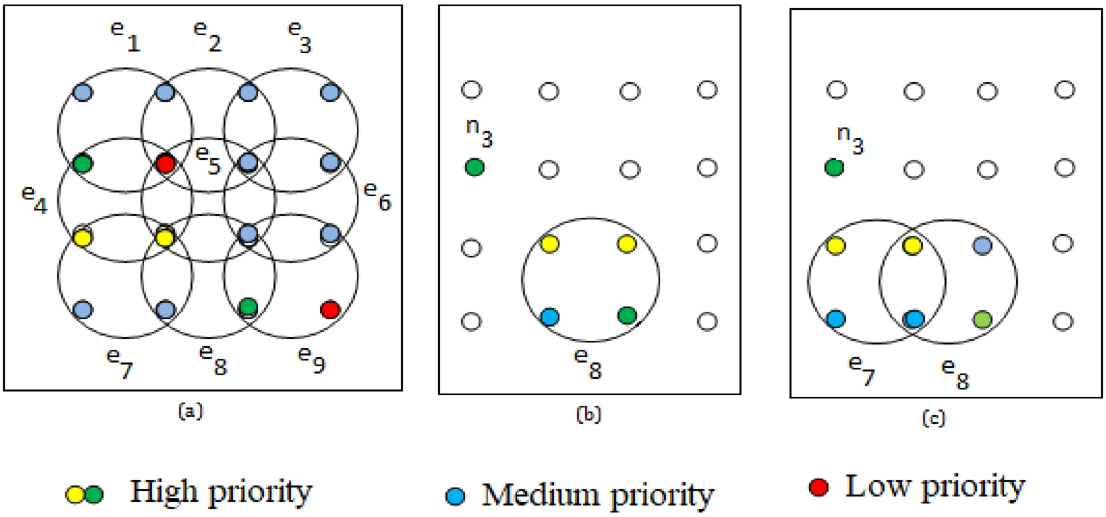


FIGURE 4.26: Intuitionistic fuzzy hypergraphs(a):H, (b): X, (c) :Y

TABLE 4.7: Details of hypergraph H_{IF}

Hyperedges	Nodes				Edge priority
e_1 (0.3, 0.7)	n_1 (0.5, 0.5)	n_2 (0.5, 0.5)	n_3 (0.7, 0.3)	n_4 (0.3, 0.7)	Low
e_2 (0.3, 0.7)	n_2 (0.5, 0.5)	n_4 (0.3, 0.7)	n_5 (0.5, 0.5)	n_7 (0.5, 0.5)	Low
e_3 (0.5, 0.5)	n_5 (0.5, 0.5)	n_6 (0.5, 0.5)	n_7 (0.5, 0.5)	n_8 (0.5, 0.5)	Medium
e_4 (0.3, 0.7)	n_3 (0.7, 0.3)	n_4 (0.3, 0.7)	n_9 (0.6, 0.4)	n_{10} (0.6, 0.4)	Low
e_5 (0.3, 0.7)	n_4 (0.3, 0.7)	n_7 (0.5, 0.5)	n_{10} (0.6, 0.4)	n_{11} (0.5, 0.5)	Low
e_6 (0.5, 0.5)	n_7 (0.5, 0.5)	n_8 (0.5, 0.5)	n_{11} (0.5, 0.5)	n_{12} (0.5, 0.5)	Medium
e_7 (0.6, 0.4)	n_9 (0.6, 0.4)	n_{10} (0.6, 0.4)	n_{13} (0.5, 0.5)	n_{14} (0.5, 0.5)	High
e_8 (0.7, 0.3)	n_{10} (0.6, 0.4)	n_{11} (0.5, 0.5)	n_{14} (0.5, 0.5)	n_{15} (0.7, 0.3)	High
e_9 (0.4, 0.6)	n_{11} (0.5, 0.5)	n_{12} (0.5, 0.5)	n_{15} (0.7, 0.3)	n_{16} (0.4, 0.6)	Low

TABLE 4.8: Details of hypergraph X_{IF}

Hyperedges	Nodes				Edge priority
e_8 (0.7, 0.3)	n_{10} (0.6, 0.4)	n_{11} (0.5, 0.5)	n_{14} (0.5, 0.5)	n_{15} (0.7, 0.3)	High
Hyperedges	Nodes				Node priority
Nil	n_3 (0.7, 0.3)				High

TABLE 4.9: Details of hypergraph Y_{IF}

Hyperedges	Nodes				Edge priority
e_7 (0.6, 0.4)	n_9 (0.6, 0.4)	n_{10} (0.6, 0.4)	n_{13} (0.5, 0.5)	n_{14} (0.5, 0.5)	High
e_8 (0.7, 0.3)	n_{10} (0.6, 0.4)	n_{11} (0.5, 0.5)	n_{14} (0.5, 0.5)	n_{15} (0.7, 0.3)	High
Hyperedges	Nodes				Node priority
Nil	n_3 (0.7, 0.3)				High

that $(\varepsilon^e, \delta^n)$ are adjunctions if

$$X^e \subseteq \varepsilon^e(Y^n) \tag{4.89}$$

and

$$\delta^n(X^e) \subseteq Y^n; X \subseteq Y. \tag{4.90}$$

Proof: Let us consider erosion operator ε^e, let e be an edge in X^e. i.e., $e \subset X^e$. We know that $\varepsilon^e(Y^n) = Y^e$. Since $X \subset Y$, we get $e \subset X^e \subset Y^e$. Therefore $X^e \subseteq \varepsilon^e(Y^n)$. This edge is a priority edge in H. Now let us consider dilation operator δ^n. Let v be a node in $\delta^n(X^e)$, i.e., $v \subset X^n$. Since $X \subseteq Y$, $v \subset Y^n$. Therefore $v \subseteq X^n \subseteq Y^n$. Therefore $\delta^n(X^e) \subseteq Y^n$. This node v is definitely a priority node of H.

Illustration: Let us check the results on IFHG by considering the H, X and Y IFHGs. Here, In R.H.S of eq(4.89), $\varepsilon^e(Y^n)$ means the set of edges in H, which consists of nodes in Y only. i.e., $\varepsilon^e(Y^n) = \{e_7, e_8\}$. This operation returns the high priority edges in H. Now we know that X^e as the hyperedges in X, i.e., $X^e = \{e_8\}$. Therefore $X^e \subseteq \varepsilon^e(Y^n)$. Now in L.H.S of eq(4.90), find $\delta^n(X^e)$ which is the set of nodes in edges of X. i.e., $\delta^n(X^e) = \{n_{10}, n_{11}, n_{14}, n_{15}\}$. This operation

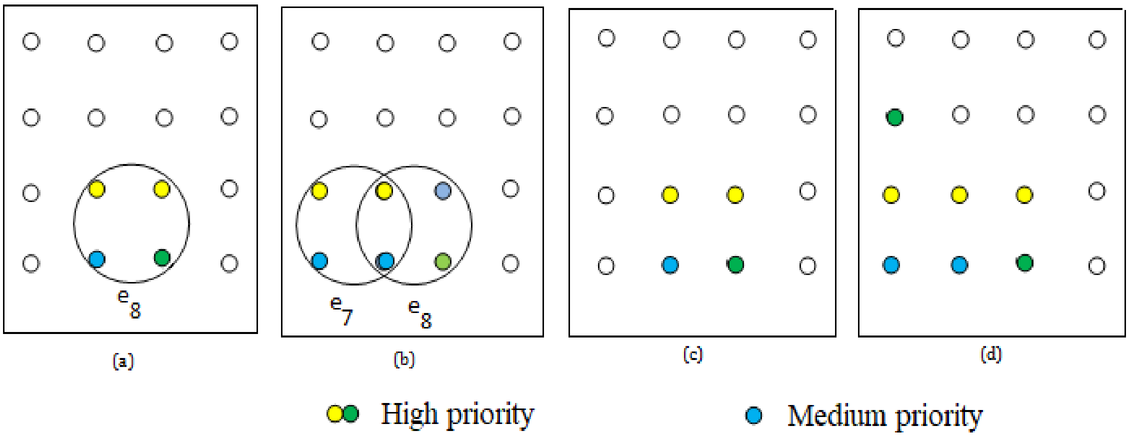


FIGURE 4.27: Results of adjunction(a):X^e, (b):$\varepsilon^e(Y^n)$, (c) :$\delta^n(X^e)$ (d) :Y^n

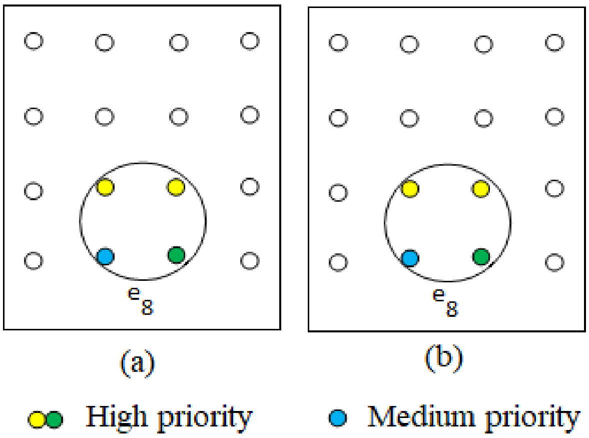


FIGURE 4.28: Complement results of adjunction(a):$(\delta^e(X^{n'}))'$, (b):$\varepsilon^e(X^n)$

returns priority nodes in H which are part of X. We get Y^n as the nodes in Y. i.e., $Y^n = \{n_3, n_9, n_{10}, n_{11}, n_{13}, n_{14}, n_{15}\}$. We find that $\delta^n(X^e) \subset Y^n$. Therefore $(\varepsilon^e, \delta^n)$ are adjunctions and the results are shown in Fig. 4.27.

Proposition 4.26: Let H be the intuitionistic fuzzy hypergraph, let X, Y be the sub-IFHGs, ε be the erosion operator and δ be the dilation operator. We observe that, If $(\varepsilon^e, \delta^n)$ are adjunctions, then

$$(\delta^e(X^{n'}))' = \varepsilon^e(X^n) \tag{4.91}$$

and

$$(\delta^n(X^{e'}))' = \varepsilon^n(X^e). \tag{4.92}$$

Proof: Let e be an edge in $(\delta^e(X^{n'}))'$. i.e.,

$$e \in (\delta^e(X^{n'}))', \tag{4.93}$$

then

$$e \notin \delta^e(X^{n'}); e \notin X'; \tag{4.94}$$

i.e.,

$$e \in X. \tag{4.95}$$

Let us consider R.H.S of eq(4.91). Let e be an edge of $\varepsilon^e(X^n)$. i.e., $e \in \varepsilon^e(X^n)$. i.e.,

$$e \in X. \tag{4.96}$$

Eq(4.91) is implied by eq(4.95) and eq(4.96). The edge e is a priority edge present in X.

Consider L.H.S of eq(4.92). Let v be a node in $(\delta^n(X^{e'}))'$ i.e., $v \in (\delta^n(X^{e'}))'$ i.e., $v \notin (\delta^n(X^{e'}))$. So we can write $v \notin X^{e'}$ or

$$v \in X^e. \tag{4.97}$$

Now consider R.H.S of eq(4.92). Let v be a node of $\varepsilon^n(X^e)$; i.e., $v \in \varepsilon^n(X^e)$. We get

$$v \in X^e. \tag{4.98}$$

Eq(4.92) is implied by eq(4.97) and eq(4.98).

Illustration: In L.H.S of eq(4.91), $\delta^e(X^{n'})$ is the set of edges in H, which consists of any node $X^{n'}$. We know that $X^{n'} = \{n_1, n_2, n_4, n_5, n_6, n_7, n_8, n_9, n_{12}, n_{13}, n_{16}\}$. Thus $\delta^e(X^{n'}) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_9\}$. Therefore $(\delta^n(X^{n'}))' = \{e_8\} = \varepsilon^e(X^n)$. Both these operations return priority edges in X and the same are shown in Fig. 4.28(a) and Fig. 4.28(b) respectively. Now consider L.H.S of eq(4.92), $\delta^n(X^{e'})$ which is the set of nodes in $X^{e'}$. i.e., $X^{e'} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_9\}$. Now $\delta^n(X^{e'})$ is the set of nodes $\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{16}\}$. Now $(\delta^n(X^{e'}))' = \phi$. We get $\varepsilon^n(X^e)$ as the set of nodes in X which are not in $X^{e'}$. i.e., $\varepsilon^n(X^e) = \phi$.

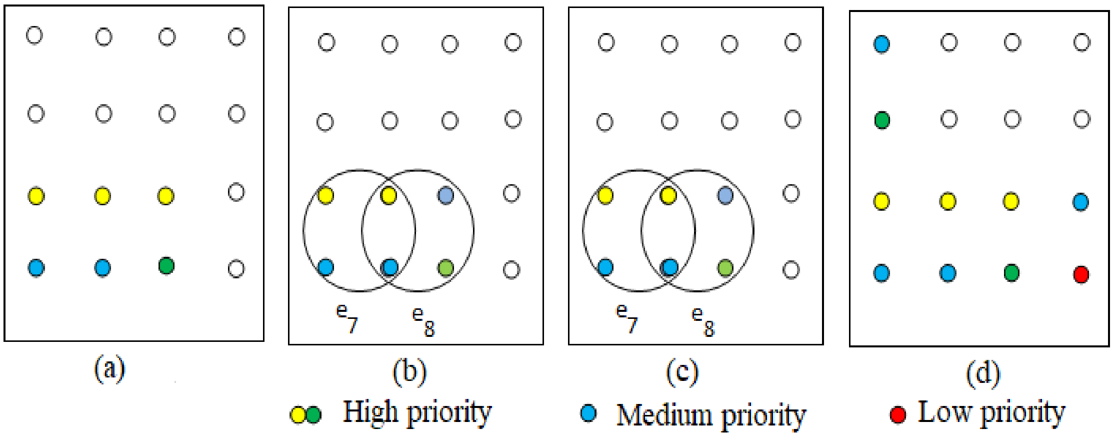


FIGURE 4.29: (a):$\delta^n(\varepsilon^e(Y^n))$ (b): $\delta^e(\varepsilon^n(Y^e))$ (c):$\varepsilon^e(\delta^n(Y^e))$ (d):$\varepsilon^n(\delta^e(Y^n))$

4.14 Materials and methods - construction of various intuitionistic fuzzy hypergraph filters

A filter is something which gives the same result if a function is repeatedly applied to it. Consider a water/sand filter where the filtrate on repeated passage through the same filter gives the same filtrate. Similarly in the case of an IFHG, a filter applied on a sub-IFHG should produce the same set of edges and nodes even if it is filtered many times. If δ is the dilation operator and ε is the erosion operator, $\gamma = \delta \circ \varepsilon$ is an opening filter and $\phi = \varepsilon \circ \delta$ is a closing filter.

- **Half opening filter w.r.to nodes -** $\delta^n(\varepsilon^e(Y^n))$

  If H is the parent IFHG, Y is the sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\gamma_{1/2} = \delta^n(\varepsilon^e(Y^n))$ is a half opening filter with respect to the nodes in Y. Here $\varepsilon^e(Y^n)$ is the set of edges in H which consists of Y^n only. i.e., $\varepsilon^e(Y^n) = \{e_7, e_8\}$. Now $\delta^n(\varepsilon^e(Y^n))$ is the set of nodes within those edges. i.e., $\delta^n(\varepsilon^e(Y^n)) = \{n_9, n_{10}, n_{11}, n_{13}, n_{14}, n_{15}\}$. This will retrieve all nodes within all edges in Y. To this result if we apply half opening again, it will retrieve the same set of nodes. Thus we can prove that half opening $\gamma_{1/2} = \delta^n(\varepsilon^e(Y^n))$ is a filter. Here only a part of Y is retrieved as shown in Fig. 4.29(a).

- **Half opening filter w.r.to hyperedges -** $\delta^e(\varepsilon^n(Y^e))$

  If H is the parent IFHG, Y is the sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\gamma_{1/2} = \delta^e(\varepsilon^n(Y^e))$ is a half opening filter with respect to the hyperedges in Y. Here $\varepsilon^n(Y^e)$ is the set of all nodes in Y but not in $Y^{e'}$. i.e., $\varepsilon^n(Y^e) = \{n_{13}, n_{14}\}$. Now $\delta^e(\varepsilon^n(Y^e))$ is the set of all hyperedges in H which consists of such nodes. i.e., $\delta^e(\varepsilon^n(Y^e)) = \{e_7, e_8\}$. Here $\{e_7, e_8\}$ is the filtrate obtained. If we repeatedly apply $\delta^e \circ \varepsilon^n$ to this filtrate, we get the same results. Thus this half opening is a filter as shown in Fig. 4.29(b).

- **Half closing filter w.r.to hyperedges -** $\varepsilon^e(\delta^n(Y^e))$

  If H is the parent IFHG, Y is the sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\phi_{1/2} = \varepsilon^e(\delta^n(Y^e))$ is a half closing filter with respect to the hyperedges in Y. Here $\delta^n(Y^e)$ is the set of nodes within the hyperedges of Y. i.e., $\delta^n(Y^e) = \{n_9, n_{10}, n_{11}, n_{13}, n_{14}, n_{15}\}$. Now $\varepsilon^e(\delta^n(Y^e))$ is the set of all edges in H which consists of the above nodes only. i.e., $\varepsilon^e(\delta^n(Y^e)) = \{e_7, e_8\}$. Here only a part of Y is retrieved as seen in Fig. 4.29(c).

- **Half closing filter w.r.to nodes -** $\varepsilon^n(\delta^e(Y^n))$

  If H is the parent IFHG, Y is the sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\phi_{1/2} = \varepsilon^n(\delta^e(Y^n))$ is a half closing filter with respect to the nodes in Y. Here $\delta^e(Y^n)$ is the set of all edges in H which has nodes in Y. i.e., $\delta^e(Y^n) = \{e_1, e_4, e_5, e_6, e_7, e_8, e_9\}$. Now $\varepsilon^n(\delta^e(Y^n))$ is the nodes not in $(\delta^e(Y^n))'$. From the given example, $(\delta^e(Y^n))' = \{e_2, e_3\}$. Now $\varepsilon^n(\delta^e(Y^n)) = \{n_1, n_3, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{16}\}$.
  The result is shown in Fig. 4.29(d).

4.15 Metric induced opening and closing filters

We can consider γ_λ as a metric induced opening where λ is a natural number which shows the number of edges/nodes to be included in the retrieved sub-IFHG after opening operation. i.e., $\gamma_\lambda = (\delta \circ \varepsilon)_\lambda$. Similarily ϕ_λ is a metric induced closing,

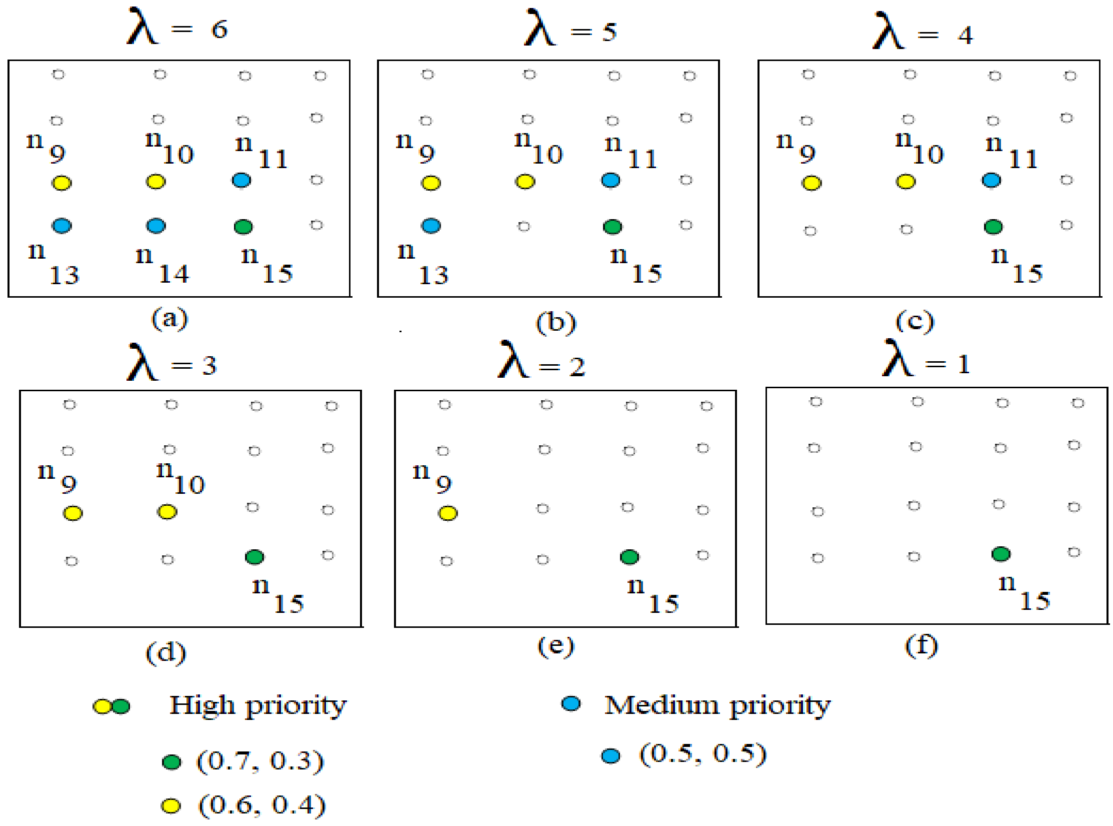


FIGURE 4.30: (a)-(f):Metric induced opening:$\gamma_\lambda = [\delta^n(\varepsilon^e(Y^n))]_\lambda$ for various λ

where λ is the number of edges/nodes to be included in the result after closing. Here λ should be from 1 to number of elements in γ. So let us see different flavours of γ_λ and ϕ_λ.

- **Metric induced opening (γ_λ) with respect to nodes**

  If H is a parent IFHG, Y is a sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\gamma_\lambda = [\delta^n(\varepsilon^e(Y^n))]_\lambda$ is a metric induced opening with respect to the nodes where top λ nodes with high membership degrees are selected. Here not all nodes in Y are retrieved. Only top priority nodes are retrieved. The results of this opening are shown in Fig. 4.30(a)-4.30(f). Here λ takes a maximum value of 6, since $\delta^n(\varepsilon^e(Y^n))$ returns a maximum of only 6 nodes with respect to IFHGs in Fig. 4.26.

- **Metric induced opening(γ_λ) with respect to hyperedges**

  If H is a parent IFHG, Y is a sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\gamma_\lambda = [\delta^e(\varepsilon^n(Y^e))]_\lambda$ is a metric induced opening

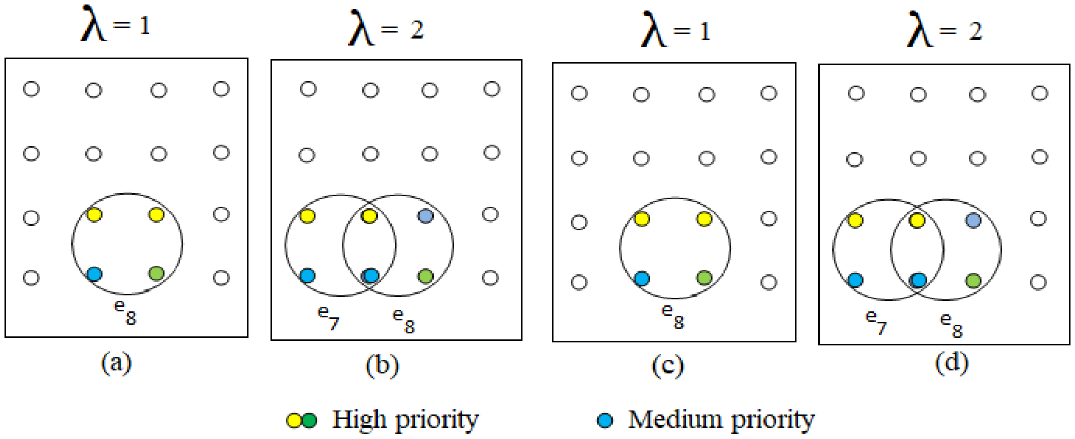


FIGURE 4.31: (a)-(b):Opening: $\gamma_\lambda = [\delta^e(\varepsilon^n(Y^e))]_\lambda$ 4.31(c)-(d):Closing: $\phi_\lambda = [\varepsilon^e(\delta^n(Y^e))]_\lambda$

with respect to the hyperedges where top λ edges with high membership degrees are selected. Here λ takes a maximum value of 2, since $\delta^e(\varepsilon^n(Y^e))$ returns only maximum of 2 edges with respect to IFHGs in Fig. 4.26. The results of this opening are shown in Fig. 4.31(a) and Fig. 4.31(b) for different values of λ.

- **Metric induced closing(ϕ_λ) with respect to hyperedges**

  If H is a parent IFHG, Y is a sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\phi_\lambda = [\varepsilon^e(\delta^n(Y^e))]_\lambda$ is a metric induced closing with respect to the hyperedges where top λ edges with high membership degrees are selected. Here λ takes a maximum value of 2, since $\varepsilon^e(\delta^n(Y^e))$ returns only 2 edges with respect to IFHGs in Fig. 4.26. The results of this closing operation are shown in Fig. 4.31(c) and Fig. 4.31(d) for different values of λ.

- **Metric induced closing(ϕ_λ) with respect to nodes**

  If H is a parent IFHG, Y is a sub-IFHG, δ is the dilation operator and ε is the erosion operator, then $\phi_\lambda = [\varepsilon^n(\delta^e(Y^n))]_\lambda$ is a metric induced closing with respect to nodes where top λ nodes from edges which contain Y^n and which do not belong to the complement edges are selected. Here λ takes a maximum value of 10, since $\varepsilon^n(\delta^e(Y^n))$ returns a maximum of 10 nodes with respect to IFHGs in Fig. 4.26. The results of this closing are shown in Fig. 4.32(a) - 4.32(j) for various values of λ.

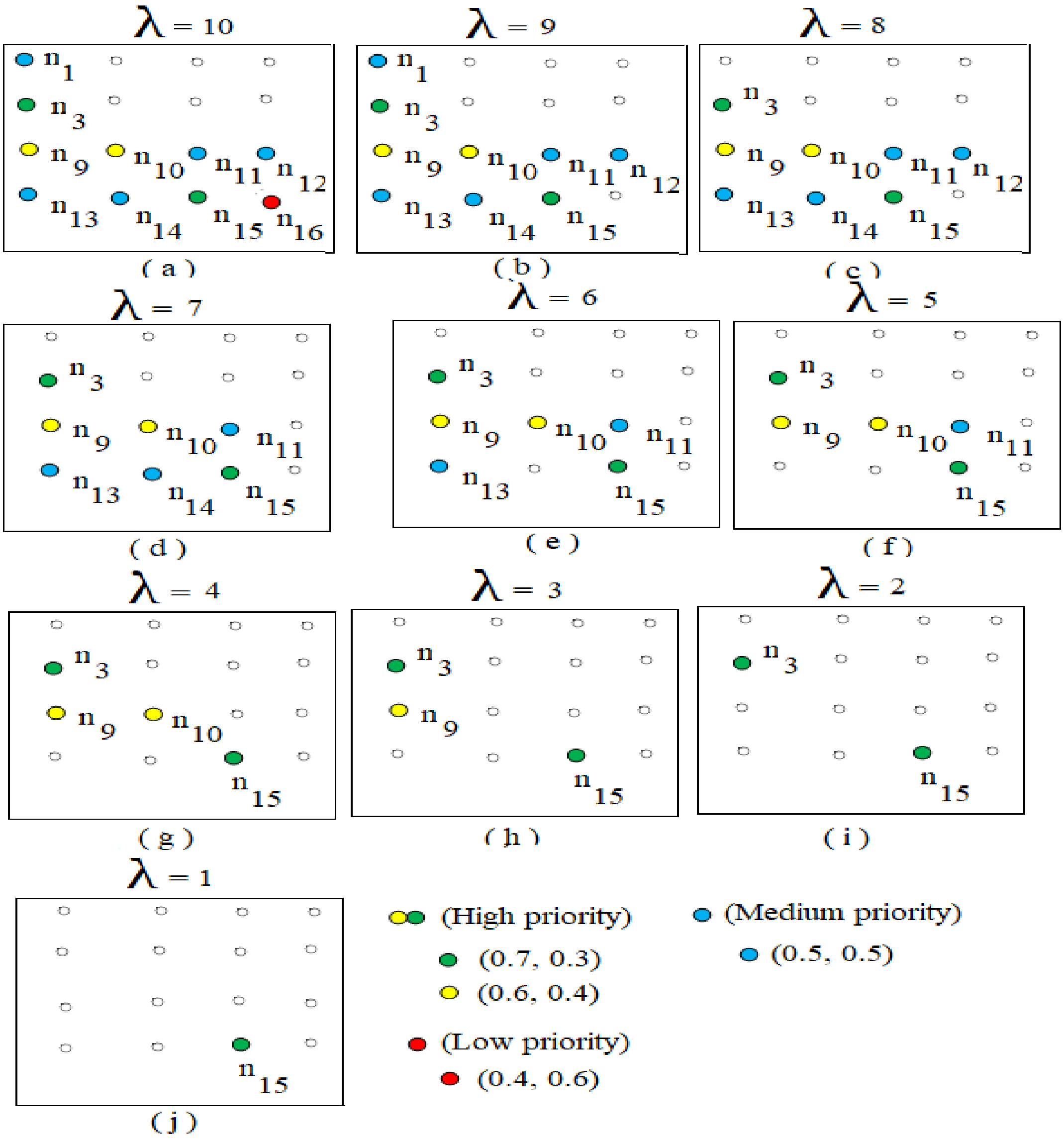
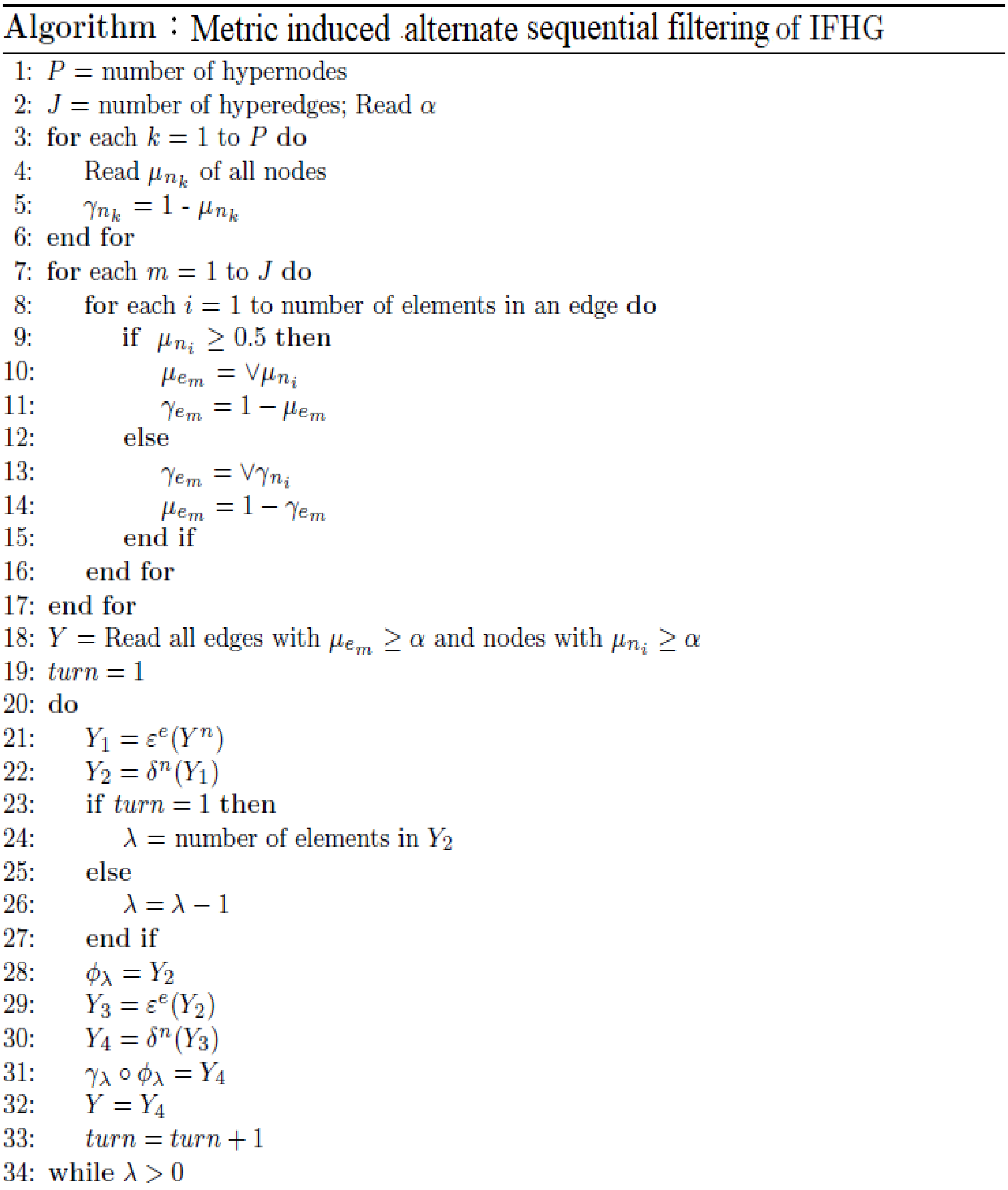


FIGURE 4.32: (a)-(j) : Metric induced closing $\phi_\lambda = [\varepsilon^n(\delta^e(Y^n))]_\lambda$

4.16 Alternate sequential filters

If H is a parent IFHG, Y is a sub-IFHG, γ_λ is an opening of the form $(\delta \circ \varepsilon)_\lambda$ and ϕ_λ is a closing operator of the form $(\varepsilon \circ \delta)_\lambda$, then $(\gamma_\lambda \circ \phi_\lambda)$ is also a filter. Now an alternate sequential filter can be obtained as $(\gamma_\lambda \circ \phi_\lambda) \circ (\gamma_\lambda \circ \phi_\lambda)$. The operations repeated n number of times will retrieve the same set of hyperedges/nodes for a particular value of λ, but we can have different results by varying the value of λ.

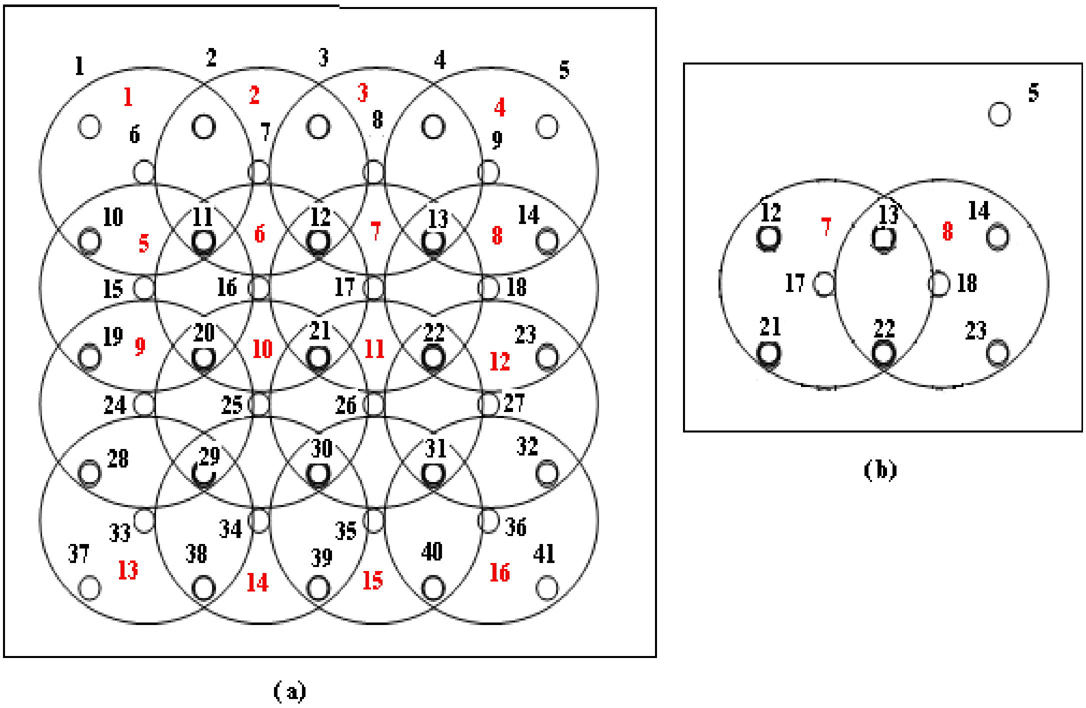
- **Illustration:** Consider H as a parent IFHG and Y as a sub-IFHG as shown in Fig. 4.34(a) and 4.34(b) respectively. Let us apply $(\gamma_\lambda \circ \phi_\lambda) \circ (\gamma_\lambda \circ \phi_\lambda)$ on these IFHGs. In Fig. 4.34, those marked in black



Algorithm : Metric induced alternate sequential filtering of IFHG

```
$P$ = number of hypernodes
$J$ = number of hyperedges; Read $\alpha$
for each $k = 1$ to $P$ do
    Read $\mu_{n_k}$ of all nodes
    $\gamma_{n_k} = 1$ - $\mu_{n_k}$
end for
for each $m = 1$ to $J$ do
    for each $i = 1$ to number of elements in an edge do
        if $\mu_{n_i} \geq 0.5$ then
            $\mu_{e_m} = \vee \mu_{n_i}$
            $\gamma_{e_m} = 1 - \mu_{e_m}$
        else
            $\gamma_{e_m} = \vee \gamma_{n_i}$
            $\mu_{e_m} = 1 - \gamma_{e_m}$
        end if
    end for
end for
$Y$ = Read all edges with $\mu_{e_m} \geq \alpha$ and nodes with $\mu_{n_i} \geq \alpha$
$turn = 1$
do
    $Y_1 = \varepsilon^e(Y^n)$
    $Y_2 = \delta^n(Y_1)$
    if $turn = 1$ then
        $\lambda$ = number of elements in $Y_2$
    else
        $\lambda = \lambda - 1$
    end if
    $\phi_\lambda = Y_2$
    $Y_3 = \varepsilon^e(Y_2)$
    $Y_4 = \delta^n(Y_3)$
    $\gamma_\lambda \circ \phi_\lambda = Y_4$
    $Y = Y_4$
    $turn = turn + 1$
while $\lambda > 0$
```

Figure 4.33: Algorithm for ASF



(a)

(b)

FIGURE 4.34: (a):H , (b):Y

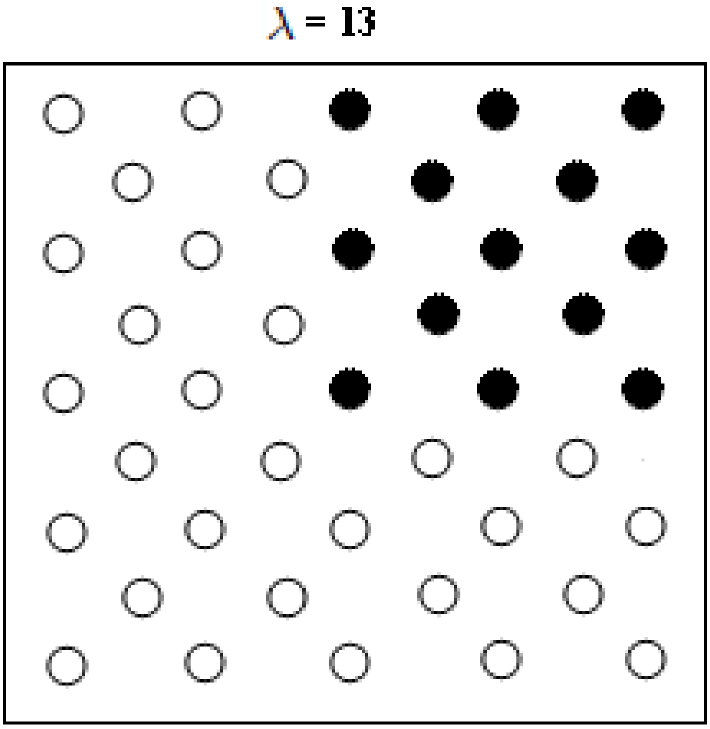


FIGURE 4.35: Node ASF $(\gamma_\lambda \circ \phi_\lambda) \circ (\gamma_\lambda \circ \phi_\lambda)$ for λ_{max}

represents the node numbers and those in red shows the edge numbers. Let $\phi_\lambda = [\varepsilon^n(\delta^e(Y^n))]_\lambda$. Since the value of λ is determined by the maximum nodes retrieved by $\varepsilon^n(\delta^e(Y^n))$, we get $\lambda = 18$ in ϕ_λ. We get $\delta^e(Y^n)$ as the set $\{e_2, e_3, e_4, e_6, e_7, e_8, e_{10}, e_{11}, e_{12}\}$. Now $[\varepsilon^n(\delta^e(Y^n))]_\lambda = \phi_\lambda$, which is the set $\{n_3, n_4, n_5, n_7, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{16}, n_{17}, n_{18}, n_{21}, n_{22}, n_{23}, n_{25}, n_{26}, n_{27}\}$. We know $(\gamma_\lambda \circ \phi_\lambda) = [\delta^n(\varepsilon^e(\phi_\lambda))]_\lambda$. We get $\varepsilon^e(\phi_\lambda) = \{e_3, e_4, e_7, e_8\}$. Now $[\delta^n(\varepsilon^e(\phi_\lambda))]_\lambda = \{n_3, n_4, n_5, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{17}, n_{18}, n_{21}, n_{22}, n_{23}\}$, where λ is 13. Applying $(\gamma_\lambda \circ \phi_\lambda)$ to these nodes will again retrieve the same set of nodes for the λ_{max} value. Different results can be obtained for $1 <= \lambda <= \lambda_{max}$ for which algorithm is shown above. The results of this ASF for λ_{max} value is shown in Fig. 4.29.

4.17 Applications

Modeling systems with intuitionistic fuzzy hypergraphs finds application in the field of Medical report processing, where a patient can be modeled as a hyperedge and the symptoms can be modeled as nodes. When multiple patients are having the same symptom, such a node forms part of multiple edges. In Fig. 4.36(a), symptom 5 is present in all the three patients. An IFHG constructed in this way can be subjected to many information retrieval operations. Membership and non membership values can be assigned to different nodes/symptoms based on the severity of the symptoms. Likewise membership and non-membership values can be assigned to patients following the rules given in section 4.12. IFHG modeling can be done in the area of social networking where a network group can be modeled as a hyperedge and the members/nodes of the network group can be converted to nodes. One member may be part of many network groups as shown in Fig. 4.36(b). They can be assigned different membership values based on their life/character background.

The systems modeled in this way can be subjected to various morphological operations like dilation, erosion, adjunction, opening, closing and filtering. An (α, β) cut can be applied on the Medical report IFHGs to find sub-IFHG X_1. Let us consider this sub-IFHG X_1 as the set of all patients with severe diseases and set of all severe symptoms. Let X_2 be the sub-IFHG of Fig. 4.36(b), which consists of all blacklisted groups and low priority members. The operations applied to this X_1 and X_2 are given in Table 4.10. All these operations can be

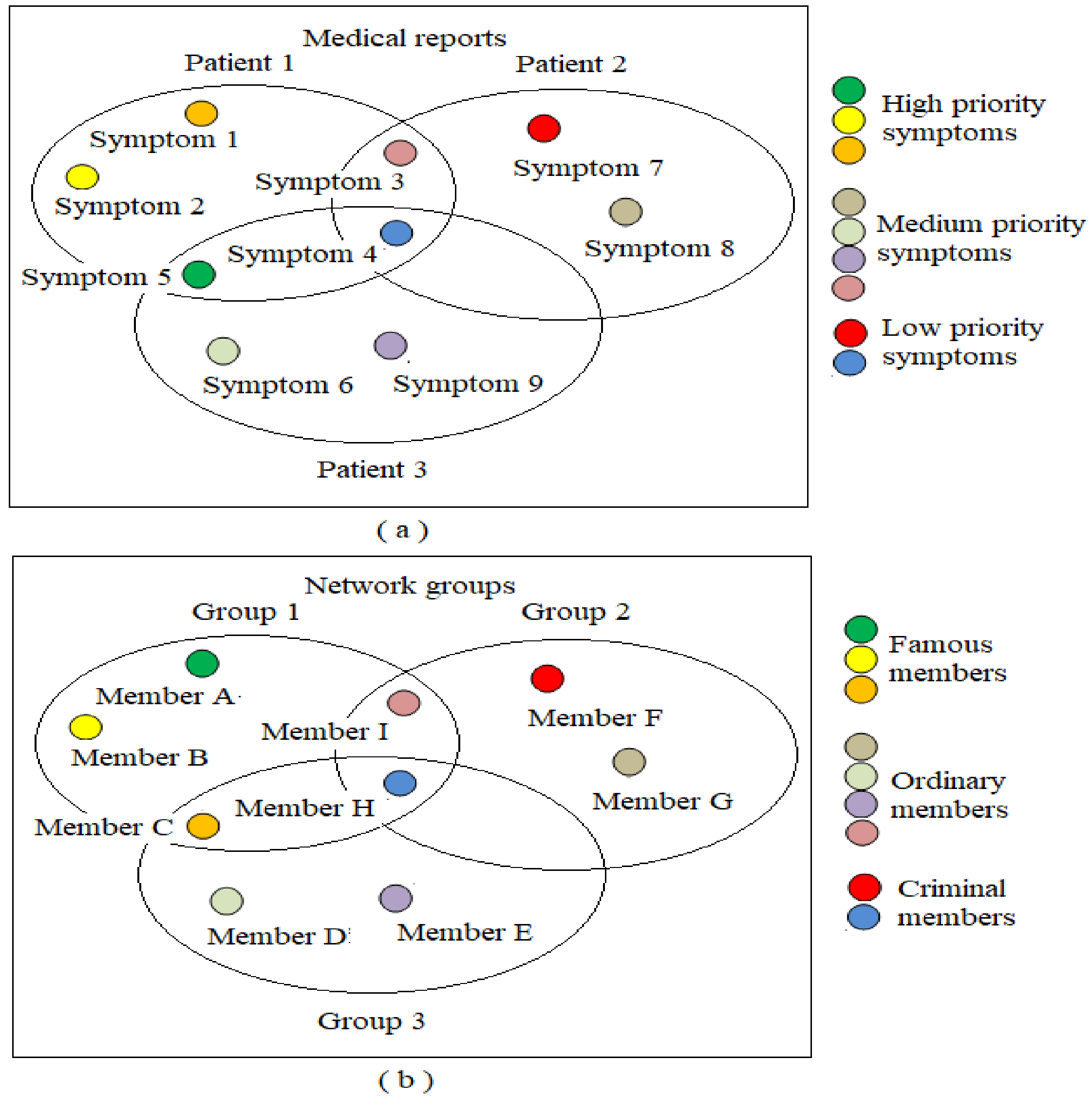


FIGURE 4.36: (a) : Medical report processing (b) : Social network analysis

further expanded to opening, closing, filtering etc. A detailed medical analysis of patients in a particular area can be done with such systems which opens a wide range of possibilities.

4.18 Data availablity, results and discussion

The filters mentioned in this work are tested on IFHGs consisting of maximum of 9,000 nodes. The method has shown 100% accurate results. The ASF algorithm designed on IFHG has a complexity of $O(n^2)$, since we are searching through the hyperedges and nodes within those hyperedges. The parameters α and β are

Table 4.10: Applications of IFHG

Medical report processing using IFHG		
Notation	Operation	Result
$\delta^n(X_1^e)$	Dilation w.r.to nodes	This operation will retrieve all the symptoms of patients with severe diseases.
$\delta^e(X_1^n)$	Dilation w.r.to hyperedges	This retrieves all the patients with atleast one symptom common with severely diseased patients.
$\varepsilon^n(X_1^e)$	Erosion w.r.to nodes	Retrieve all the symptoms which are seen only in severely diseased patients.
$\varepsilon^e(X_1^n)$	Erosion w.r.to hyperedges	Retrieval of all patients with severe diseases.
Social network analysis using IFHG		
Notation	Operation	Result
$\delta^n(X_2^e)$	Dilation w.r.to nodes	This operation will retrieve all the members of black listed groups
$\delta^e(X_2^n)$	Dilation w.r.to hyperedges	This retrieves all the groups with atleast one criminal member
$\varepsilon^n(X_2^e)$	Erosion w.r.to nodes	Retrieve all the members which are seen only in black listed groups.
$\varepsilon^e(X_2^n)$	Erosion w.r.to hyperedges	Retrieval of all groups which are blacklisted.

working as filter parameters, since a high value of these parameters results in less amount of filterate and low value of these parameters result in large amount of filterate. With respect to text processing application using medical reports, patients with "minor", "moderate", "major" and "extreme" medical conditions are retrieved, when we vary the (α, β) cut. The algorithm to find the optimal number of nodes/hyperedges in ASF iterates till the following condition is satisfied:

$$|\gamma_\lambda \circ \phi_\lambda| \geq \epsilon, \tag{4.99}$$

where γ_λ is an opening filter, ϕ_λ is a closing filter, $\gamma_\lambda \circ \phi_\lambda$ is an ASF and $|\gamma_\lambda \circ \phi_\lambda|$ is the cardinality of the filter. In eq(4.99), ϵ is a positive number. The algorithm converges when $\epsilon \to 0$. Also the algorithm may exit without an output if no sub-IFHG is obtained after (α, β) cut. i.e., in terms of medical report analysis we can say that, if we have set (α, β) cut such as to retrieve patients with "extreme" medical condition and if the area considered for analysis is not having such patients, then the algorithm exits without generating an output. In such a case we have to reduce the level of (α, β) cut such as to retrieve all patients in that area with "major" medical conditions. Likewise when we set the level of (α, β) cut such as to retrieve patients in the area with "minor" medical conditions and get an empty sub-IFHG, this implies that, the area under consideration is the one with "good" medical conditions.

4.19 Conclusion

Here we have successfully defined the morphological operations like adjunction, opening, closing, half opening, half closing and alternate sequential filter on IFHG. The results have been substantiated with sample parent IFHG and sub-IFHG. Such filter designs find applications in image processing, text processing, computer networks etc. The (α, β) cut used to generate the sub-hypergraphs can be varied with different values of (α, β). Different sub-hypergraphs with varying (α, β) cut when applied with the above morphological operators will produce results accordingly with various priority ranges of hyperdges/nodes. One who is working with text/image processing and network analysis can find numerous applications with these operations. A filter designed on text results in text summary. Such applications are explained in chapters 5, 6 and 7.

Chapter 5

Text IFHG and morphological operators

In the previous chapter we have seen various morphological operations that are applied on an IFHG. Aim of this chapter is to model text as an IFHG and apply various morphological operations like dilation and erosion on it. The results are verified with the help of a sample text.

5.1 Modeling text using IFHG

Let [H_{IF}, (μ_n, γ_n), (μ_e, γ_e), H^n, H^e] be an intuitionistic fuzzy hypergraph with membership degree μ_n and non-membership degree γ_n defined on the set of nodes H^n; membership degree μ_e and non-membership degree γ_e defined on a set of hyperedges H^e of H_{IF}. While using the concept of hypergraphs in document modeling, the sentences in the document forms the hyperedges H^e and the words in the document forms the nodes H^n. The same method can be used in the case of an IFHG where it includes membership and non-membership degrees for nodes and hyperedges. The membership value μ_n of a node H^n is the term priority p_n of a word. i.e., the membership value of a word depends on the priority of the word. The words which are having less priority will have a high non-membership value, so also the node H^n which represents that word will have a less membership value μ_n and high non-membership value γ_n. The words which are having high priority will have a high membership value, so also the nodes H^n which represent those words will have a high membership value μ_n and less non-membership value γ_n.

Table 5.1: Priority set - words in various domains with high membership values

Domain	Sports	Domain	Health
words	Membership	words	Membership
board	0.6	disease/illness	0.8
indian	0.6	problem	0.7
failure	0.8	severe	0.7
success	0.8	result	0.7
score	0.7	medicine	0.6
team	0.7	medical	0.6
amount	0.6	medicine	0.8
player	0.6	treatment	0.7
cricket	0.7	harmful	0.7
football	0.8	reason	0.8
Reception	0.8	severe	0.7
Domain	Travel	Domain	Politics
words	Membership	words	Membership
bus	0.8	failure	0.8
metro	0.8	success	0.8
distance	0.7	election	0.7
kilometer	0.7	chief minister	0.7
hotel	0.6	minister	0.7
road	0.6	prime minister	0.8
rail	0.6	panchayat	0.6
plane/flight	0.6	municipality	0.6
train	0.8	corporation	0.6
history	0.7	result	0.7
nature	0.8	state/country	0.7

The membership and non-membership values of the words are assigned according to Table 5.1, Table 5.2 and Table 5.3 respectively. All other words in the document other than those given in Table 5.1, Table 5.2 and Table 5.3 will have $\mu_n = 0.5$ and $\gamma_n = 0.5$. Those words are medium words whose presence won't affect the result of morphological operations which are defined on sub-IFHG X_{IF} of H_{IF}.

TABLE 5.2: Priority set - words with high membership values

Domain	Automobile	Domain	Gadgets
words	Membership	words	Membership
new	0.8	model	0.8
engine	0.8	price	0.8
company	0.8	market	0.7
market	0.7	memory	0.7
speed	0.7	speed	0.7
metro	0.6	storage	0.7

5.2 Assigning membership and non-membership degrees

The membership degree $\mu(n_i)$ of some node H^n is the sum of normalized term frequency and membership value(given in Table 5.1 and Table 5.2) of the word. For such words, non-membership degree is $<= 1 - \mu(n_i)$. The non-membership degree $\gamma(n_i)$ of some of the node H^n is the sum of normalized term frequency and non-membership value of the node(given in Table 5.3). Here the normalized term frequency is the count of the word in the document / number of words in the document. For such words, the membership degree is $<= 1 - \gamma(n_i)$. The membership degree of a hyperedge can be written as

$$\mu(e_j) = \vee_{\forall i,j}\{\mu(n_i)/n_i \in e_j \cap n_i \in P_j\}. \tag{5.1}$$

As per Eq 5.1, The membership degree $\mu(e_j)$ of the hyperedge H^e is the supremum of the membership degrees of all the nodes H^n in it, provided all H^n in it belong to the priority set P_j. The non-membership degree $\gamma(e_j)$ of such a hyperedge H^e *is* $<= 1 - \mu(e_j)$. The non-membership degree $\gamma(e_j)$ of a hyperedge H^e can be written as

$$\gamma(e_j) = \vee_{\forall i,j}\{\{\gamma(n_i)/n_i \in e_j\} \cap \{\exists n_i/n_i \in nP_j\}\}. \tag{5.2}$$

It is the supremum of the non-membership degrees of all the nodes H^n in it, provided at least one H^n belongs to the non-priority set nP_j. The membership degree of such edges will be $<= 1 - \gamma(e_j)$. Let us illustrate this IFHG modeling with a small sample text. The text under consideration as in Fig. 5.1 is a

Table 5.3: Non priority set-words with high non-membership values

Domain	Sports	Domain	Health
Words	Non-membership	Words	Non-membership
medicine	0.8	surgery	0.8
drugs	0.8	delivery	0.7
police	0.7	cancer	0.7
custody	0.7	death	0.7
arrest	0.7	failure	0.7
Domain	Travel	Domain	Politics
Words	Non-membership	Words	Non-membership
disaster	0.8	strike	0.8
accident	0.8	police	0.7
death	0.8	expel	0.7
deep	0.7	arrest	0.7
expensive	0.6	court	0.7
luxurious	0.6	strike	0.8
expense	0.6	harthal	0.8
Domain	Automobile	Domain	Gadgets
Words	Non-Membership	Words	Non-Membership
bike	0.6	expensive	0.8
lorry	0.7	expense	0.8
bus	0.7	old	0.8
minibus	0.7	tablet	0.7
railer	0.8	ipod	0.7
expensive	0.8	earphone	0.7
luxurious	0.8	outdated	0.8
old	0.8	cheap	0.8

preprocessed one from which the stop words are removed and which is subjected to lemmatization.

This sample text consists of seven sentences. The membership and the non-membership values of these words are calculated from Table 5.1, Table 5.2 and Table 5.3. This membership/non-membership value along with the normalized term frequency give the membership and non-membership degree. For all words other than those in the above tables, the membership and non-membership values are 0.5. Here we consider that the sum of the membership degree and non-membership degree of the node (word) is less than or equal to 1. i.e., $\mu(n_i) + \gamma(n_i) <= 1$ [95]. So also the sum of the membership degree and non-membership degree of the hyperedge (sentence) is <= 1; i.e.,

”He is an Indian cricket board player. The board has seen his arrest for using drugs. Still the success was with Indian team. The team scored an amount of 20,00,000. The cricket player was arrested on 25/10/17.Police has stopped the reception. Well, the next match is in the city......”

- $indian - n_1$, $cricket - n_2$, $board - n_9$, $player - n_8$.
- $board - n_9$, $drugs - n_{11}$, $arrest - n_{15}$.
- $indian - n_1$, $team - n_6$, $success - n_4$.
- $team - n_6$, $score - n_5$, $amount - n_7$.
- $cricket - n_2$, $player - n_8$, $arrest - n_{15}$.
- $receipt - n_{16}$, $police - n_{13}$, $stop - n_{17}$.
- $next - n_{18}$, $match - n_{19}$, $city - n_{20}$.

FIGURE 5.1: The sample text to be modeled as intuitionistic fuzzy hypergraph.

$\mu(e_i) + \gamma(e_i) <= 1$ [95]. The IFHG for the above sample text can be drawn as in Fig. 5.2.

In Fig. 5.2, we can see sentences modeled as hyperedges and words modeled as nodes. Nodes are having both membership degree $\mu(e_i)$ and non-membership degree $\gamma(e_i)$. The hyperedges are also having both membership degree $\mu(e_i)$ and non-membership degree $\gamma(e_i)$. Since there are seven sentences in the sample text in Fig. 5.1, there are seven hyperedges in Fig. 5.2. The hyperedge having the nodes n_1, n_2, n_8 and n_9 is an edge with only priority words so that it is having good membership degree. Due to the presence of nodes n_{11} and n_{15} which are having high non-membership degrees, the corresponding hyperedge is having less membership degree and high non-membership degree; I.e., the presence of a single word with high non-membership degree $\gamma(e_i)$ influences the non-membership degree of the hyperedge.

5.3 (α, β) cut on text IFHG

Here $X_{IF} \subset H_{IF}$, such that X_{IF} consists of nodes with membership degree > 0.5. The hyperedges in X_{IF} has at least one node with membership degree > 0.5 and it should not contain any node with non-membership degree > 0.5. i.e., the membership degree can be greater than 0.5, but the non-membership degree should

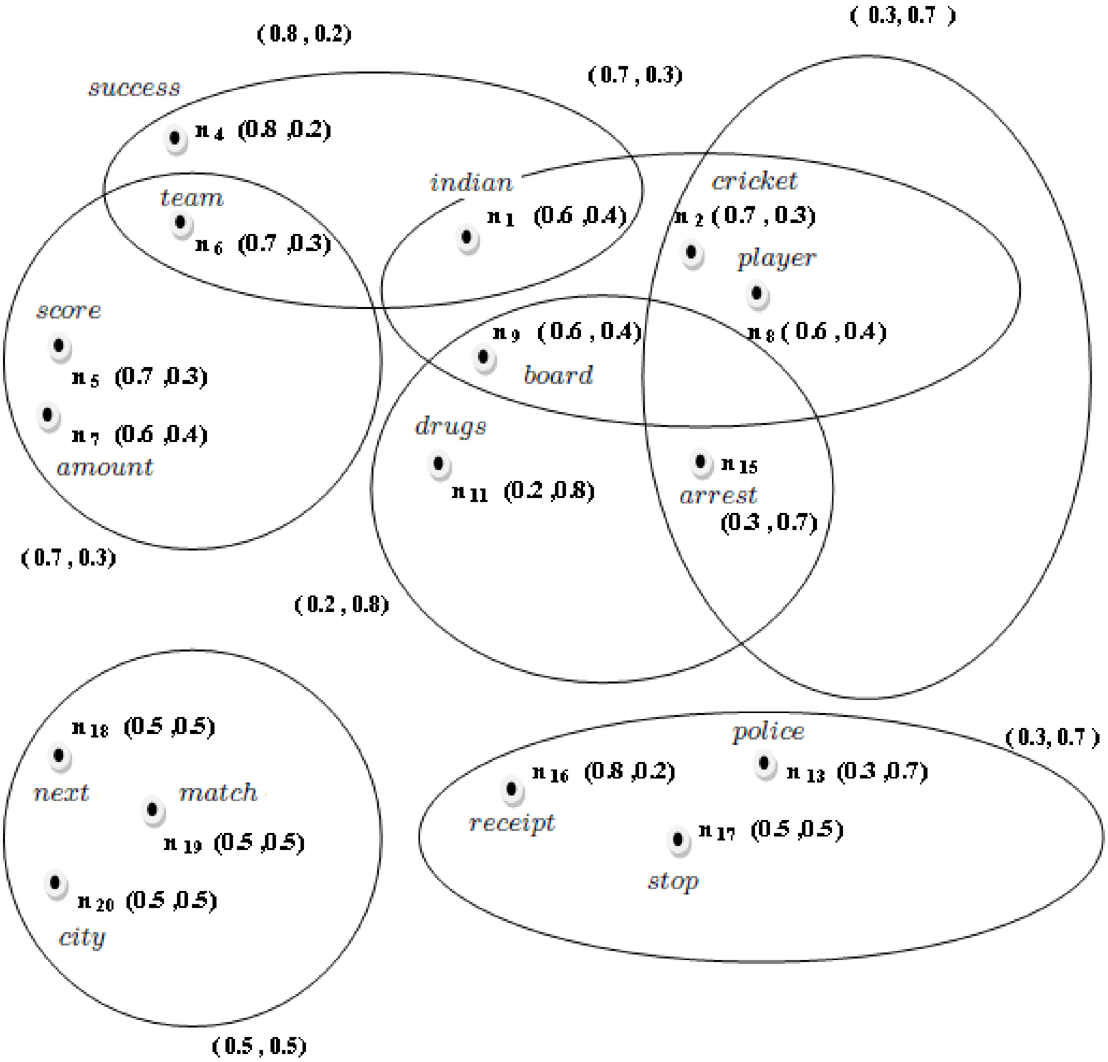


Figure 5.2: Text modeled as hypergraph

be less than 0.5. Now X_{IF} is a collection of priority sentences and priority words as given in Fig. 5.3.

Now let us apply morphological operations [102], [104], [105] on this X_{IF}. Let X^n be the node set in X_{IF} and X^e be the edge set in X_{IF}.

5.4 Morphological operators on text IFHG

- **Dilation with respect to nodes-$\delta^n(X^e)$**

  This morphological operation is defined as

$$\delta^n(X^e) = \{n_i / n_i \in X^e\}. \tag{5.3}$$

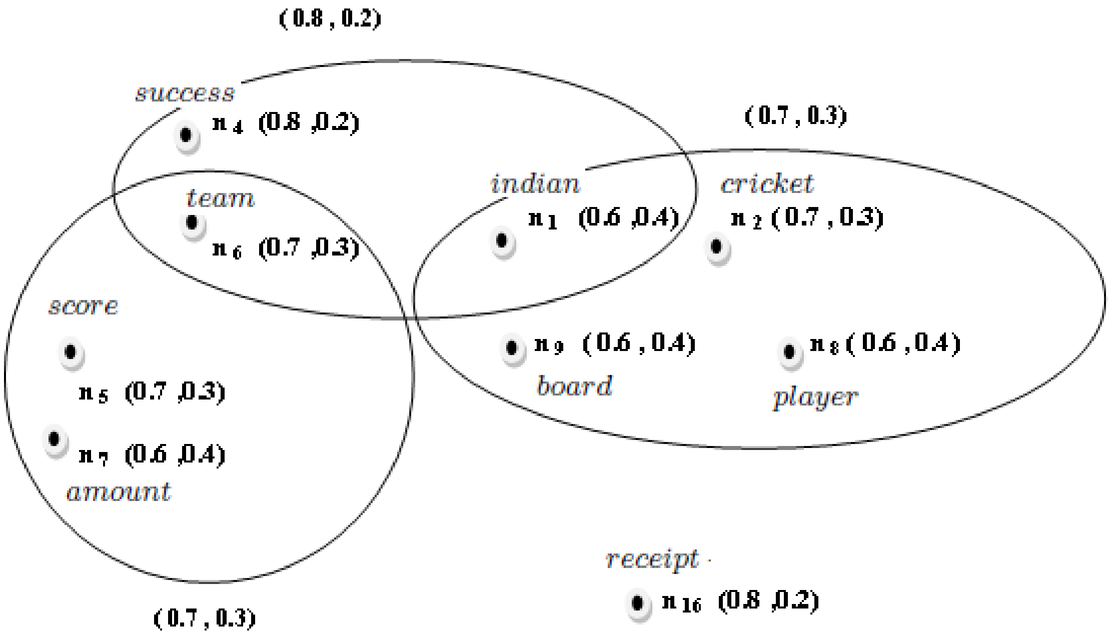


FIGURE 5.3: Hypergraph X_{IF} obtained after (α, β) cut on H_{IF}

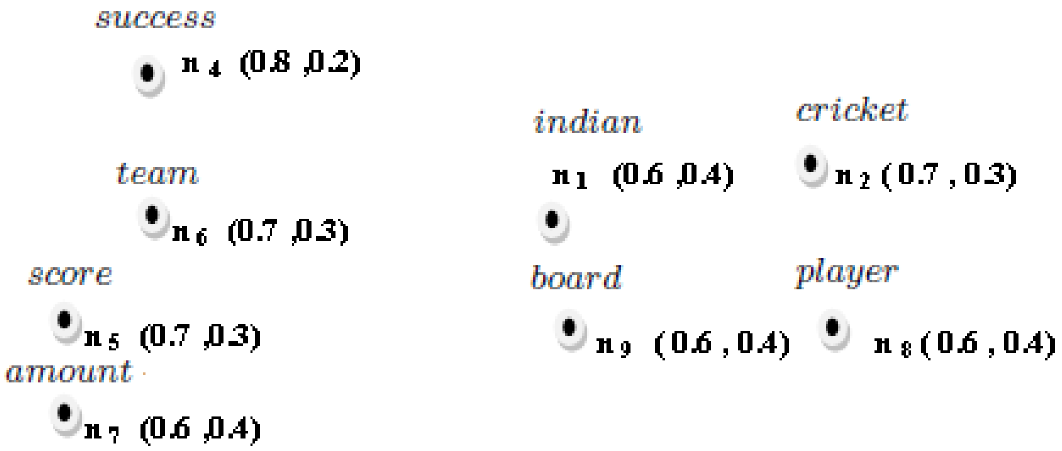


FIGURE 5.4: Hypergraph obtained after dilation on X_{IF}

Take all edges in X_{IF}. This will result in X^e. Take all nodes X^n in X^e. Here we are selecting all hyperedges from H_{IF}, which have at least one node with membership degree > 0.5 and which does not contain any node with non-membership degree > 0.5. Once we select such edges, we select the nodes in it with membership degree > 0.5. This will ultimately give $\delta^n(X^e)$. This retrieves a collection of priority words within priority sentences as shown in Fig. 5.4.

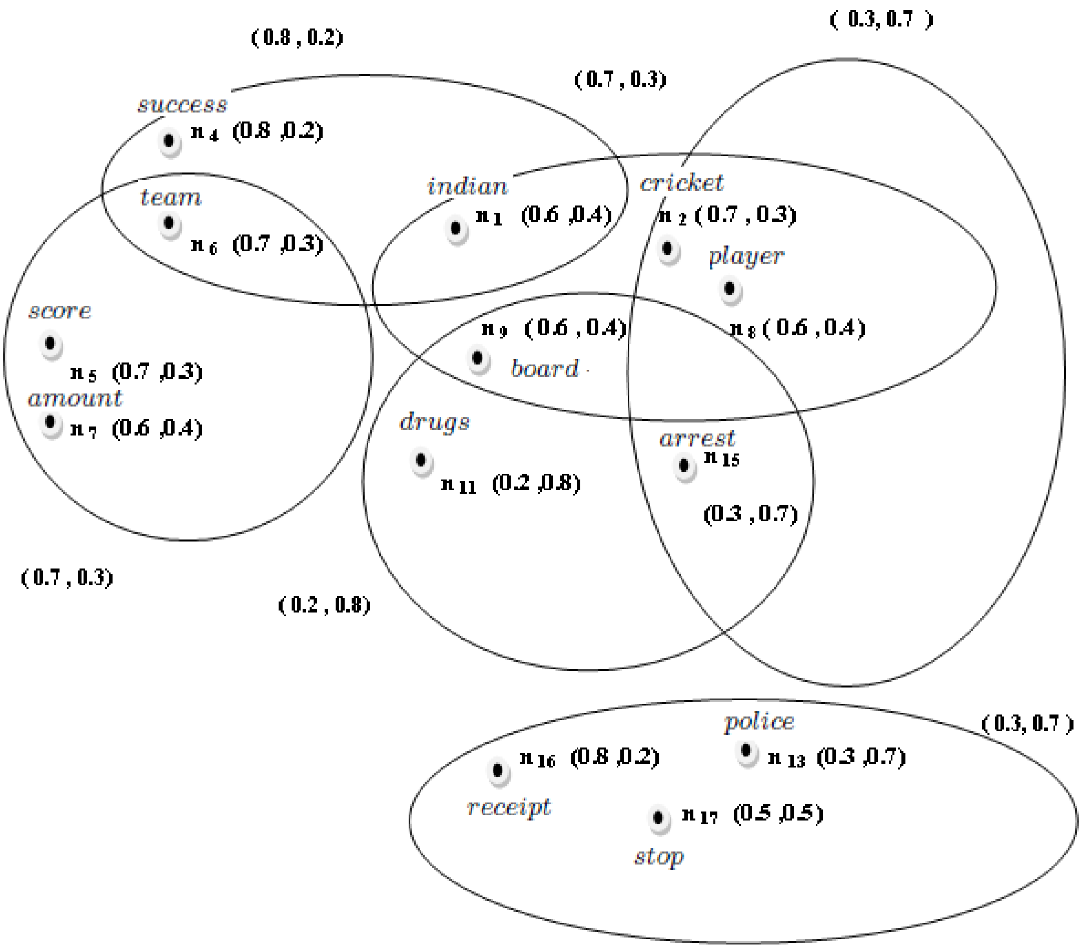


FIGURE 5.5: Dilation w.r.to hyperedge

- **Dilation with respect to hyperedge-** $\delta^e(X^n)$

  This dilation can be written as

$$\delta^e(X^n) = \{e_i/e_i \in H^e \cap \{\exists n_i \in e_i/n_i \in X^n\}\}. \qquad (5.4)$$

Take all nodes X^n. Find from H_{IF} all the hyperedges which include X^n. Here we select from X_{IF} all nodes with membership degree > 0.5. Find from H_{IF} all hyperedges which contain those nodes. This will give all hyperedges which contain at least one node with membership degree > 0.5. These hyperedges may or may not contain nodes with non-membership degree > 0.5. This dilation selects all texts which has at least one priority word as shown in Fig. 5.5.

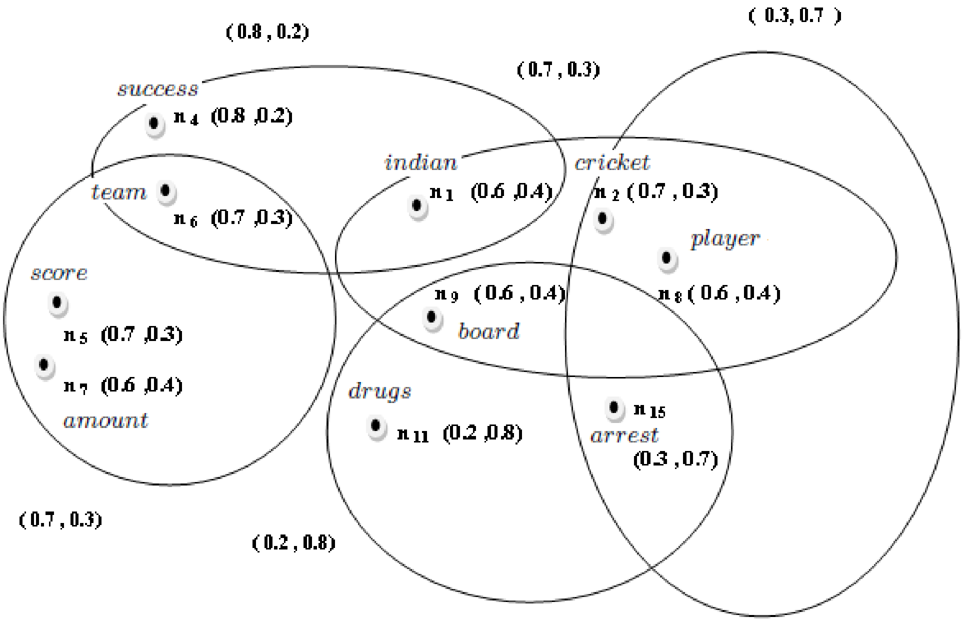


FIGURE 5.6: $\delta(X^n)$-Dilation

- **Node dilation-** $\delta(X_n)$
  This dilation can be written as

  $$\delta(X_n) = \{e_i/e_i \in H^e \cap \{\exists n_i \in e_i/e_i \in X^e\}\}. \tag{5.5}$$

  Take all hyperedges X^e. Take all nodes in X^e. Find all the hyperedges with respect to H_{IF} which contain these nodes. This dilation gives all sentences in H_{IF} which overlap with the priority sentences. This is shown in Fig. 5.6.

- **Dilation-**$\Delta(X^e)$
  This dilation can be written as

  $$\Delta(X^e) = \{e_i/e_i \in H^e \cap \{\exists n_i/n_i \in \{X^e \cap H^e\}\}\}. \tag{5.6}$$

  Find all hyperedges X^e. Find all nodes in X^e. Let it be X^{n1}. Find all hyperedges H^e and the nodes in it. Let it be H^{n1}. For all $X^{n1} \cap H^{n1} \neq \emptyset$, find the hyperedges from H_{IF}. This will retrieve all sentences which has at least one priority word in priority sentences of X_{IF}. The same is represented in Fig. 5.7.

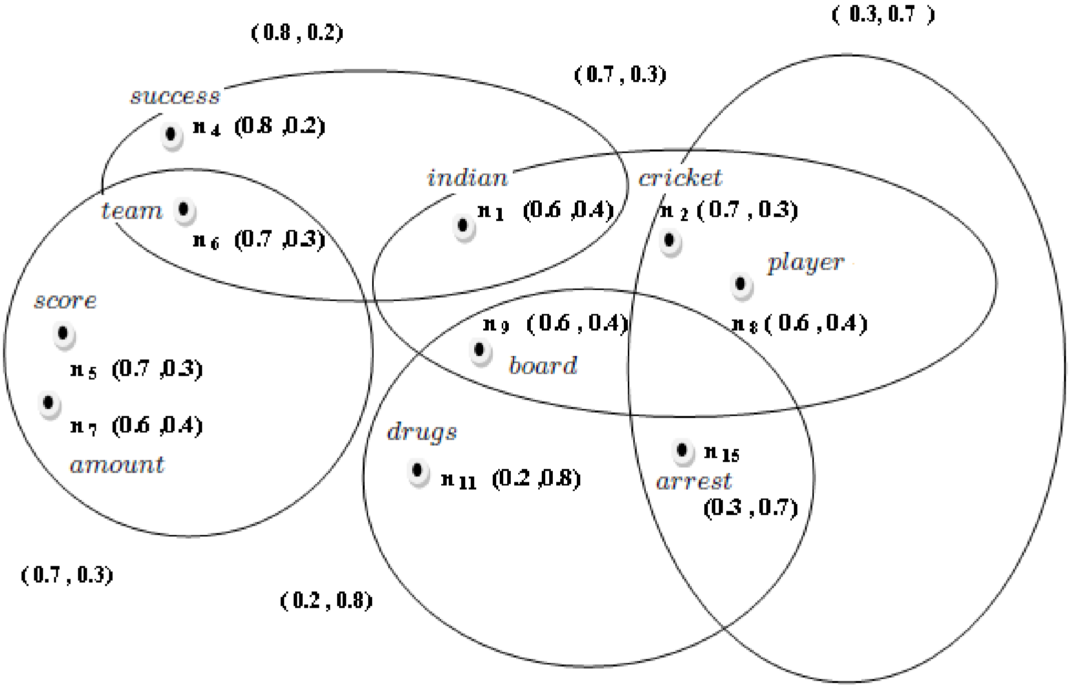


FIGURE 5.7: $\Delta(X^e)$-Dilation

- **Erosion w.r.to hyperedge-$\varepsilon^e(X^n)$**

  So far we have seen dilation operations of X_{IF}. Now let us see how different types of erosion can be defined on X_{IF}. The erosion $\varepsilon^e(X^n)$ can be defined as

$$\varepsilon^e(X^n) = \{e_i/e_i \in H^e \cap \{\forall n_i/\{n_i \in e_i \cap n_i \in X^n\}\}\}. \qquad (5.7)$$

  Take all nodes X^n in X_{IF}. Take all hyperedges in H_{IF} which consists of these nodes only. This erosion as seen in Fig. 5.8 strictly retrieves priority sentences.

- **Erosion w.r.to node- $\varepsilon^n(X^e)$**

  The erosion $\varepsilon^n(X^e)$ can be written as

$$\varepsilon^n(X^e) = \{n_i/\{n_i \notin \{X^e \cap X^{e'}\}/X^{e'} = H_{IF} - X^e\}\}\}. \qquad (5.8)$$

  Take all hyperedges X^e. Take its complement edges $X^{e'}$ in H_{IF}. Take all nodes X^n which are not in $X^e \cap X^{e'}$. This will retrieve all priority sentences which do not overlap with any non priority sentences. Now take the priority words in it as shown in Fig. 5.9.

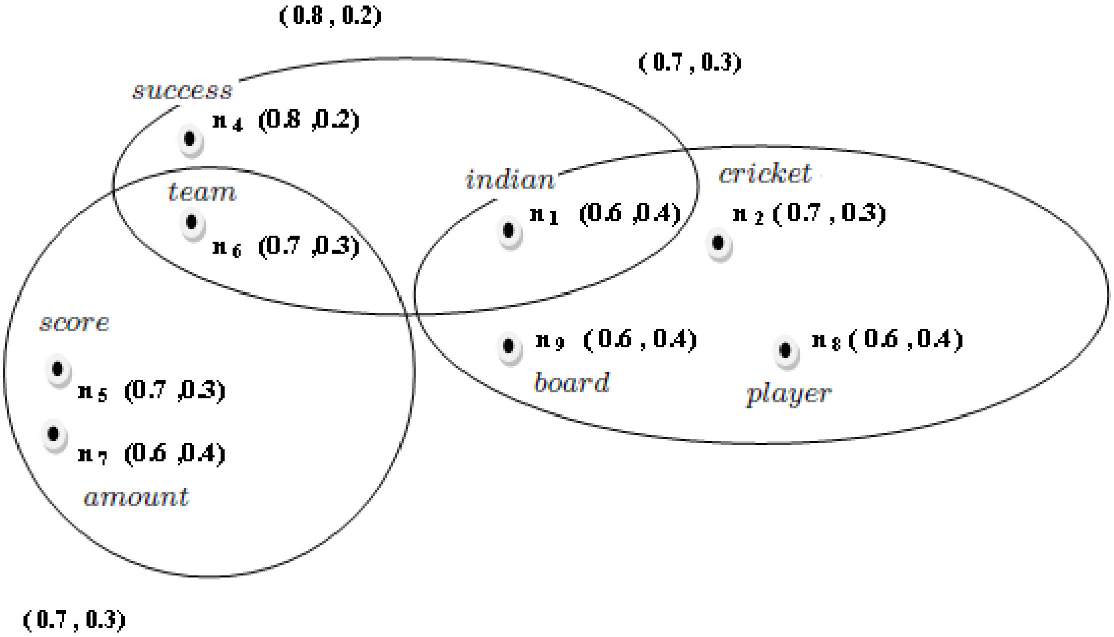


Figure 5.8: Erosion with respect to hyperedge

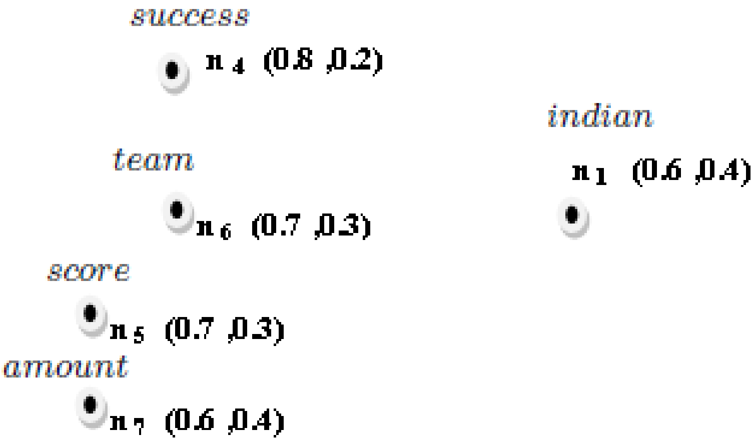


Figure 5.9: Erosion with respect to node

- **Hyperedge erosion-$\varepsilon(X^e)$**

  The erosion $\varepsilon(X^e)$ is defined as

$$\varepsilon(X^e) = \{e_i/e_i \in H^e \cap \{n_i \in e_i \cap n_i \in \varepsilon^n(X^e)\}\} \tag{5.9}$$

Take all nodes in $\varepsilon^n(X^e)$. Take all edges from X_{IF} which fully contains these nodes. This will retrieve all priority sentences which do not overlap with the non-priority sentences. This is illustrated in Fig. 5.10.

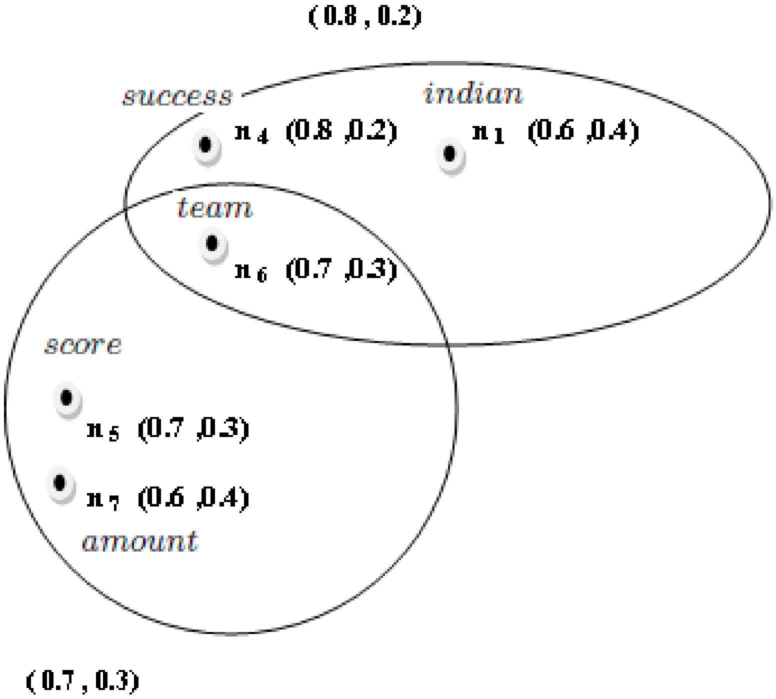


FIGURE 5.10: $\varepsilon(X^e)$-Erosion

- **Dilation-**$[\delta, \Delta](X_{IF})$
  This dilation can be written as

$$[\delta, \Delta](X_{IF}) = \{(e_i, n_i)/\{e_i \in \{\Delta(X^e) \cap \delta^e(X^n)\}\} \cap \{n_i \in e_i \notin \{\Delta(X^e) \cap \delta^e(X^n)\}\}\} \tag{5.10}$$

As seen in Fig. 5.11, this is obtained by joining $\Delta(X^e)$ and $\delta^e(X^n)$. Take all edges which are common in $\delta^e(X^n)$ and $\Delta(X^e)$. Include all such hyperedges and its nodes as output. For other edges in $\delta^e(X^n)$, include only nodes in it. This will retrieve all sentences which overlaps with the priority sentences and the words in it. It also retrieves all words in sentences which have both priority and non-priority words and which do not overlap with others.

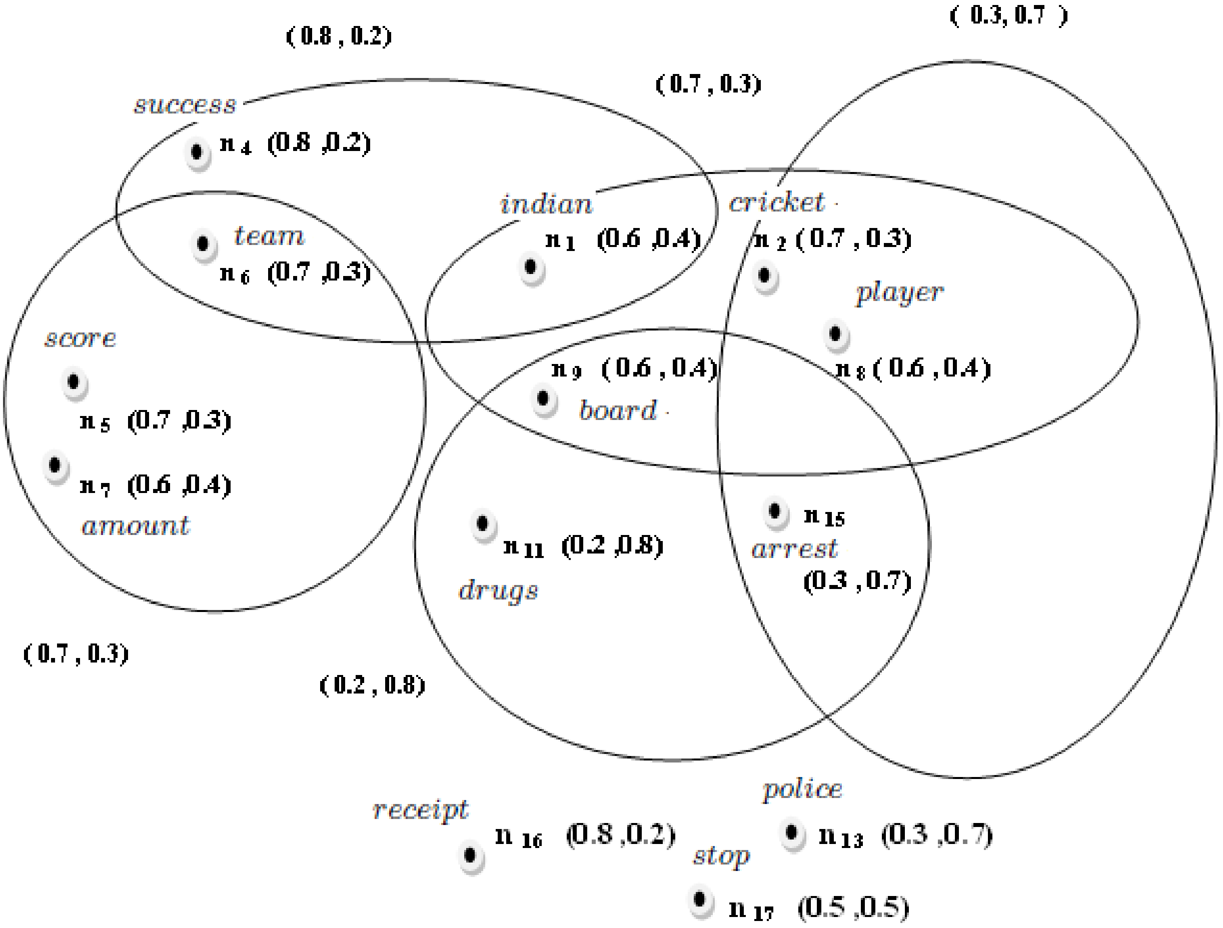


Figure 5.11: $[\delta, \Delta](X_{IF})$-Dilation

5.5 Conclusion

This chapter has successfully modeled text using an IFHG, by converting sentences as hyperedges and words as nodes. Membership and non-membership values are assigned for words, from which those of hyperdges are also calculated. It has also shown the results of various dilations and erosions on text IFHG. Next chapter shows how morphological filter operation is done on a text IFHG.

Chapter 6

Design of summary filter using IFHG

6.1 Implementation

The implementation of the summarization as shown in Fig. 6.1 and algorithm 7, is done with the help of a filter system developed in python for input English news taken from online news sites. The English news related to various topics are being subjected to stop word removal and stemming. The preprocessed text is then represented as a weighted hypergraph [107]. The weighted hypergraph is subjected to spectral partitioning. Spectral partitions lead to text clusters. The summary filter is then applied to each cluster formed. The sentences which do not fall under any of the clusters are treated as outliers and are removed. A Malayalam summarization system is also developed using the same method, where a Malayalam lemmatizer [106] is used to stem the words.

6.1.1 Filter design

Filter is an operator which is idempotent and increasing defined on domain D. Let X_{IF} be the sub-hypergraph defined in section 5.3; then if $f(f(X_{IF})) = f(X_{IF})$, then f is idempotent. If X and Y are sub-hypergraphs then if $f(X) \subset f(Y)$, then f is increasing. F is a filter if both of these are satisfied. Let ε be the erosion operator and δ be the dilation operator. Then let $\varepsilon \circ \delta$ be an operator and if

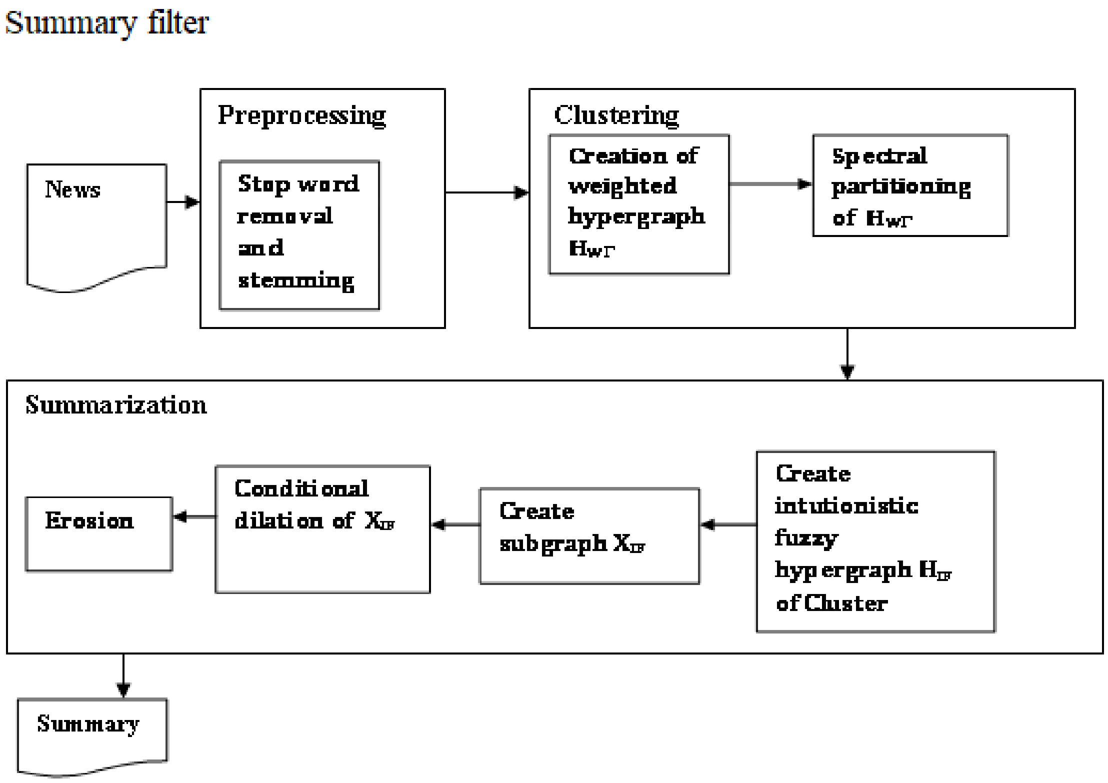

Summary filter
News
Preprocessing
Stop word removal and stemming
Clustering
Creation of weighted hypergraph H_WΓ
Spectral partitioning of H_WΓ
Summarization
Create intutionistic fuzzy hypergraph H_IF of Cluster
Create subgraph X_IF
Conditional dilation of X_IF
Erosion
Summary

FIGURE 6.1: Architecture of summarization system

$\varepsilon \circ \delta(\varepsilon \circ \delta(X)) = \varepsilon \circ \delta(X)$ then $\varepsilon \circ \delta$ is a filter. That is, here filter consists of a erosion which is composed of dilation or we can say that we have dilation followed by erosion. Such a filter can be used for text summarization. Text summarization basically can be considered as a filter which removes all unwanted sentences from a text. We can also call summarization as a filter operator which selects only the needed sentences from the given text.

6.1.2 Summary filter

Text summarization can be done with the help of this filter operator which is applied to the IFHG created from the text under consideration. This filter is designed as a combination of two morphological operators namely dilation and erosion. Here dilation is designed as a conditional one and erosion is designed as the one which performs complement operation. For implementing this conditional dilation, let us assume that our text consists of certain star words, whose occurrence in sentences are valid even if they co-occur with non priority words. So for this summary filter, let us assume that our text consists of words which are of high priority, words which are of low priority, words with neutral

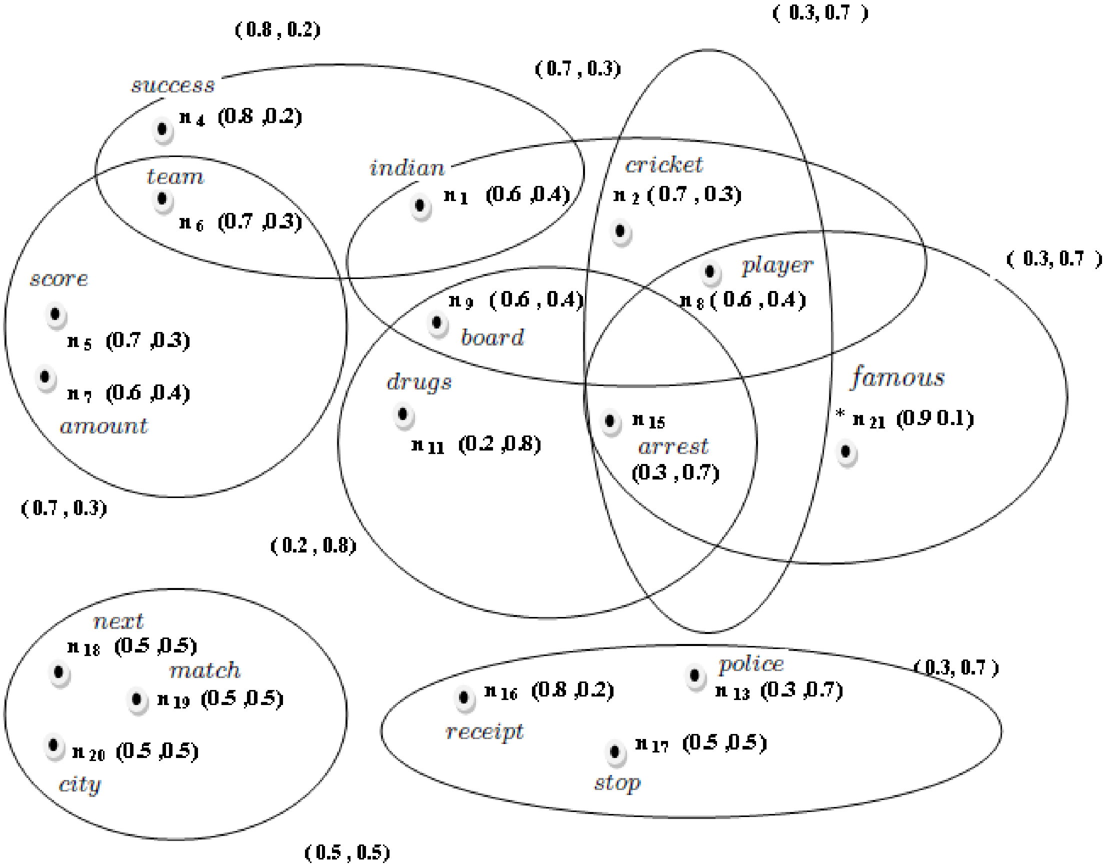


FIGURE 6.2: Modified intuitionistic fuzzy hypergraph H_{IF}

priority, and star words. Let us redefine the intuitionistic fuzzy hypergraph as $[H_{IF}$, $(\mu_n, \gamma_n\ )$, (μ_e, γ_e), H^n, H^{*n}, H^e, $H^{*e}]$, where H^{*n} is the star node and H^{*e} is the edge which has the star node H^{*n}. These star words are domain independent. Some of the star words are given in Table 6.1. Sentences which contain star words are definitely included in the summary text. To illustrate this, let us add one more sentence to our sample text as the following:

" The arrest of the famous player .....".

Now this will result in new hyperedge with the following nodes. famous- n_{21} player- n_8 arrest-n_{15}.

The modified intuitionistic fuzzy hypergraph after the addition of the above sentence is given in Fig. 6.2. The sub-hypergraph X_{IF} is also getting modified since it will have the star nodes also in it. The modified X_{IF} can be shown as in Fig. 6.3.

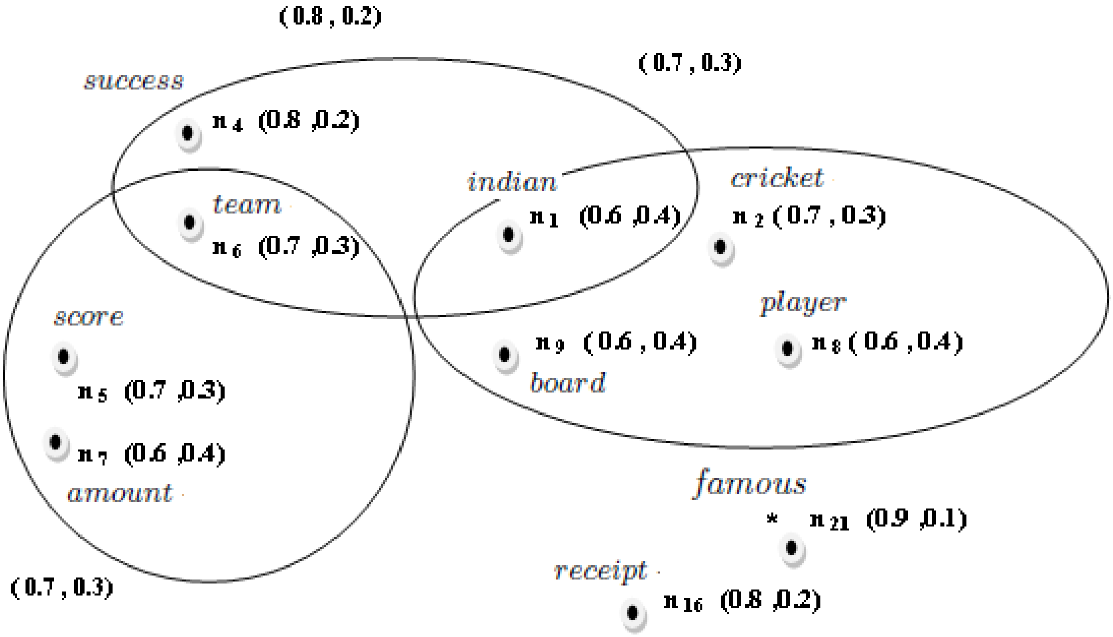


Figure 6.3: Modified X_{IF}

Table 6.1: Star words irrespective of the domain of the text

Words	Membership	Words	Membership
famous	0.9	excel	0.9
fame	0.9	excellent	0.9
well known	0.9	attract	0.9
famed	0.9	attractive	0.9
popular	0.9	pleasing	0.9
important	0.9	pretty	0.9
prominent	0.9	alluring	0.9
main	0.9	good	0.9
chief	0.9	handsome	0.9
major	0.9	significant	0.9
key	0.9	powerful	0.9
foremost	0.9	urgent	0.9
supreme	0.9	influential	0.9
overriding	0.9	momentous	0.9
essential	0.9	indispensable	0.9

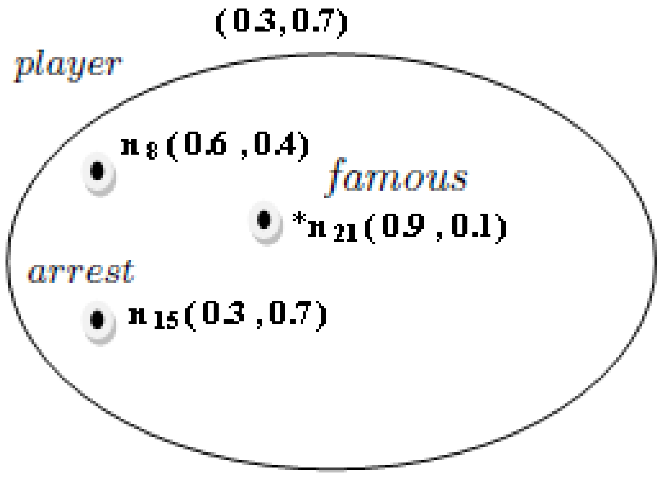


FIGURE 6.4: Conditional dilation on X_{IF}

6.1.3 Conditional dilation for summary filter - $\delta^c(X_{IF})$

This conditional dilation is applied such that while dilating the sub-IFHG X_{IF} we consider the condition specified by c, where c is designed such that it selects all hyperedges in H which consists of star nodes given in Table 6.1.

$$\delta^c(X_{IF}) = \{e_i/e_i \in H^{*e}\}. \tag{6.1}$$

- This conditional dilation will retrieve all edges from the intuitionistic fuzzy hypergraph, such that it consists of all edges H^{*e}, which consists of star nodes H^{*n} as given in Fig. 6.4. Even though the non membership degree of the edge H^{*e} is 0.7, it is retrieved in the dilation operation which is applied, since it contains the star node H^{*n}.

6.1.4 Erosion for summary filter - $\varepsilon(H^{*e}, X^e)$

This erosion will retrieve all edges ε' from H_{IF} which are not in H^{*e}. Also take all edges ε'' from H_{IF} which are not in X^e. The intersection of the two will result in the retrieval of non priority edges. Now the complement of this will yield the priority edges from the hypergraph H_{IF}. This erosion will eliminate all duplicate edges from H^{*e} and X^e and retrieve us the most important sentences which itself is the required feature of a summary. This erosion can be written as

$$\varepsilon(H^{*e}, X^e) = \{e_i/e_i \in [H_{IF} - [H^{*e'} \cap X^{e'}]]\}, \tag{6.2}$$

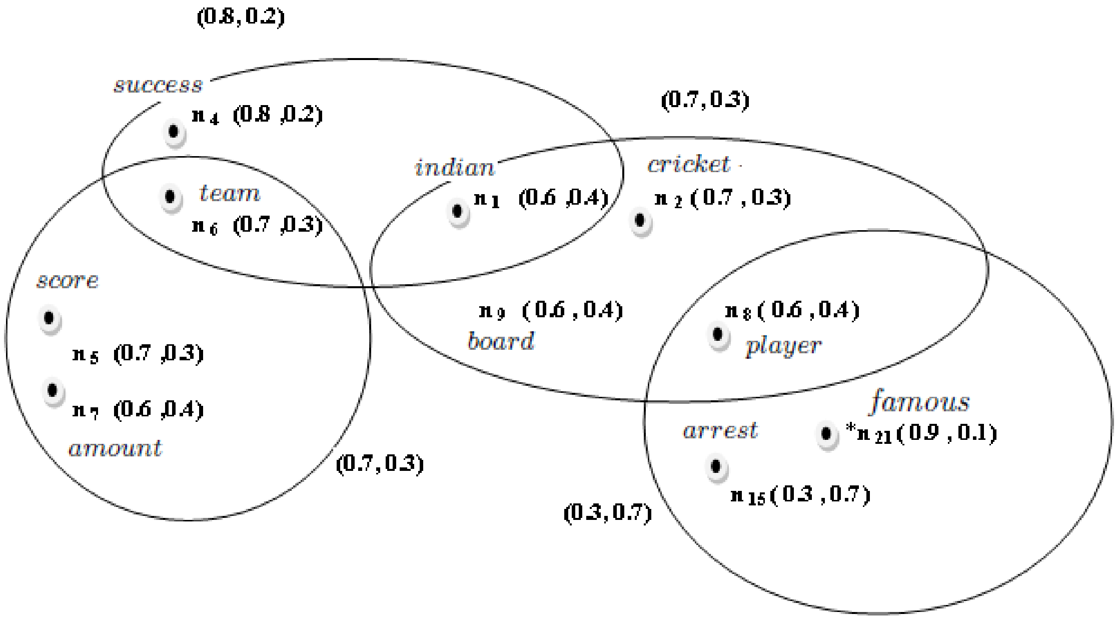


FIGURE 6.5: Summary filter

where $H^{*e'}$ is the complement of H^{*e} and $X^{e'}$ is the complement of X^e. The intuitionistic fuzzy sub-hypergraph retrieved after filter can be shown as in the Fig. 6.5.

Algorithm 7: Summarization of text

```
collect news related to various topics from online sites;
preprocess the sentences by subjecting to stop word removal and stemming;
create weighted hypergraph $H_{w\tau}$ of the text $\tau$;
cluster the text $\tau$ using spectral partitioning of hypergraph $H_{w\tau}$;
for each cluster $C_i$ do
    assign $\mu(n_j)$ and $\gamma(n_j)$ for all words $C_i$;
    assign $\mu(e_j)$ and $\gamma(e_j)$ for all sentences in $C_i$;
    create intuitionistic fuzzy hypergraph $H_{IF}$ with nodes $H^n$ having ($\mu(n_j)$,
      $\gamma(n_j)$) and hyperedges $H^e$ having ($\mu(e_j)$, $\gamma(e_j)$);
    create subgraph $X_{IF}$ of $H_{IF}$ with hyperedges $X^e$ having $\mu(e_j) > 0.5$ and
      nodes $X^n$ having $\mu(n_j) > 0.5$ ;
    apply conditional dilation $H^{*e} = \delta^c(X_{IF})$ ;
    apply erosion $\varepsilon(H^{*e}, X^e)$ to form the summary;
end
```

6.2 Advantages over existing systems

The summarization system which is designed here as a filter applied on IFHG has many advantages over existing summarization methods developed so far. They can be listed as follows:

- **Variety of summary filters**
  As we all know, a filter is basically a composition of dilation and erosion or erosion and dilation. The proposed new method helps in the creation of series of different types of filters by combining the morphological operators like dilation and erosion discussed in section 5.4. Using these different types of filters, different types of summaries can be generated. Some of the filter designs other than the one discussed in section 6.1 are shown below.

  - **Filter 1 -** $\delta(\varepsilon^n(X^e))$
    This filter is a composition of erosion $\varepsilon^n(X^e)$ and dilation δ. The erosion will retrieve all nodes in $X^e \cap X^{e'}$. Now the dilation operation will retrieve all hyperedges H^e which contains the nodes retrieved by the erosion operator. This summary filter will retrieve all sentences from the text with at least one priority word. But this summary will consider star words only if they are part of priority edges in X. Well, this summary is not that short.
  - **Filter 2 -** $\varepsilon(\delta^n(X^e))$ This is a composition of dilation $\delta^n(X^e)$ and erosion ε. The dilation operator retrieves the collection of priority nodes within priority edges. The erosion operator will retrieve all hyperedges H^e in H which consists of only the nodes returned by the dilation operator. This summary retrieves only pure priority sentences that has no non-priority words in it. This is a very short summary.
  - **Filter 3 -** $\varepsilon(\delta^e(X^n))$ This is a composition of dilation $\delta^e(X^n)$ and erosion ε. The dilation defined by $\delta^e(X^n)$ takes all nodes in X and retrieves all edges from H which consists of these nodes. The erosion will take the double complement of $\delta^e(X^n)$. This is also a very short summary. More number of filters can be designed by combining the morphological operators defined in chapter 5, resulting in the generation of different types of summaries.

- **Customized summary** The summary generated by the filter is a customized one as it requires the priority of the user to be submitted before the summary being generated. Thus the summary generated is not a blind one as it takes in to consideration the preferences of the reader. The reader can give as input the priority and non-priority words and the summary will be generated accordingly. So the summary report will definitely be a one which satisfies the reader.

6.3 Result analysis

The system is tested on google cloud platform with 8 cores, 30 GB memory. A comparison of the proposed system with the existing online text summary systems like tools4noobs, summarization.net, splitbrain.org/services is done for various data set. The data set consists of English news taken from online news sites. The news belongs to various domains like travel, politics, health, sports, gadgets etc. The same is uploaded in Mendeley repository. First of all, the news is subjected to clustering and then to summary generation using IFHG method. The summaries generated by each of the above system is compared with human summaries created. About 50 human summarizers are asked to create summaries for each of the data set. The maximum repeating sentences among all the 50 summaries are output to create the final human summary with which the existing systems and the IFHG method are compared. The Rouge-L, Rouge-2 and Rouge-1 scores are calculated and summarized in Table 6.2, Table 6.3 and Table 6.4. In the following tables 'P' stands for the Precision, 'R' stands for recall and 'F' stands for F-measure. The proposed work has shown an average precision of 0.88 , average recall of 0.84 and average F-measure of 0.86. The similarity of the output of the proposed system and the three online systems are compared with the human summaries as shown in Fig. 6.6. For all the data sets, the proposed system generated summaries having more than 90% similarity with human summaries. The method has a time complexity of $O(n^2)$.

6.4 Conclusion

The system developed here has successfully modeled text using IFHG, where words become nodes and sentences become hyperedges. Membership and non-membership degrees are assigned for nodes. Based on that, membership degrees and non-membership degrees of hyperedges are calculated. Various morphological operations are defined and applied on IFHG. Summary of the text is created by applying a filter operator on IFHG. The system has given a better

Table 6.2: Rouge-L score

Data set	IFHG			Tools4noobs		
size (words)	P	R	F	P	R	F
600	0.89	0.88	0.88	0.46	0.39	0.42
1071	0.81	0.78	0.79	0.33	0.23	0.25
2774	0.95	0.95	0.95	0.25	0.23	0.24
5044	0.67	0.69	0.68	0.19	0.16	0.17
6436	0.97	0.72	0.79	0.39	0.19	0.21
Data set	Summarization.net			Splitbrain.org		
size(words)	P	R	F	P	R	F
600	0.27	0.31	0.29	0.49	0.51	0.50
1071	0.17	0.22	0.19	0.21	0.26	0.23
2774	0.22	0.35	0.24	0.36	0.48	0.39
5044	0.33	0.27	0.29	0.19	0.19	0.19
6436	0.29	0.27	0.28	0.29	0.32	0.31

Table 6.3: Rouge-2 score

Data set	IFHG			Tools4noobs		
size(words)	P	R	F	P	R	F
600	0.87	0.88	0.88	0.51	0.45	0.48
1071	0.79	0.76	0.77	0.41	0.29	0.34
2774	0.97	0.97	0.97	0.60	0.55	0.58
5044	0.71	0.72	0.72	0.23	0.19	0.21
6436	0.95	0.69	0.79	0.31	0.17	0.23
Data set	Summarization.net			Splitbrain.org		
size(words)	P	R	F	P	R	F
600	0.31	0.36	0.33	0.51	0.56	0.53
1071	0.07	0.09	0.08	0.17	0.20	0.18
2774	0.39	0.60	0.47	0.46	0.62	0.53
5044	0.46	0.41	0.43	0.20	0.21	0.21
6436	0.28	0.29	0.29	0.24	0.27	0.25

performance when compared to other existing systems. The summary filter has shown more similarity with human summaries generated. The system combines multiple text and treat it as a single one. The system can also be extended with multiple documents, where important words can be modeled as nodes and documents as hyperedges. In our system, there is only a single sub-hypergraph with which morphological operations are defined. Other enhancements like creating more than one sub-hypergraph and morphological operations with intersection/union of those are also possible. These are explained in the next chapter.

Table 6.4: Rouge-1 score

Data set	IFHG			Tools4noobs		
size(words)	P	R	F	P	R	F
600	0.88	0.92	0.91	0.58	0.55	0.57
1071	0.81	0.79	0.79	0.49	0.37	0.42
2774	0.97	0.97	0.97	0.69	0.66	0.67
5044	0.79	0.78	0.78	0.37	0.34	0.36
6436	0.97	0.74	0.84	0.49	0.34	0.39
Data set	Summarization.net			Splitbrain.org		
size(words)	P	R	F	P	R	F
600	0.40	0.46	0.43	0.55	0.65	0.59
1071	0.19	0.25	0.22	0.29	0.35	0.32
2774	0.52	0.69	0.59	0.54	0.73	0.61
5044	0.55	0.5	0.52	0.34	0.38	0.36
6436	0.42	0.48	0.45	0.40	0.49	0.44

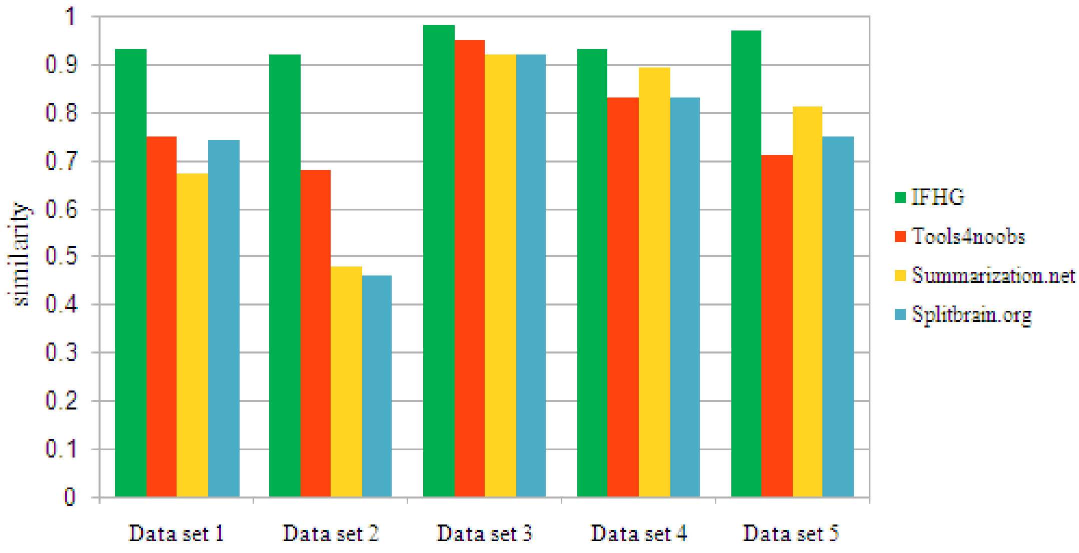


Figure 6.6: Similarity with human summary

Chapter 7

Filtering of IFHG and multi-document summarization

A morphological filter [108] is nothing but an opening filter γ which is obtained as $\delta \circ \varepsilon$ or a closing filter ϕ which is obtained as $\varepsilon \circ \delta$.
An opening filter can again be classified as the following:

- Opening filter w.r.to nodes $\delta^n(\varepsilon^e(Y^n))$ for which the filtrate are nodes.
- Opening filter w.r.to hyperedges $\delta^e(\varepsilon^n(Y^e))$ for which the filtrate are hyperedges.

A closing filter can be classified as the following:

- Closing filter w.r.to nodes $\varepsilon^n(\delta^e(Y^n))$ for which the results are nodes.
- Closing filter w.r.to hyperedges $\varepsilon^e(\delta^n(Y^e))$ for which the results are hyperedges.

Now let us apply algebraic operations like union, intersection and complement operations on these filters.

7.1 Algebra of filter

Let us define $H_{IF} = [H^n, H^e, \mu_n, \mu_e, \gamma_n, \gamma_e]$, where H^n are the hypernodes $[n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{16}, n_{17}, n_{18}, n_{19}, n_{20}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}]$ and $H^e = [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}]$ as given in Fig. 7.1. Some nodes are of low priority ($\mu_n < 0.5$), some nodes are of medium priority ($\mu_n = 0.5$) and few others are of high priority ($\mu_n > 0.5$). Let X_{IF} be obtained by (α, β) cut on $H_{IF}/0.5 < \alpha \leq 0.7; \beta \leq 1 - \alpha$. Let Y_{IF} be obtained by (α, β) cut on $H_{IF}/\alpha \geq 0.7; \beta \leq 1 - \alpha$. Here α corresponds to membership degree and β corresponds to non-membership degree. The details of the hypergraphs H_{IF}, X_{IF} and Y_{IF} are given in Fig. 7.1. Having given IFHGs H, X and Y, let us define $\delta^n(Y^e)$ as the set of all nodes within the hyperedges in Y. Now $\delta^e(Y^n)$ is the set of hyperedges in H which consists of nodes in Y. Let us define erosion with the help of the operator ε. Now $\varepsilon^e(Y^n)$ is the set of all hyperedges in H which consists of nodes of Y only. Similarly $\varepsilon^n(Y^e)$ is the set of all nodes in Y but not in $Y^{e'}$. Now let us apply algebraic operations like union, intersection and complement on these filters.

Proposition *A***:** Let H be a parent IFHG, X and Y be sub-IFHGs, δ be the dilation operator and ε be the erosion operator, then

$$\delta^e(\varepsilon^n(X \cup Y)^e = \delta^e(\varepsilon^n(X^e)) \cup \delta^e(\varepsilon^n(Y^e)), \tag{7.1}$$

where $\delta^e(\varepsilon^n(X \cup Y)^e$ is an opening filter w.r.to hyperedges

Proof: Consider L.H.S of eq(7.1). Let v be an arbitrary node in $\varepsilon^n(X \cup Y)^e$. i.e.,

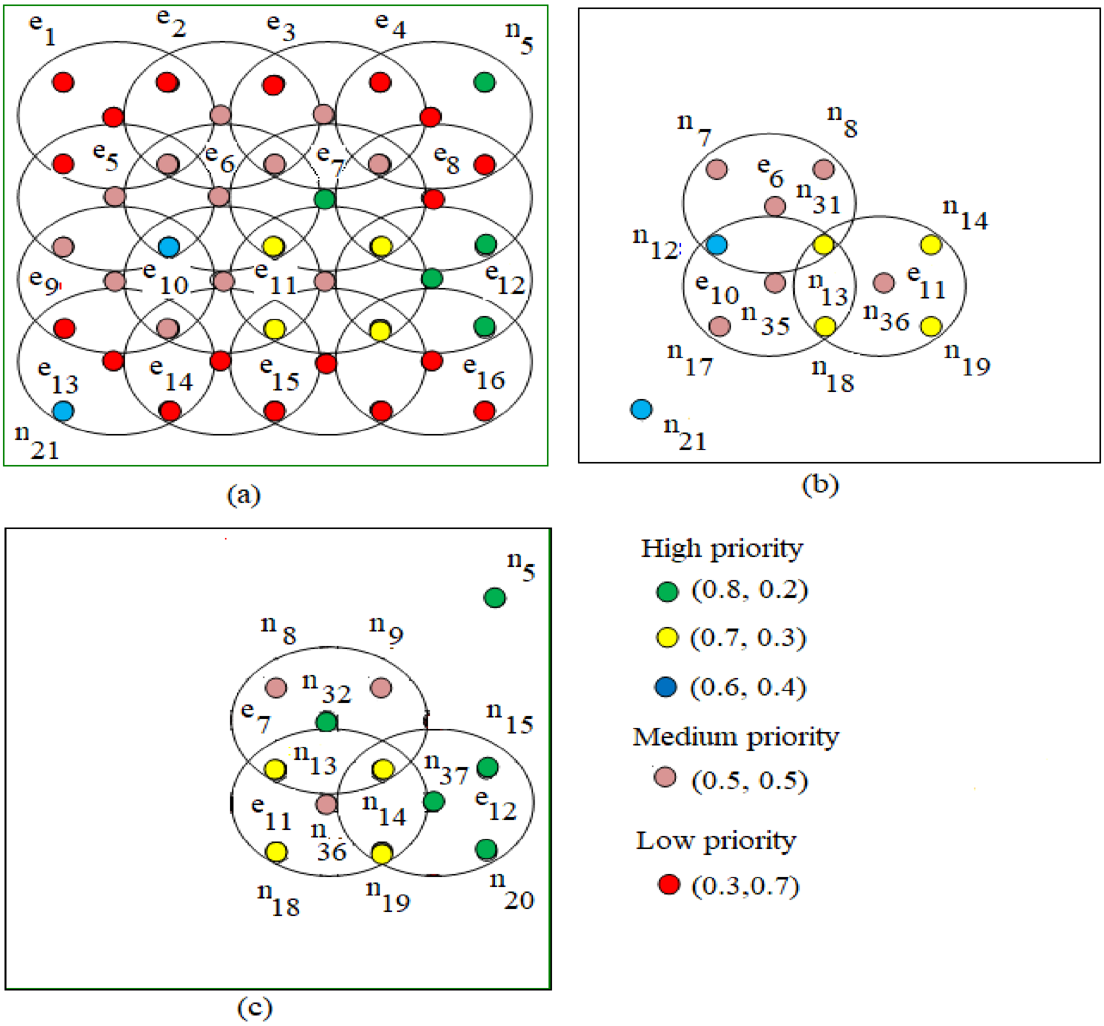
$$v \notin (X \cup Y)^{e'}; v \in (X \cup Y)^e. \tag{7.2}$$

Let e be the edge which contains this v. i.e.,

$$e \subseteq \delta^e(\varepsilon^n(X \cup Y)^e. \tag{7.3}$$

Since $v \in (X \cup Y)^e$, this e may be present either in X or in Y. Consider R.H.S of eq(7.1). Let v be a node in $\varepsilon^n(X^e)$; i.e.,

$$v \notin X^{e'}; v \in X^e. \tag{7.4}$$



(a)

(b)

(c)

Figure 7.1: (a) H, (b) X, (c) Y

Now $\delta^e(\varepsilon^n(X^e)) = \{e_1, e_2 ..., e_p\}$ which contain v. Let u be a node in $\varepsilon^n(Y^e)$. i.e.,

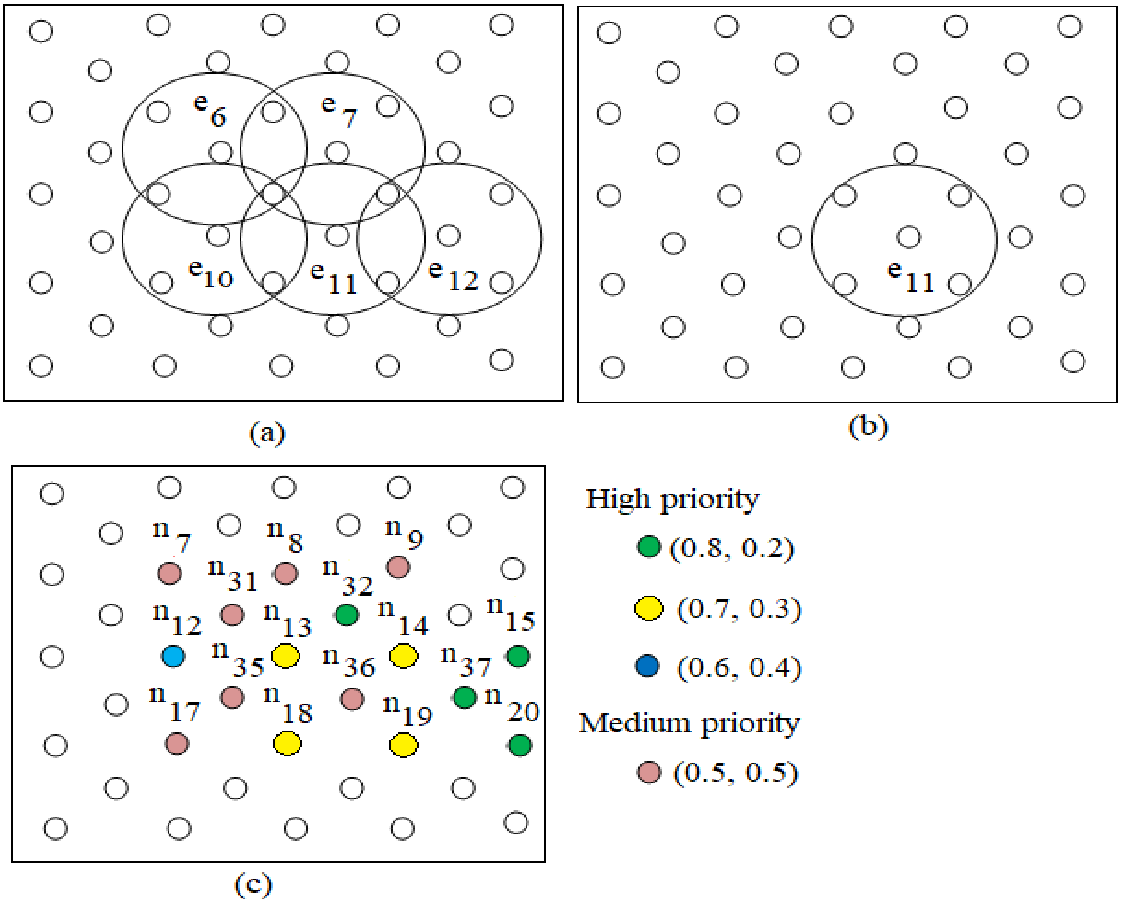
$$u \notin Y^{e'}; u \in Y^e. \tag{7.5}$$

Now $\delta^e(\varepsilon^n(Y)^e) = \{e_{p+1}, ..., e_k\}$: let e be an arbitrary edge in $\{e_1, e_2 ..., e_k\}$. This e can be either a member of $\delta^e(\varepsilon^n(X)^e)$ or $\delta^e(\varepsilon^n(Y)^e)$. i.e.,

$$e \subseteq \delta^e(\varepsilon^n(X)^e) \cup \delta^e(\varepsilon^n(Y^e)). \tag{7.6}$$

Hence Eq(7.1) is implied by eq(7.3) and eq(7.6).

Example 7.1 Consider the IFHGs given in Fig. 7.1. Let us find the R.H.S of eq(7.1), where we get $\varepsilon^n(X^e) = \{n_{31}, n_{35}, n_{36}\}$. Thus we get



(a) (b) (c)

FIGURE 7.2: (a) Proposition A (b)Proposition B (c)Proposition C

$\delta^e(\varepsilon^n(X^e)) = \{e_6, e_{10}, e_{11}\}$. Again in R.H.S of eq(7.1) we can find $\varepsilon^n(Y^e) = \{n_{32}, n_{36}, n_{37}\}$. Thus we get $\delta^e(\varepsilon^n(Y^e))$ as the set $\{e_7, e_{11}, e_{12}\}$. Now $\delta^e(\varepsilon^n(X)^e) \cup \delta^e(\varepsilon^n(Y^e)) = \{e_6, e_7, e_{10}, e_{11}, e_{12}\}$. The L.H.S of eq(7.1), $\delta^e(\varepsilon^n(X \cup Y)^e)$ gives the set $\{e_6, e_7, e_{10}, e_{11}, e_{12}\}$.

Proposition *B*: Let H be a parent IFHG, X and Y be sub-IFHGs, δ be the dilation operator and ε be the erosion operator, then

$$\delta^e(\varepsilon^n(X \cap Y)^e) = \delta^e(\varepsilon^n(X^e)) \cap \delta^e(\varepsilon^n(Y^e)), \tag{7.7}$$

where $\delta^e(\varepsilon^n(X \cap Y)^e)$ is an opening filter w.r.to hyperedges

Proof: Consider L.H.S of eq(7.7). Let v be an arbitrary node in $\varepsilon^n(X \cap Y)^e$. i.e.,

$$v \in (X \cap Y)^e; v \notin (X \cap Y)^{e'}. \tag{7.8}$$

Let e be the edge which contains this v. Now we can say that

$$e \subseteq \delta^e(\varepsilon^n(X \cap Y)^e). \tag{7.9}$$

Consider the R.H.S of eq(7.7). Let $\delta^e(\varepsilon^n(X^e)) = \{e_1, ..., e_p\}$. Let $\delta^e(\varepsilon^n(Y^e)) = \{e_{p+1}, ..., e_k\}$. There will be an edge e in $\{e_1, e_2, ..., e_k\}$ which will be a member of both $\delta^e(\varepsilon^n(X^e))$ and $\delta^e(\varepsilon^n(Y^e))$. i.e.,

$$e \subseteq \delta^e(\varepsilon^n(X^e)) \cap \delta^e(\varepsilon^n(Y^e)). \tag{7.10}$$

Eq(7.7) is implied by eq(7.9) and eq(7.10).

Example 7.2 Consider the IFHGs in Fig. 7.1. Let us find the R.H.S of eq(7.7), where we get $\delta^e(\varepsilon^n(X^e)) = \{e_6, e_{10}, e_{11}\}$. Also $\delta^e(\varepsilon^n(Y^e)) = \{e_7, e_{11}, e_{12}\}$. Now $\delta^e(\varepsilon^n(X^e)) \cap \delta^e(\varepsilon^n(Y^e)) = \{e_{11}\}$. Now considering L.H.S of eq(7.7) we get $\varepsilon^n(X \cap Y)^e = \{n_{28}\}$. Thus $\delta^e(\varepsilon^n(X \cap Y)^e) = \{e_{11}\}$.

Proposition *C***:** Let H be a parent IFHG, X and Y be sub-IFHGs, δ be the dilation operator and ε be the erosion operator, then $(\delta \circ \varepsilon)$ is defined as:

$$\delta^n(\varepsilon^e(X \cup Y)^n) = \delta^n(\varepsilon^e(X^n) \cup \delta^n(\varepsilon^e(Y^n), \tag{7.11}$$

which is an opening filter w.r.to nodes.

Consider L.H.S of eq(7.11). Let e be an arbitrary edge in $\varepsilon^e(X \cup Y)^n)$. Let v be an arbitrary node in e. By definition of δ^n, this v can be a node either of X or of Y. i.e., $v \in X$ or $v \in Y$. Consider R.H.S of eq(7.11). Let $\{v_1, v_2.....v_p\}$ be the nodes in $\delta^n(\varepsilon^e(X^n)$. By definition of δ^n , a node v is an edge of X. i.e., $v \in X$. Let $\{v_{p+1}, v_{p+2}.....v_k\}$ be the nodes in $\delta^n(\varepsilon^e(Y^n)$. By definition of δ^n, a node v is an edge of Y. i.e., $v \in Y$ Let v be a node in $\{v_1, v_2.....v_k\}$. Thus $v \in X$ or $v \in Y$. This implies eq(7.11).

Example 7.3 Consider IFHGs in Fig. 7.1. Let us find the R.H.S of eq(7.3), where we get $\varepsilon^e(X^n)$ is the set of hyperedges $\{e_6, e_{10}, e_{11}\}$. Thus $\delta^n(\varepsilon^e(X^n)$ is the set of nodes $\{n_7, n_8, n_{12}, n_{13}, n_{14}, n_{17}, n_{18}, n_{19}, n_{31}, n_{35}, n_{36}\}$. Again $\varepsilon^e(Y^n)$ is the set of edges $\{e_7, e_{11}, e_{12}\}$. Thus we get $\delta^n(\varepsilon^e(Y^n)$ as the set of nodes $\{n_8, n_9, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{32}, n_{36}, n_{37}\}$. Thus $\delta^n(\varepsilon^e(X^n) \cup \delta^n(\varepsilon^e(Y^n)$ gives the set $\{n_7, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{15}, n_{17}, n_{18}, n_{19}, n_{20}, n_{31}, n_{32}, n_{35}, n_{36}, n_{37}\}$. Considering L.H.S, we get $\varepsilon^e(X \cup Y)^n$ as the set of hyperedges $\{e_6, e_7, e_{10}, e_{11}, e_{12}\}$. Thus $\delta^n(\varepsilon^e(X \cup Y)^n)$ results in the set of nodes $\{n_7, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{15}, n_{17}, n_{18}, n_{19}, n_{20}, n_{31}, n_{32}, n_{35}, n_{36}, n_{37}\}$.

Proposition *D***:** Let H be the parent IFHG, X, Y be the sub-IFHGs, δ be the dilation operator and ε be the erosion operator. Then

$$\delta^n(\varepsilon^e(X \cap Y)^n) \subset (\delta^n(\varepsilon^e(X^n)) \cap \delta^n(\varepsilon^e(Y^n))). \tag{7.12}$$

Example 7.4: Consider the IFHGs given in Fig. 7.1. Let us find the L.H.S of eq(7.12). Here $\varepsilon^e(X \cap Y)^n$ is the set of edges which contains $(X \cap Y)^n$ only. i.e., $\varepsilon^e(X \cap Y)^n$ is the set $\{e_{11}\}$. Now $\delta^n(\varepsilon^e(X \cap Y)^n)$ is the set of nodes within e_{11}. Thus $\delta^n(\varepsilon^e(X \cap Y)^n) = \{n_{13}, n_{14}, n_{18}, n_{19}\}$. Now consider R.H.S of eq(7.12). Here $\delta^n(\varepsilon^e(X^n))$ is the set $\{n_7, n_8, n_{12}, n_{13}, n_{14}, n_{17}, n_{18}, n_{19}, n_{26}, n_{27}, n_{28}\}$. Also $\delta^n(\varepsilon^e(Y^n)) = \{n_8, n_9, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{28}, n_{29}, n_{30}\}$. Thus $\delta^n(\varepsilon^e(X^n)) \cap \delta^n(\varepsilon^e(Y^n)) = \{n_8, n_{13}, n_{14}, n_{18}, n_{19}, n_{28}\}$. Thus we can see that L.H.S $\subset$ R.H.S.

Proposition *E***:** Let H be the parent IFHG, X, Y be the sub-IFHGs, δ be the dilation operator and ε be the erosion operator. Then

$$\varepsilon^e(\delta^n(X \cup Y)^e) = \varepsilon^e(\delta^n(X^e)) \cup \varepsilon^e(\delta^n(Y^e)). \tag{7.13}$$

Proof: Consider the L.H.S of eq(7.13) Let v be an arbitrary node in $\delta^n(X \cup Y)^e$. According to the definition of δ^n, $v \in X$ or $v \in Y$. Let h be an edge which contains this v. By definition of ε^e, this $h \in X$ or $h \in Y$. Consider R.H.S of eq(7.13). Let $\{h_1, h_2....h_p\}$ be the edges in $\varepsilon^e(\delta^n(X^e))$. By definition of ε^e, an edge h in $\{h_1, h_2....h_p\}$ is an element of x. i.e., $h \in X$. Let $\{h_{p+1}, h_{p+2}....h_k\}$ be edges in $\varepsilon^e(\delta^n(Y^e))$. By definition of ε^e, an edge h in $\{h_1, h_2....h_k\}$ will be edge of X or Y. i.e., $h \in X$ or $h \in Y$. Eq(7.13) is implied by this.

Example 7.5: Consider the IFHGs given in Fig.7.1. In L.H.S of eq(7.13), $\delta^n(X \cup Y)^e$ is the set of nodes within $(X \cup Y)^e$. i.e., $\delta^n(X \cup Y)^e = \{n_7, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{15}, n_{17}, n_{18}, n_{19}, n_{20}, n_{31}, n_{32}, n_{35}, n_{36}, n_{37}\}$.

Now $\varepsilon^e(\delta^n(X \cup Y)^e) = \{e_6, e_7, e_{10}, e_{11}, e_{12}\}$. Consider R.H.S of eq(7.13). Here $\delta^n(X^e)$ is the nodes within X^e. i.e., $\delta^n(X^e)$ is the set of nodes $\{n_7, n_8, n_9, n_{12}, n_{13}, n_{14}, n_{15}, n_{17}, n_{18}, n_{19}, n_{31}, n_{35}, n_{36}\}$. Thus $\varepsilon^e(\delta^n(X^e))$ is the set of edges which consists of these nodes only. i.e., $\varepsilon^e(\delta^n(X^e)) = \{e_6, e_{10}, e_{11}\}$. Now $\delta^n(Y^e)$ is the nodes in Y^e. i.e., $\varepsilon^e(\delta^n(Y^e))$ is the set $\{n_8, n_9, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{32}, n_{36}, n_{37}\}$. Thus $\varepsilon^e(\delta^n(Y^e))$ is the set of edges which contains the above nodes only. i.e., $\varepsilon^e(\delta^n(Y^e)) = \{e_7, e_{11}, e_{12}\}$. Therefore $\varepsilon^e(\delta^n(X^e)) \cup \varepsilon^e(\delta^n(Y^e)) = \{e_6, e_{10}, e_{11}\} \cup \{e_7, e_{11}, e_{12}\} = \{e_6, e_7, e_{10}, e_{11}, e_{12}\}$.

Proposition F**:** Let H be the parent IFHG, X, Y be the sub-IFHGs, δ be the dilation operator and ε be the erosion operator. Then

$$\varepsilon^e(\delta^n(X \cap Y)^e) = \varepsilon^e(\delta^n(X^e)) \cap \varepsilon^e(\delta^n(Y^e)), \tag{7.14}$$

where $\varepsilon^e(\delta^n(X \cap Y)^e)$ is a closing filter w.r.to hyperedges.

Proof: Consider L.H.S of eq(7.14). Let v be a node in $\delta^n(X \cap Y)^e$. By definition of δ^n, it is a node in both X and Y. Let e be an edge which contains this v. By definition of ε^e, this is an edge in both X and Y. i.e., $e \in X$; $e \in Y$. Consider R.H.S of eq(7.14). Let $\{u_1, u_2....u_p\}$ be nodes in $\delta^n(X^e)$. By definition of δ^n, these nodes are in X. Let $\{e_1, e_2......e_p\}$ be the edges which consist of these nodes. By definition of ε^e, these edges are in X. Let $\{u_{p+1}, u_{p+2}....u_k\}$ be nodes in $\delta^n(Y^e)$. By definition of δ^n, these nodes are in Y. Let $\{e_{p+1}, e_{p+2}......e_k\}$ be the edges which consist of these nodes. By definition of ε^e, these edge is in Y. Now an edge e in $\{e_1, e_2......e_k\}$ is present both in X and in Y; i.e., $e \in X$, $e \in Y$.

Example 7.6: Consider L.H.S of eq(7.14). There $\delta^n(X \cap Y)^e$ yields the set of nodes $\{n_{13}, n_{14}, n_{18}, n_{19}, n_{36}\}$. Now $\varepsilon^e(\delta^n(X \cap Y)^e)$ is the set of edges which contains the above nodes only. i.e., $\varepsilon^e(\delta^n(X \cap Y)^e) = \{e_{11}\}$. Consider R.H.S of eq(7.14). $\delta^n(X^e)$ is the set of nodes in X^e. i.e., $\delta^n(X^e) = \{n_7, n_8, n_{12}, n_{13}, n_{14}, n_{17}, n_{18}, n_{19}, n_{31}, n_{35}, n_{36}\}$. Now $\varepsilon^e(\delta^n(X^e))$ is the edges which contain the above nodes only. i.e., $\varepsilon^e(\delta^n(X^e)) = \{e_6, e_{10}, e_{11}\}$. Likewise $\delta^n(Y^e)$ is the set of nodes in Y^e only which gives the set $\{n_8, n_9, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{32}, n_{36}, n_{37}\}$. Now $\varepsilon^e(\delta^n(Y^e)$ is the set of edges which contains these nodes only. i.e., $\varepsilon^e(\delta^n(Y^e) = \{e_7, e_{11}, e_{12}\}$. Now $\varepsilon^e(\delta^n(X^e)) \cap \varepsilon^e(\delta^n(Y^e)) = \{e_{11}\}$.

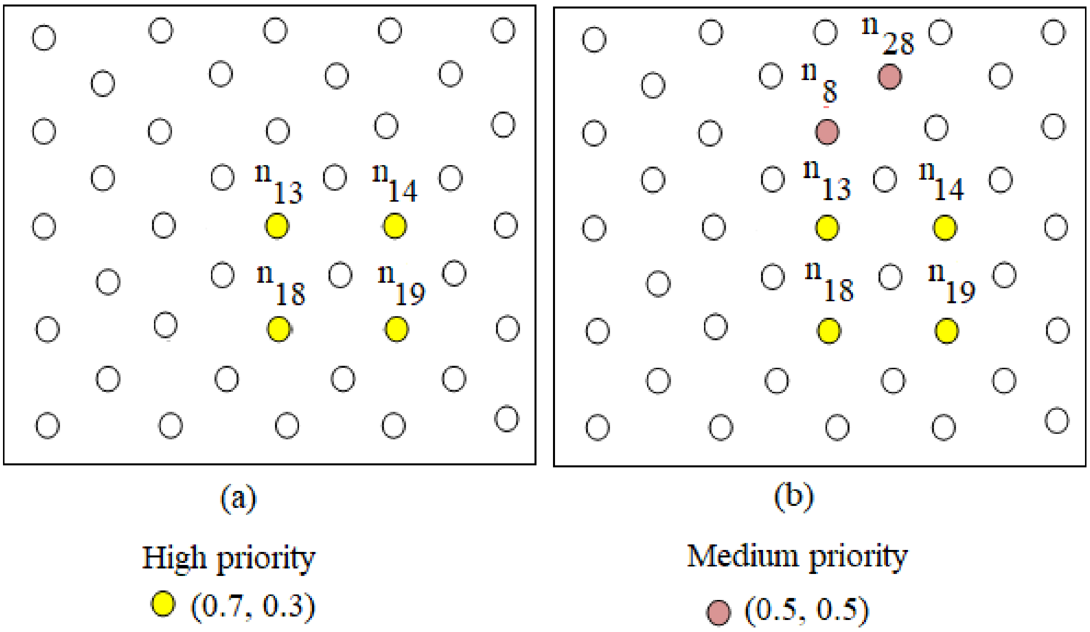


FIGURE 7.3: Proposition D

Proposition G**:** Let H be the parent IFHG, X, Y be the sub-IFHGs, δ be the dilation operator and ε be the erosion operator. Then

$$\varepsilon^n(\delta^e(X \cup Y)^n) = \varepsilon^n(\delta^e(X^n)) \cup \varepsilon^n(\delta^e(Y^n)), \tag{7.15}$$

where $\varepsilon^n(\delta^e(X \cup Y)^n)$ is a closing filter w.r.to nodes.

Proof: Consider L.H.S of eq(7.15). Let $\{e_1, e_2...., e_k\}$ be the edges in $\delta^e(X \cup Y)^n$. Let v be an arbitrary node in $\{e_1, e_2...., e_k\}$. According to the definition of $\varepsilon^n(\delta^e(X \cup Y)^n))$, this v should be contained in $\{e_1, e_2...., e_k\}$ but not in $\{e_1, e_2...., e_k\}'$. Consider R.H.S of eq(7.15). Let $\{e_1, e_2...., e_p\}$ be the edges in $\delta^e(X^n)$. Let $\{n_1, n_2, ..., n_p\}$ be the nodes in $\{e_1, e_2..., e_p\}$. According to the definition of $\varepsilon^n(\delta^e(X^n))$ these nodes are present in $\{e_1, e_2, .., e_p\}$ but not in $\{e_1, e_2, .., e_p\}'$. Let $\{e_{p+1}, e_{p+2}..., e_k\}$ be the edges in $\delta^e(Y^n)$. Let $\{n_{p+1}, n_{p+2}, .., n_k\}$ be the nodes in $\{e_{p+1}, e_{p+2}, .., e_k\}$. According to the definition of $\varepsilon^n(\delta^e(Y^n))$ these nodes are present in $\{e_{p+1}, e_{p+2}..., e_k\}$ but not in $\{e_{p+1}, e_{p+2}..., e_k\}'$. i.e., any node v in $\{e_1, e_2...., e_k\}$ is present in $\{e_1, e_2...., e_k\}$ but not in $\{e_1, e_2..., e_k\}$.

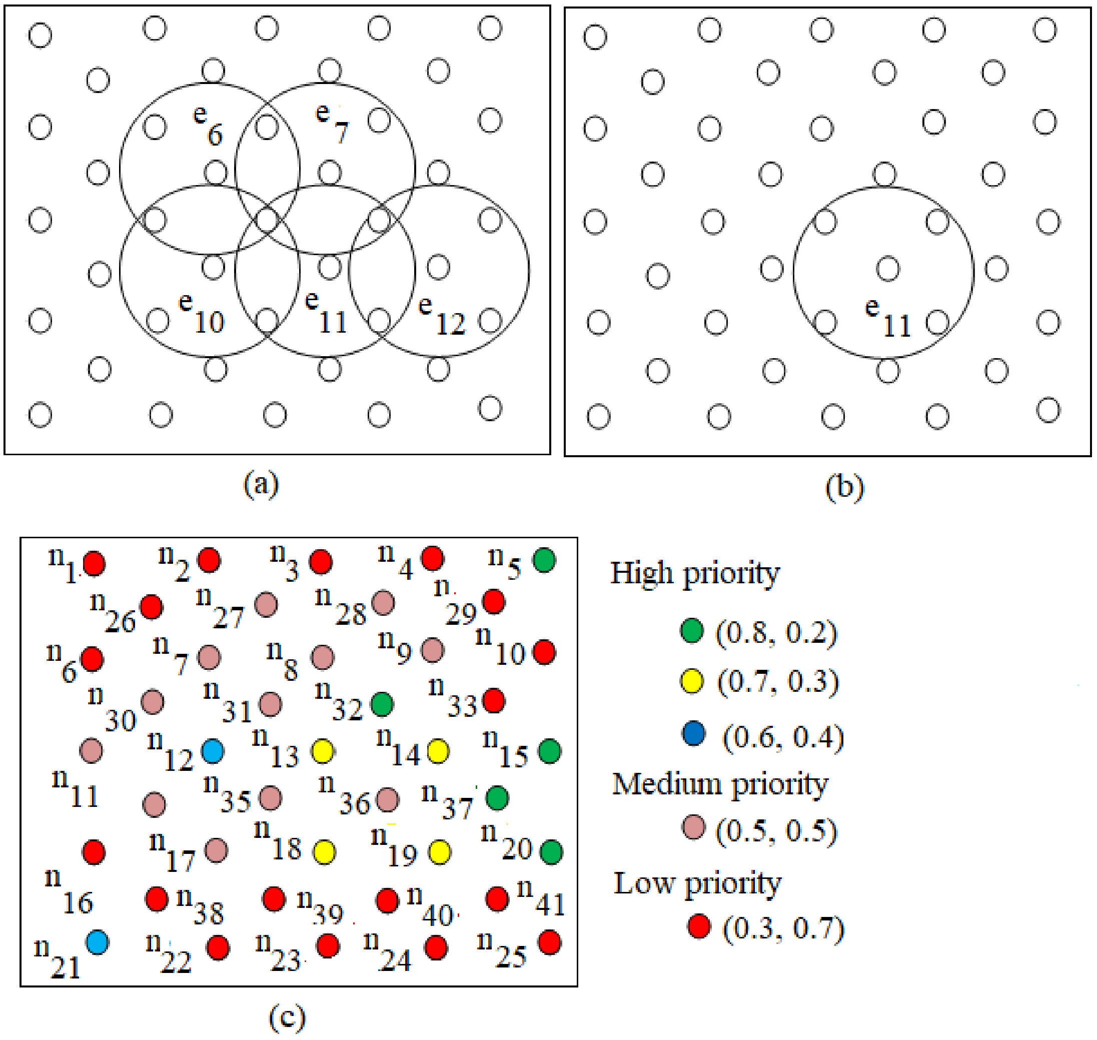


FIGURE 7.4: (a) Proposition E (b)Proposition F (c)Proposition G

Example 7.7: Consider L.H.S of eq(7.15). Here $\delta^e(X \cup Y)^n$ is the set of edges which contain nodes in $(X \cup Y)^n$ which gives the set $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\}$. Thus $\varepsilon^n(\delta^e(X \cup Y)^n)$ is the nodes in the above edges but not present in their complement edges = all nodes in H. Consider R.H.S of eq(7.15). Here $\delta^e(X^n)$ is the set of edges which contain nodes in X^n = all edges in H. Now $\varepsilon^n(\delta^e(X^n))$ is the set of nodes in above edges but not in their complement = all nodes in H. Now $\delta^e(Y^n)$ is the set of edges which contains nodes in Y^n which gives $\{e_2, e_3, e_4, e_6, e_7, e_8, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\}$. Hence $\varepsilon^n(\delta^e(Y^n))$ is the nodes in the above edges, but not in their complement which is the set $\{n_3, n_4, n_5, n_8, n_9, n_{10}, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}, n_{27}, n_{28}, n_{29}, n_{31}, n_{32}, n_{33}, n_{35}, n_{36}, n_{37}, n_{39}, n_{40}, n_{41}\}$. Hence $\varepsilon^n(\delta^e(X^n)) \cup \varepsilon^n(\delta^e(Y^n))$ = all nodes in H.

Proposition H: Let H be the parent IFHG, X, Y be the sub-IFHGs, δ be the dilation operator and ε be the erosion operator. Then

$$\varepsilon^n(\delta^e(X \cap Y)^n) \subset \varepsilon^n(\delta^e(X^n)) \cap \varepsilon^n(\delta^e(Y^n)), \tag{7.16}$$

where $\varepsilon^n(\delta^e(X \cap Y)^n)$ is a closing filter w.r.to nodes.

Example 7.8: Consider L.H.S of eq(7.16). Here $\delta^e(X \cap Y)^n$ is the set of edges which consists of any element in $(X \cap Y)^n = \{e_2, e_3, e_4, e_6, e_7, e_8, e_{10}, e_{11}, e_{12}\}$. Now $\varepsilon^n(\delta^e(X \cap Y)^n)$ is the set of nodes in the above edges but not in their complement edges which gives the set of nodes $\{n_3, n_4, n_5, n_8, n_9, n_{10}, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{27}, n_{28}, n_{29}, n_{31}, n_{32}, n_{33}, n_{35}, n_{36}, n_{37}\}$. Now consider R.H.S of eq(7.16). $\delta^e(X^n)$ is the set of edges which contain any node in X^n which gives all edges in H. Now $\varepsilon^n(\delta^e(X^n))$ = all nodes in H. Now $\delta^e(Y^n)$ is the set of edges which contain nodes in $Y^n = \{e_2, e_3, e_4, e_6, e_7, e_8, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\}$. Hence $\varepsilon^n(\delta^e(Y^n))$ is the nodes in the above edges, but not in their complement edges which gives the set $\{n_3, n_4, n_5, n_8, n_9, n_{10}, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}, n_{27}, n_{28}, n_{29}, n_{31}, n_{32}, n_{33}, n_{35}, n_{36}, n_{37}, n_{39}, n_{40}, n_{41}\}$. Hence $\varepsilon^n(\delta^e(X^n)) \cap \varepsilon^n(\delta^e(Y^n))$ is the set $\{n_3, n_4, n_5, n_8, n_9, n_{10}, n_{13}, n_{14}, n_{15}, n_{18}, n_{19}, n_{20}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}, n_{27}, n_{28}, n_{29}, n_{31}, n_{32}, n_{33}, n_{35}, n_{36}, n_{37}, n_{39}, n_{40}, n_{41}\}$.

7.2 Application of filter in multi-document summary

Multi-document summarization is an important area in the field of NLP. There are many summarization methods developed so far. In this section let us see how multiple documents can be represented using an IFHG assigning membership and non-membership values to the nodes and hyperedges.

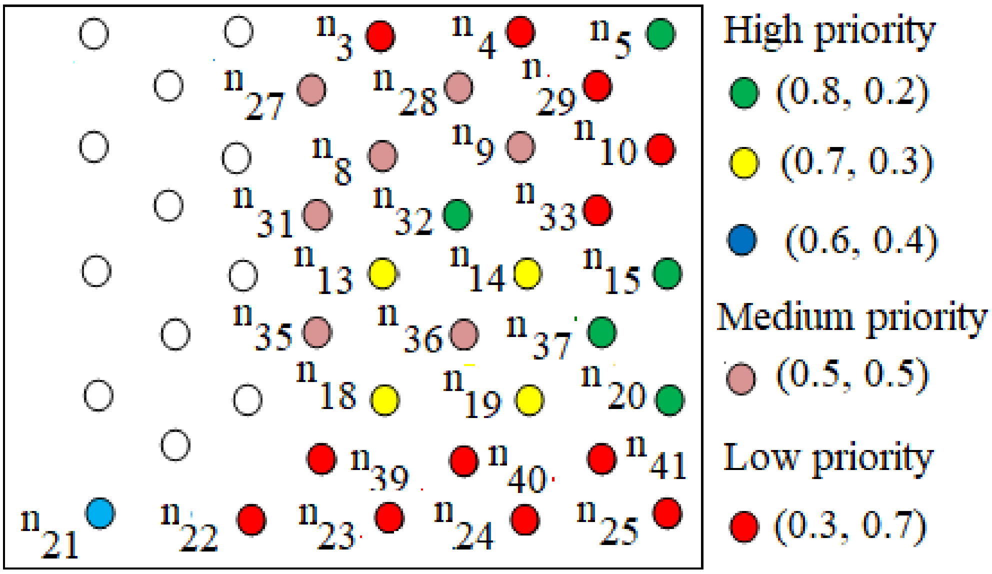


FIGURE 7.5: Proposition H

7.2.1 Modeling multiple documents using IFHG

Let $[H_{IF}, H^n, H^e, \mu_n, \gamma_n, \mu_e, \gamma_e]$ be an IFHG as mentioned in Section 7.1. Let H^e be the hyperedges, where a hyperedge e_i represents a document D_i. Let H^n be the nodes where a node v_j represents a keyword w_j in the document. The same is shown in Fig. 7.6. Therefore the number of hyperedges in the H_{IF} will be same as the number of documents considered for summarization and the number of nodes in a hyperedge will be same as the number of keywords in that document. The membership value for a node v_j depends on the normalized $TF - IDF$ value and the priority of that word. Irrespective of the domain there are words which are having high priority. For example words like *important*, *famous*, *beautiful*, *attractive*, *relevant* etc are given high priority. So these words will have $\mu_n > 0.5$. For such words $\gamma_n = 1 - \mu_n$. There are words which are having very low priority. Some sample words include *notorious*, *expensive*, *least*, *badly* etc. These words are having $\gamma_n < 0.5$. So for such words $\mu_n = 1 - \gamma_n$. Rest of the words are given $\mu_n = \gamma_n = 0.5$ as they are medium priority words. Once the nodes are assigned with both membership degree (μ_n) and non-membership degree (γ_n), the hyperedges are also assigned membership degree (μ_e) and non-membership degree (γ_e) as per the rules in Section 7.1.

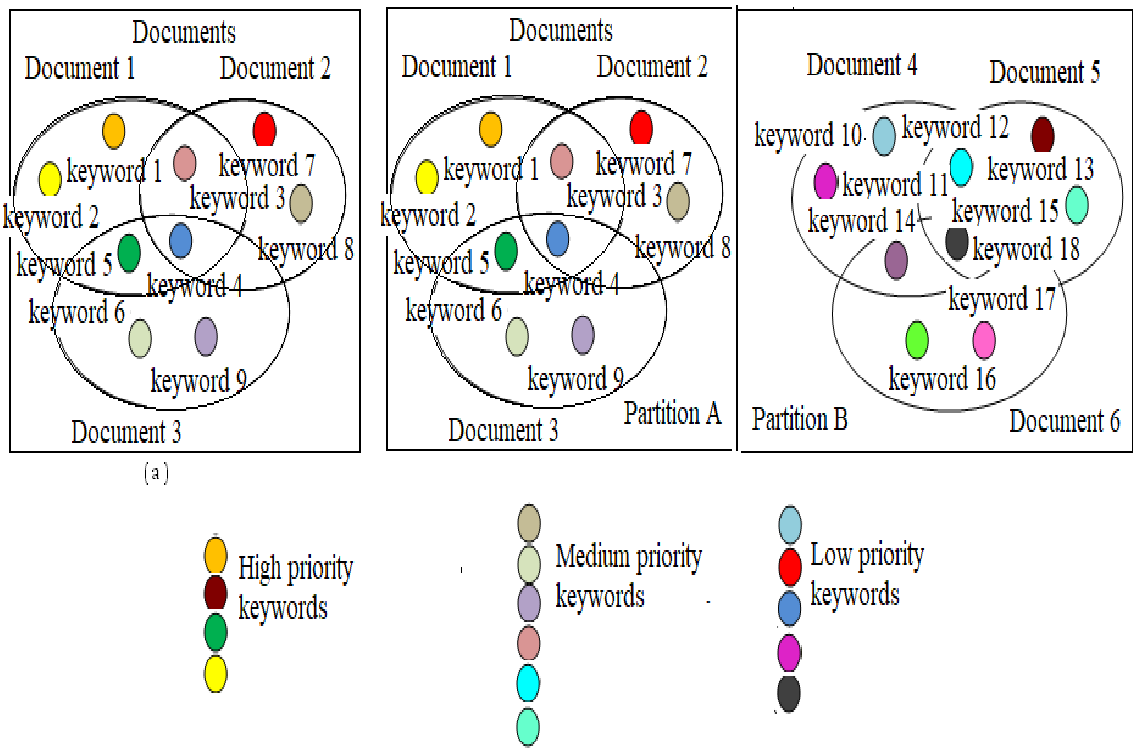


FIGURE 7.6: Multiple documents as IFHG

7.2.2 System architecture

For the formation of IFHG, all the documents are subjected to preprocessing steps like stop word removal and lemmatization. keywords in a document are found out by considering the $tf - idf$ of the words. Only words with $tf - idf$ above a threshold θ are considered for the construction of the IFHG. Once an IFHG is formed as mentioned in Section 7.2.1, it can be partitioned in to sub-IFHGs based on the absence of overlapping nodes between the hyperedges. Such partitions are now document clusters. For each cluster, sub-IFHG X_i is created by applying an (α, β) cut. This X_i is subjected to opening and closing filters as mentioned in Section 7.1. The results of filter w.r.to hyperedges will be yield good documents and results of filter w.r.to nodes will yield priority keywords. These priority words combined with priority documents yield good summaries. If needed, the number of parameters considered for assigning membership degree can be increased. The architecture of the system is shown in Fig. 7.7. The algorithm is given in Fig. 7.8.

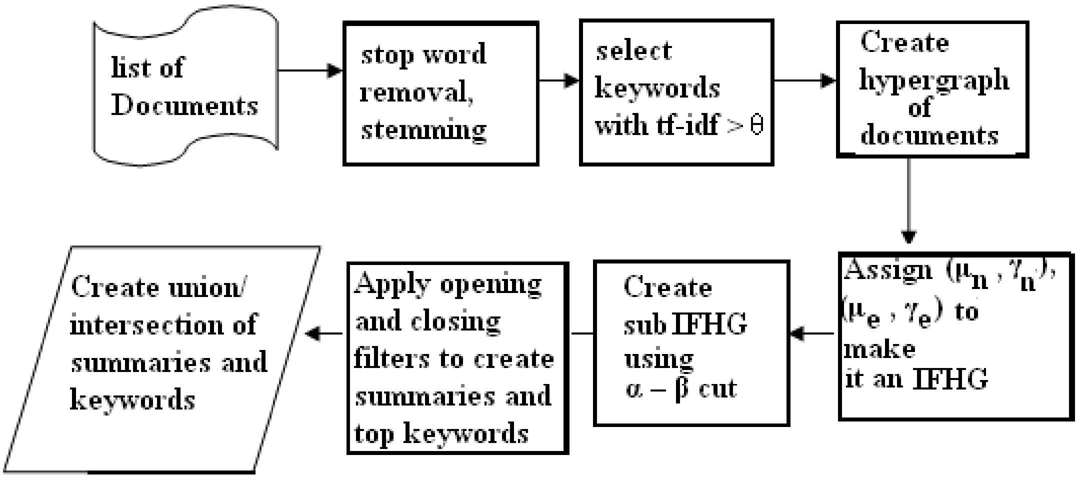
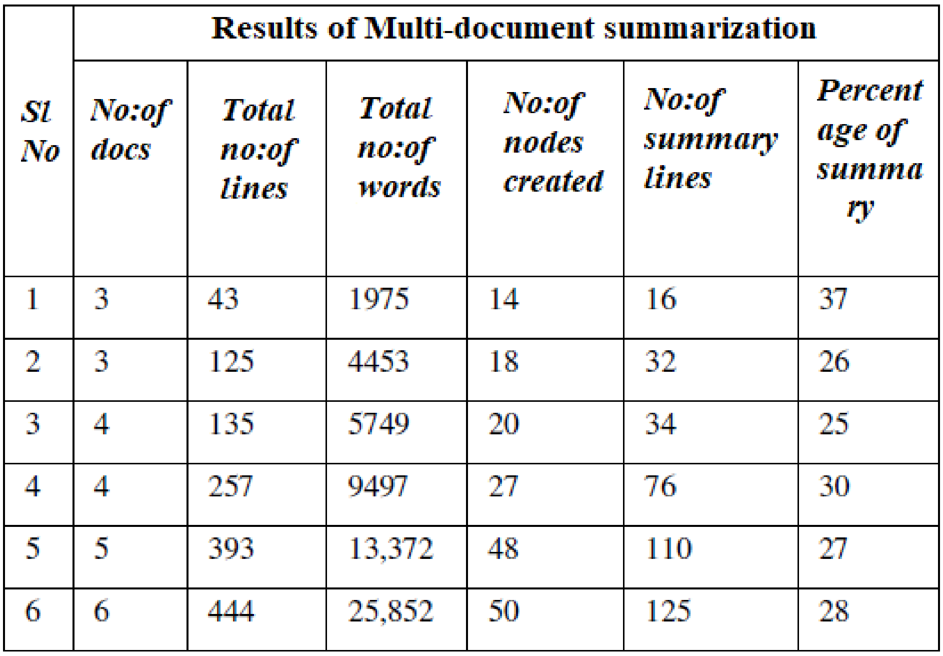

list of Documents
stop word removal, stemming
select keywords with tf-idf > θ
Create hypergraph of documents
Assign (μ_n, γ_n), (μ_e, γ_e) to make it an IFHG
Create sub IFHG using $\alpha - \beta$ cut
Apply opening and closing filters to create summaries and top keywords
Create union/ intersection of summaries and keywords

FIGURE 7.7: Architecture of multi-document summary

Algorithm - Multi-document summary using IFHG

```
Input : n documents, $\theta$, $\alpha_n$
Output : Summary
for each document $D_i$; $i = 1$ to $n$ do
    Remove stop words
    Perform Stemming of words
    Find $tf - idf$
    Find all words $w_j$ in $D_i$ with $tf - idf > \theta$
end for
$P$ = number of hypernodes = Total Number of words in all the $D_i$
$J$ = number of hyperedges = n;
for each $k = 1$ to $P$ do
    Assign $\mu_{n_k}$, $\gamma_{n_k}$ of all nodes as per rules
end for
for each $m = 1$ to $J$ do
    Assign $\mu_{e_k}$, $\gamma_{e_k}$ for all edges as per rules
end for
$t$ Clusters = Find sub IFHGs with non overlapping hypernodes
for each cluster $C_i$, $i = 1$ to $t$ do
    Find sub IFHG $X_i$ with $\alpha - \beta$ cut where $\alpha >= \alpha_n$
    Apply opening filter w.r.to nodes $\delta^n(\varepsilon^e(X_i)^n)$ to find keywords in summary.
    Apply opening filter w.r.to hyperedges $\delta^e(\varepsilon^n(X_i)^e)$ to find relevant documents.
    Find $Opening\_Summary\_Filter(C_i) = Combine(\delta^e(\varepsilon^n(X_i)^e), \delta^n(\varepsilon^e(X_i)^n))$
    Apply closing filter w.r.to nodes $\varepsilon^n(\delta^e(X_i)^n)$ to find keywords in summary.
    Apply closing filter w.r.to edges $\varepsilon^e(\delta^n(X_i)^e)$ to find relevant documents.
    Find $Closing\_Summary\_Filter(C_i) = Combine(\varepsilon^e(\delta^n(X_i)^e), \varepsilon^n(\delta^e(X_i)^n)$.
end for
for two clusters $C_i$, $C_j$ do
    Find $Opening\_Summary\_Filter(C_i) \cup Opening\_Summary\_Filter(C_j)$ to create union of two summaries
    Find $\delta^n(\varepsilon^e(X_i)^n) \cup \delta^n(\varepsilon^e(X_j)^n)$ to create union of keywords in summaries
    Find $Closing\_Summary\_Filter(C_i) \cup Closing\_Summary\_Filter(C_j)$ to create union of two summaries
    Find $\varepsilon^n(\delta^e(X_i)^n) \cup \varepsilon^n(\delta^e(X_j)^n)$ to find all the keywords of two summaries
end for
```

FIGURE 7.8: Algorithm for multi-document summary filter

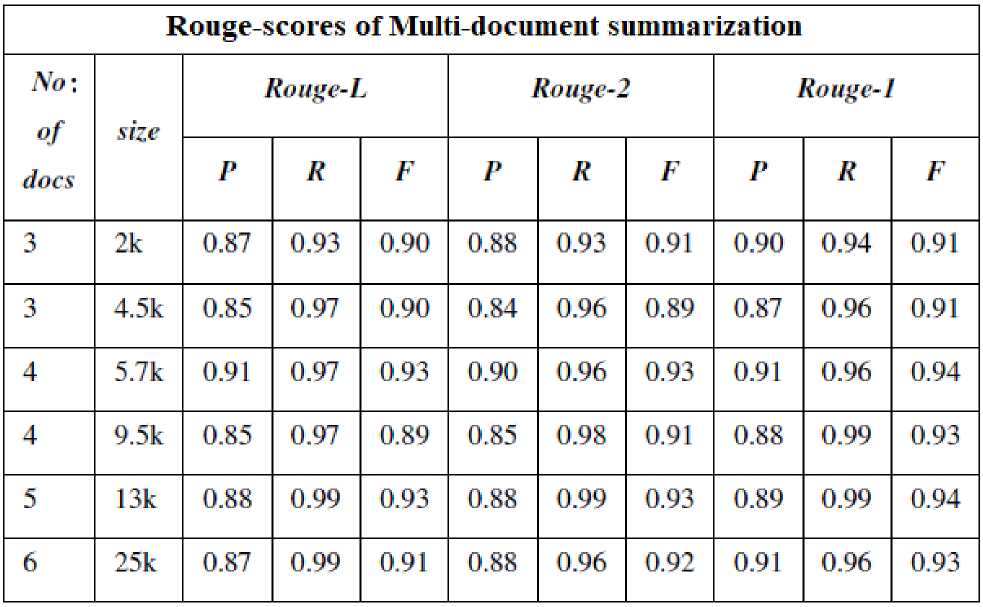


Sl No	Results of Multi-document summarization					
	No:of docs	Total no:of lines	Total no:of words	No:of nodes created	No:of summary lines	Percentage of summary
1	3	43	1975	14	16	37
2	3	125	4453	18	32	26
3	4	135	5749	20	34	25
4	4	257	9497	27	76	30
5	5	393	13,372	48	110	27
6	6	444	25,852	50	125	28

Figure 7.9: Summary results

7.3 Results of multi-document Summary

The proposed system with multiple documents modeled as IFHG was tested with different number of input documents. The system developed in python, and the hypergraph created using python pygraph works for both Malayalam and English documents. The difference between two systems lie in the stemming phase, where in Malayalam, a stemmer developed using tree based method [106] is used. Porter stemmer is being used for English documents. Rests of the modules are common for Malayalam and English summarization system. The results obtained with various test cases are given in Fig. 7.9. The results of the IFHG system is compared with results of 50 human summarizers and Rouge-L, Rouge-2, Rouge-1 scores are calculated and shown in Fig. 7.10.



Rouge-scores of Multi-document summarization										
No: of docs	***size***	***Rouge-L***			***Rouge-2***			***Rouge-1***		
		P	***R***	***F***	***P***	***R***	***F***	***P***	***R***	***F***
3	2k	0.87	0.93	0.90	0.88	0.93	0.91	0.90	0.94	0.91
3	4.5k	0.85	0.97	0.90	0.84	0.96	0.89	0.87	0.96	0.91
4	5.7k	0.91	0.97	0.93	0.90	0.96	0.93	0.91	0.96	0.94
4	9.5k	0.85	0.97	0.89	0.85	0.98	0.91	0.88	0.99	0.93
5	13k	0.88	0.99	0.93	0.88	0.99	0.93	0.89	0.99	0.94
6	25k	0.87	0.99	0.91	0.88	0.96	0.92	0.91	0.96	0.93

Figure 7.10: Rouge-scores

7.4 Conclusion

This chapter has presented a novel method which models multiple documents using IFHG. While chapter 5 has modeled sentences as hyperedges and words as nodes, this chapter has extended it to multiple documents by treating documents as hyperedges and words as nodes. The system is tested with documents of varying size and has shown better results when compared to human summaries with a time complexity of $O(n^2)$. Documents in various domains are considered for summarization. The priority levels, (α, β) cuts, range oriented (α, β) cuts and union / intersection operations applied on the filters give different types of summaries suitable for different applications. Here IFHG takes only less space since care is taken to reduce the number of nodes. We have also developed a system which models sentences as hyperedges and words as nodes. Such IFHG modeling and filtering can be done in other areas like mobile networking, social networking, image processing etc. Modeling medical reports as IFHG and creating medical report summary is a future enhancement of this work.

Chapter 8

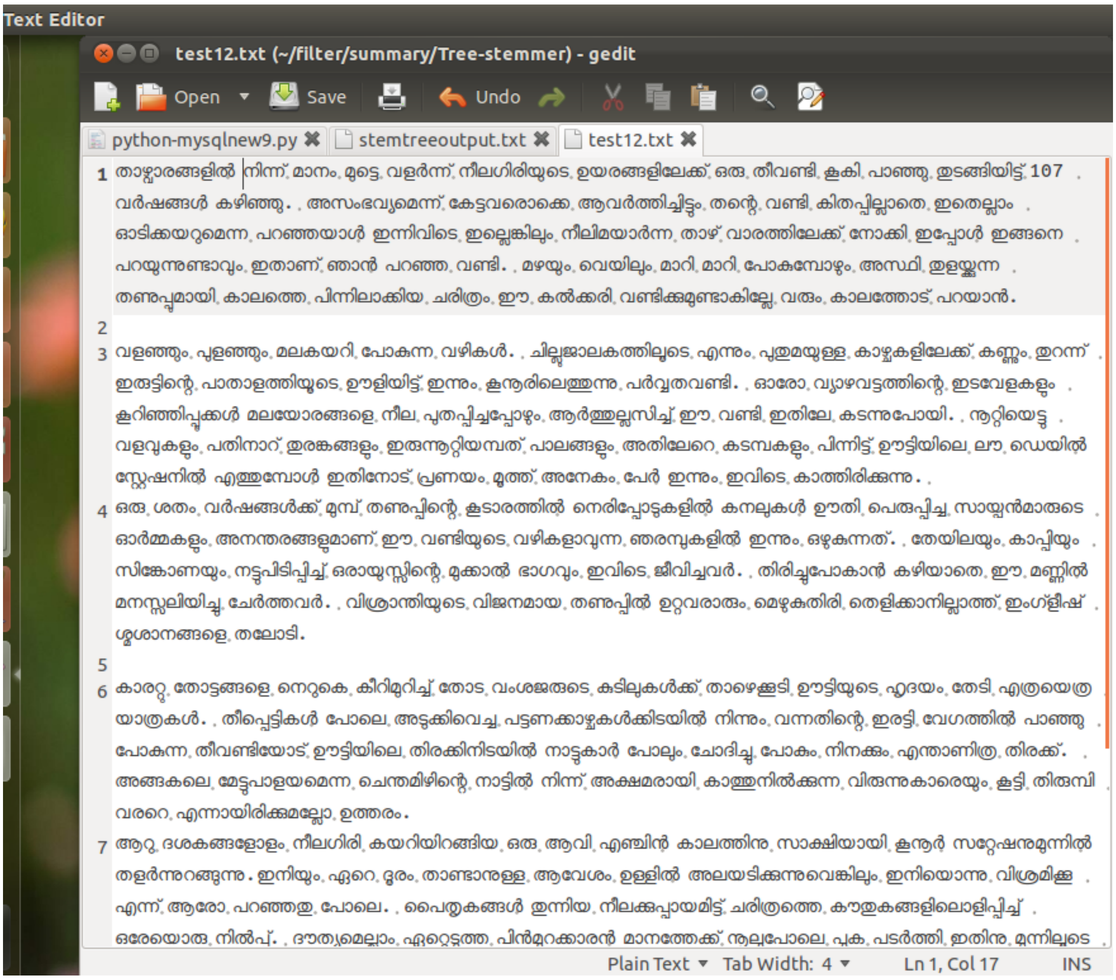
Results and Screenshots

Here two systems are developed, one which summarizes Malayalam documents and another which summarizes English documents. Both the systems have the same sub modules like preprocessing, stemming, clustering and summary generation.

The main difference between the two systems lies in the stemming phase.

8.1 Lemmatization phase

In the case of English summary system, the stemmer used is porter stemmer, while in the case of Malayalam summary system, a tree based Malayalam lemmatizer is developed.



താഴ്വാരങ്ങളിൽ നിന്ന് മാനം മുട്ടെ വളർന്ന് നീലഗിരിയുടെ ഉയരങ്ങളിലേക്ക് ഒരു തീവണ്ടി കൂകി പാഞ്ഞു തുടങ്ങിയിട്ട് 107 വർഷങ്ങൾ കഴിഞ്ഞു. അസംഭവ്യമെന്ന് കേട്ടവരൊക്കെ ആവർത്തിച്ചിട്ടും തന്റെ വണ്ടി കിതപ്പില്ലാതെ ഇതെല്ലാം ഓടിക്കയറുമെന്ന പറഞ്ഞയാൾ ഇന്നിവിടെ ഇല്ലെങ്കിലും നീലിമയാർന്ന താഴ് വാരത്തിലേക്ക് നോക്കി ഇപ്പോൾ ഇങ്ങനെ പറയുന്നുണ്ടാവും. ഇതാണ് ഞാൻ പറഞ്ഞ വണ്ടി. മഴയും വെയിലും മാറി മാറി പോകുമ്പോഴും അസ്ഥി തുളയ്ക്കുന്ന തണുപ്പുമായി കാലത്തെ പിന്നിലാക്കിയ ചരിത്രം ഈ കൽക്കരി വണ്ടിക്കുമുണ്ടാകില്ലേ വരും കാലത്തോട് പറയാൻ.

വളഞ്ഞും പുളഞ്ഞും മലകയറി പോകുന്ന വഴികൾ. ചില്ലുജാലകത്തിലൂടെ എന്നും പുതുമയുള്ള കാഴ്ചകളിലേക്ക് കണ്ണും തുറന്ന് ഇരുട്ടിന്റെ പാതാളത്തിയൂടെ ഊളിയിട്ട് ഇന്നും കൂനൂരിലെത്തുന്ന പർവ്വതവണ്ടി. ഓരോ വ്യാഴവട്ടത്തിന്റെ ഇടവേളകളും കുറിഞ്ഞിപ്പൂക്കൾ മലയോരങ്ങളെ നീല പുതപ്പിച്ചപ്പോഴും ആർത്തുല്ലസിച്ച് ഈ വണ്ടി ഇതിലേ കടന്നുപോയി. നൂറ്റിയെട്ട് വളവുകളും പതിനാറ് തുരങ്കങ്ങളും ഇരുന്നൂറ്റിയമ്പത് പാലങ്ങളും അതിലേറെ കടമ്പകളും പിന്നിട്ട് ഊട്ടിയിലെ ലൗ ഡെയിൽ സ്റ്റേഷനിൽ എത്തുമ്പോൾ ഇതിനോട് പ്രണയം മൂത്ത് അനേകം പേർ ഇന്നും ഇവിടെ കാത്തിരിക്കുന്നു.

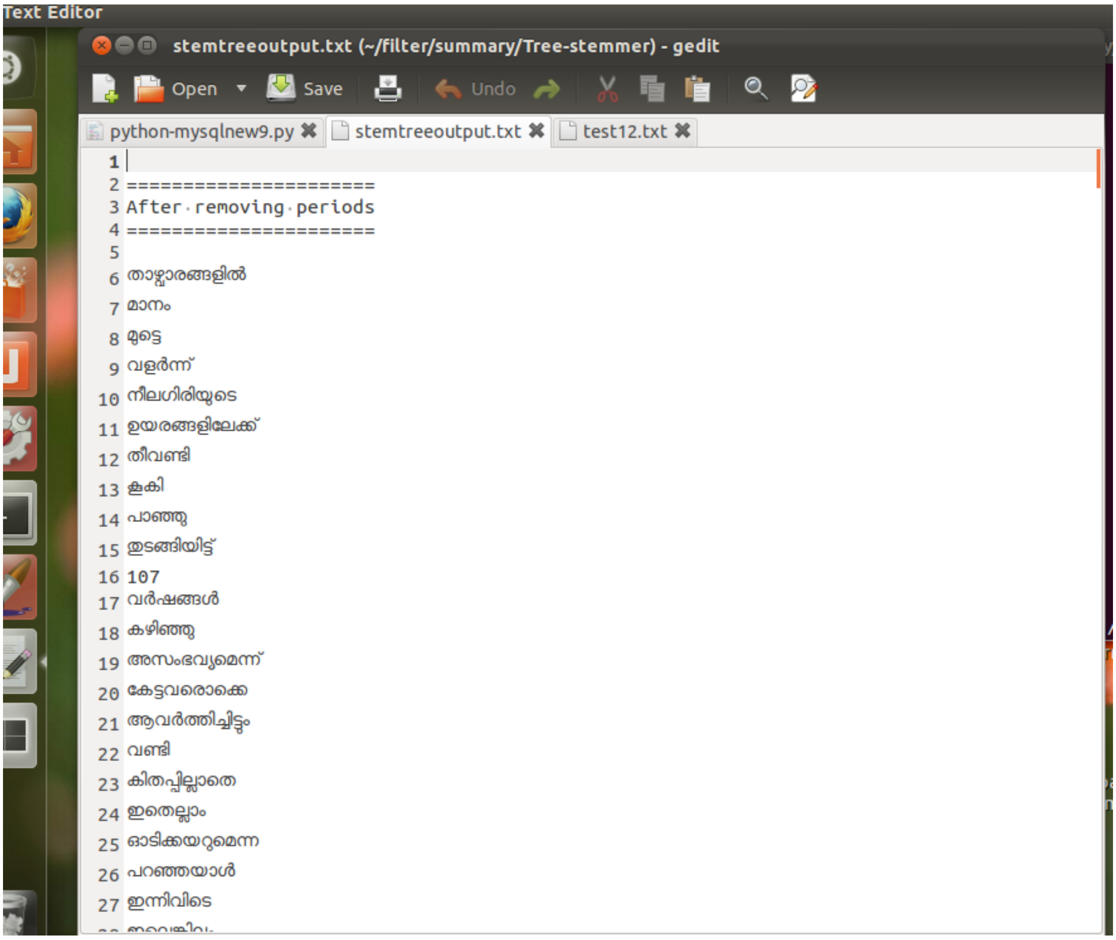
ഒരു ശതം വർഷങ്ങൾക്ക് മുമ്പ് തണുപ്പിന്റെ കൂടാരത്തിൽ നെരിപ്പോട്ടുകളിൽ കനലുകൾ ഊതി പെരുപ്പിച്ച സായ്പൻമാരുടെ ഓർമ്മകളും അനന്തരങ്ങളുമാണ് ഈ വണ്ടിയുടെ വഴികളാവുന്ന ഞരമ്പുകളിൽ ഇന്നും ഒഴുകുന്നത്. തേയിലയും കാപ്പിയും സിങ്കോണയും നട്ടുപിടിപ്പിച്ച് ഒരായുസ്സിന്റെ മുക്കാൽ ഭാഗവും ഇവിടെ ജീവിച്ചവർ. തിരിച്ചുപോകാൻ കഴിയാതെ ഈ മണ്ണിൽ മനസ്സലിയിച്ചു ചേർത്തവർ. വിശ്രാന്തിയുടെ വിജനമായ തണുപ്പിൽ ഉറ്റവരാരും മെഴുകുതിരി തെളിക്കാനില്ലാത്ത് ഇംഗ്ലീഷ് ശ്മശാനങ്ങളെ തലോടി.

കാരറ്റ് തോട്ടങ്ങളെ നെറുകെ കീറിമുറിച്ച് തോട വംശജരുടെ കുടിലുകൾക്ക് താഴെക്കൂടി ഊട്ടിയുടെ ഹൃദയം തേടി എത്രയെത്ര യാത്രകൾ. തീപ്പെട്ടികൾ പോലെ അടുക്കിവെച്ച പട്ടണക്കാഴ്ചകൾക്കിടയിൽ നിന്നും വന്നതിന്റെ ഇരട്ടി വേഗത്തിൽ പാഞ്ഞു പോകുന്ന തീവണ്ടിയോട് ഊട്ടിയിലെ തിരക്കിനിടയിൽ നാട്ടുകാർ പോലും ചോദിച്ചു പോകും നിനക്കും എന്താണിത്ര തിരക്ക്. അങ്ങകലെ മേട്ടുപാളയമെന്ന ചെന്തമിഴിന്റെ നാട്ടിൽ നിന്ന് അക്ഷമരായി കാത്തുനിൽക്കുന്ന വിരുന്നുകാരെയും കൂട്ടി തിരുമ്പി വരറെ എന്നായിരിക്കുമല്ലോ ഉത്തരം.

ആറു ദശകങ്ങളോളം നീലഗിരി കയറിയിറങ്ങിയ ഒരു ആവി എഞ്ചിൻ കാലത്തിനു സാക്ഷിയായി കൂനൂർ സ്റ്റേഷനുമുന്നിൽ തളർന്നുറങ്ങുന്നു. ഇനിയും ഏറെ ദൂരം താണ്ടാനുള്ള ആവേശം ഉള്ളിൽ അലയടിക്കുന്നുവെങ്കിലും ഇനിയൊന്നു വിശ്രമിക്കു എന്ന് ആരോ പറഞ്ഞതു പോലെ. പൈതൃകങ്ങൾ തുന്നിയ നീലക്കുപ്പായമിട്ട് ചരിത്രത്തെ കൗതുകങ്ങളിലൊളിപ്പിച്ച് ഒരേയൊരു നിൽപ്പ്. ദൗത്യമെല്ലാം ഏറ്റെടുത്ത പിൻമുറക്കാരൻ മാനത്തേക്ക് നൂലുപോലെ പുക പടർത്തി ഇതിനു മുന്നിലൂടെ

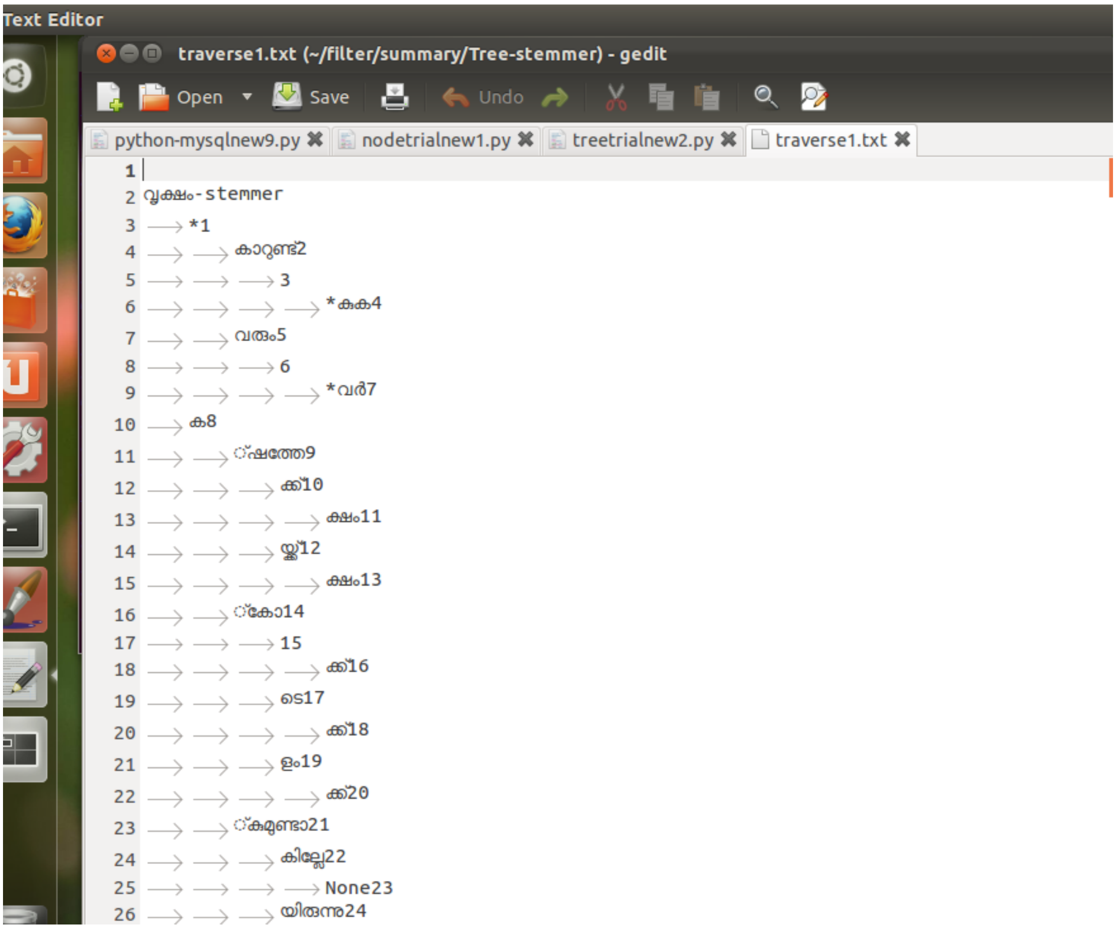
FIGURE 8.1: Malayalam input file

Fig. 8.1 shows a Malayalam input file which is input to the system. Preprocessing of the document is done which involves removal of periods, commas, white spaces, stop words etc. Tokenization and removal of periods result in an output shown in Fig. 8.2. A snapshot of stopword file is shown in fig. 8.3.



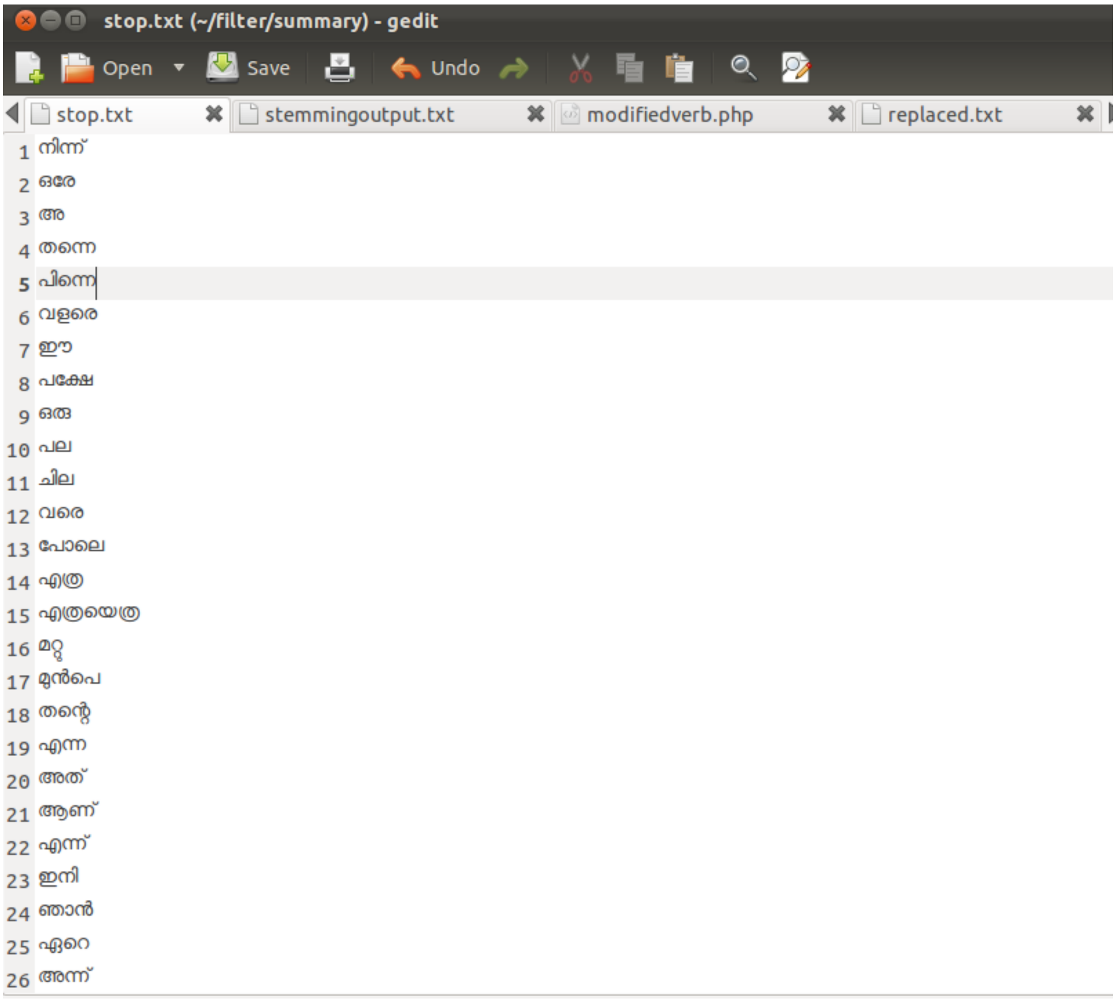
```

======================
After·removing·periods
======================

താഴ്വാരങ്ങളിൽ
മാനം
മുട്ടെ
വളർന്ന്
നീലഗിരിയുടെ
ഉയരങ്ങളിലേക്ക്
തീവണ്ടി
കൂകി
പാഞ്ഞു
തുടങ്ങിയിട്ട്
107
വർഷങ്ങൾ
കഴിഞ്ഞു
അസംഭവ്യമെന്ന്
കേട്ടവരൊക്കെ
ആവർത്തിച്ചിട്ടും
വണ്ടി
കിതപ്പില്ലാതെ
ഇതെല്ലാം
ഓടിക്കയറുമെന്ന
പറഞ്ഞയാൾ
ഇന്നിവിടെ
```

Figure 8.2: Removal of periods



```

വൃക്ഷം-stemmer
→ *1
→ → കാറുണ്ട്2
→ → → 3
→ → → → *കക4
→ → വരും5
→ → → 6
→ → → → *വർ7
→ ക8
→ → ്ഷത്തെ9
→ → → ക്ക്10
→ → → → ക്ഷം11
→ → → യ്ക്ക്12
→ → → → ക്ഷം13
→ → ്കോ14
→ → → 15
→ → → → ക്ക്16
→ → → ടെ17
→ → → → ക്ക്18
→ → → ളം19
→ → → → ക്ക്20
→ → ്കമുണ്ടാ21
→ → → കില്ലേ22
→ → → → None23
→ → → യിരുന്നു24
```

Figure 8.6: Suffix-replacement-tree



```
നിന്ന്
ഒരേ
അ
തന്നെ
പിന്നെ
വളരെ
ഈ
പക്ഷേ
ഒരു
പല
ചില
വരെ
പോലെ
എത്ര
എത്രയെത്ര
മറ്റു
മുൻപെ
തന്റെ
എന്ന
അത്
ആണ്
എന്ന്
ഇനി
ഞാൻ
ഏറെ
അന്ന്
```

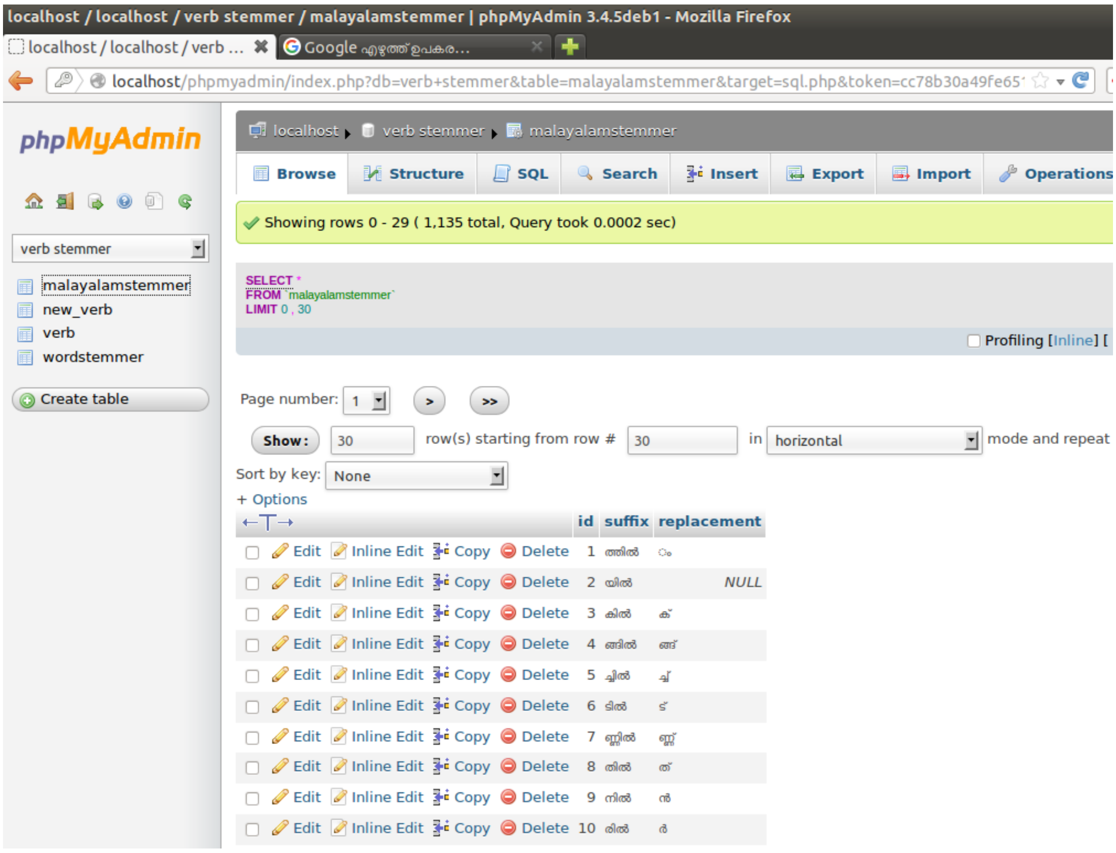
FIGURE 8.3: Stop word file

Tokens are subjected to lemmatization. The suffix replacement rules used to build the tree based lemmatizer is shown in Fig. 8.4 and Fig. 8.5. As seen in Fig. 8.6, a path in the tree will be a suffix and the leaves are storing the replacement. When a path match with suffix is obtained, suffix is replaced with the respective leaf. The lemmatized output and the time taken for lemmatizing each word is shown in Fig. 8.7. We can see the following replacements:

- $thazhvarangalil \xrightarrow{(angalil:am)} thazhvaram$

- $valarnnu \xrightarrow{(rnnu:ruka)} valaruka$

- $neelagiriyude \xrightarrow{(yude:null)} neelagiri$


localhost / localhost / verb stemmer / malayalamstemmer | phpMyAdmin 3.4.5deb1 - Mozilla Firefox
localhost / localhost / verb ...
Google എഴുത്ത് ഉപകര...
localhost/phpmyadmin/index.php?db=verb+stemmer&table=malayalamstemmer&target=sql.php&token=cc78b30a49fe65
phpMyAdmin
verb stemmer
malayalamstemmer
new_verb
verb
wordstemmer
Create table
localhost
verb stemmer
malayalamstemmer
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

Showing rows 0 - 29 (1,135 total, Query took 0.0002 sec)

```
SELECT *
FROM `malayalamstemmer`
LIMIT 0 , 30
```

□ Profiling [Inline] [

Page number: 1 > >>

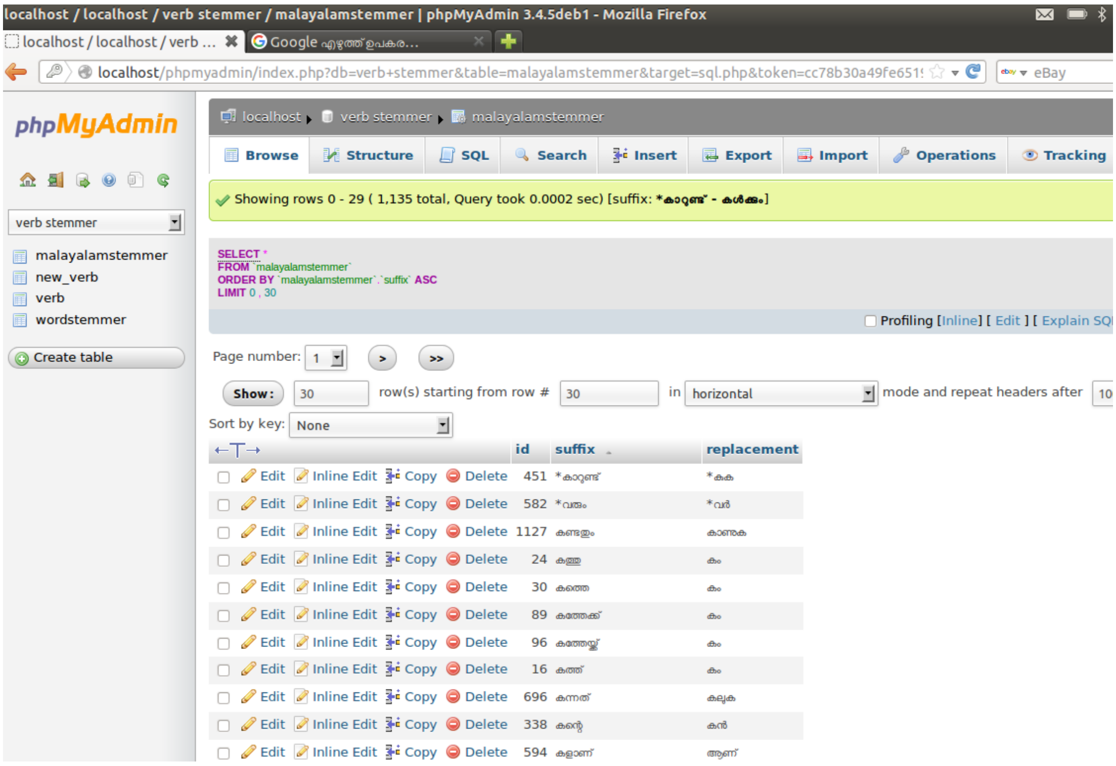
Show : 30 row(s) starting from row # 30 in horizontal mode and repeat

Sort by key: None

+ Options

←T→	id	suffix	replacement
□ Edit Inline Edit Copy Delete	1	ത്തിൽ	ം
□ Edit Inline Edit Copy Delete	2	യിൽ	NULL
□ Edit Inline Edit Copy Delete	3	കിൽ	ക്
□ Edit Inline Edit Copy Delete	4	ങ്ങിൽ	ങ്ങ്
□ Edit Inline Edit Copy Delete	5	ച്ചിൽ	ച്ച്
□ Edit Inline Edit Copy Delete	6	ടിൽ	ട്
□ Edit Inline Edit Copy Delete	7	ണ്ണിൽ	ണ്ണ്
□ Edit Inline Edit Copy Delete	8	തിൽ	ത്
□ Edit Inline Edit Copy Delete	9	നിൽ	ൻ
□ Edit Inline Edit Copy Delete	10	രിൽ	ർ

Figure 8.4: Malayalam lemmatizer in phpmyadmin


localhost / localhost / verb stemmer / malayalamstemmer | phpMyAdmin 3.4.5deb1 - Mozilla Firefox
localhost / localhost / verb ...
Google എഴുത്ത് ഉപകര...
localhost/phpmyadmin/index.php?db=verb+stemmer&table=malayalamstemmer&target=sql.php&token=cc78b30a49fe651
eBay
phpMyAdmin
verb stemmer
malayalamstemmer
new_verb
verb
wordstemmer
Create table
localhost
verb stemmer
malayalamstemmer
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Tracking

Showing rows 0 - 29 (1,135 total, Query took 0.0002 sec) [suffix: *കാറുണ്ട് - കൾക്കം]

```
SELECT *
FROM `malayalamstemmer`
ORDER BY `malayalamstemmer`.`suffix` ASC
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQ

Page number: 1 > >>

Show : 30 row(s) starting from row # 30 in horizontal mode and repeat headers after 10

Sort by key: None

←T→	id	suffix	replacement
□ Edit Inline Edit Copy Delete	451	*കാറുണ്ട്	*കക
□ Edit Inline Edit Copy Delete	582	*വരും	*വർ
□ Edit Inline Edit Copy Delete	1127	കണ്ടതും	കാണുക
□ Edit Inline Edit Copy Delete	24	കള്ള	കം
□ Edit Inline Edit Copy Delete	30	കത്തെ	കം
□ Edit Inline Edit Copy Delete	89	കത്തേക്ക്	കം
□ Edit Inline Edit Copy Delete	96	കത്തേയ്ക്ക്	കം
□ Edit Inline Edit Copy Delete	16	കത്ത്	കം
□ Edit Inline Edit Copy Delete	696	കന്നത്	കലുക
□ Edit Inline Edit Copy Delete	338	കന്റെ	കൻ
□ Edit Inline Edit Copy Delete	594	കളാണ്	ആണ്

Figure 8.5: Malayalam lemmatizer sorted in phpmyadmin

- $uyarangalilekku \xrightarrow{(lilekku:l)} uyarangal$

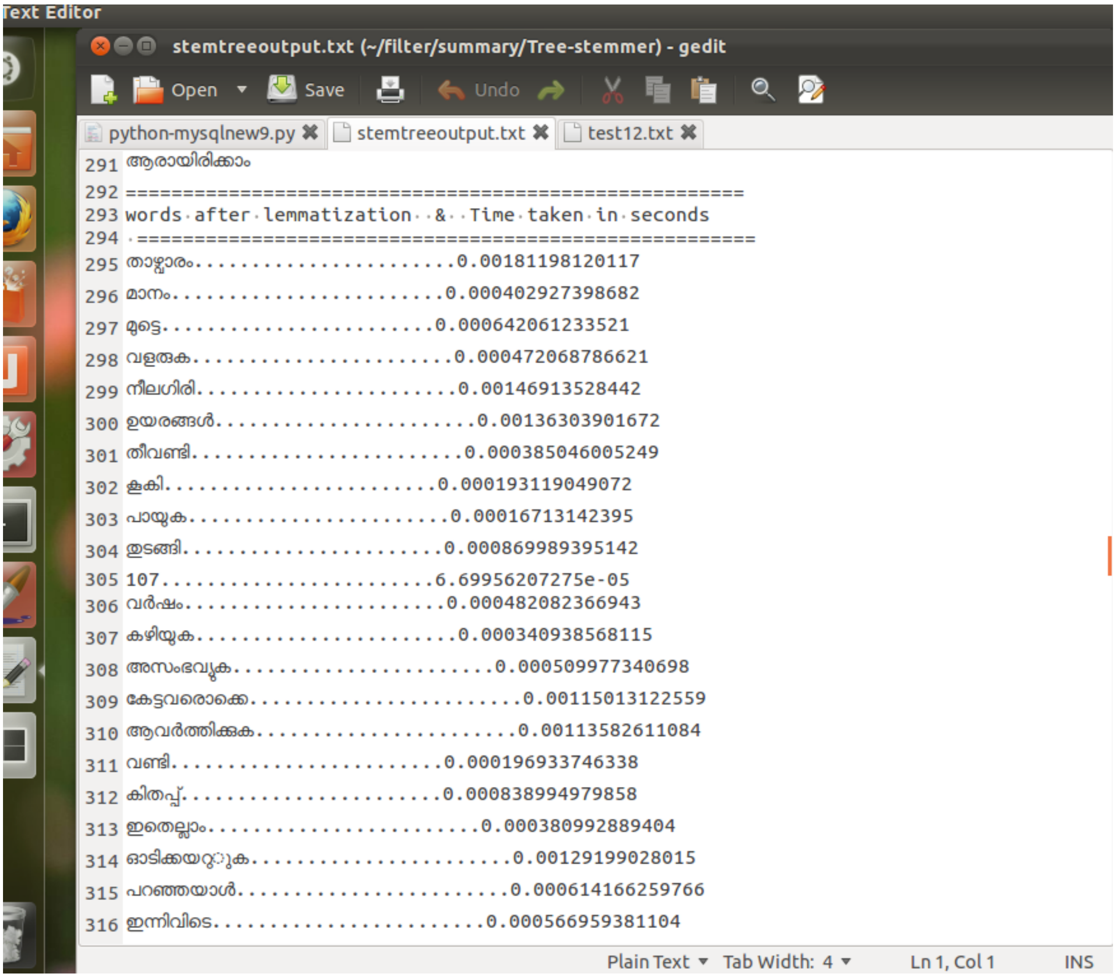
- $thudangiyittu \xrightarrow{(yittu:null)} thudangi$

- $panju \xrightarrow{(u:yuka)} payuka$

- $kazhinju \xrightarrow{(u:yuka)} kazhiyuka$

- $avarthichittum \xrightarrow{(chittum:kkuka)} avarthikkuka$

Here we can see that on finding a matching suffix, it is replaced with replacement. But stemmers developed in many languages just strip the suffix, which yield the stem word. Since the proposed lemmatizer is yielding the root word rather than the stem word, it is more meaningful. The suffix replacement rules are placed in database using phpmyadmin and tree is created using python. Python pickling technique makes the tree permanent.

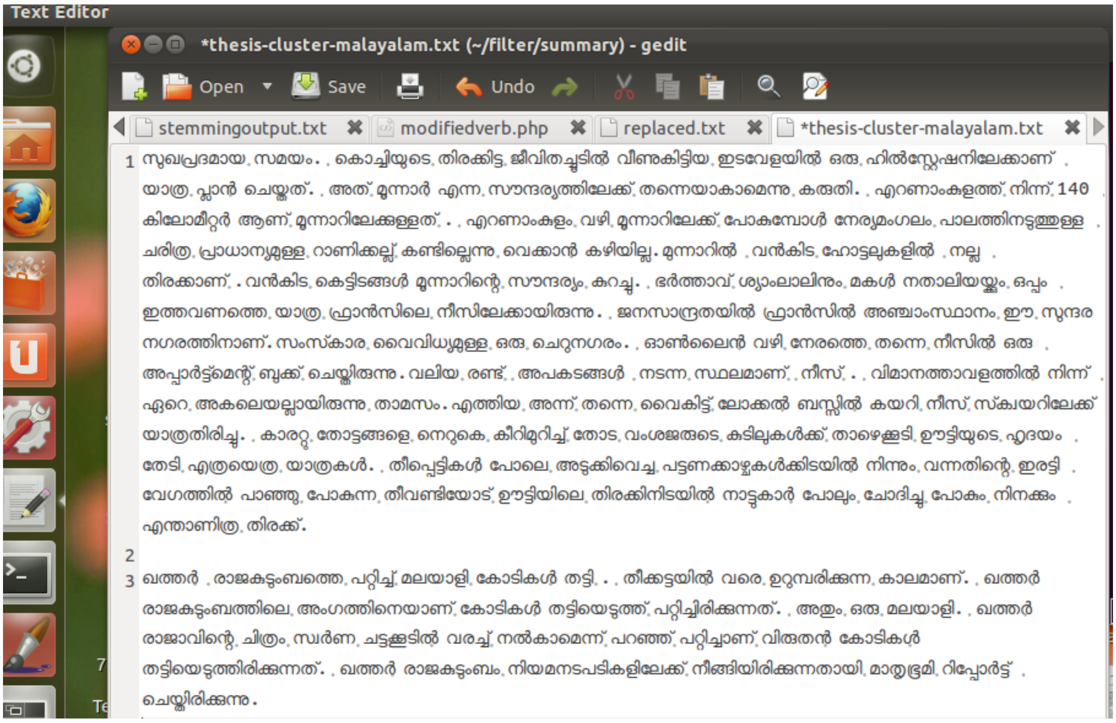


```
ആരായിരിക്കാം
=====================================================
words after lemmatization  &  Time taken in seconds
 =====================================================
താഴ്വാരം.......................0.00181198120117
മാനം.......................0.000402927398682
മുട്ടെ.......................0.000642061233521
വളരുക.......................0.000472068786621
നീലഗിരി.......................0.00146913528442
ഉയരങ്ങൾ.......................0.00136303901672
തീവണ്ടി.......................0.000385046005249
ക്ലകി.......................0.000193119049072
പായുക.......................0.00016713142395
തുടങ്ങി.......................0.000869989395142
107.......................6.69956207275e-05
വർഷം.......................0.000482082366943
കഴിയുക.......................0.000340938568115
അസംഭവ്യക.......................0.000509977340698
കേട്ടവരൊക്കെ.......................0.00115013122559
ആവർത്തിക്കുക.......................0.00113582611084
വണ്ടി.......................0.000196933746338
കിതപ്പ്.......................0.000838994979858
ഇതെല്ലാം.......................0.000380992889404
ഓടിക്കയറ്◌ുക.......................0.00129199028015
പറഞ്ഞയാൾ.......................0.000614166259766
ഇന്നിവിടെ.......................0.000566959381104
```

FIGURE 8.7: Lemmatizer output and the time taken for each word

8.2 Clustering phase

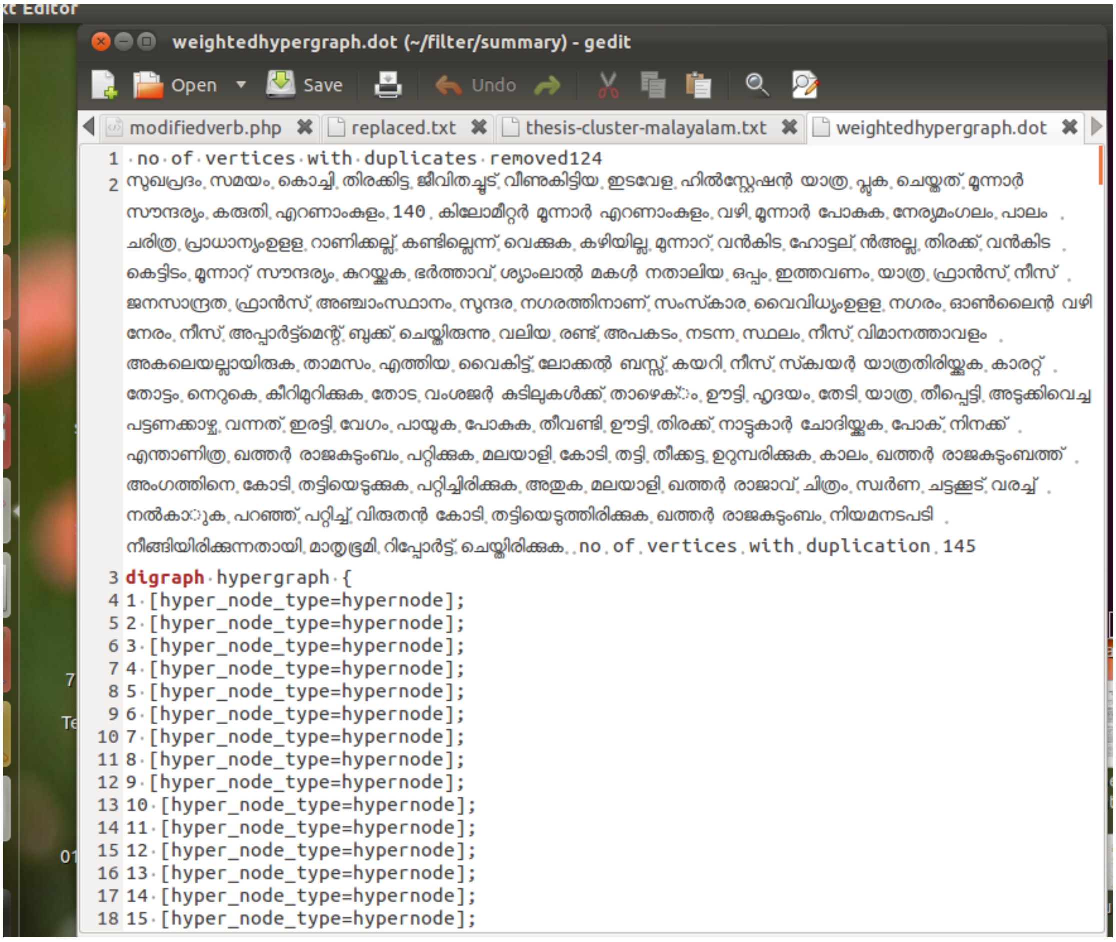
Now let us see the results of clustering phase, where the text is modeled as a weighted hypergraph. Clustering is done by spectral partitioning of the similarity matrix of the hypergraph. Fig. 8.8 shows a sample Malayalam input file given for clustering. It consists of a text pertaining to two topics. Some sentences are related to travel and others related to politics. Number of unique words in the text are found out. It is taken as the number of nodes in the hypergraph. The number of sentences in the text form the number of hyperedges. The hypergraph formed is shown in Fig. 8.9 and Fig. 8.10. The eigen values calculated for the similarity matrix is shown in Fig. 8.11. The eigen vector of the eigen value with maximum absolute value is shown in Fig. 8.12. Splitting the eigen vector based on positive and negative values are illustrated in Fig. 8.13. The process iterates until there are no change in sign or number of elements is less than 2. The cluster output is shown in Fig. 8.14.


Text Editor
*thesis-cluster-malayalam.txt (~/filter/summary) - gedit
Open
Save
Undo
stemmingoutput.txt
modifiedverb.php
replaced.txt
*thesis-cluster-malayalam.txt

സുഖപ്രദമായ സമയം. കൊച്ചിയുടെ തിരക്കിട്ട ജീവിതച്ചൂടിൽ വീണുകിട്ടിയ ഇടവേളയിൽ ഒരു ഹിൽസ്റ്റേഷനിലേക്കാണ് യാത്ര പ്ലാൻ ചെയ്തത്. അത് മൂന്നാർ എന്ന സൗന്ദര്യത്തിലേക്ക് തന്നെയാകാമെന്നു കരുതി. എറണാംകുളത്ത് നിന്ന് 140 കിലോമീറ്റർ ആണ് മൂന്നാറിലേക്കുള്ളത് . എറണാംകുളം വഴി മൂന്നാറിലേക്ക് പോകുമ്പോൾ നേര്യമംഗലം പാലത്തിനടുത്തുള്ള ചരിത്ര പ്രാധാന്യമുള്ള റാണിക്കല്ല് കണ്ടില്ലെന്നു വെക്കാൻ കഴിയില്ല. മൂന്നാറിൽ വൻകിട ഹോട്ടലുകളിൽ നല്ല തിരക്കാണ് . വൻകിട കെട്ടിടങ്ങൾ മൂന്നാറിന്റെ സൗന്ദര്യം കുറച്ചു. ഭർത്താവ് ശ്യാംലാലിനും മകൾ നതാലിയയ്ക്കും ഒപ്പം ഇത്തവണത്തെ യാത്ര ഫ്രാൻസിലെ നീസിലേക്കായിരുന്നു. ജനസാന്ദ്രതയിൽ ഫ്രാൻസിൽ അഞ്ചാംസ്ഥാനം ഈ സുന്ദര നഗരത്തിനാണ്. സംസ്കാര വൈവിധ്യമുള്ള ഒരു ചെറുനഗരം. ഓൺലൈൻ വഴി നേരത്തെ തന്നെ നീസിൽ ഒരു അപ്പാർട്ട്മെന്റ് ബുക്ക് ചെയ്തിരുന്നു. വലിയ രണ്ട് അപകടങ്ങൾ നടന്ന സ്ഥലമാണ് നീസ് . വിമാനത്താവളത്തിൽ നിന്ന് ഏറെ അകലെയല്ലായിരുന്നു താമസം. എത്തിയ അന്ന് തന്നെ വൈകിട്ട് ലോക്കൽ ബസ്സിൽ കയറി നീസ് സ്ക്വയറിലേക്ക് യാത്രതിരിച്ചു. കാരറ്റ് തോട്ടങ്ങളെ നെറുകെ കീറിമുറിച്ച് തോട വംശജരുടെ കുടിലുകൾക്ക് താഴെക്കൂടി ഊട്ടിയുടെ ഹൃദയം തേടി എത്രയെത്ര യാത്രകൾ. തീപ്പെട്ടികൾ പോലെ അടുക്കിവെച്ച പട്ടണക്കാഴ്ചകൾക്കിടയിൽ നിന്നും വന്നതിന്റെ ഇരട്ടി വേഗത്തിൽ പാഞ്ഞു പോകുന്ന തീവണ്ടിയോട് ഊട്ടിയിലെ തിരക്കിനിടയിൽ നാട്ടുകാർ പോലും ചോദിച്ചു പോകും നിനക്കും എന്താണിത്ര തിരക്ക്.

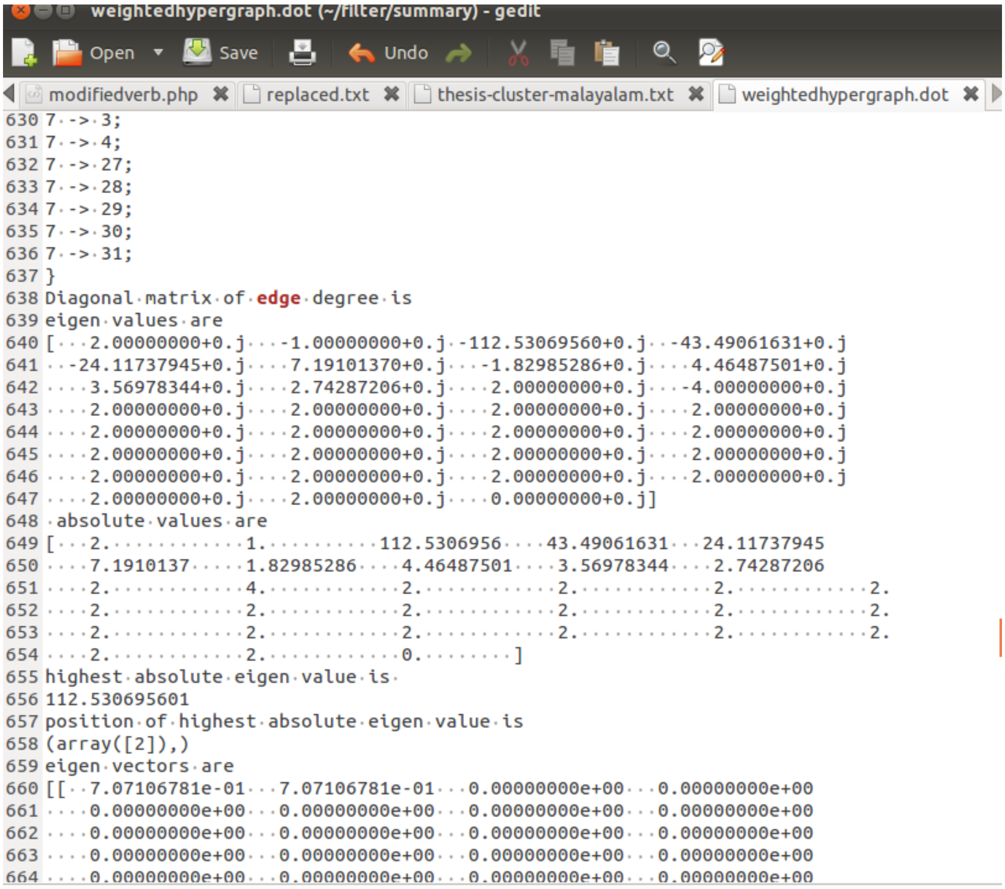
ഖത്തർ രാജകുടുംബത്തെ പറ്റിച്ച് മലയാളി കോടികൾ തട്ടി. . തീക്കട്ടയിൽ വരെ ഉറുമ്പരിക്കുന്ന കാലമാണ്. ഖത്തർ രാജകുടുംബത്തിലെ അംഗത്തിനെയാണ് കോടികൾ തട്ടിയെടുത്ത് പറ്റിച്ചിരിക്കുന്നത്. അതും ഒരു മലയാളി. ഖത്തർ രാജാവിന്റെ ചിത്രം സ്വർണ ചട്ടക്കൂടിൽ വരച്ച് നൽകാമെന്ന് പറഞ്ഞ് പറ്റിച്ചാണ് വിരുതൻ കോടികൾ തട്ടിയെടുത്തിരിക്കുന്നത്. ഖത്തർ രാജകുടുംബം നിയമനടപടികളിലേക്ക് നീങ്ങിയിരിക്കുന്നതായി മാതൃഭൂമി റിപ്പോർട്ട് ചെയ്തിരിക്കുന്നു.

Figure 8.8: Malayalam input for clustering


weightedhypergraph.dot (~/filter/summary) - gedit
Open
Save
Undo
modifiedverb.php
replaced.txt
thesis-cluster-malayalam.txt
weightedhypergraph.dot

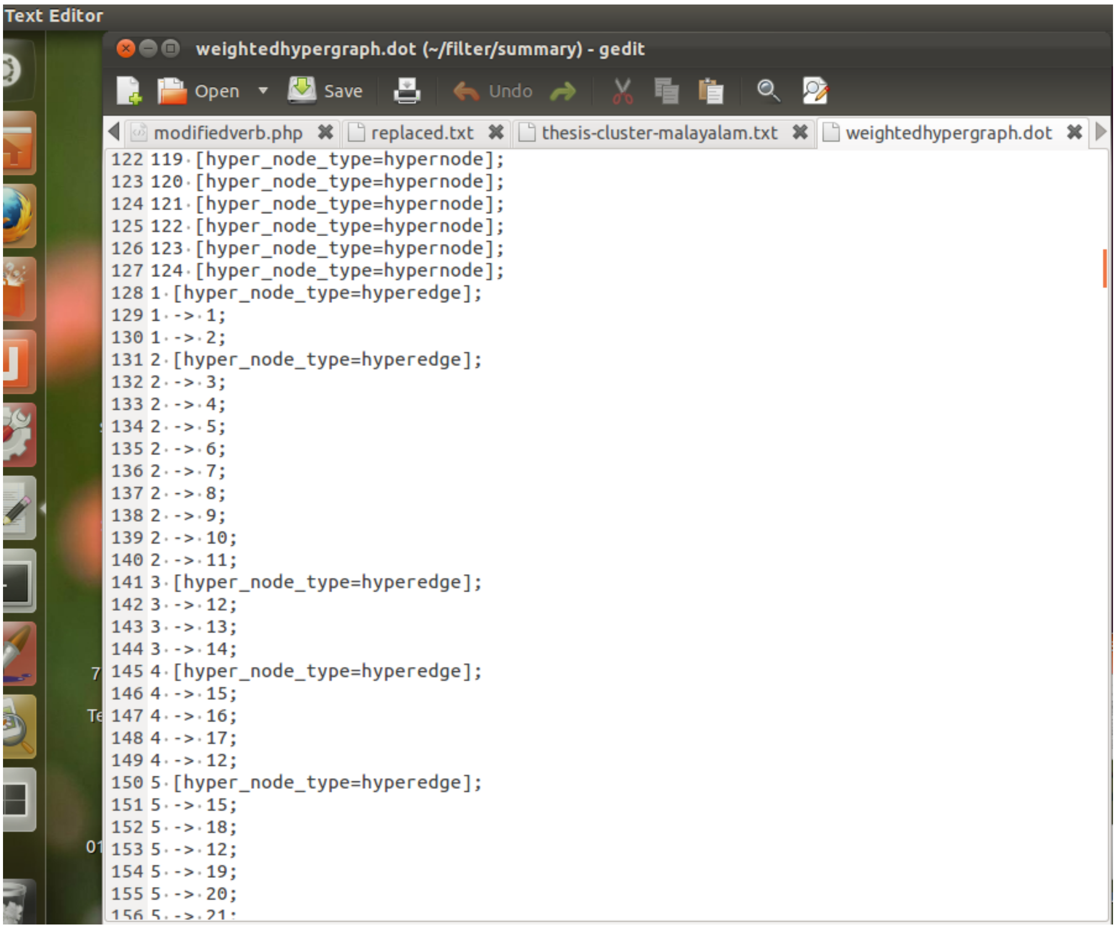
```
 no of vertices with duplicates removed124
സുഖപ്രദം സമയം കൊച്ചി തിരക്കിട്ട ജീവിതച്ചൂട് വീണുകിട്ടിയ ഇടവേള ഹിൽസ്റ്റേഷൻ യാത്ര പ്ലക ചെയ്യുത് മൂന്നാർ സൗന്ദര്യം കരുതി എറണാംകുളം 140 കിലോമീറ്റർ മൂന്നാർ എറണാംകുളം വഴി മൂന്നാർ പോകുക നേര്യമംഗലം പാലം ചരിത്ര പ്രാധാന്യംഉള്ള റാണിക്കല്ല് കണ്ടില്ലെന്ന് വെക്കുക കഴിയില്ല മൂന്നാറ് വൻകിട ഹോട്ടല് ൻഅല്ല തിരക്ക് വൻകിട കെട്ടിടം മൂന്നാറ് സൗന്ദര്യം കുറയ്ക്കുക ഭർത്താവ് ശ്യാംലാൽ മകൾ നതാലിയ ഒപ്പം ഇത്തവണം യാത്ര ഫ്രാൻസ് നീസ് ജനസാന്ദ്രത ഫ്രാൻസ് അഞ്ചാംസ്ഥാനം സുന്ദര നഗരത്തിനാണ് സംസ്കാര വൈവിധ്യംഉള്ള നഗരം ഓൺലൈൻ വഴി നേരം നീസ് അപ്പാർട്ട്മെന്റ് ബുക്ക് ചെയ്യിരുന്നു വലിയ രണ്ട് അപകടം നടന്ന സ്ഥലം നീസ് വിമാനത്താവളം അകലെയല്ലായിരുക താമസം എത്തിയ വൈകിട്ട് ലോക്കൽ ബസ്സ് കയറി നീസ് സ്ക്വയർ യാത്രതിരിയ്ക്കുക കാരറ്റ് തോട്ടം നെറുകെ കീറിമുറിക്കുക തോട വംശജർ കടില്യകൾക്ക് താഴെക്ം ഊട്ടി ഹൃദയം തേടി യാത്ര തീപ്പെട്ടി അടുക്കിവെച്ച പട്ടണക്കാഴ്ച വന്നത് ഇരട്ടി വേഗം പായുക പോകുക തീവണ്ടി ഊട്ടി തിരക്ക് നാട്ടുകാർ ചോദിയ്ക്കുക പോക് നിനക്ക് എന്താണിത്ര ഖത്തർ രാജകുടുംബം പറ്റിക്കുക മലയാളി കോടി തട്ടി തീക്കട്ട ഉറുമ്പരിക്കുക കാലം ഖത്തർ രാജകുടുംബത്ത് അംഗത്തിനെ കോടി തട്ടിയെടുക്കുക പറ്റിച്ചിരിക്കുക അതുക മലയാളി ഖത്തർ രാജാവ് ചിത്രം സ്വർണ ചട്ടക്കൂട് വരച്ച് നൽകാുക പറഞ്ഞ് പറ്റിച്ച് വിരുതൻ കോടി തട്ടിയെടുത്തിരിക്കുക ഖത്തർ രാജകുടുംബം നിയമനടപടി നീങ്ങിയിരിക്കുന്നതായി മാതൃഭൂമി റിപ്പോർട്ട് ചെയ്യിരിക്കുക  no of vertices with duplication 145
digraph hypergraph {
1 [hyper_node_type=hypernode];
2 [hyper_node_type=hypernode];
3 [hyper_node_type=hypernode];
4 [hyper_node_type=hypernode];
5 [hyper_node_type=hypernode];
6 [hyper_node_type=hypernode];
7 [hyper_node_type=hypernode];
8 [hyper_node_type=hypernode];
9 [hyper_node_type=hypernode];
10 [hyper_node_type=hypernode];
11 [hyper_node_type=hypernode];
12 [hyper_node_type=hypernode];
13 [hyper_node_type=hypernode];
14 [hyper_node_type=hypernode];
15 [hyper_node_type=hypernode];
```

FIGURE 8.9: Hypergraph modeling of the input


weightedhypergraph.dot (~/filter/summary) - gedit
Open
Save
Undo
modifiedverb.php
replaced.txt
thesis-cluster-malayalam.txt
weightedhypergraph.dot

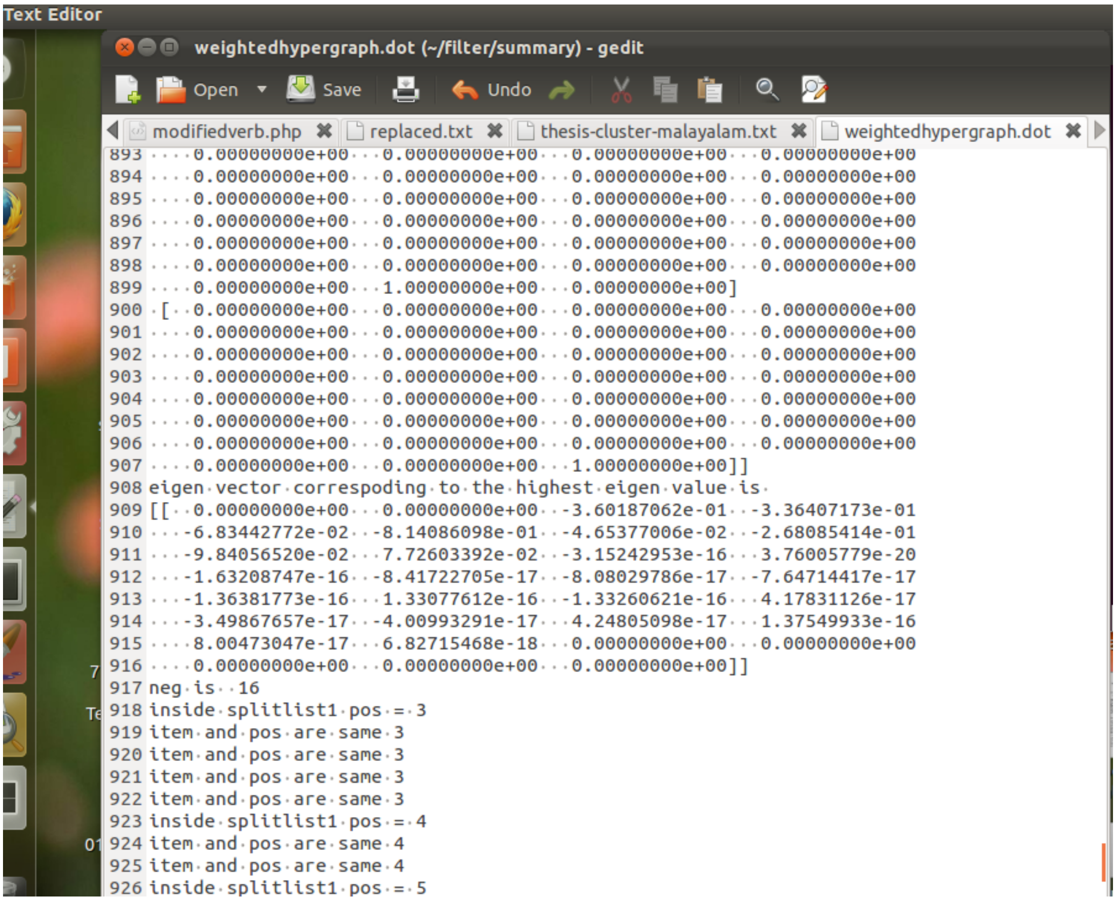
```
7 -> 3;
7 -> 4;
7 -> 27;
7 -> 28;
7 -> 29;
7 -> 30;
7 -> 31;
}
Diagonal matrix of edge degree is
eigen values are
[   2.00000000+0.j   -1.00000000+0.j -112.53069560+0.j  -43.49061631+0.j
  -24.11737945+0.j    7.19101370+0.j   -1.82985286+0.j    4.46487501+0.j
    3.56978344+0.j    2.74287206+0.j    2.00000000+0.j   -4.00000000+0.j
    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j
    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j
    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j
    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j    2.00000000+0.j
    2.00000000+0.j    2.00000000+0.j    0.00000000+0.j]
 absolute values are
[   2.            1.          112.5306956    43.49061631   24.11737945
    7.1910137     1.82985286    4.46487501    3.56978344    2.74287206
    2.            4.            2.            2.            2.            2.
    2.            2.            2.            2.            2.            2.
    2.            2.            2.            2.            2.            2.
    2.            2.            0.        ]
highest absolute eigen value is 
112.530695601
position of highest absolute eigen value is
(array([2]),)
eigen vectors are
[[  7.07106781e-01   7.07106781e-01   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
```

FIGURE 8.11: Eigen values of the similarity matrix


Text Editor
weightedhypergraph.dot (~/filter/summary) - gedit
Open
Save
Undo
modifiedverb.php
replaced.txt
thesis-cluster-malayalam.txt
weightedhypergraph.dot

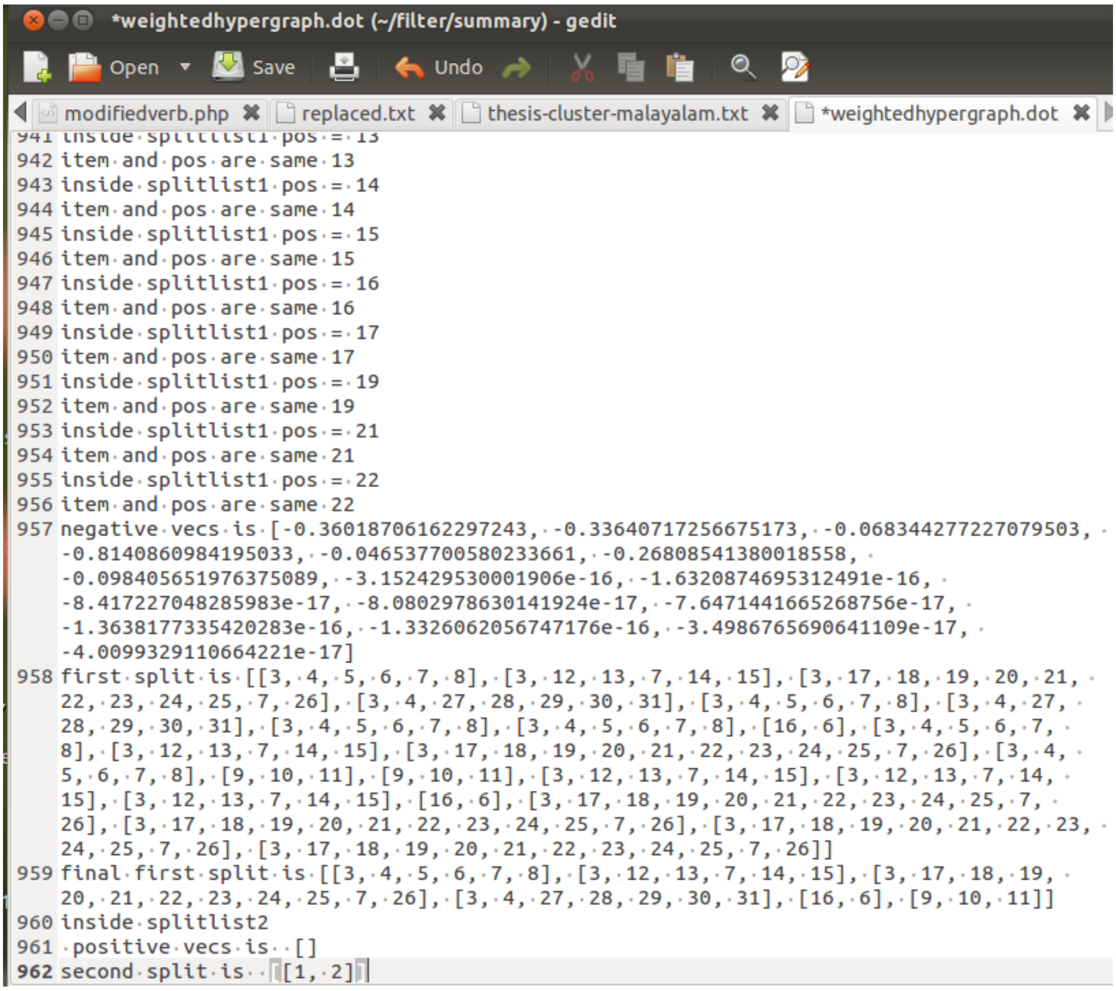
```
119 [hyper_node_type=hypernode];
120 [hyper_node_type=hypernode];
121 [hyper_node_type=hypernode];
122 [hyper_node_type=hypernode];
123 [hyper_node_type=hypernode];
124 [hyper_node_type=hypernode];
1 [hyper_node_type=hyperedge];
1 -> 1;
1 -> 2;
2 [hyper_node_type=hyperedge];
2 -> 3;
2 -> 4;
2 -> 5;
2 -> 6;
2 -> 7;
2 -> 8;
2 -> 9;
2 -> 10;
2 -> 11;
3 [hyper_node_type=hyperedge];
3 -> 12;
3 -> 13;
3 -> 14;
4 [hyper_node_type=hyperedge];
4 -> 15;
4 -> 16;
4 -> 17;
4 -> 12;
5 [hyper_node_type=hyperedge];
5 -> 15;
5 -> 18;
5 -> 12;
5 -> 19;
5 -> 20;
5 -> 21;
```

FIGURE 8.10: Hypergraph edges and nodes


Text Editor
weightedhypergraph.dot (~/filter/summary) - gedit
Open
Save
Undo
modifiedverb.php
replaced.txt
thesis-cluster-malayalam.txt
weightedhypergraph.dot

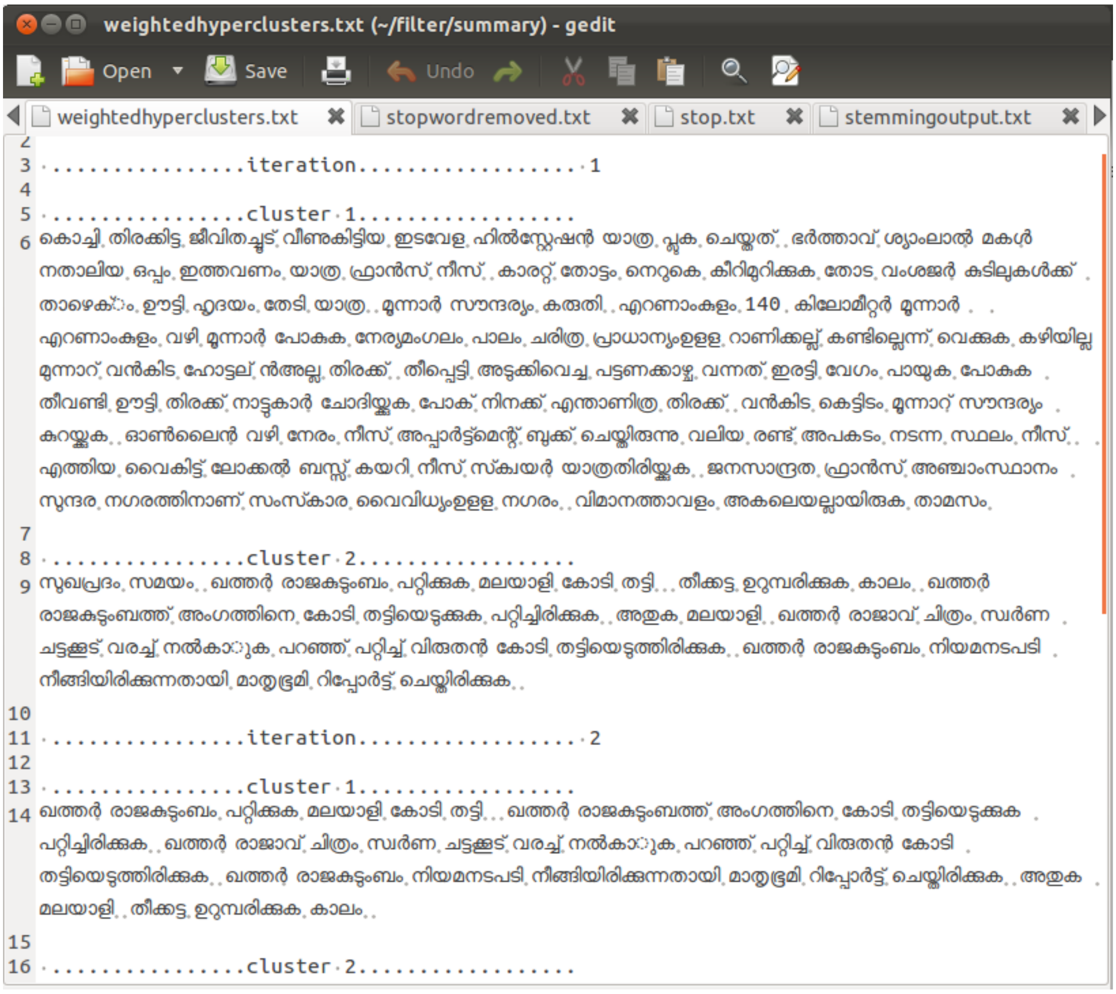
```
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   1.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   1.00000000e+00]]
eigen vector correspoding to the highest eigen value is 
[[  0.00000000e+00   0.00000000e+00  -3.60187062e-01  -3.36407173e-01
   -6.83442772e-02  -8.14086098e-01  -4.65377006e-02  -2.68085414e-01
   -9.84056520e-02   7.72603392e-02  -3.15242953e-16   3.76005779e-20
   -1.63208747e-16  -8.41722705e-17  -8.08029786e-17  -7.64714417e-17
   -1.36381773e-16   1.33077612e-16  -1.33260621e-16   4.17831126e-17
   -3.49867657e-17  -4.00993291e-17   4.24805098e-17   1.37549933e-16
    8.00473047e-17   6.82715468e-18   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00]]
neg is  16
inside splitlist1 pos = 3
item and pos are same 3
item and pos are same 3
item and pos are same 3
item and pos are same 3
inside splitlist1 pos = 4
item and pos are same 4
item and pos are same 4
inside splitlist1 pos = 5
```

FIGURE 8.12: Eigen vector


*weightedhypergraph.dot (~/filter/summary) - gedit
Open
Save
Undo
modifiedverb.php
replaced.txt
thesis-cluster-malayalam.txt
*weightedhypergraph.dot

```
inside splitlist1 pos = 13
item and pos are same 13
inside splitlist1 pos = 14
item and pos are same 14
inside splitlist1 pos = 15
item and pos are same 15
inside splitlist1 pos = 16
item and pos are same 16
inside splitlist1 pos = 17
item and pos are same 17
inside splitlist1 pos = 19
item and pos are same 19
inside splitlist1 pos = 21
item and pos are same 21
inside splitlist1 pos = 22
item and pos are same 22
negative vecs is [-0.36018706162297243, -0.33640717256675173, -0.068344277227079503, 
-0.8140860984195033, -0.046537700580233661, -0.26808541380018558, 
-0.098405651976375089, -3.152429530001906e-16, -1.6320874695312491e-16, 
-8.417227048285983e-17, -8.0802978630141924e-17, -7.6471441665268756e-17, 
-1.3638177335420283e-16, -1.3326062056747176e-16, -3.4986765690641109e-17, 
-4.0099329110664221e-17]
first split is [[3, 4, 5, 6, 7, 8], [3, 12, 13, 7, 14, 15], [3, 17, 18, 19, 20, 21, 
22, 23, 24, 25, 7, 26], [3, 4, 27, 28, 29, 30, 31], [3, 4, 5, 6, 7, 8], [3, 4, 27, 
28, 29, 30, 31], [3, 4, 5, 6, 7, 8], [3, 4, 5, 6, 7, 8], [16, 6], [3, 4, 5, 6, 7, 
8], [3, 12, 13, 7, 14, 15], [3, 17, 18, 19, 20, 21, 22, 23, 24, 25, 7, 26], [3, 4, 
5, 6, 7, 8], [9, 10, 11], [9, 10, 11], [3, 12, 13, 7, 14, 15], [3, 12, 13, 7, 14, 
15], [3, 12, 13, 7, 14, 15], [16, 6], [3, 17, 18, 19, 20, 21, 22, 23, 24, 25, 7, 
26], [3, 17, 18, 19, 20, 21, 22, 23, 24, 25, 7, 26], [3, 17, 18, 19, 20, 21, 22, 23, 
24, 25, 7, 26], [3, 17, 18, 19, 20, 21, 22, 23, 24, 25, 7, 26]]
final first split is [[3, 4, 5, 6, 7, 8], [3, 12, 13, 7, 14, 15], [3, 17, 18, 19, 
20, 21, 22, 23, 24, 25, 7, 26], [3, 4, 27, 28, 29, 30, 31], [16, 6], [9, 10, 11]]
inside splitlist2
 positive vecs is  []
second split is  [[1, 2]]
```

Figure 8.13: Splits in eigen vector

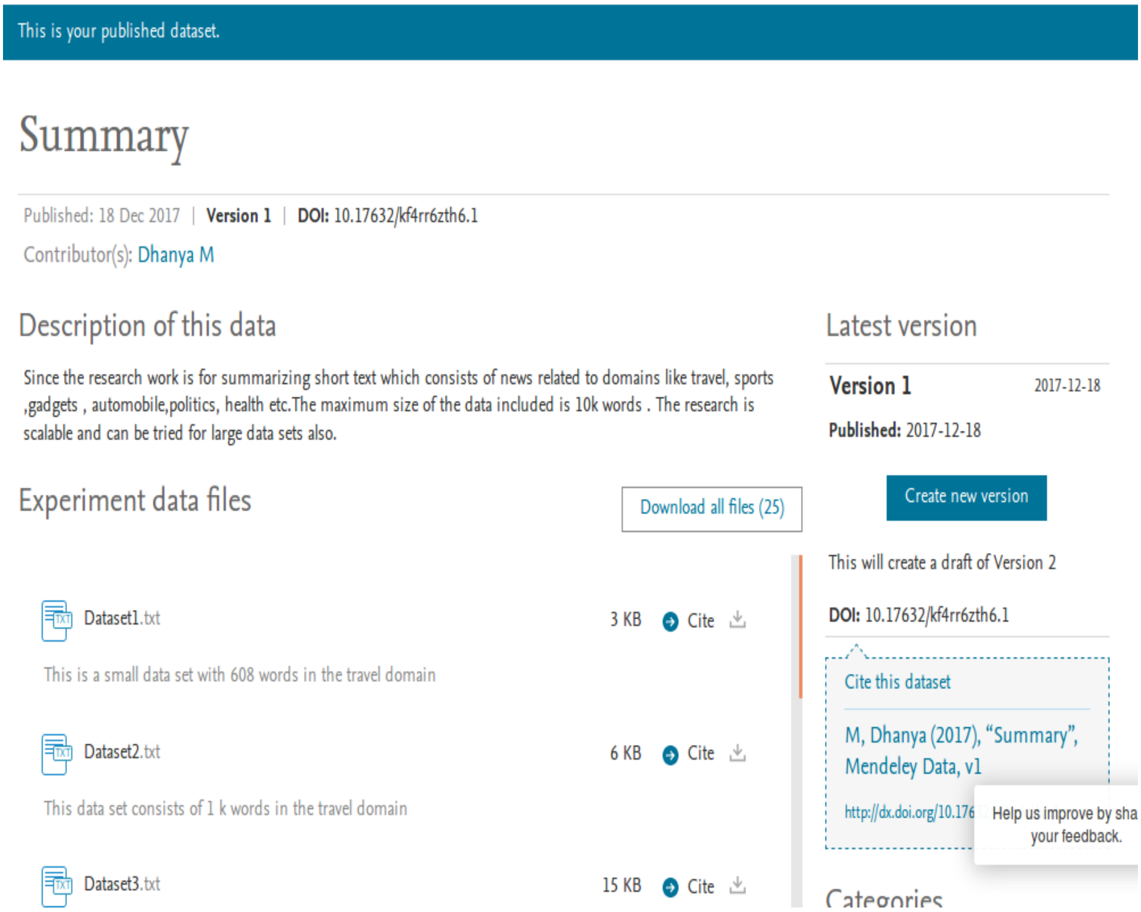

weightedhyperclusters.txt (~/filter/summary) - gedit
Open
Save
Undo
weightedhyperclusters.txt
stopwordremoved.txt
stop.txt
stemmingoutput.txt

```
.................iteration..................1

.................cluster 1..................
കൊച്ചി തിരക്കിട്ട ജീവിതച്ചൂട് വീണുകിട്ടിയ ഇടവേള ഹിൽസ്റ്റേഷൻ യാത്ര പ്ലക ചെയ്യുത്  ഭർത്താവ് ശ്യാംലാൽ മകൾ നതാലിയ ഒപ്പം ഇത്തവണം യാത്ര ഫ്രാൻസ് നീസ്  കാരറ്റ് തോട്ടം നെറുകെ കീറിമുറിക്കുക തോട വംശജർ കുടിലുകൾക്ക്  താഴെക്◌ം ഊട്ടി ഹൃദയം തേടി യാത്ര  മൂന്നാർ സൗന്ദര്യം കരുതി  എറണാംകുളം 140  കിലോമീറ്റർ മൂന്നാർ    എറണാംകുളം വഴി മൂന്നാർ പോകുക നേര്യമംഗലം പാലം ചരിത്ര പ്രാധാന്യംഉള്ള റാണിക്കല്ല് കണ്ടില്ലെന്ന് വെക്കുക കഴിയില്ല മൂന്നാറ് വൻകിട ഹോട്ടല് ൻഅല്ല തിരക്ക്  തീപ്പെട്ടി അട്ടുക്കിവെച്ച പട്ടണക്കാഴ്ച വന്നത് ഇരട്ടി വേഗം പായുക പോകുക  തീവണ്ടി ഊട്ടി തിരക്ക് നാട്ടുകാർ ചോദിയ്ക്കുക പോക് നിനക്ക് എന്താണിത്ര തിരക്ക്  വൻകിട കെട്ടിടം മൂന്നാറ് സൗന്ദര്യം  കുറയ്ക്കുക  ഓൺലൈൻ വഴി നേരം നീസ് അപ്പാർട്ട്മെന്റ് ബുക്ക് ചെയ്യിരുന്നു വലിയ രണ്ട് അപകടം നടന്ന സ്ഥലം നീസ്   എത്തിയ വൈകിട്ട് ലോക്കൽ ബസ്സ് കയറി നീസ് സ്ക്വയർ യാത്രതിരിയ്ക്കുക  ജനസാന്ദ്രത ഫ്രാൻസ് അഞ്ചാംസ്ഥാനം  സുന്ദര നഗരത്തിനാണ് സംസ്കാര വൈവിധ്യംഉള്ള നഗരം  വിമാനത്താവളം അകലെയല്ലായിരുക താമസം

.................cluster 2..................
സുഖപ്രദം സമയം  ഖത്തർ രാജകുടുംബം പറ്റിക്കുക മലയാളി കോടി തട്ടി   തീക്കട്ട ഉറുമ്പരിക്കുക കാലം  ഖത്തർ രാജകുടുംബത്ത് അംഗത്തിനെ കോടി തട്ടിയെടുക്കുക പറ്റിച്ചിരിക്കുക  അതുക മലയാളി  ഖത്തർ രാജാവ് ചിത്രം സ്വർണ  ചട്ടക്കൂട് വരച്ച് നൽകാ◌ുക പറഞ്ഞ് പറ്റിച്ച് വിരുതൻ കോടി തട്ടിയെടുത്തിരിക്കുക  ഖത്തർ രാജകുടുംബം നിയമനടപടി  നീങ്ങിയിരിക്കുന്നതായി മാതൃഭൂമി റിപ്പോർട്ട് ചെയ്യിരിക്കുക

.................iteration..................2

.................cluster 1..................
ഖത്തർ രാജകുടുംബം പറ്റിക്കുക മലയാളി കോടി തട്ടി   ഖത്തർ രാജകുടുംബത്ത് അംഗത്തിനെ കോടി തട്ടിയെടുക്കുക  പറ്റിച്ചിരിക്കുക  ഖത്തർ രാജാവ് ചിത്രം സ്വർണ ചട്ടക്കൂട് വരച്ച് നൽകാ◌ുക പറഞ്ഞ് പറ്റിച്ച് വിരുതൻ കോടി  തട്ടിയെടുത്തിരിക്കുക  ഖത്തർ രാജകുടുംബം നിയമനടപടി നീങ്ങിയിരിക്കുന്നതായി മാതൃഭൂമി റിപ്പോർട്ട് ചെയ്യിരിക്കുക  അതുക  മലയാളി  തീക്കട്ട ഉറുമ്പരിക്കുക കാലം

.................cluster 2..................
```

FIGURE 8.14: Iterative cluster formation

8.3 English system

Now we will see the results of English clustering system developed using IFHG. The system is being run in google cloud system with 30GB memory with 8 cores. The data set is uploaded in Mendeley Data set available as $http://dx.doi.org/10.17632/kf4rr6zth6.1$ The data set is shown in Fig. 8.15 and 8.16. It consists of data belonging to various domain like travel, politics, medical, gadgets and automobile. Here also the text is first subjected to preprocessing, stemming using porter stemmer and clustering using spectral partitioning of weighted hypergraph. Fig. 8.17 is a snapshot of the English input given in google cloud for clustering. Fig. 8.18 is the word frequency result and Fig. 8.19 depicts the IFHG formed from the text, Fig. 8.20 shows the conditional dilation, Fig. 8.21 reveals the erosion and summary and Fig. 8.22 shows the summary of each cluster output. As in the figure there are two clusters. For each cluster two outputs are seen. First one contains cluster with stemmed words. In the second, stemmed words are mapped back to original.



This is your published dataset.

Summary

Published: 18 Dec 2017 | **Version 1** | **DOI:** 10.17632/kf4rr6zth6.1

Contributor(s): Dhanya M

Description of this data

Since the research work is for summarizing short text which consists of news related to domains like travel, sports ,gadgets , automobile,politics, health etc.The maximum size of the data included is 10k words . The research is scalable and can be tried for large data sets also.

Experiment data files

Dataset1.txt 3 KB

This is a small data set with 608 words in the travel domain

Dataset2.txt 6 KB

This data set consists of 1 k words in the travel domain

Dataset3.txt 15 KB

Latest version

Version 1 2017-12-18

Published: 2017-12-18

This will create a draft of Version 2

DOI: 10.17632/kf4rr6zth6.1

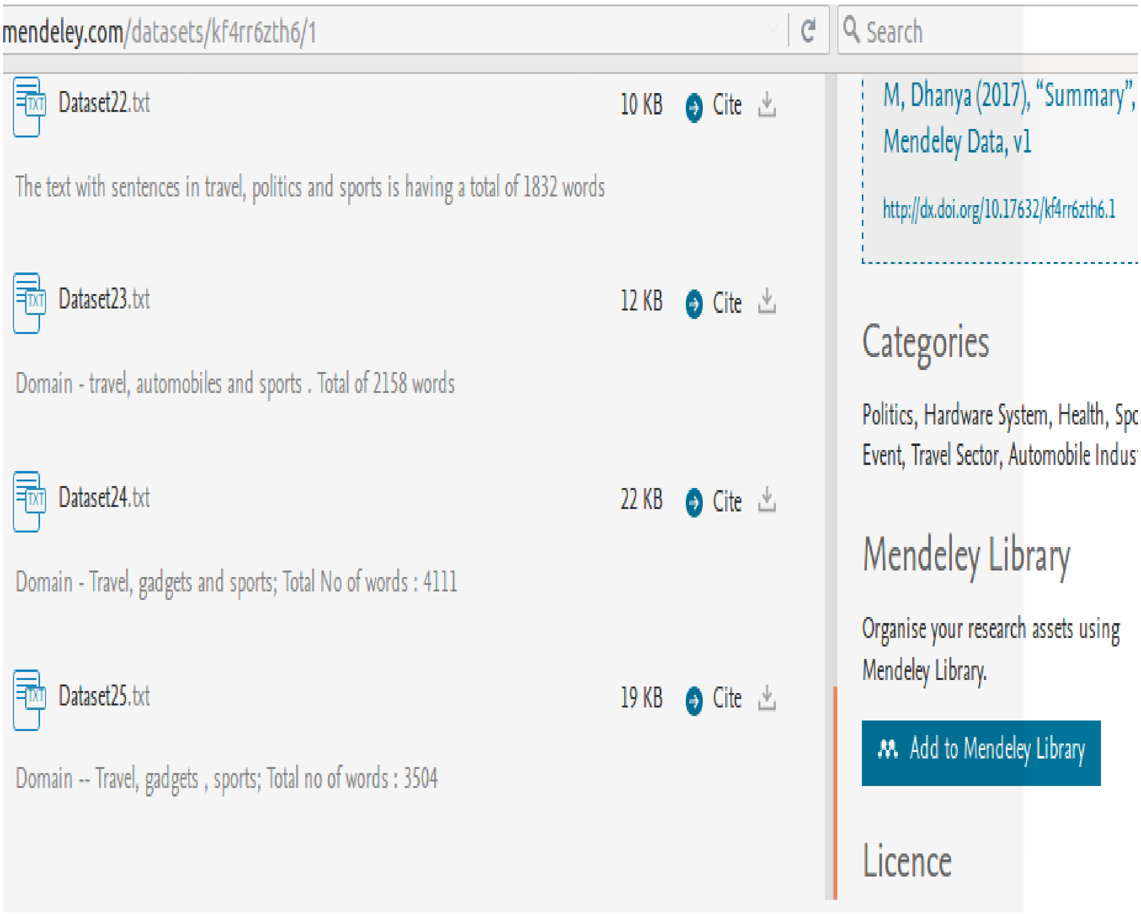
Cite this dataset

M, Dhanya (2017), "Summary", Mendeley Data, v1

http://dx.doi.org/10.176

Categories

FIGURE 8.15: English input in mendeley data set



Dataset22.txt 10 KB

The text with sentences in travel, politics and sports is having a total of 1832 words

Dataset23.txt 12 KB

Domain - travel, automobiles and sports . Total of 2158 words

Dataset24.txt 22 KB

Domain - Travel, gadgets and sports; Total No of words : 4111

Dataset25.txt 19 KB

Domain -- Travel, gadgets , sports; Total no of words : 3504

M, Dhanya (2017), "Summary", Mendeley Data, v1

http://dx.doi.org/10.17632/kf4rr6zth6.1

Categories

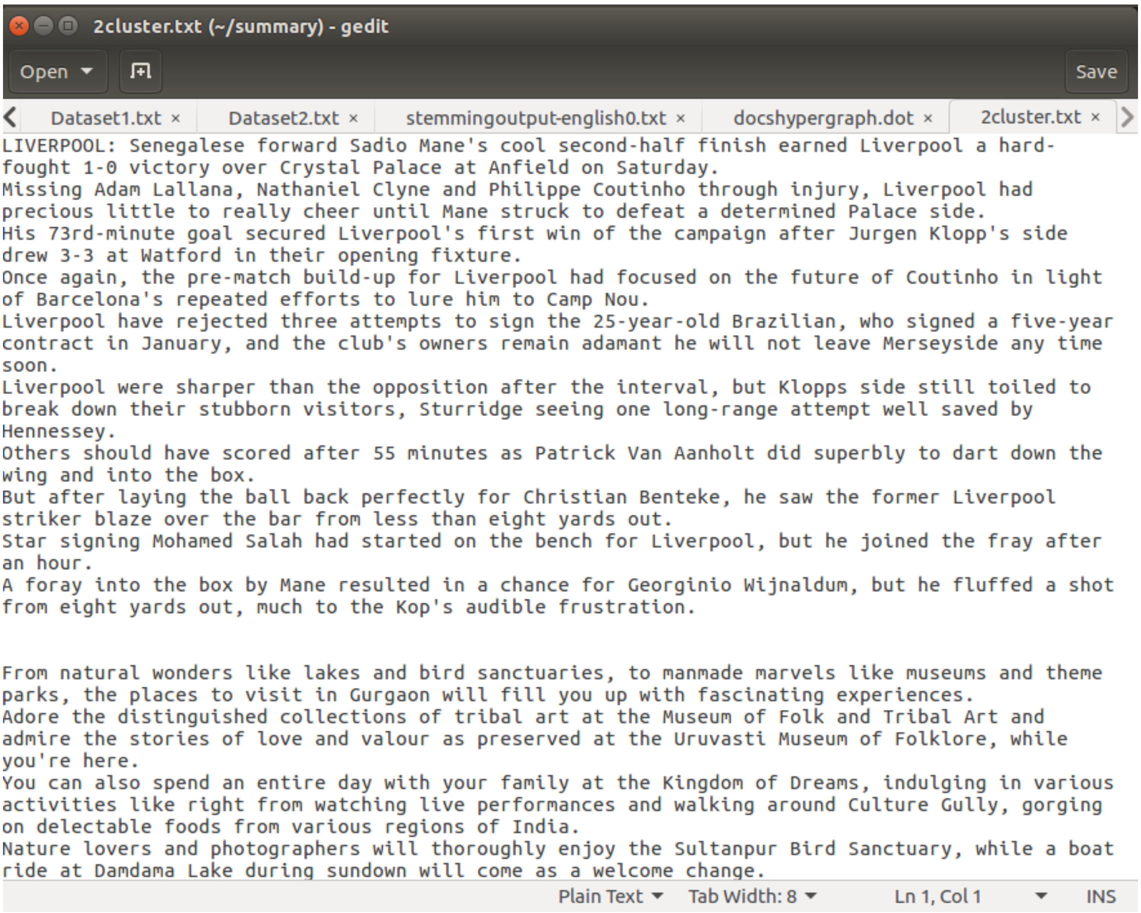
Politics, Hardware System, Health, Spo
Event, Travel Sector, Automobile Indus

Mendeley Library

Organise your research assets using Mendeley Library.

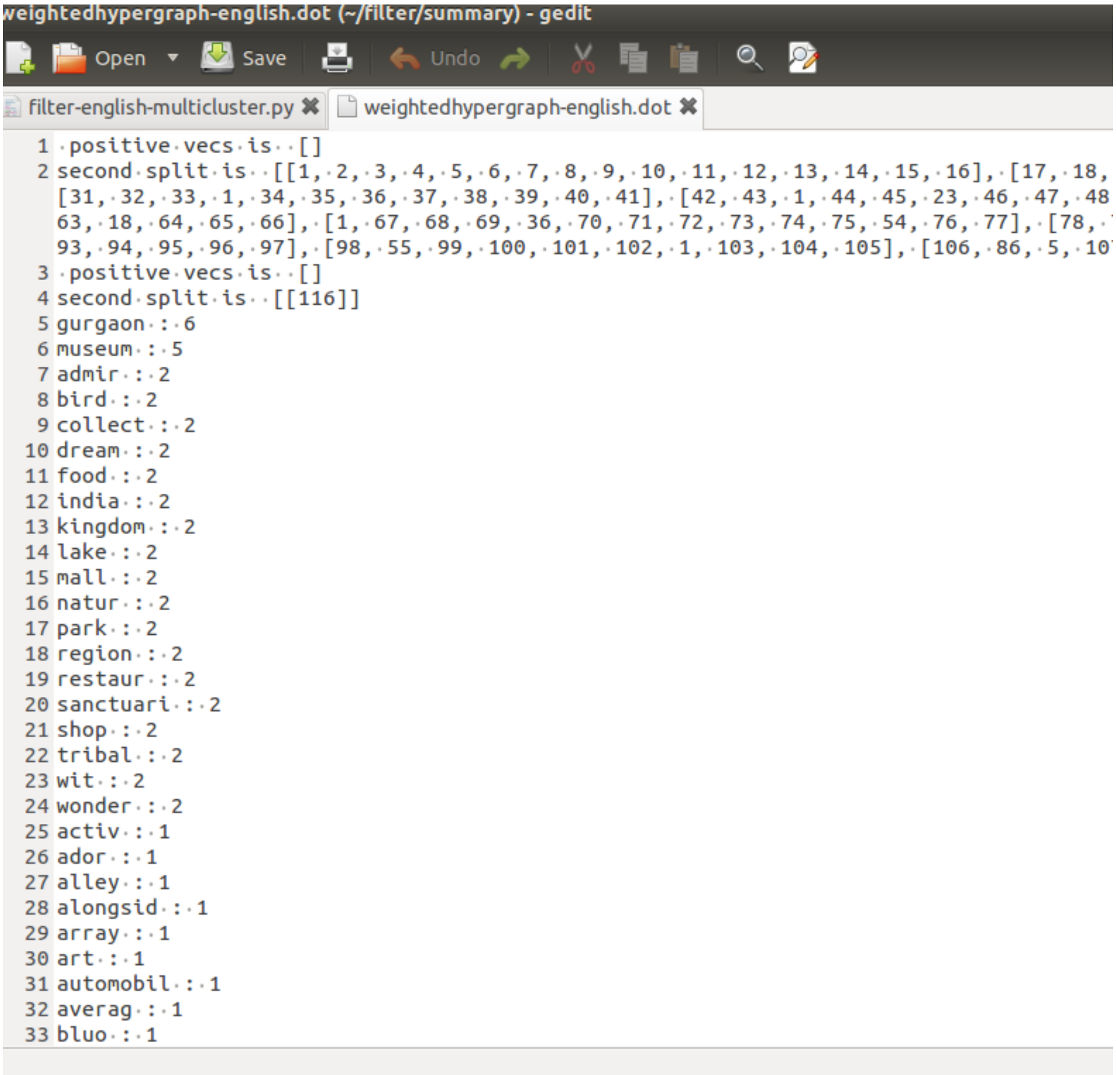
Licence

FIGURE 8.16: English input in mendeley data set



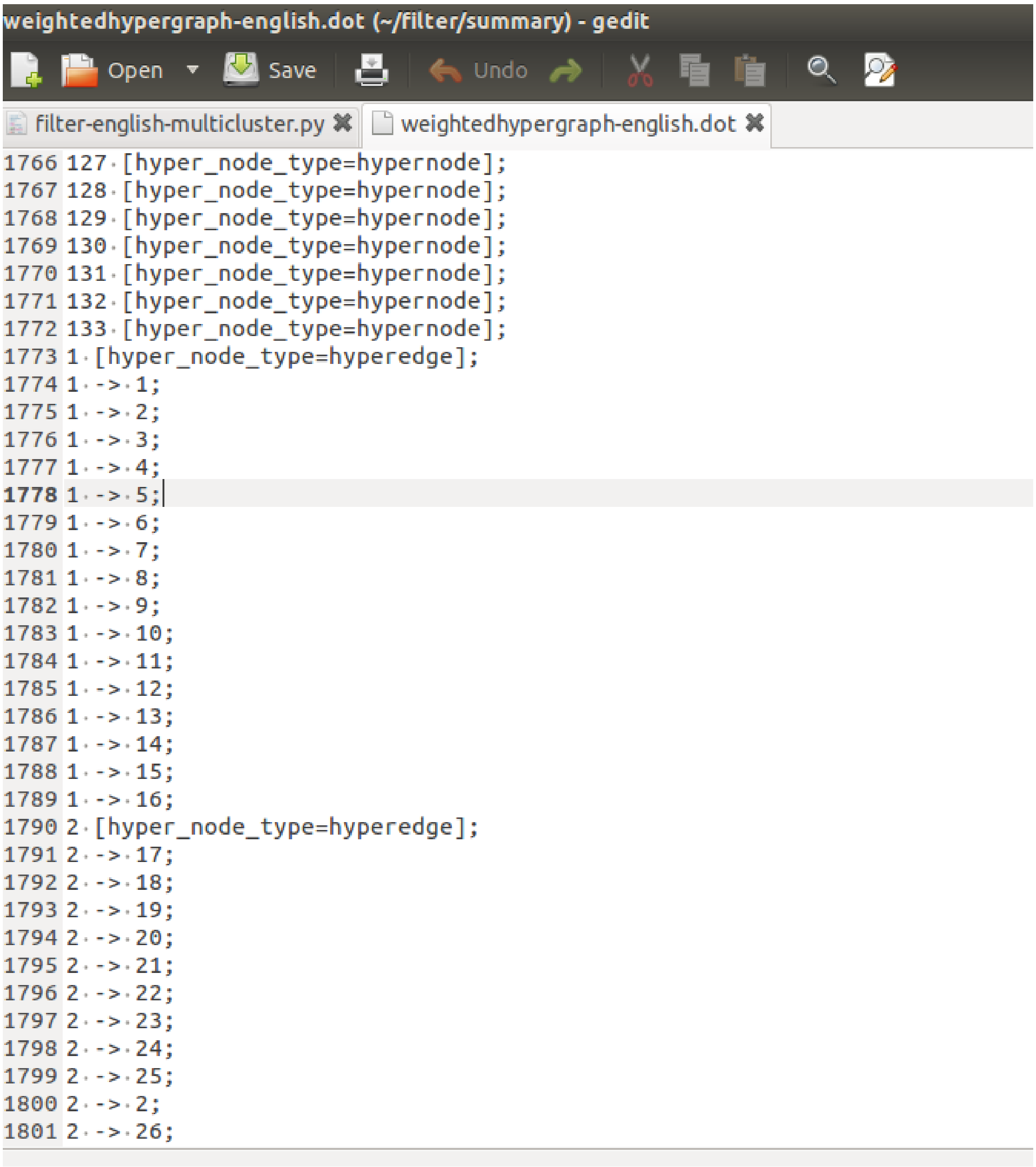
```
LIVERPOOL: Senegalese forward Sadio Mane's cool second-half finish earned Liverpool a hard-
fought 1-0 victory over Crystal Palace at Anfield on Saturday.
Missing Adam Lallana, Nathaniel Clyne and Philippe Coutinho through injury, Liverpool had
precious little to really cheer until Mane struck to defeat a determined Palace side.
His 73rd-minute goal secured Liverpool's first win of the campaign after Jurgen Klopp's side
drew 3-3 at Watford in their opening fixture.
Once again, the pre-match build-up for Liverpool had focused on the future of Coutinho in light
of Barcelona's repeated efforts to lure him to Camp Nou.
Liverpool have rejected three attempts to sign the 25-year-old Brazilian, who signed a five-year
contract in January, and the club's owners remain adamant he will not leave Merseyside any time
soon.
Liverpool were sharper than the opposition after the interval, but Klopps side still toiled to
break down their stubborn visitors, Sturridge seeing one long-range attempt well saved by
Hennessey.
Others should have scored after 55 minutes as Patrick Van Aanholt did superbly to dart down the
wing and into the box.
But after laying the ball back perfectly for Christian Benteke, he saw the former Liverpool
striker blaze over the bar from less than eight yards out.
Star signing Mohamed Salah had started on the bench for Liverpool, but he joined the fray after
an hour.
A foray into the box by Mane resulted in a chance for Georginio Wijnaldum, but he fluffed a shot
from eight yards out, much to the Kop's audible frustration.


From natural wonders like lakes and bird sanctuaries, to manmade marvels like museums and theme
parks, the places to visit in Gurgaon will fill you up with fascinating experiences.
Adore the distinguished collections of tribal art at the Museum of Folk and Tribal Art and
admire the stories of love and valour as preserved at the Uruvasti Museum of Folklore, while
you're here.
You can also spend an entire day with your family at the Kingdom of Dreams, indulging in various
activities like right from watching live performances and walking around Culture Gully, gorging
on delectable foods from various regions of India.
Nature lovers and photographers will thoroughly enjoy the Sultanpur Bird Sanctuary, while a boat
ride at Damdama Lake during sundown will come as a welcome change.
```

FIGURE 8.17: Multi-cluster input in English


weightedhypergraph-english.dot (~/filter/summary) - gedit
Open
Save
Undo
filter-english-multicluster.py
weightedhypergraph-english.dot

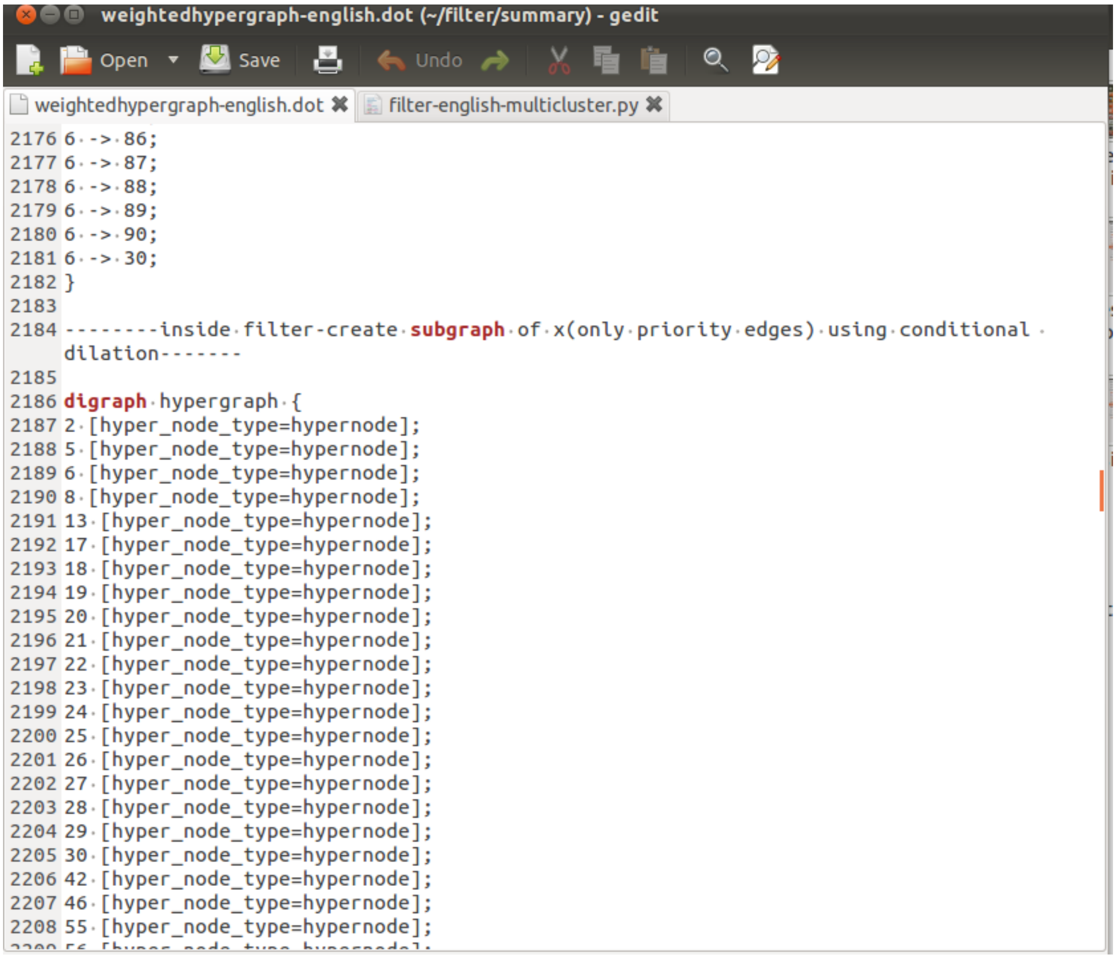
```
 positive vecs is  []
second split is  [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16], [17, 18,
[31, 32, 33, 1, 34, 35, 36, 37, 38, 39, 40, 41], [42, 43, 1, 44, 45, 23, 46, 47, 48
63, 18, 64, 65, 66], [1, 67, 68, 69, 36, 70, 71, 72, 73, 74, 75, 54, 76, 77], [78,
93, 94, 95, 96, 97], [98, 55, 99, 100, 101, 102, 1, 103, 104, 105], [106, 86, 5, 10
 positive vecs is  []
second split is  [[116]]
gurgaon : 6
museum : 5
admir : 2
bird : 2
collect : 2
dream : 2
food : 2
india : 2
kingdom : 2
lake : 2
mall : 2
natur : 2
park : 2
region : 2
restaur : 2
sanctuari : 2
shop : 2
tribal : 2
wit : 2
wonder : 2
activ : 1
ador : 1
alley : 1
alongsid : 1
array : 1
art : 1
automobil : 1
averag : 1
bluo : 1
```

FIGURE 8.18: Term frequency of input English text


weightedhypergraph-english.dot (~/filter/summary) - gedit
Open
Save
Undo
filter-english-multicluster.py
weightedhypergraph-english.dot

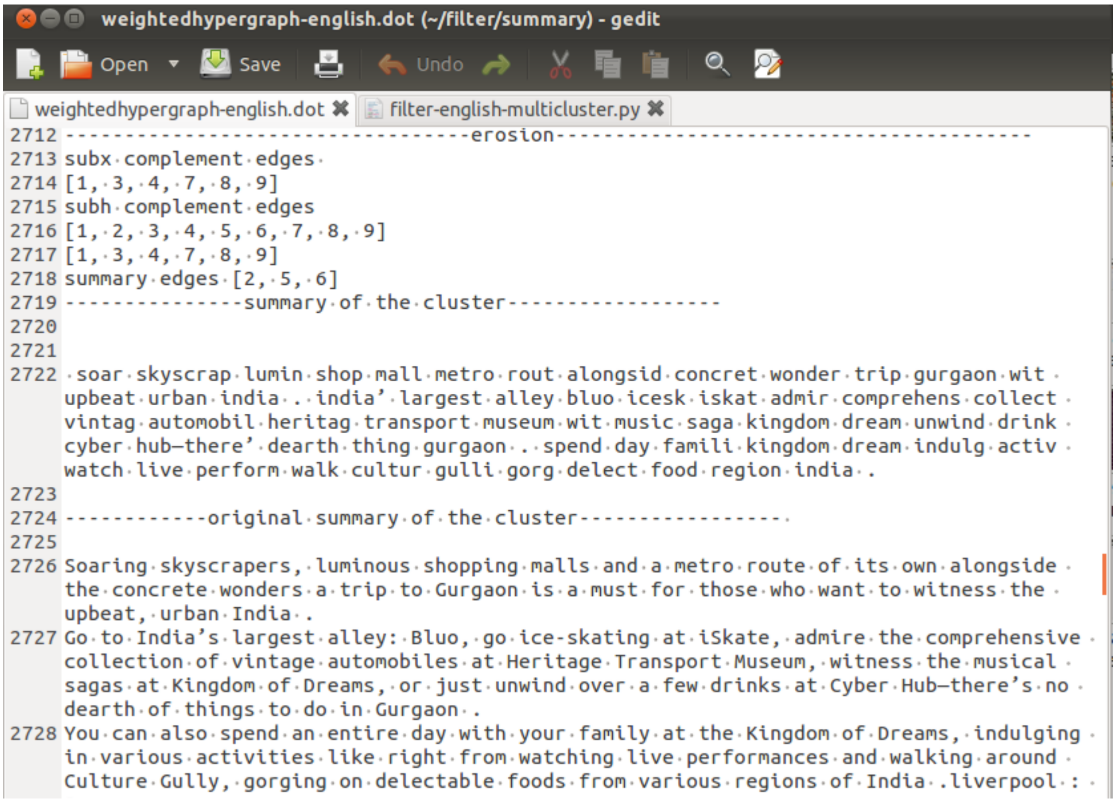
```
127 [hyper_node_type=hypernode];
128 [hyper_node_type=hypernode];
129 [hyper_node_type=hypernode];
130 [hyper_node_type=hypernode];
131 [hyper_node_type=hypernode];
132 [hyper_node_type=hypernode];
133 [hyper_node_type=hypernode];
1 [hyper_node_type=hyperedge];
1 -> 1;
1 -> 2;
1 -> 3;
1 -> 4;
1 -> 5;
1 -> 6;
1 -> 7;
1 -> 8;
1 -> 9;
1 -> 10;
1 -> 11;
1 -> 12;
1 -> 13;
1 -> 14;
1 -> 15;
1 -> 16;
2 [hyper_node_type=hyperedge];
2 -> 17;
2 -> 18;
2 -> 19;
2 -> 20;
2 -> 21;
2 -> 22;
2 -> 23;
2 -> 24;
2 -> 25;
2 -> 2;
2 -> 26;
```

FIGURE 8.19: IFHG of input English text


weightedhypergraph-english.dot (~/filter/summary) - gedit
Open
Save
Undo
weightedhypergraph-english.dot
filter-english-multicluster.py

```
6 -> 86;
6 -> 87;
6 -> 88;
6 -> 89;
6 -> 90;
6 -> 30;
}

--------inside filter-create subgraph of x(only priority edges) using conditional dilation-------

digraph hypergraph {
2 [hyper_node_type=hypernode];
5 [hyper_node_type=hypernode];
6 [hyper_node_type=hypernode];
8 [hyper_node_type=hypernode];
13 [hyper_node_type=hypernode];
17 [hyper_node_type=hypernode];
18 [hyper_node_type=hypernode];
19 [hyper_node_type=hypernode];
20 [hyper_node_type=hypernode];
21 [hyper_node_type=hypernode];
22 [hyper_node_type=hypernode];
23 [hyper_node_type=hypernode];
24 [hyper_node_type=hypernode];
25 [hyper_node_type=hypernode];
26 [hyper_node_type=hypernode];
27 [hyper_node_type=hypernode];
28 [hyper_node_type=hypernode];
29 [hyper_node_type=hypernode];
30 [hyper_node_type=hypernode];
42 [hyper_node_type=hypernode];
46 [hyper_node_type=hypernode];
55 [hyper_node_type=hypernode];
56 [hyper_node_type=hypernode];
```

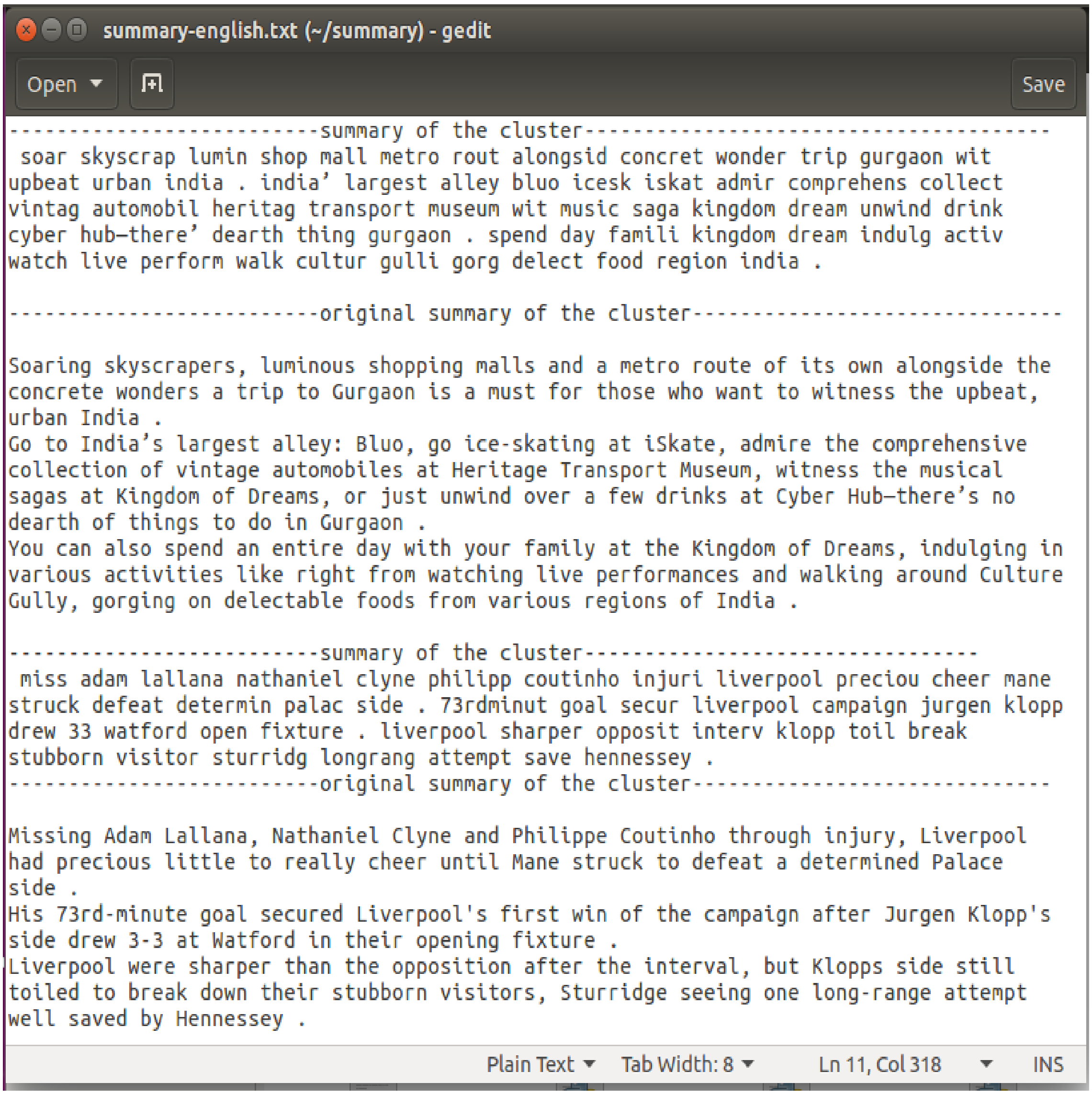
Figure 8.20: Conditional dilation of X



weightedhypergraph-english.dot (~/filter/summary) - gedit
Open
Save
Undo
weightedhypergraph-english.dot
filter-english-multicluster.py

```
---------------------------------erosion---------------------------------------
subx complement edges 
[1, 3, 4, 7, 8, 9]
subh complement edges
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 3, 4, 7, 8, 9]
summary edges [2, 5, 6]
---------------summary of the cluster-----------------


 soar skyscrap lumin shop mall metro rout alongsid concret wonder trip gurgaon wit 
upbeat urban india . india’ largest alley bluo icesk iskat admir comprehens collect 
vintag automobil heritag transport museum wit music saga kingdom dream unwind drink 
cyber hub—there’ dearth thing gurgaon . spend day famili kingdom dream indulg activ 
watch live perform walk cultur gulli gorg delect food region india .

------------original summary of the cluster-----------------

Soaring skyscrapers, luminous shopping malls and a metro route of its own alongside 
the concrete wonders a trip to Gurgaon is a must for those who want to witness the 
upbeat, urban India .
Go to India’s largest alley: Bluo, go ice-skating at iSkate, admire the comprehensive 
collection of vintage automobiles at Heritage Transport Museum, witness the musical 
sagas at Kingdom of Dreams, or just unwind over a few drinks at Cyber Hub—there’s no 
dearth of things to do in Gurgaon .
You can also spend an entire day with your family at the Kingdom of Dreams, indulging 
in various activities like right from watching live performances and walking around 
Culture Gully, gorging on delectable foods from various regions of India .liverpool : 
```

Figure 8.21: Erosion of X to generate summary

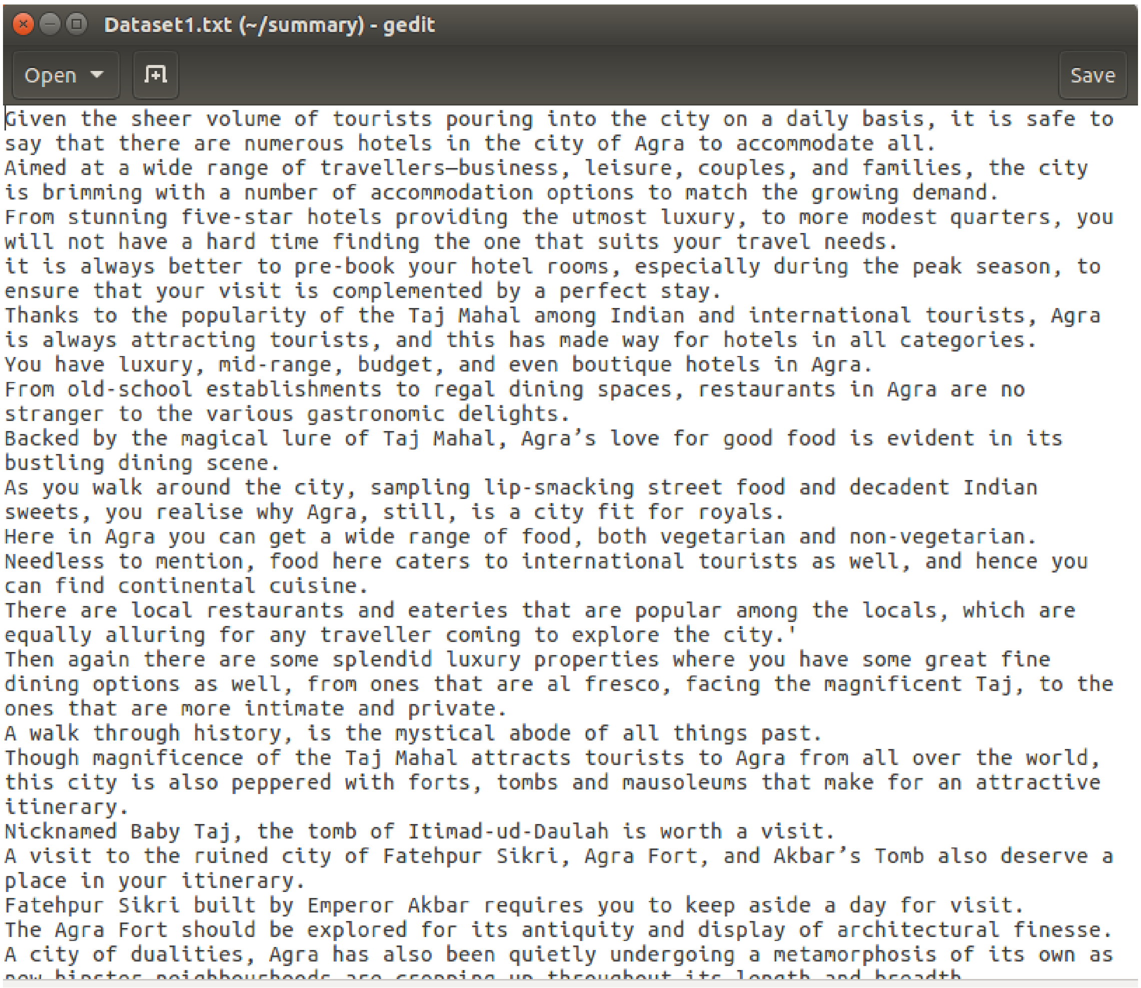


```
--------------------------summary of the cluster---------------------------------------
 soar skyscrap lumin shop mall metro rout alongsid concret wonder trip gurgaon wit
upbeat urban india . india’ largest alley bluo icesk iskat admir comprehens collect
vintag automobil heritag transport museum wit music saga kingdom dream unwind drink
cyber hub–there’ dearth thing gurgaon . spend day famili kingdom dream indulg activ
watch live perform walk cultur gulli gorg delect food region india .

--------------------------original summary of the cluster-------------------------------

Soaring skyscrapers, luminous shopping malls and a metro route of its own alongside the
concrete wonders a trip to Gurgaon is a must for those who want to witness the upbeat,
urban India .
Go to India’s largest alley: Bluo, go ice-skating at iSkate, admire the comprehensive
collection of vintage automobiles at Heritage Transport Museum, witness the musical
sagas at Kingdom of Dreams, or just unwind over a few drinks at Cyber Hub–there’s no
dearth of things to do in Gurgaon .
You can also spend an entire day with your family at the Kingdom of Dreams, indulging in
various activities like right from watching live performances and walking around Culture
Gully, gorging on delectable foods from various regions of India .

--------------------------summary of the cluster---------------------------------
 miss adam lallana nathaniel clyne philipp coutinho injuri liverpool preciou cheer mane
struck defeat determin palac side . 73rdminut goal secur liverpool campaign jurgen klopp
drew 33 watford open fixture . liverpool sharper opposit interv klopp toil break
stubborn visitor sturridg longrang attempt save hennessey .
--------------------------original summary of the cluster------------------------------

Missing Adam Lallana, Nathaniel Clyne and Philippe Coutinho through injury, Liverpool
had precious little to really cheer until Mane struck to defeat a determined Palace
side .
His 73rd-minute goal secured Liverpool's first win of the campaign after Jurgen Klopp's
side drew 3-3 at Watford in their opening fixture .
Liverpool were sharper than the opposition after the interval, but Klopps side still
toiled to break down their stubborn visitors, Sturridge seeing one long-range attempt
well saved by Hennessey .
```

Figure 8.22: Multi-cluster output in English

8.4 Multi-document IFHG

IFHG modeling is also applied to multiple documents where documents are modeled as hyperedges and keywords as nodes. In multi-document IFHG documents form the hyperedges and keywords form the nodes. In the previous modeling sentences were the hyperedges and words were the nodes. Let us see some outputs obtained when this multiple document IFHG is run in google cloud platform. Fig. 8.23 and Fig. 8.24 are two input documents given for summary generation. Fig. 8.25 is the multi-document IFHG and 8.26 gives the sub-IFHG formed. The morphological operations applied on this IFHG is given in Fig. 8.27, the final summary result id shown in Fig. 8.28.



Given the sheer volume of tourists pouring into the city on a daily basis, it is safe to say that there are numerous hotels in the city of Agra to accommodate all.

Aimed at a wide range of travellers–business, leisure, couples, and families, the city is brimming with a number of accommodation options to match the growing demand.

From stunning five-star hotels providing the utmost luxury, to more modest quarters, you will not have a hard time finding the one that suits your travel needs.

it is always better to pre-book your hotel rooms, especially during the peak season, to ensure that your visit is complemented by a perfect stay.

Thanks to the popularity of the Taj Mahal among Indian and international tourists, Agra is always attracting tourists, and this has made way for hotels in all categories.

You have luxury, mid-range, budget, and even boutique hotels in Agra.

From old-school establishments to regal dining spaces, restaurants in Agra are no stranger to the various gastronomic delights.

Backed by the magical lure of Taj Mahal, Agra’s love for good food is evident in its bustling dining scene.

As you walk around the city, sampling lip-smacking street food and decadent Indian sweets, you realise why Agra, still, is a city fit for royals.

Here in Agra you can get a wide range of food, both vegetarian and non-vegetarian.

Needless to mention, food here caters to international tourists as well, and hence you can find continental cuisine.

There are local restaurants and eateries that are popular among the locals, which are equally alluring for any traveller coming to explore the city.'

Then again there are some splendid luxury properties where you have some great fine dining options as well, from ones that are al fresco, facing the magnificent Taj, to the ones that are more intimate and private.

A walk through history, is the mystical abode of all things past.

Though magnificence of the Taj Mahal attracts tourists to Agra from all over the world, this city is also peppered with forts, tombs and mausoleums that make for an attractive itinerary.

Nicknamed Baby Taj, the tomb of Itimad-ud-Daulah is worth a visit.

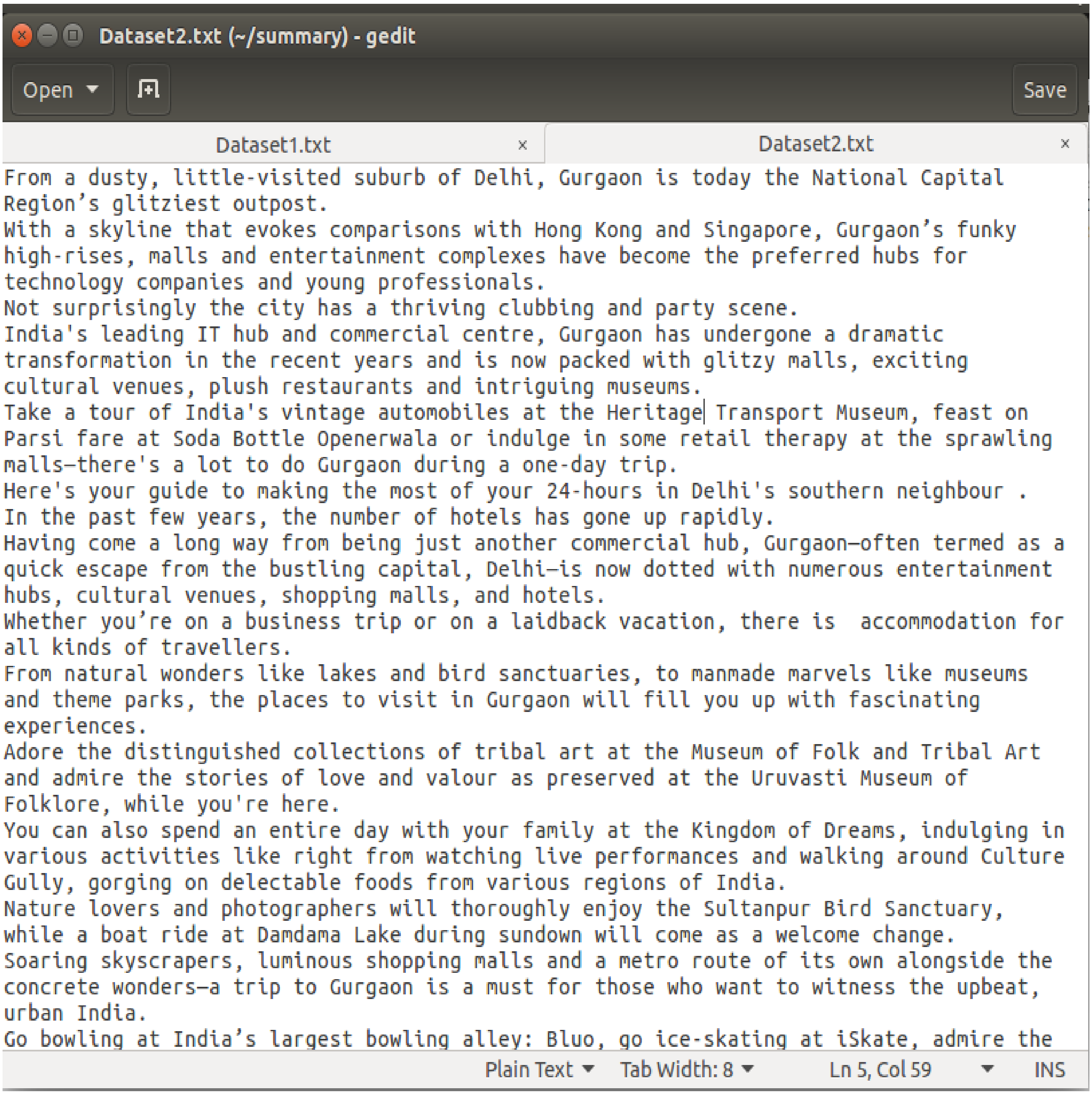
A visit to the ruined city of Fatehpur Sikri, Agra Fort, and Akbar’s Tomb also deserve a place in your itinerary.

Fatehpur Sikri built by Emperor Akbar requires you to keep aside a day for visit.

The Agra Fort should be explored for its antiquity and display of architectural finesse.

A city of dualities, Agra has also been quietly undergoing a metamorphosis of its own as new hipster neighbourhoods are cropping up throughout its length and breadth

FIGURE 8.23: English input1 for multi-document IFHG



From a dusty, little-visited suburb of Delhi, Gurgaon is today the National Capital Region’s glitziest outpost.

With a skyline that evokes comparisons with Hong Kong and Singapore, Gurgaon’s funky high-rises, malls and entertainment complexes have become the preferred hubs for technology companies and young professionals.

Not surprisingly the city has a thriving clubbing and party scene.

India's leading IT hub and commercial centre, Gurgaon has undergone a dramatic transformation in the recent years and is now packed with glitzy malls, exciting cultural venues, plush restaurants and intriguing museums.

Take a tour of India's vintage automobiles at the Heritage Transport Museum, feast on Parsi fare at Soda Bottle Openerwala or indulge in some retail therapy at the sprawling malls—there's a lot to do Gurgaon during a one-day trip.

Here's your guide to making the most of your 24-hours in Delhi's southern neighbour .

In the past few years, the number of hotels has gone up rapidly.

Having come a long way from being just another commercial hub, Gurgaon—often termed as a quick escape from the bustling capital, Delhi—is now dotted with numerous entertainment hubs, cultural venues, shopping malls, and hotels.

Whether you’re on a business trip or on a laidback vacation, there is accommodation for all kinds of travellers.

From natural wonders like lakes and bird sanctuaries, to manmade marvels like museums and theme parks, the places to visit in Gurgaon will fill you up with fascinating experiences.

Adore the distinguished collections of tribal art at the Museum of Folk and Tribal Art and admire the stories of love and valour as preserved at the Uruvasti Museum of Folklore, while you're here.

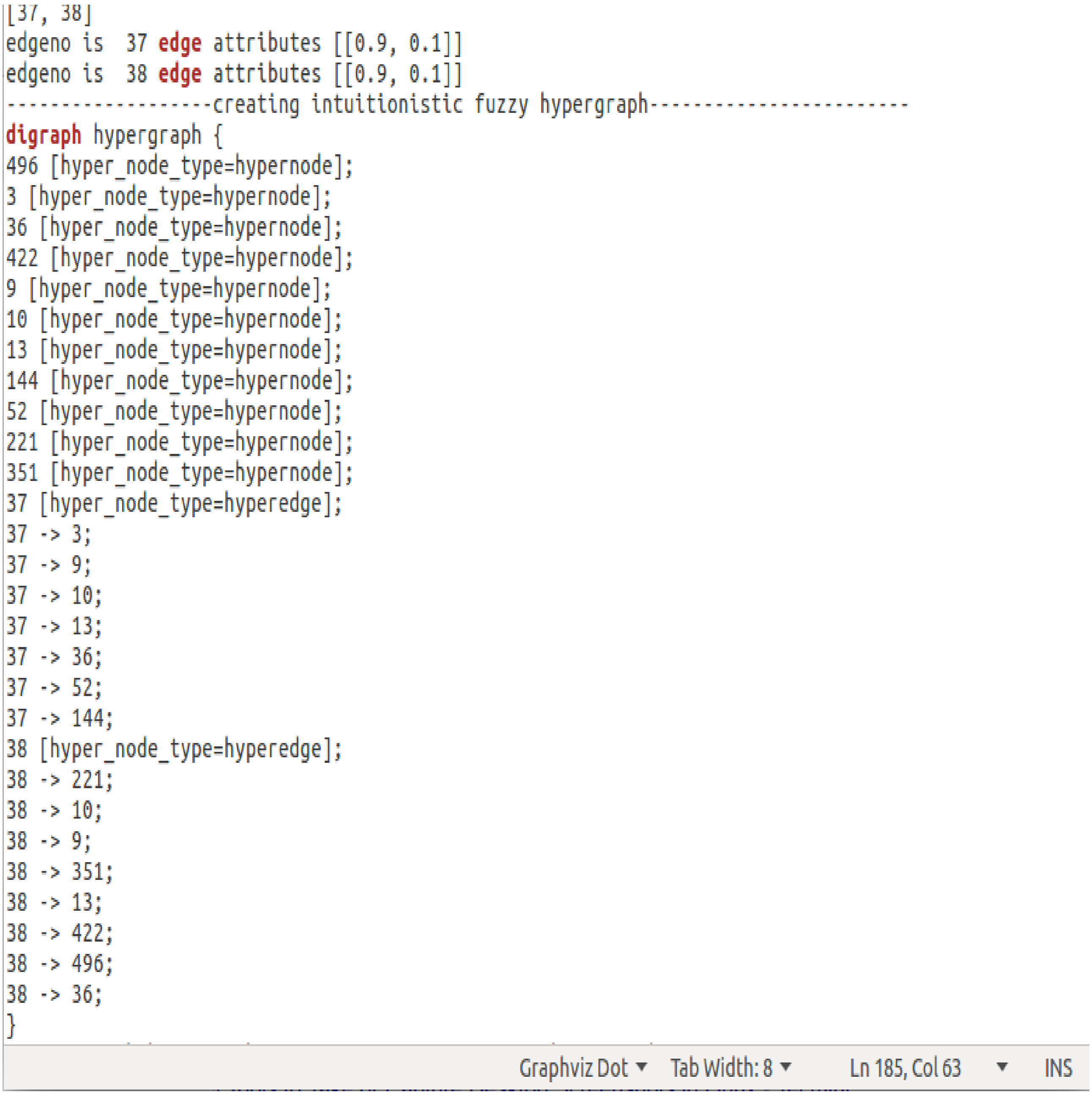
You can also spend an entire day with your family at the Kingdom of Dreams, indulging in various activities like right from watching live performances and walking around Culture Gully, gorging on delectable foods from various regions of India.

Nature lovers and photographers will thoroughly enjoy the Sultanpur Bird Sanctuary, while a boat ride at Damdama Lake during sundown will come as a welcome change.

Soaring skyscrapers, luminous shopping malls and a metro route of its own alongside the concrete wonders—a trip to Gurgaon is a must for those who want to witness the upbeat, urban India.

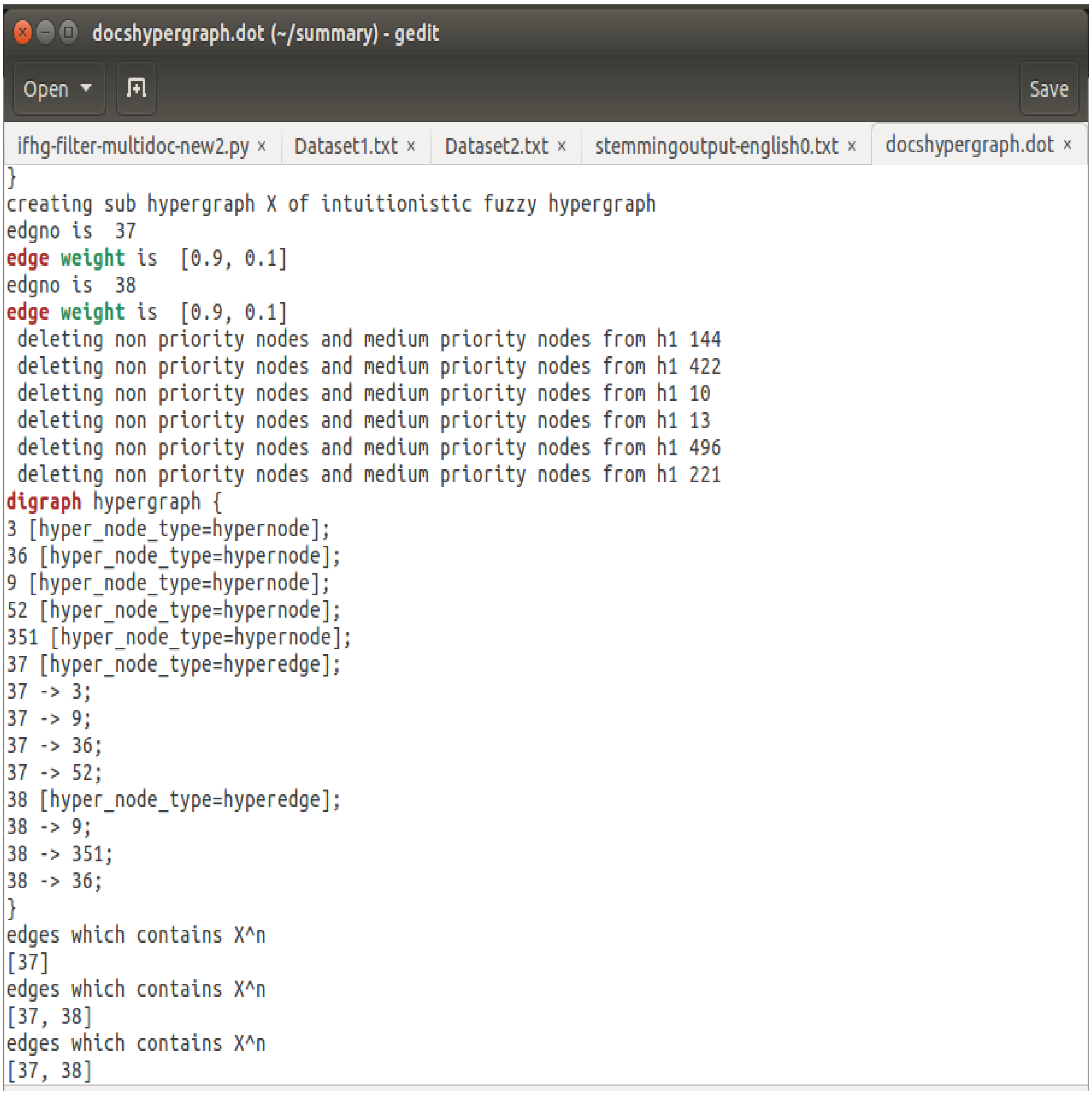
Go bowling at India’s largest bowling alley: Bluo, go ice-skating at iSkate, admire the

FIGURE 8.24: English input2 for multi-document IFHG



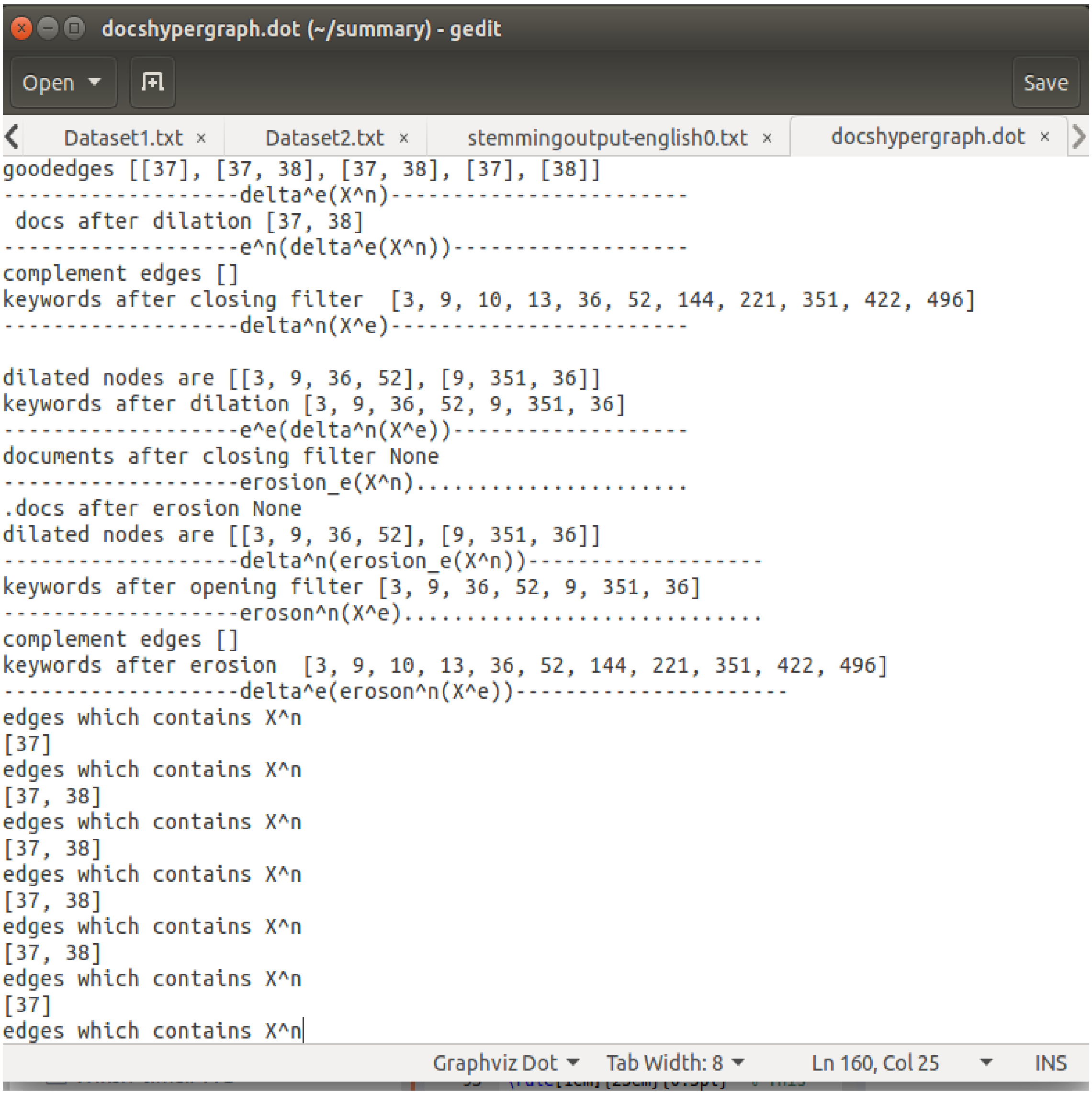
```
[37, 38]
edgeno is  37 edge attributes [[0.9, 0.1]]
edgeno is  38 edge attributes [[0.9, 0.1]]
-------------------creating intuitionistic fuzzy hypergraph-----------------------
digraph hypergraph {
496 [hyper_node_type=hypernode];
3 [hyper_node_type=hypernode];
36 [hyper_node_type=hypernode];
422 [hyper_node_type=hypernode];
9 [hyper_node_type=hypernode];
10 [hyper_node_type=hypernode];
13 [hyper_node_type=hypernode];
144 [hyper_node_type=hypernode];
52 [hyper_node_type=hypernode];
221 [hyper_node_type=hypernode];
351 [hyper_node_type=hypernode];
37 [hyper_node_type=hyperedge];
37 -> 3;
37 -> 9;
37 -> 10;
37 -> 13;
37 -> 36;
37 -> 52;
37 -> 144;
38 [hyper_node_type=hyperedge];
38 -> 221;
38 -> 10;
38 -> 9;
38 -> 351;
38 -> 13;
38 -> 422;
38 -> 496;
38 -> 36;
}
```

Figure 8.25: Multi-document IFHG formed from the input


docshypergraph.dot (~/summary) - gedit
Open
Save
ifhg-filter-multidoc-new2.py ×
Dataset1.txt ×
Dataset2.txt ×
stemmingoutput-english0.txt ×
docshypergraph.dot ×

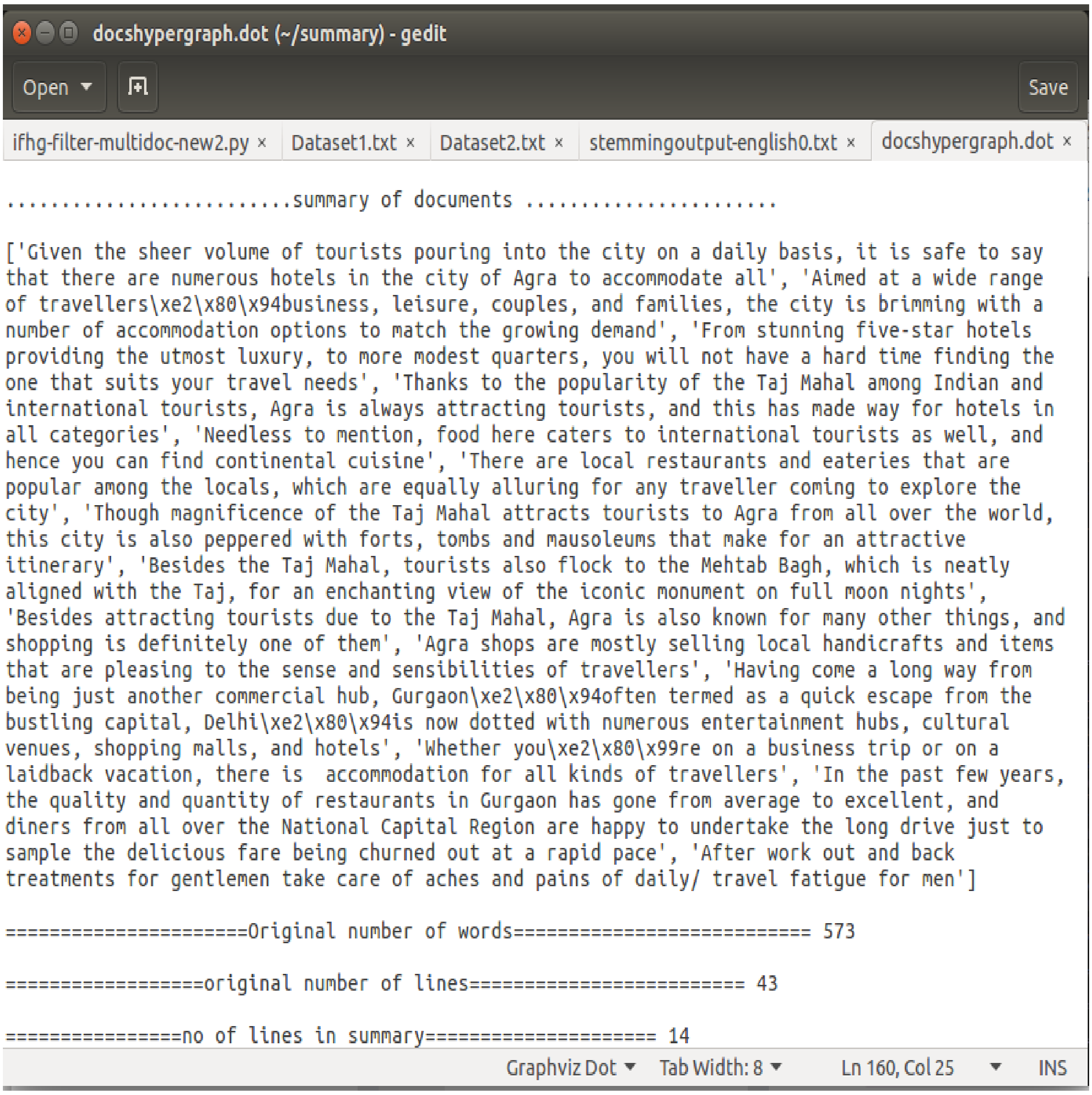
```
}
creating sub hypergraph X of intuitionistic fuzzy hypergraph
edgno is  37
edge weight is  [0.9, 0.1]
edgno is  38
edge weight is  [0.9, 0.1]
 deleting non priority nodes and medium priority nodes from h1 144
 deleting non priority nodes and medium priority nodes from h1 422
 deleting non priority nodes and medium priority nodes from h1 10
 deleting non priority nodes and medium priority nodes from h1 13
 deleting non priority nodes and medium priority nodes from h1 496
 deleting non priority nodes and medium priority nodes from h1 221
digraph hypergraph {
3 [hyper_node_type=hypernode];
36 [hyper_node_type=hypernode];
9 [hyper_node_type=hypernode];
52 [hyper_node_type=hypernode];
351 [hyper_node_type=hypernode];
37 [hyper_node_type=hyperedge];
37 -> 3;
37 -> 9;
37 -> 36;
37 -> 52;
38 [hyper_node_type=hyperedge];
38 -> 9;
38 -> 351;
38 -> 36;
}
edges which contains X^n
[37]
edges which contains X^n
[37, 38]
edges which contains X^n
[37, 38]
```

Figure 8.26: Sub-IFHG formed


docshypergraph.dot (~/summary) - gedit
Open
Save
Dataset1.txt ×
Dataset2.txt ×
stemmingoutput-english0.txt ×
docshypergraph.dot ×

```
goodedges [[37], [37, 38], [37, 38], [37], [38]]
-------------------delta^e(X^n)------------------------
 docs after dilation [37, 38]
-------------------e^n(delta^e(X^n))-------------------
complement edges []
keywords after closing filter  [3, 9, 10, 13, 36, 52, 144, 221, 351, 422, 496]
-------------------delta^n(X^e)------------------------

dilated nodes are [[3, 9, 36, 52], [9, 351, 36]]
keywords after dilation [3, 9, 36, 52, 9, 351, 36]
-------------------e^e(delta^n(X^e))-------------------
documents after closing filter None
-------------------erosion_e(X^n)......................
.docs after erosion None
dilated nodes are [[3, 9, 36, 52], [9, 351, 36]]
-------------------delta^n(erosion_e(X^n))------------------
keywords after opening filter [3, 9, 36, 52, 9, 351, 36]
-------------------eroson^n(X^e)............................
complement edges []
keywords after erosion  [3, 9, 10, 13, 36, 52, 144, 221, 351, 422, 496]
-------------------delta^e(eroson^n(X^e))----------------------
edges which contains X^n
[37]
edges which contains X^n
[37, 38]
edges which contains X^n
[37, 38]
edges which contains X^n
[37, 38]
edges which contains X^n
[37, 38]
edges which contains X^n
[37]
edges which contains X^n
```

Figure 8.27: Morphological operations on IFHG


docshypergraph.dot (~/summary) - gedit
Open
Save
ifhg-filter-multidoc-new2.py ×
Dataset1.txt ×
Dataset2.txt ×
stemmingoutput-english0.txt ×
docshypergraph.dot ×
Graphviz Dot
Tab Width: 8
Ln 160, Col 25
INS

```
.........................summary of documents ......................

['Given the sheer volume of tourists pouring into the city on a daily basis, it is safe to say
that there are numerous hotels in the city of Agra to accommodate all', 'Aimed at a wide range
of travellers\xe2\x80\x94business, leisure, couples, and families, the city is brimming with a
number of accommodation options to match the growing demand', 'From stunning five-star hotels
providing the utmost luxury, to more modest quarters, you will not have a hard time finding the
one that suits your travel needs', 'Thanks to the popularity of the Taj Mahal among Indian and
international tourists, Agra is always attracting tourists, and this has made way for hotels in
all categories', 'Needless to mention, food here caters to international tourists as well, and
hence you can find continental cuisine', 'There are local restaurants and eateries that are
popular among the locals, which are equally alluring for any traveller coming to explore the
city', 'Though magnificence of the Taj Mahal attracts tourists to Agra from all over the world,
this city is also peppered with forts, tombs and mausoleums that make for an attractive
itinerary', 'Besides the Taj Mahal, tourists also flock to the Mehtab Bagh, which is neatly
aligned with the Taj, for an enchanting view of the iconic monument on full moon nights',
'Besides attracting tourists due to the Taj Mahal, Agra is also known for many other things, and
shopping is definitely one of them', 'Agra shops are mostly selling local handicrafts and items
that are pleasing to the sense and sensibilities of travellers', 'Having come a long way from
being just another commercial hub, Gurgaon\xe2\x80\x94often termed as a quick escape from the
bustling capital, Delhi\xe2\x80\x94is now dotted with numerous entertainment hubs, cultural
venues, shopping malls, and hotels', 'Whether you\xe2\x80\x99re on a business trip or on a
laidback vacation, there is  accommodation for all kinds of travellers', 'In the past few years,
the quality and quantity of restaurants in Gurgaon has gone from average to excellent, and
diners from all over the National Capital Region are happy to undertake the long drive just to
sample the delicious fare being churned out at a rapid pace', 'After work out and back
treatments for gentlemen take care of aches and pains of daily/ travel fatigue for men']

=====================Original number of words========================== 573

=================original number of lines======================== 43

===============no of lines in summary==================== 14
```

FIGURE 8.28: Multi-document summary from IFHG filter

8.5 Advantages of IFHG modeling and summary

1. Hypergraph structure of the text is more meaningful when compared to the normal graph structure of text.

2. Hypergraph actually shows, how the entire text is structured and how the sentences are related to each other.

3. The membership and non-membership values given to the nodes and hyperedges show the wantedness and unwantedness of the words and sentences.

4. Multi-document IFHG created is having only less number of nodes. This reduces the space required for storing the data structure.

5. Priority levels are not limited to high, medium and low. Different levels of priorities like very high, high, medium high, low, very low can be set with the help of the membership and non-membership degrees of the words as well as the documents.

6. Increasing the threshold of (α, β) cut results in a very short summary, where by reducing it results in a lengthy summary.

7. Range oriented (α, β) cuts like $value1 < \alpha < value2$ are also possible which results in different sets of summaries.

8. Union/intersection of filters result in union/intersection of keywords. Since filters w.r.to hyperedges result in selection of good documents, they combined with node filters result in summary. Union /intersection of summaries are also possible.

8.6 Disadvantages of the system

1. Since word sense disambiguation is not implemented, the system expects unambiguous text as input.

2. If the input is ambiguous, it may affect the performance of the clustering phase.

3. Since all words except stop words are subjected to lemmatization, the performance depends on lemmatizer also.

4. Lemmatizer makes the system language depended, even though the rest of the modules does not depend on the language. If the system need to be extended to other languages, a stemmer in their own language is required.

8.7 Conclusion

This is a premier work which models text as IFHG with words modeled as nodes and sentences modeled as hyperedges. This is a novel work which applies morphological filtering on IFHG. This is a premier work which creates document summary using morphological filter. This is also the first work which models multiple documents as IFHG where nodes are the keywords and hyperedges are the documents. Better results are obtained when compared with other online summary systems. The summary generated by these two systems have shown more similarity with human summaries. Further extensions of the work are possible with other graph operations on IFHG. System is yet to explore the application of many hypergraph operations like graph join, cartesian product etc., on text IFHG. This IFHG modeling can be extended to other areas like image processing, networking etc.

8.8 Future enhancement

This IFHG modelling can be extended to modeling of crimes in various police stations across Kerala and developing a Criminal Policing System, which provides alerts based on crime history. The work in this direction has already been started as a project funded by Kerala Technical University - Centre for Research and Development(KTU-CERD). The project aims in developing a fuzzy based crime/theft alert system for criminal policing system. The goals of the project are the following:

- Predict probability of crime by combining information from multiple police stations.
- Give alerts to public and authorities.
- Create a website which can be used by both police and public.
- Create a mobile app for both police and public which is connected to Google map.

This project models an intuitionistic fuzzy hypergraph by considering the crime categories, individual crimes and the criminals involved in each crime.

As specific examples of wide applicability of IFHG models, we have already mentioned medical report processing and social network analysis in chapter 4.

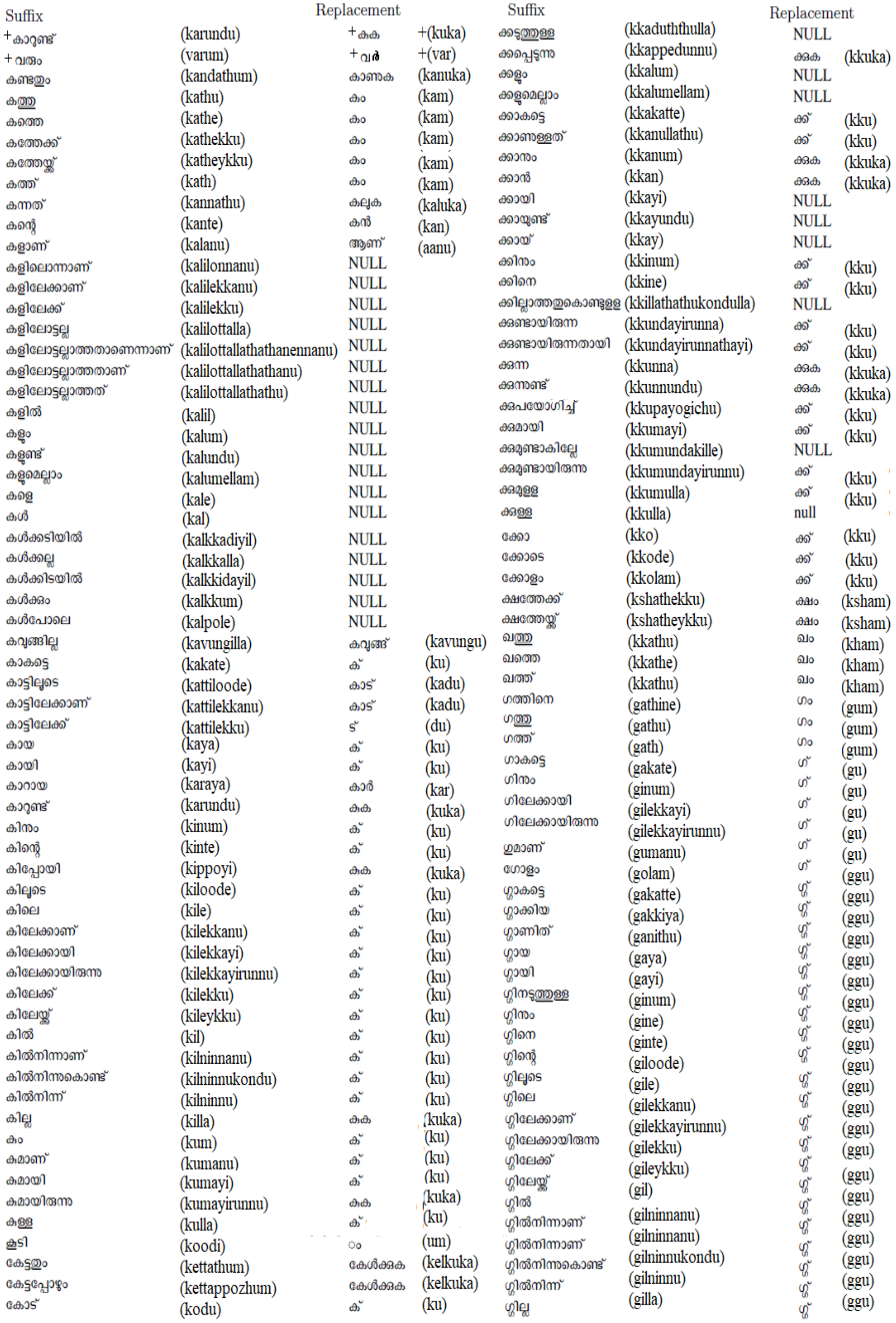
Appendix A

Malayalam Lemmatizer

The Malayalam Lemmatizer developed using php-myadmin uses nearly 1135 suffix replacement rules. The lemmatizer is developed in two methods:

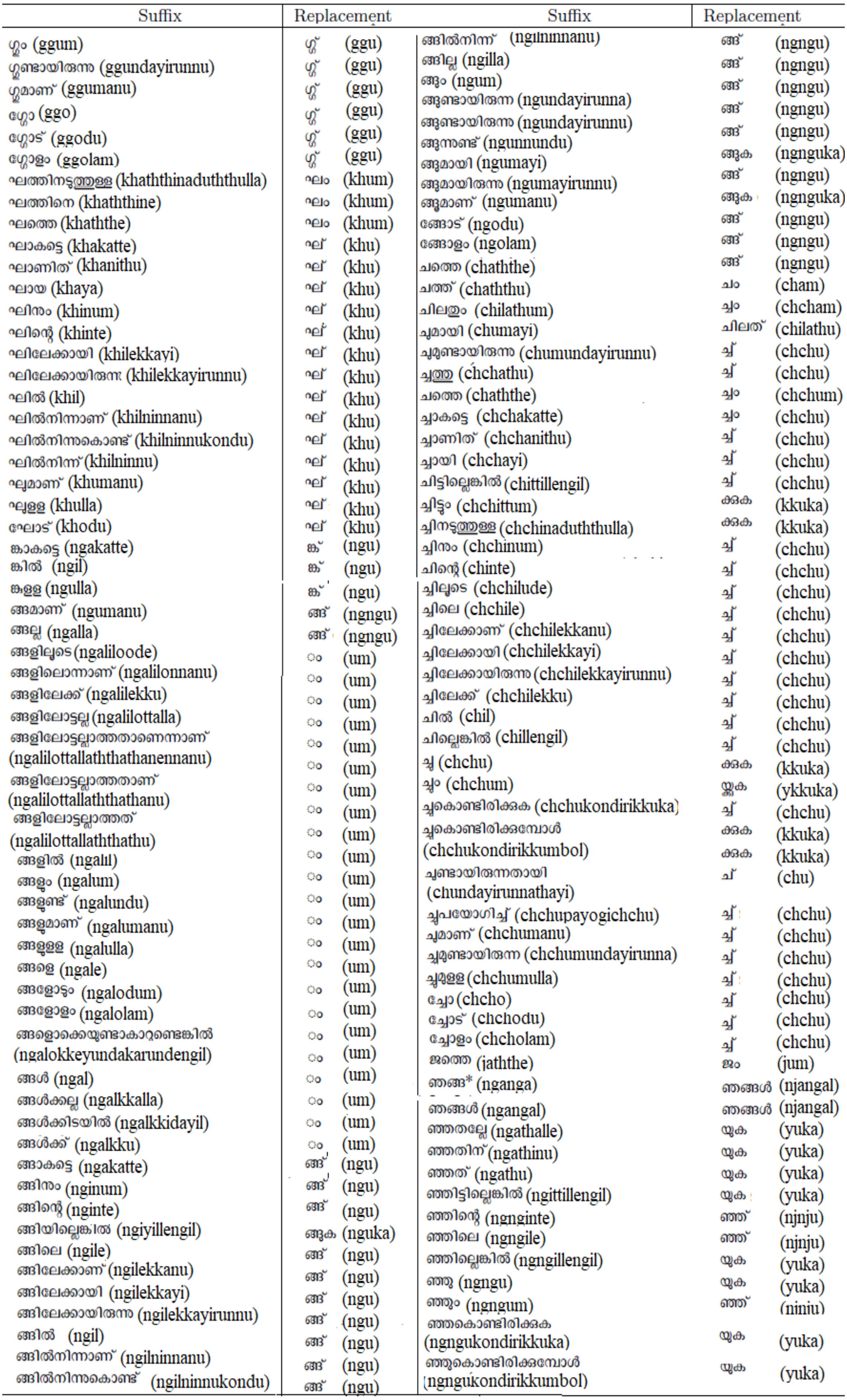
- Dictionary Based method
- Tree based method

The suffix replacement rules are sorted alphabetically in the order of suffix. They are shown in the following figures.



Suffix		Replacement		Suffix		Replacement	
+കാറുണ്ട്	(karundu)	+കുക	+(kuka)	ക്കടുത്തുള്ള	(kkaduththulla)	NULL	
+വരും	(varum)	+വർ	+(var)	ക്കപ്പെടുന്നു	(kkappedunnu)	ക്കുക	(kkuka)
കണ്ടതും	(kandathum)	കാണുക	(kanuka)	ക്കളും	(kkalum)	NULL	
കത്തു	(kathu)	കം	(kam)	ക്കളുമെല്ലാം	(kkalumellam)	NULL	
കത്തെ	(kathe)	കം	(kam)	ക്കാകട്ടെ	(kkakatte)	ക്ക്	(kku)
കത്തേക്ക്	(kathekku)	കം	(kam)	ക്കാണുള്ളത്	(kkanullathu)	ക്ക്	(kku)
കത്തേയ്ക്ക്	(katheykku)	കം	(kam)	ക്കാനും	(kkanum)	ക്കുക	(kkuka)
കത്ത്	(kath)	കം	(kam)	ക്കാൻ	(kkan)	ക്കുക	(kkuka)
കന്നത്	(kannathu)	കലുക	(kaluka)	ക്കായി	(kkayi)	NULL	
കന്റെ	(kante)	കൻ	(kan)	ക്കായുണ്ട്	(kkayundu)	NULL	
കളാണ്	(kalanu)	ആണ്	(aanu)	ക്കായ്	(kkay)	NULL	
കളിലൊന്നാണ്	(kalilonnanu)	NULL		ക്കിനും	(kkinum)	ക്ക്	(kku)
കളിലേക്കാണ്	(kalilekkanu)	NULL		ക്കിനെ	(kkine)	ക്ക്	(kku)
കളിലേക്ക്	(kalilekku)	NULL		ക്കില്ലാത്തതുകൊണ്ടുള്ള	(kkillathathukondulla)	NULL	
കളിലോട്ടല്ല	(kalilottalla)	NULL		ക്കുണ്ടായിരുന്ന	(kkundayirunna)	ക്ക്	(kku)
കളിലോട്ടല്ലാത്തതാണെന്നാണ്	(kalilottallathathanennanu)	NULL		ക്കുണ്ടായിരുന്നതായി	(kkundayirunnathayi)	ക്ക്	(kku)
കളിലോട്ടല്ലാത്തതാണ്	(kalilottallathathanu)	NULL		ക്കുന്ന	(kkunna)	ക്കുക	(kkuka)
കളിലോട്ടല്ലാത്തത്	(kalilottallathathu)	NULL		ക്കുന്നുണ്ട്	(kkunnundu)	ക്കുക	(kkuka)
കളിൽ	(kalil)	NULL		ക്കുപയോഗിച്ച്	(kkupayogichu)	ക്ക്	(kku)
കളും	(kalum)	NULL		ക്കുമായി	(kkumayi)	ക്ക്	(kku)
കളുണ്ട്	(kalundu)	NULL		ക്കുമുണ്ടാകില്ലേ	(kkumundakille)	NULL	
കളുമെല്ലാം	(kalumellam)	NULL		ക്കുമുണ്ടായിരുന്നു	(kkumundayirunnu)	ക്ക്	(kku)
കളെ	(kale)	NULL		ക്കുമുള്ള	(kkumulla)	ക്ക്	(kku)
കൾ	(kal)	NULL		ക്കുള്ള	(kkulla)	null	
കൾക്കടിയിൽ	(kalkkadiyil)	NULL		ക്കോ	(kko)	ക്ക്	(kku)
കൾക്കല്ല	(kalkkalla)	NULL		ക്കോടെ	(kkode)	ക്ക്	(kku)
കൾക്കിടയിൽ	(kalkkidayil)	NULL		ക്കോളം	(kkolam)	ക്ക്	(kku)
കൾക്കും	(kalkkum)	NULL		ക്ഷത്തേക്ക്	(kshathekku)	ക്ഷം	(ksham)
കൾപോലെ	(kalpole)	NULL		ക്ഷത്തേയ്ക്ക്	(kshatheykku)	ക്ഷം	(ksham)
കവുങ്ങില്ല	(kavungilla)	കവുങ്ങ്	(kavungu)	ഖത്തു	(kkathu)	ഖം	(kham)
കാകട്ടെ	(kakate)	ക്	(ku)	ഖത്തെ	(kkathe)	ഖം	(kham)
കാട്ടിലൂടെ	(kattiloode)	കാട്	(kadu)	ഖത്ത്	(kkathu)	ഖം	(kham)
കാട്ടിലേക്കാണ്	(kattilekkanu)	കാട്	(kadu)	ഗത്തിനെ	(gathine)	ഗം	(gum)
കാട്ടിലേക്ക്	(kattilekku)	ട്	(du)	ഗത്തു	(gathu)	ഗം	(gum)
കായ	(kaya)	ക്	(ku)	ഗത്ത്	(gath)	ഗം	(gum)
കായി	(kayi)	ക്	(ku)	ഗാകട്ടെ	(gakate)	ഗ്	(gu)
കാറായ	(karaya)	കാർ	(kar)	ഗിനും	(ginum)	ഗ്	(gu)
കാറുണ്ട്	(karundu)	കുക	(kuka)	ഗിലേക്കായി	(gilekkayi)	ഗ്	(gu)
കിനും	(kinum)	ക്	(ku)	ഗിലേക്കായിരുന്നു	(gilekkayirunnu)	ഗ്	(gu)
കിന്റെ	(kinte)	ക്	(ku)	ഗുമാണ്	(gumanu)	ഗ്	(gu)
കിപ്പോയി	(kippoyi)	കുക	(kuka)	ഗോളം	(golam)	ഗ്	(ggu)
കിലൂടെ	(kiloode)	ക്	(ku)	ഗ്ഗാകട്ടെ	(gakatte)	ഗ്ഗ്	(ggu)
കിലെ	(kile)	ക്	(ku)	ഗ്ഗാക്കിയ	(gakkiya)	ഗ്ഗ്	(ggu)
കിലേക്കാണ്	(kilekkanu)	ക്	(ku)	ഗ്ഗാണിത്	(ganithu)	ഗ്ഗ്	(ggu)
കിലേക്കായി	(kilekkayi)	ക്	(ku)	ഗ്ഗായ	(gaya)	ഗ്ഗ്	(ggu)
കിലേക്കായിരുന്നു	(kilekkayirunnu)	ക്	(ku)	ഗ്ഗായി	(gayi)	ഗ്ഗ്	(ggu)
കിലേക്ക്	(kilekku)	ക്	(ku)	ഗ്ഗിനടുത്തുള്ള	(ginum)	ഗ്ഗ്	(ggu)
കിലേയ്ക്ക്	(kileykku)	ക്	(ku)	ഗ്ഗിനും	(gine)	ഗ്ഗ്	(ggu)
കിൽ	(kil)	ക്	(ku)	ഗ്ഗിനെ	(ginte)	ഗ്ഗ്	(ggu)
കിൽനിന്നാണ്	(kilninnanu)	ക്	(ku)	ഗ്ഗിന്റെ	(giloode)	ഗ്ഗ്	(ggu)
കിൽനിന്നുകൊണ്ട്	(kilninnukondu)	ക്	(ku)	ഗ്ഗിലൂടെ	(gile)	ഗ്ഗ്	(ggu)
കിൽനിന്ന്	(kilninnu)	ക്	(ku)	ഗ്ഗിലെ	(gilekkanu)	ഗ്ഗ്	(ggu)
കില്ല	(killa)	കുക	(kuka)	ഗ്ഗിലേക്കാണ്	(gilekkayirunnu)	ഗ്ഗ്	(ggu)
കും	(kum)	ക്	(ku)	ഗ്ഗിലേക്കായിരുന്നു	(gilekku)	ഗ്ഗ്	(ggu)
കുമാണ്	(kumanu)	ക്	(ku)	ഗ്ഗിലേക്ക്	(gileykku)	ഗ്ഗ്	(ggu)
കുമായി	(kumayi)	ക്	(ku)	ഗ്ഗിലേയ്ക്ക്	(gil)	ഗ്ഗ്	(ggu)
കുമായിരുന്നു	(kumayirunnu)	കുക	(kuka)	ഗ്ഗിൽ	(gilninnanu)	ഗ്ഗ്	(ggu)
കുള്ള	(kulla)	ക്	(ku)	ഗ്ഗിൽനിന്നാണ്	(gilninnanu)	ഗ്ഗ്	(ggu)
കൂടി	(koodi)	◌ം	(um)	ഗ്ഗിൽനിന്നാണ്	(gilninnukondu)	ഗ്ഗ്	(ggu)
കേട്ടതും	(kettathum)	കേൾക്കുക	(kelkuka)	ഗ്ഗിൽനിന്നുകൊണ്ട്	(gilninnu)	ഗ്ഗ്	(ggu)
കേട്ടപ്പോഴും	(kettappozhum)	കേൾക്കുക	(kelkuka)	ഗ്ഗിൽനിന്ന്	(gilla)	ഗ്ഗ്	(ggu)
കോട്	(kodu)	ക്	(ku)	ഗ്ഗില്ല		ഗ്ഗ്	(ggu)

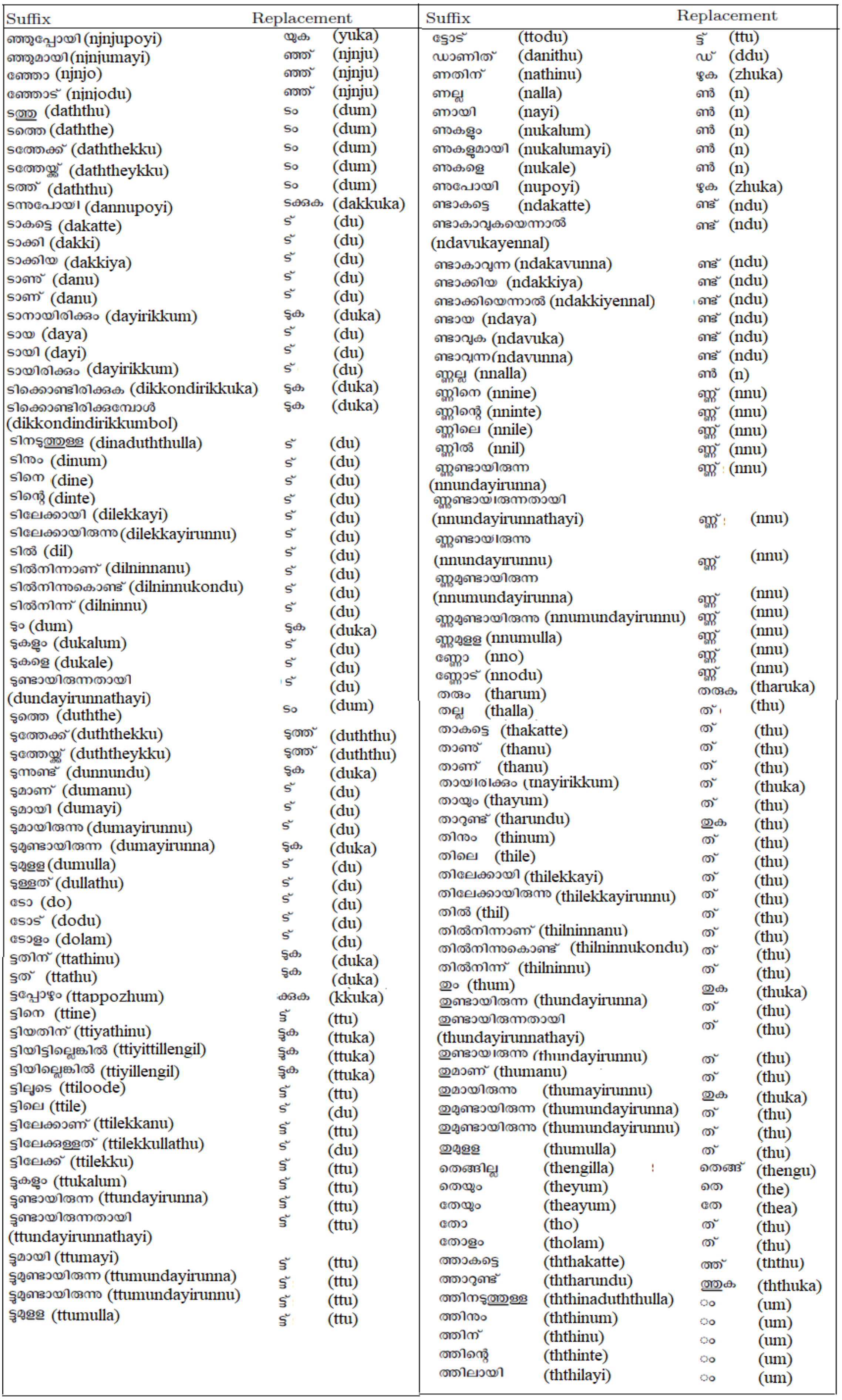
FIGURE A.1: Suffix Replacement rules starting with 'ka'/'ga'



Suffix	Replacement
ഗ്ഗും (ggum)	ഗ്ഗ് (ggu)
ഗ്ഗുണ്ടായിരുന്നു (ggundayirunnu)	ഗ്ഗ് (ggu)
ഗ്ഗുമാണ് (ggumanu)	ഗ്ഗ് (ggu)
ഗ്ഗോ (ggo)	ഗ്ഗ് (ggu)
ഗ്ഗോട് (ggodu)	ഗ്ഗ് (ggu)
ഗ്ഗോളം (ggolam)	ഗ്ഗ് (ggu)
ഘത്തിനടുത്തുള്ള (khaththinaduththulla)	ഘം (khum)
ഘത്തിനെ (khaththine)	ഘം (khum)
ഘത്തെ (khaththe)	ഘം (khum)
ഘാകട്ടെ (khakatte)	ഘ് (khu)
ഘാണിത് (khanithu)	ഘ് (khu)
ഘായ (khaya)	ഘ് (khu)
ഘിനും (khinum)	ഘ് (khu)
ഘിന്റെ (khinte)	ഘ് (khu)
ഘിലേക്കായി (khilekkayi)	ഘ് (khu)
ഘിലേക്കായിരുന്നു (khilekkayirunnu)	ഘ് (khu)
ഘിൽ (khil)	ഘ് (khu)
ഘിൽനിന്നാണ് (khilninnanu)	ഘ് (khu)
ഘിൽനിന്നുകൊണ്ട് (khilninnukondu)	ഘ് (khu)
ഘിൽനിന്ന് (khilninnu)	ഘ് (khu)
ഘുമാണ് (khumanu)	ഘ് (khu)
ഘുള്ള (khulla)	ഘ് (khu)
ഘോട് (khodu)	ഘ് (khu)
ങാകട്ടെ (ngakatte)	ങ് (ngu)
ങിൽ (ngil)	ങ് (ngu)
ങുള്ള (ngulla)	ങ് (ngu)
ങ്ങമാണ് (ngumanu)	ങ്ങ് (ngngu)
ങ്ങല്ല (ngalla)	ങ്ങ് (ngngu)
ങ്ങളിലൂടെ (ngaliloode)	◌ം (um)
ങ്ങളിലൊന്നാണ് (ngalilonnanu)	◌ം (um)
ങ്ങളിലേക്ക് (ngalilekku)	◌ം (um)
ങ്ങളിലോട്ടല്ല (ngalilottalla)	◌ം (um)
ങ്ങളിലോട്ടല്ലാത്തതാണെന്നാണ് (ngalilottallaththathanennanu)	◌ം (um) ◌ം (um)
ങ്ങളിലോട്ടല്ലാത്തതാണ് (ngalilottallaththathanu)	◌ം (um) ◌ം (um)
ങ്ങളിലോട്ടല്ലാത്തത് (ngalilottallaththathu)	◌ം (um) ◌ം (um)
ങ്ങളിൽ (ngalil)	◌ം (um)
ങ്ങളും (ngalum)	◌ം (um)
ങ്ങളുണ്ട് (ngalundu)	◌ം (um)
ങ്ങളുമാണ് (ngalumanu)	◌ം (um)
ങ്ങളുള്ള (ngalulla)	◌ം (um)
ങ്ങളെ (ngale)	◌ം (um)
ങ്ങളോടും (ngalodum)	◌ം (um)
ങ്ങളോളം (ngalolam)	◌ം (um)
ങ്ങളൊക്കെയുണ്ടാകാറുണ്ടെങ്കിൽ (ngalokkeyundakarundengil)	◌ം (um) ◌ം (um)
ങ്ങൾ (ngal)	◌ം (um)
ങ്ങൾക്കല്ല (ngalkkalla)	◌ം (um)
ങ്ങൾക്കിടയിൽ (ngalkkidayil)	◌ം (um)
ങ്ങൾക്ക് (ngalkku)	◌ം (um)
ങ്ങാകട്ടെ (ngakatte)	ങ്ങ് (ngu)
ങ്ങിനും (nginum)	ങ്ങ് (ngu)
ങ്ങിന്റെ (nginte)	ങ്ങ് (ngu)
ങ്ങിയില്ലെങ്കിൽ (ngiyillengil)	ങ്ങുക (nguka)
ങ്ങിലെ (ngile)	ങ്ങ് (ngu)
ങ്ങിലേക്കാണ് (ngilekkanu)	ങ്ങ് (ngu)
ങ്ങിലേക്കായി (ngilekkayi)	ങ്ങ് (ngu)
ങ്ങിലേക്കായിരുന്നു (ngilekkayirunnu)	ങ്ങ് (ngu)
ങ്ങിൽ (ngil)	ങ്ങ് (ngu)
ങ്ങിൽനിന്നാണ് (ngilninnanu)	ങ്ങ് (ngu)
ങ്ങിൽനിന്നുകൊണ്ട് (ngilninnukondu)	ങ്ങ് (ngu)

Suffix	Replacement
ങ്ങിൽനിന്ന് (ngilninnanu)	ങ്ങ് (ngngu)
ങ്ങില്ല (ngilla)	ങ്ങ് (ngngu)
ങ്ങും (ngum)	ങ്ങ് (ngngu)
ങ്ങുണ്ടായിരുന്ന (ngundayirunna)	ങ്ങ് (ngngu)
ങ്ങുണ്ടായിരുന്നു (ngundayirunnu)	ങ്ങ് (ngngu)
ങ്ങുന്നുണ്ട് (ngunnundu)	ങ്ങുക (ngnguka)
ങ്ങുമായി (ngumayi)	
ങ്ങുമായിരുന്നു (ngumayirunnu)	ങ്ങ് (ngngu)
ങ്ങുമാണ് (ngumanu)	ങ്ങുക (ngnguka)
ങ്ങോട് (ngodu)	ങ്ങ് (ngngu)
ങ്ങോളം (ngolam)	ങ്ങ് (ngngu)
ചത്തെ (chaththe)	ങ്ങ് (ngngu)
ചത്ത് (chaththu)	ചം (cham)
ചിലതും (chilathum)	ച്ചം (chcham)
ചുമായി (chumayi)	ചിലത് (chilathu)
ചുമുണ്ടായിരുന്നു (chumundayirunnu)	ച്ച് (chchu)
ച്ചത്ത (chchathu)	ച്ച് (chchu)
ചത്തെ (chaththe)	ച്ചം (chchum)
ച്ചാകട്ടെ (chchakatte)	ച്ചം (chchu)
ച്ചാണിത് (chchanithu)	ച്ച് (chchu)
ച്ചായി (chchayi)	ച്ച് (chchu)
ചിട്ടില്ലെങ്കിൽ (chittillengil)	ച്ച് (chchu)
ച്ചിട്ടും (chchittum)	ക്കുക (kkuka)
ച്ചിനടുത്തുള്ള (chchinaduththulla)	ക്കുക (kkuka)
ച്ചിനും (chchinum)	ച്ച് (chchu)
ചിന്റെ (chinte)	ച്ച് (chchu)
ച്ചിലൂടെ (chchilude)	ച്ച് (chchu)
ച്ചിലെ (chchile)	ച്ച് (chchu)
ച്ചിലേക്കാണ് (chchilekkanu)	ച്ച് (chchu)
ച്ചിലേക്കായി (chchilekkayi)	ച്ച് (chchu)
ച്ചിലേക്കായിരുന്നു (chchilekkayirunnu)	ച്ച് (chchu)
ച്ചിലേക്ക് (chchilekku)	ച്ച് (chchu)
ചിൽ (chil)	ച്ച് (chchu)
ചില്ലെങ്കിൽ (chillengil)	ച്ച് (chchu)
ച്ചു (chchu)	ക്കുക (kkuka)
ച്ചും (chchum)	യ്ക്കുക (ykkuka)
ച്ചുകൊണ്ടിരിക്കുക (chchukondirikkuka)	ച്ച് (chchu)
ച്ചുകൊണ്ടിരിക്കുമ്പോൾ (chchukondirikkumbol)	ക്കുക (kkuka) ക്കുക (kkuka)
ചുണ്ടായിരുന്നതായി (chundayirunnathayi)	ച് (chu)
ച്ചുപയോഗിച്ച് (chchupayogichchu)	ച്ച് (chchu)
ച്ചുമാണ് (chchumanu)	ച്ച് (chchu)
ച്ചുമുണ്ടായിരുന്ന (chchumundayirunna)	ച്ച് (chchu)
ച്ചുമുള്ള (chchumulla)	ച്ച് (chchu)
ച്ചോ (chcho)	ച്ച് (chchu)
ച്ചോട് (chchodu)	ച്ച് (chchu)
ച്ചോളം (chcholam)	ച്ച് (chchu)
ജത്തെ (jaththe)	ജം (jum)
ഞങ്ങ* (nganga)	ഞങ്ങൾ (njangal)
ഞങ്ങൾ (ngangal)	ഞങ്ങൾ (njangal)
ഞ്ഞതല്ലേ (ngathalle)	യുക (yuka)
ഞ്ഞതിന് (ngathinu)	യുക (yuka)
ഞ്ഞത് (ngathu)	യുക (yuka)
ഞ്ഞിട്ടില്ലെങ്കിൽ (ngittillengil)	യുക (yuka)
ഞ്ഞിന്റെ (ngnginte)	ഞ്ഞ് (njnju)
ഞ്ഞിലെ (ngngile)	ഞ്ഞ് (njnju)
ഞ്ഞില്ലെങ്കിൽ (ngngillengil)	യുക (yuka)
ഞ്ഞു (ngngu)	യുക (yuka)
ഞ്ഞും (ngngum)	ഞ്ഞ് (njnju)
ഞ്ഞകൊണ്ടിരിക്കുക (ngngukondirikkuka)	യുക (yuka)
ഞ്ഞുകൊണ്ടിരിക്കുമ്പോൾ (ngngukondirikkumbol)	യുക (yuka)

FIGURE A.2: Suffix Replacement rules starting with 'ga'-'njnja'



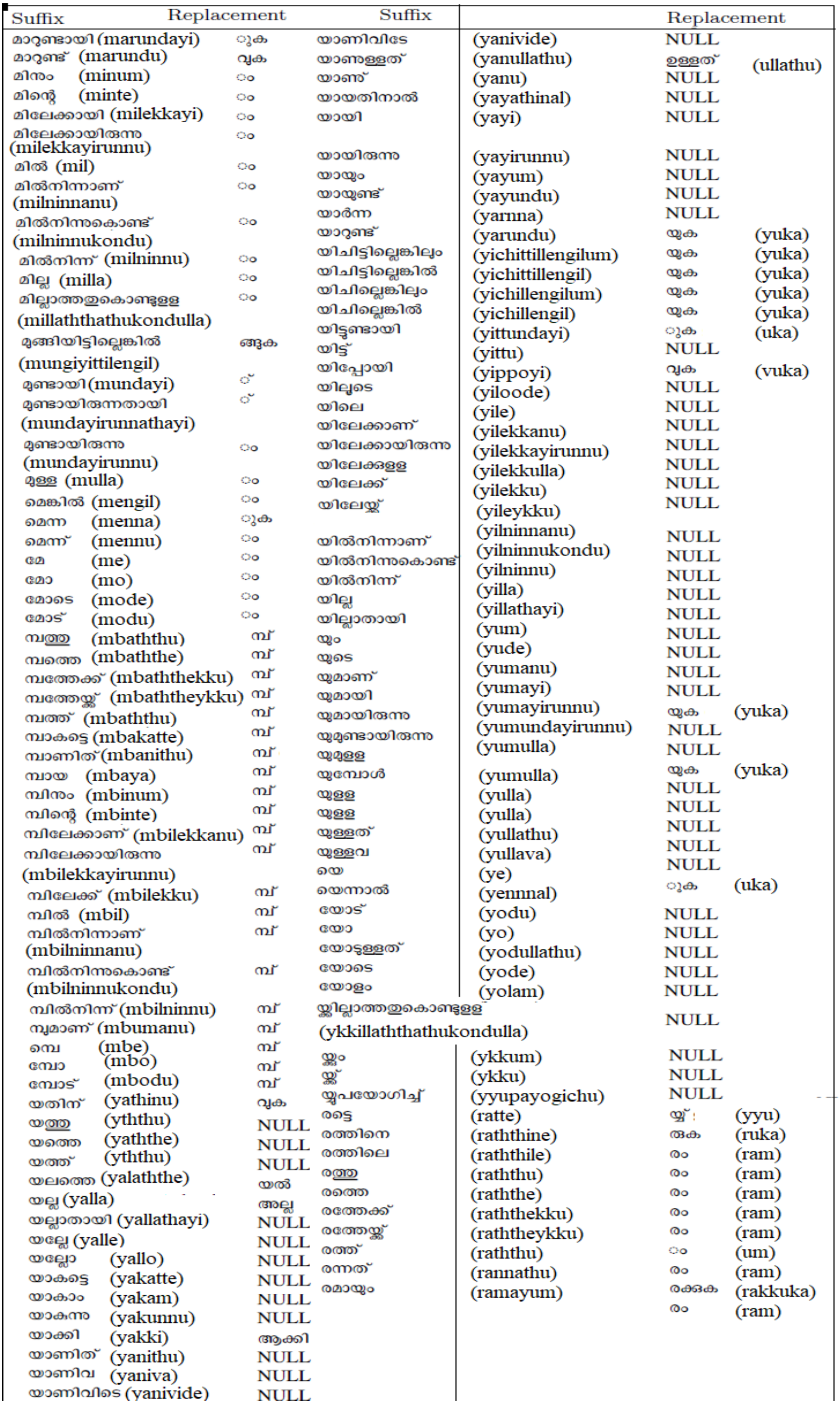
Suffix	Replacement
ഞ്ഞുപ്പോയി (njnjupoyi)	യുക (yuka)
ഞ്ഞുമായി (njnjumayi)	ഞ്ഞ് (njnju)
ഞ്ഞോ (njnjo)	ഞ്ഞ് (njnju)
ഞ്ഞോട് (njnjodu)	ഞ്ഞ് (njnju)
ടത്തു (daththu)	ടം (dum)
ടത്തെ (daththe)	ടം (dum)
ടത്തേക്ക് (daththekku)	ടം (dum)
ടത്തേയ്ക്ക് (daththeykku)	ടം (dum)
ടത്ത് (daththu)	ടം (dum)
ടന്നുപോയി (dannupoyi)	ടക്കുക (dakkuka)
ടാകട്ടെ (dakatte)	ട് (du)
ടാക്കി (dakki)	ട് (du)
ടാക്കിയ (dakkiya)	ട് (du)
ടാണ് (danu)	ട് (du)
ടാണ് (danu)	ട് (du)
ടാനായിരിക്കും (dayirikkum)	ടുക (duka)
ടായ (daya)	ട് (du)
ടായി (dayi)	ട് (du)
ടായിരിക്കും (dayirikkum)	ട് (du)
ടിക്കൊണ്ടിരിക്കുക (dikkondirikkuka)	ടുക (duka)
ടിക്കൊണ്ടിരിക്കുമ്പോൾ (dikkondindirikkumbol)	ടുക (duka)
ടിനടുത്തുള്ള (dinaduththulla)	ട് (du)
ടിനും (dinum)	ട് (du)
ടിനെ (dine)	ട് (du)
ടിന്റെ (dinte)	ട് (du)
ടിലേക്കായി (dilekkayi)	ട് (du)
ടിലേക്കായിരുന്നു (dilekkayirunnu)	ട് (du)
ടിൽ (dil)	ട് (du)
ടിൽനിന്നാണ് (dilninnanu)	ട് (du)
ടിൽനിന്നുകൊണ്ട് (dilninnukondu)	ട് (du)
ടിൽനിന്ന് (dilninnu)	ട് (du)
ടും (dum)	ടുക (duka)
ടുകളും (dukalum)	ട് (du)
ടുകളെ (dukale)	ട് (du)
ടുണ്ടായിരുന്നതായി (dundayirunnathayi)	ട് (du)
ടുത്തെ (duththe)	ടം (dum)
ടുത്തേക്ക് (duththekku)	ടുത്ത് (duththu)
ടുത്തേയ്ക്ക് (duththeykku)	ടുത്ത് (duththu)
ടുന്നുണ്ട് (dunnundu)	ടുക (duka)
ടുമാണ് (dumanu)	ട് (du)
ടുമായി (dumayi)	ട് (du)
ടുമായിരുന്നു (dumayirunnu)	ട് (du)
ടുമുണ്ടായിരുന്ന (dumayirunna)	ടുക (duka)
ടുമുള്ള (dumulla)	ട് (du)
ടുള്ളത് (dullathu)	ട് (du)
ടോ (do)	ട് (du)
ടോട് (dodu)	ട് (du)
ടോളം (dolam)	ട് (du)
ട്ടതിന് (ttathinu)	ടുക (duka)
ട്ടത് (ttathu)	ടുക (duka)
ട്ടപ്പോഴും (ttappozhum)	ക്കുക (kkuka)
ട്ടിനെ (ttine)	ട്ട് (ttu)
ട്ടിയതിന് (ttiyathinu)	ട്ടുക (ttuka)
ട്ടിയിട്ടില്ലെങ്കിൽ (ttiyittillengil)	ട്ടുക (ttuka)
ട്ടിയില്ലെങ്കിൽ (ttiyillengil)	ട്ടുക (ttuka)
ട്ടിലൂടെ (ttiloode)	ട്ട് (ttu)
ട്ടിലെ (ttile)	ട് (du)
ട്ടിലേക്കാണ് (ttilekkanu)	ട്ട് (ttu)
ട്ടിലേക്കുള്ളത് (ttilekkullathu)	ട് (du)
ട്ടിലേക്ക് (ttilekku)	ട്ട് (ttu)
ട്ടുകളും (ttukalum)	ട്ട് (ttu)
ട്ടുണ്ടായിരുന്ന (ttundayirunna)	ട്ട് (ttu)
ട്ടുണ്ടായിരുന്നതായി (ttundayirunnathayi)	ട്ട് (ttu)
ട്ടുമായി (ttumayi)	ട്ട് (ttu)
ട്ടുമുണ്ടായിരുന്ന (ttumundayirunna)	ട്ട് (ttu)
ട്ടുമുണ്ടായിരുന്നു (ttumundayirunnu)	ട്ട് (ttu)
ട്ടുമുള്ള (ttumulla)	ട്ട് (ttu)
ട്ടോട് (ttodu)	ട്ട് (ttu)
ഡാണിത് (danithu)	ഡ് (ddu)
ണതിന് (nathinu)	ഴുക (zhuka)
ണല്ല (nalla)	ൺ (n)
ണായി (nayi)	ൺ (n)
ണുകളും (nukalum)	ൺ (n)
ണുകളുമായി (nukalumayi)	ൺ (n)
ണുകളെ (nukale)	ൺ (n)
ണുപോയി (nupoyi)	ഴുക (zhuka)
ണ്ടാകട്ടെ (ndakatte)	ണ്ട് (ndu)
ണ്ടാകാവുകയെന്നാൽ (ndavukayennal)	ണ്ട് (ndu)
ണ്ടാകാവുന്ന (ndakavunna)	ണ്ട് (ndu)
ണ്ടാക്കിയ (ndakkiya)	ണ്ട് (ndu)
ണ്ടാക്കിയെന്നാൽ (ndakkiyennal)	ണ്ട് (ndu)
ണ്ടായ (ndaya)	ണ്ട് (ndu)
ണ്ടാവുക (ndavuka)	ണ്ട് (ndu)
ണ്ടാവുന്ന (ndavunna)	ണ്ട് (ndu)
ണ്ണല്ല (nnalla)	ൺ (n)
ണ്ണിനെ (nnine)	ണ്ണ് (nnu)
ണ്ണിന്റെ (nninte)	ണ്ണ് (nnu)
ണ്ണിലെ (nnile)	ണ്ണ് (nnu)
ണ്ണിൽ (nnil)	ണ്ണ് (nnu)
ണ്ണുണ്ടായിരുന്ന (nnundayirunna)	ണ്ണ് (nnu)
ണ്ണുണ്ടായിരുന്നതായി (nnundayirunnathayi)	ണ്ണ് (nnu)
ണ്ണുണ്ടായിരുന്നു (nnundayirunnu)	ണ്ണ് (nnu)
ണ്ണുമുണ്ടായിരുന്ന (nnumundayirunna)	ണ്ണ് (nnu)
ണ്ണുമുണ്ടായിരുന്നു (nnumundayirunnu)	ണ്ണ് (nnu)
ണ്ണുമുള്ള (nnumulla)	ണ്ണ് (nnu)
ണ്ണോ (nno)	ണ്ണ് (nnu)
ണ്ണോട് (nnodu)	ണ്ണ് (nnu)
തരും (tharum)	തരുക (tharuka)
തല്ല (thalla)	ത് (thu)
താകട്ടെ (thakatte)	ത് (thu)
താണ് (thanu)	ത് (thu)
താണ് (thanu)	ത് (thu)
തായിരിക്കും (tnayirikkum)	ത് (thuka)
തായും (thayum)	ത് (thu)
താറുണ്ട് (tharundu)	തുക (thu)
തിനും (thinum)	ത് (thu)
തിലെ (thile)	ത് (thu)
തിലേക്കായി (thilekkayi)	ത് (thu)
തിലേക്കായിരുന്നു (thilekkayirunnu)	ത് (thu)
തിൽ (thil)	ത് (thu)
തിൽനിന്നാണ് (thilninnanu)	ത് (thu)
തിൽനിന്നുകൊണ്ട് (thilninnukondu)	ത് (thu)
തിൽനിന്ന് (thilninnu)	ത് (thu)
തും (thum)	തുക (thuka)
തുണ്ടായിരുന്ന (thundayirunna)	ത് (thu)
തുണ്ടായിരുന്നതായി (thundayirunnathayi)	ത് (thu)
തുണ്ടായിരുന്നു (thundayirunnu)	ത് (thu)
തുമാണ് (thumanu)	ത് (thu)
തുമായിരുന്നു (thumayirunnu)	തുക (thuka)
തുമുണ്ടായിരുന്ന (thumundayirunna)	ത് (thu)
തുമുണ്ടായിരുന്നു (thumundayirunnu)	ത് (thu)
തുമുള്ള (thumulla)	ത് (thu)
തെങ്ങില്ല (thengilla)	തെങ്ങ് (thengu)
തെയും (theyum)	തെ (the)
തേയും (theayum)	തേ (thea)
തോ (tho)	ത് (thu)
തോളം (tholam)	ത് (thu)
ത്താകട്ടെ (ththakatte)	ത്ത് (ththu)
ത്താറുണ്ട് (ththarundu)	ത്തുക (ththuka)
ത്തിനടുത്തുള്ള (ththinaduththulla)	◌ം (um)
ത്തിനും (ththinum)	◌ം (um)
ത്തിന് (ththinu)	◌ം (um)
ത്തിന്റെ (ththinte)	◌ം (um)
ത്തിലായി (ththilayi)	◌ം (um)

FIGURE A.3: Suffix-replacement rules starting with 'da' - 'tha'

Suffix		Replacement	
ത്തിലുമുണ്ട്	(ththilumundu)	◌ം	(um)
ത്തിലൂടെ	(ththiloode)	◌ം	(um)
ത്തിലേക്കായി	(ththilekkayi)	◌ം	(um)
ത്തിലേക്കായിരുന്നു	(ththilekkayirunnu)	◌ം	(um)
ത്തിലേക്കുള്ള	(ththilekkulla)	◌ം	(um)
ത്തിലേക്കുള്ളത്	(ththilekkullathu)	◌ം	(um)
ത്തിലേക്ക്	(ththilekku)	◌ം	(um)
ത്തിലേയ്ക്കു്	(ththileykku)	◌ം	(um)
ത്തിൽ	(ththil)	◌ം	(um)
ത്തു	(ththu)	ക്കുക	(kkuka)
ത്തുമാണ്	(ththumanu)	ത്ത്	(ththu)
ത്തെ	(ththe)	◌ം	(um)
ത്തേക്കായിരിക്കും	(ththekkayirikkum)	◌ം	(um)
ത്തേക്കുള്ള	(ththekkulla)	◌ം	(um)
ത്തോടെ	(ththode)	◌ം	(um)
ത്തോട്	(ththodu)	◌്	(u)
ത്തോടുകൂടി	(ththodukoodi)	◌ം	(um)
ത്തോട്	(ththodu)	◌ം	(um)
ത്തോളം	(ththolam)	ത്ത്	(ththu)
നല്ല	(nalla)	ൻ	(n)
നല്ലാ	(nallaa)	ൻ	(n)
നല്ലാത്ത	(nallaththa)	ൻ	(n)
നാകട്ടെ	(nakatte)	ൻ	(n)
നാക്കി	(nakki)	ൻ	(n)
നാണിത്	(nanithu)	ൻ	(n)
നാണ്	(nanu)	ൻ	(n)
നാണ്	(nanu)	ൻ	(n)
നായി	(nayi)	ൻ	(n)
നായിരിക്കും	(nayirikkum)	ൻ	(n)
നായും	(nayum)	ൻ	(n)
നാലാണ്	(nalanu)	ൻ	(n)
നിനക്കായ്	(ninakkay)	നീ	(nee)
നിന്ന	(ninna)	നിൽക്കുക	(nilkkuka)
നിലൂടെ	(niloode)	ൻ	(n)
നിലെ	(nile)	ൻ	(n)
നിലേക്കാണ്	(nilekkanu)	ൻ	(n)
നിലേക്കുള്ള	(nilekkulla)	ൻ	(n)
നിലേക്ക്	(nilekku)	ൻ	(n)
നിൽ	(nil)	ൻ	(n)
നിൽനിന്നാണ്	(nilninnanu)	ൻ	(n)
നിൽനിന്നുകൊണ്ട്	(nilninnukondu)	ൻ	(n)
നിൽനിന്ന്	(nilninnu)	ൻ	(n)
നില്ല	(nilla)	ൻ	(n)
നില്ലാത്തത്	(nillaththathu)	◌ുക	(uka)
നും	(num)	ൻ	(n)
നുടെ	(nude)	ൻ	(n)
നുടേ	(nude)	ൻ	(n)
നുണ്ടായിരുന്ന	(nundayirunna)	ൻ	(n)
നുണ്ടായിരുന്നതായി	(nundayirunnathayi)	ൻ	
നുണ്ടായിരുന്നു	(nundayirunnu)	ൻ	(n)
നുമായിരുന്നു	(numayirunnu)	ൻ	(n)
നുമുണ്ടായിരുന്ന	(numundayirunna)	ൻ	(n)
നുമുണ്ടായിരുന്നു	(numundayirunnu)	ൻ	(n)
നുമുള്ള	(numulla)	ൻ	(n)
നുള്ളത്	(nullathu)	ൻ	(n)
നെ	(ne)	ൻ	(n)
നോ	(no)	ൻ	(n)
നോടുകൂടി	(nodukoodi)	ൻ	(n)
നോടുള്ളത്	(nodullathu)	ൻ	(n)
നോടു്	(nodu)	ൻ	(n)
നോട്	(nodu)	ൻ	(n)
നോളം	(nolam)	ൻ	(n)
ൻഡാകട്ടെ	(ndakatte)	ൻഡ്	(ndu)
ൻഡിനും	(ndinum)	ൻഡ്	(ndu)

Suffix		Replacement	
ൻഡിലേക്കായി	(ndilekkayi)	ൻഡ്	(ndu)
ൻഡിലേക്കായിരുന്നു	(ndilekkayirunnu)	ൻഡ്	(ndu)
ൻഡിൽ	(ndil)	ൻഡ്	(ndu)
ൻഡിൽനിന്നാണ്	(ndilninnanu)	ൻഡ്	(ndu)
ൻഡിൽനിന്നുകൊണ്ട്	(ndilninnukondu)	ൻഡ്	(ndu)
ൻഡിൽനിന്ന്	(ndilninnu)	ൻഡ്	(ndu)
ൻഡോളം	(ndolam)	ൻഡ്	(ndu)
ന്നാണുണ്ടാവുക	(nnanundavuka)	ന്ന്	(nnu)
ന്നാണുള്ളത്	(nnanullathu)	ന്ന്	(nnu)
ന്നിന്റെ	(nninte)	ന്ന്	(nnu)
ന്നുപോയി	(nnupoyi)	രുക	(ruka)
ന്നുള്ളൂ	(nnullu)	ക	(ka)
ന്റെ	(nte)	ൻ	(n)
പലതും	(palathum)	പലത്	(palathu)
പഴുതും	(pazhuthum)	പഴുത്	(pazhuthu)
പാർക്ക്	(parkku)	പാർക്ക്	(park)
പ്പല്ല	(ppalla)	പ്പ്	(ppu)
പ്പാകട്ടെ	(ppakatte)	പ്പ്	(ppu)
പ്പാക്കിയ	(ppakkiya)	പ്പ്	(ppu)
പ്പാണിത്	(ppanithu)	പ്പ്	(ppu)
പ്പാണുണ്ടാവുക	(ppanundavuka)	പ്പ്	(ppu)
പ്പാണുള്ളത്	(ppanullathu)	പ്പ്	(ppu)
പ്പിനും	(ppinum)	പ്പ്	(ppu)
പ്പിനെ	(ppine)	പ്പ്	(ppu)
പ്പിലൂടെ	(ppiloode)	പ്പ്	(ppu)
പ്പിലെ	(ppile)	പ്പ്	(ppu)
പ്പിലേക്കാണ്	(ppilekkanu)	പ്പ്	(ppu)
പ്പിലേക്കായി	(ppilekkayi)	പ്പ്	(ppu)
പ്പിലേക്കായിരുന്നു	(ppilekkayirunnu)	പ്പ്	(ppu)
പ്പിലേക്ക്	(ppilekku)	പ്പ്	(ppu)
പ്പിൽ	(ppil)	പ്പ്	(ppu)
പ്പിൽനിന്നാണ്	(ppilninnanu)	പ്പ്	(ppu)
പ്പിൽനിന്നുകൊണ്ട്	(ppilninnukondu)	പ്പ്	(ppu)
പ്പിൽനിന്ന്	(ppılnınnu)	പ്പ്	(ppu)
പ്പില്ല	(ppilla)	പ്പ്	(ppu)
പ്പും	(ppum)	പ്പ്	(ppu)
പ്പുണ്ടായിരുന്നു	(ppundayirunnu)	പ്പ്	(ppu)
പ്പുമാണ്	(ppumanu)	പ്പ്	(ppu)
പ്പുള്ള	(ppulla)	പ്പ്	(ppu)
പ്പെ	(ppe)	പ്പ്	(ppu)
പ്പോ	(ppo)	പ്പ്	(ppu)
പ്പോടെ	(ppode)	പ്പ്	(ppu)
പ്പോട്	(ppodu)	പ്പ്	(ppu)
പ്പോളം	(ppolam)	പ്പ്	(ppu)
ഫാണിത്	(phanithu)	ഫ്	(phu)
മകൾ	(makal)	മകൾ	(makal)
മല്ല	(malla)	◌ം	(um)
മല്ലാത്ത	(mallaththa)	◌ം	(um)
മാകട്ടെ	(makatte)	◌ം	(um)
മാകാം	(makam)	◌ം	(um)
മാക്കി	(makki)	◌ം	(um)
മാക്കിയ	(makkiya)	◌ം	(um)
മാടി	(madi)	മാടുക	(maduka)
മാണിത്	(manithu)	◌ം	(um)
മാണിവിടെ	(manivide)	◌ം	(um)
മാണ്	(manu)	◌ം	(um)
മാണ്	(manu)	◌ം	(um)
മായ	(maya)	◌ം	(um)
മായി	(mayi)	◌ം	(um)
മായിരിക്കും	(mayirikkum)	◌ുക	(uka)
മായിരുന്നു	(mayirunnu)	◌ം	(um)
മായുണ്ട്	(mayundu)	◌ം	(um)
മാർക്കടിയിൽ	(markadiyi)	NULL	
മാർക്ക്	(mark)	മാർക്ക്	(mark)

FIGURE A.4: Suffix-replacement rules starting with 'na' - 'ma'



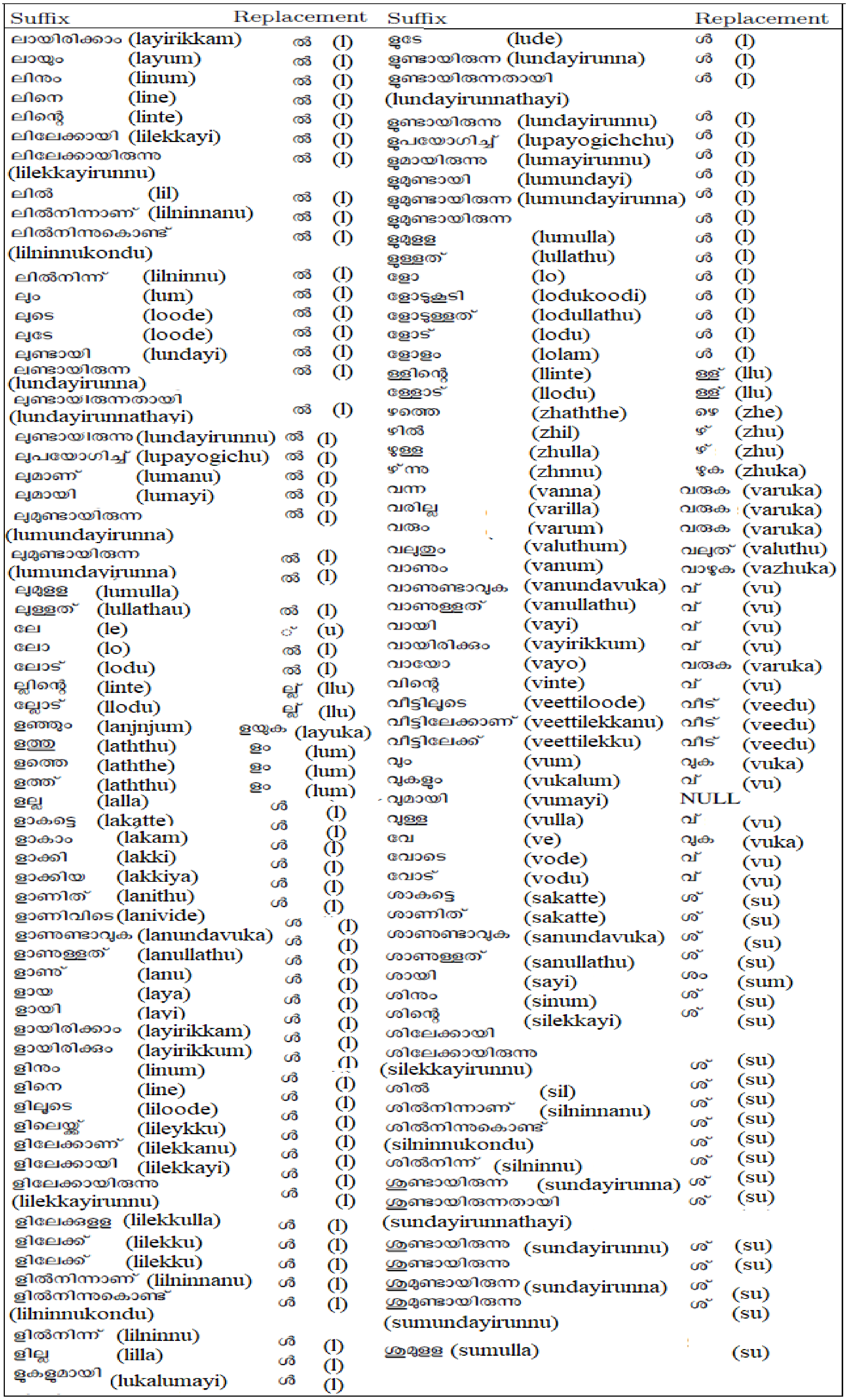
Suffix	Replacement
മാറുണ്ടായി (marundayi)	◌ുക
മാറുണ്ട് (marundu)	വുക
മിനും (minum)	◌ം
മിന്റെ (minte)	◌ം
മിലേക്കായി (milekkayi)	◌ം
മിലേക്കായിരുന്നു (milekkayirunnu)	◌ം
മിൽ (mil)	◌ം
മിൽനിന്നാണ് (milninnanu)	◌ം
മിൽനിന്നുകൊണ്ട് (milninnukondu)	◌ം
മിൽനിന്ന് (milninnu)	◌ം
മില്ല (milla)	◌ം
മില്ലാത്തതുകൊണ്ടുള്ള (millaththathukondulla)	◌ം
മുങ്ങിയിട്ടില്ലെങ്കിൽ (mungiyittilengil)	ങ്ങുക
മുണ്ടായി (mundayi)	◌്
മുണ്ടായിരുന്നതായി (mundayirunnathayi)	◌്
മുണ്ടായിരുന്നു (mundayirunnu)	◌ം
മുള്ള (mulla)	◌ം
മെങ്കിൽ (mengil)	◌ം
മെന്ന (menna)	◌ുക
മെന്ന് (mennu)	◌ം
മേ (me)	◌ം
മോ (mo)	◌ം
മോടെ (mode)	◌ം
മോട് (modu)	◌ം
മ്പത്തു (mbaththu)	മ്പ്
മ്പത്തെ (mbaththe)	മ്പ്
മ്പത്തേക്ക് (mbaththekku)	മ്പ്
മ്പത്തേയ്ക്ക് (mbaththeykku)	മ്പ്
മ്പത്ത് (mbaththu)	മ്പ്
മ്പാകട്ടെ (mbakatte)	മ്പ്
മ്പാണിത് (mbanithu)	മ്പ്
മ്പായ (mbaya)	മ്പ്
മ്പിനും (mbinum)	മ്പ്
മ്പിന്റെ (mbinte)	മ്പ്
മ്പിലേക്കാണ് (mbilekkanu)	മ്പ്
മ്പിലേക്കായിരുന്നു (mbilekkayirunnu)	മ്പ്
മ്പിലേക്ക് (mbilekku)	മ്പ്
മ്പിൽ (mbil)	മ്പ്
മ്പിൽനിന്നാണ് (mbilninnanu)	മ്പ്
മ്പിൽനിന്നുകൊണ്ട് (mbilninnukondu)	മ്പ്
മ്പിൽനിന്ന് (mbilninnu)	മ്പ്
മ്പുമാണ് (mbumanu)	മ്പ്
മ്പെ (mbe)	മ്പ്
മ്പോ (mbo)	മ്പ്
മ്പോട് (mbodu)	മ്പ്
യതിന് (yathinu)	വുക
യത്തു (yththu)	NULL
യത്തെ (yaththe)	NULL
യത്ത് (yththu)	NULL
യലത്തെ (yalaththe)	യൽ
യല്ല (yalla)	അല്ല
യല്ലാതായി (yallathayi)	NULL
യല്ലേ (yalle)	NULL
യല്ലോ (yallo)	NULL
യാകട്ടെ (yakatte)	NULL
യാകാം (yakam)	NULL
യാകുന്നു (yakunnu)	NULL
യാക്കി (yakki)	ആക്കി
യാണിത് (yanithu)	NULL
യാണിവ (yaniva)	NULL
യാണിവിടെ (yanivide)	NULL

Suffix		Replacement	
യാണിവിടെ	(yanivide)	NULL	
യാണുള്ളത്	(yanullathu)	ഉള്ളത്	(ullathu)
യാണ്	(yanu)	NULL	
യായതിനാൽ	(yayathinal)	NULL	
യായി	(yayi)	NULL	
യായിരുന്നു	(yayirunnu)	NULL	
യായും	(yayum)	NULL	
യായുണ്ട്	(yayundu)	NULL	
യാർന്ന	(yarnna)	NULL	
യാറുണ്ട്	(yarundu)	യുക	(yuka)
യിചിട്ടില്ലെങ്കിലും	(yichittillengilum)	യുക	(yuka)
യിചിട്ടില്ലെങ്കിൽ	(yichittillengil)	യുക	(yuka)
യിചില്ലെങ്കിലും	(yichillengilum)	യുക	(yuka)
യിചില്ലെങ്കിൽ	(yichillengil)	യുക	(yuka)
യിട്ടുണ്ടായി	(yittundayi)	◌ുക	(uka)
യിട്ട്	(yittu)	NULL	
യിപ്പോയി	(yippoyi)	വുക	(vuka)
യിലൂടെ	(yiloode)	NULL	
യിലെ	(yile)	NULL	
യിലേക്കാണ്	(yilekkanu)	NULL	
യിലേക്കായിരുന്നു	(yilekkayirunnu)	NULL	
യിലേക്കുള്ള	(yilekkulla)	NULL	
യിലേക്ക്	(yilekku)	NULL	
യിലേയ്ക്ക്	(yileykku)	NULL	
യിൽനിന്നാണ്	(yilninnanu)	NULL	
യിൽനിന്നുകൊണ്ട്	(yilninnukondu)	NULL	
യിൽനിന്ന്	(yilninnu)	NULL	
യില്ല	(yilla)	NULL	
യില്ലാതായി	(yillathayi)	NULL	
യും	(yum)	NULL	
യുടെ	(yude)	NULL	
യുമാണ്	(yumanu)	NULL	
യുമായി	(yumayi)	NULL	
യുമായിരുന്നു	(yumayirunnu)	യുക	(yuka)
യുമുണ്ടായിരുന്നു	(yumundayirunnu)	NULL	
യുമുള്ള	(yumulla)	NULL	
യുമ്പോൾ	(yumulla)	യുക	(yuka)
യുള്ള	(yulla)	NULL	
യുള്ള	(yulla)	NULL	
യുള്ളത്	(yullathu)	NULL	
യുള്ളവ	(yullava)	NULL	
യെ	(ye)	NULL	
യെന്നാൽ	(yennal)	◌ുക	(uka)
യോട്	(yodu)	NULL	
യോ	(yo)	NULL	
യോടുള്ളത്	(yodullathu)	NULL	
യോടെ	(yode)	NULL	
യോളം	(yolam)	NULL	
യ്ക്കില്ലാത്തതുകൊണ്ടുള്ള (ykkillaththathukondulla)		NULL	
യ്ക്കും	(ykkum)	NULL	
യ്ക്ക്	(ykku)	NULL	
യ്യുപയോഗിച്ച്	(yyupayogichu)	NULL	
രട്ടെ	(ratte)	യ്യ്	(yyu)
രത്തിനെ	(raththine)	രുക	(ruka)
രത്തിലെ	(raththile)	രം	(ram)
രത്തു	(raththu)	രം	(ram)
രത്തെ	(raththe)	രം	(ram)
രത്തേക്ക്	(raththekku)	രം	(ram)
രത്തേയ്ക്ക്	(raththeykku)	രം	(ram)
രത്ത്	(raththu)	◌ം	(um)
രന്നത്	(rannathu)	രം	(ram)
രമായും	(ramayum)	രക്കുക	(rakkuka)
		രം	(ram)

FIGURE A.5: Suffix-replacement rules starting with 'mba' - 'ra'

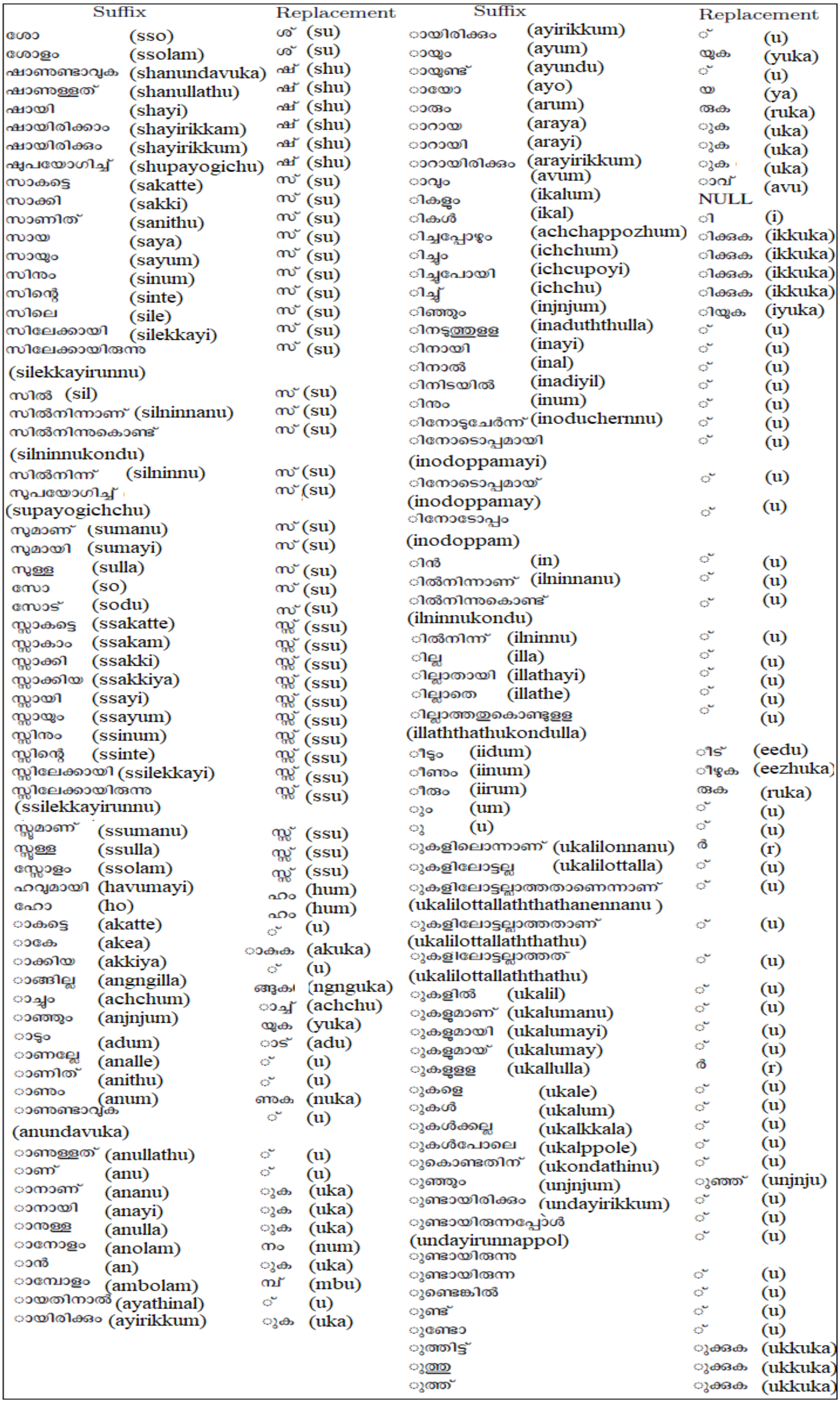
Suffix	Replacement	Suffix	Replacement
രല്ല (ralla)	ർ (r)	റാകാം (rakaam)	ുക (uka)
രവായി (ravayi)	രുക (ruka)	റാക്കി (rakki)	റ് (u)
രവുമായി (ravumayi)	രും (rum)	റാക്കിയ (ralkkiya)	റ് (u)
രാകട്ടെ (rakatte)	ർ (r)	റാണിത് (ranithu)	ർ (r)
രാക്കി (rakki)	ർ (r)	റാണുണ്ടാവുക (ranundavuka)	ർ (r)
രാടി (radi)	രാടുക (raduka)	റാണുള്ളത് (ranullathu)	ർ (r)
രാണിത് (ranithu)	ർ (r)	റാണുള്ളത്	ർ (r)
രാണുണ്ടാവുക (ranundavuka)	ർ (r)	റായിരിക്കും (rayirikkum)	ർ (r)
രാണുള്ളത് (ranullathu)	ർ (r)	റായിരുന്നു (rayirunnu)	ുക (uka)
രാണ് (ranu)	ർ (r)	റാവുക (ravuka)	ുക (uka)
രാനായി (ranayi)	രുക (ruka)	റാവുകയെന്നാൽ (ravukayennal)	ുക (uka)
രായ (raya)	ർ (r)	റാവുന്ന (ravunna)	ുക (uka)
രായതിനാൽ (rayathinal)	രുക (ruka)	റിക്കൊണ്ടിരിക്കുക (rikondirikkuka)	റുക (ruka)
രായി (rayi)	ർ (r)	റിക്കൊണ്ടിരിക്കുമ്പോൾ (rikondirikkumbol)	റുക (ruka)
രായിരിക്കും (rayirikkum)	ർ (r)	റിനടുത്തുള്ള (rinaduththulla)	റ് (ru)
രാരും (rarum)	ർ (r)	റിനും (rinum)	ർ (r)
രാൽ (ral)	ർ (r)	റിനെ (rine)	ർ (r)
രിലൂടെ (riloode)	ർ (r)	റിന്റെ (rinte)	റ് (ru)
രിലെ (rile)	ർ (r)	റിപ്പോയി (rippoyi)	ുക (uka)
രിലേക്കാണ് (rilekkanu)	ർ (r)	റിയിട്ടില്ലെങ്കിൽ (riyittillengil)	റുക (ruka)
രിലേക്കുള്ള (rilekkulla)	ർ (r)	റിയില്ലെങ്കിൽ (riyillengil)	റുക (ruka)
രിലേക്ക് (rilekku)	ർ (r)	റിലൂടെ (riloode)	ർ (r)
രിലേയ്ക്ക് (rileykku)	ർ (r)	റിലെ (rile)	ർ (r)
രിൽ (ril)	ർ (r)	റിലേക്കാണ് (rilekkanu)	ർ (r)
രിൽനിന്നാണ് (rilninnanu)	ർ (r)	റിലേക്കായി (rilekkayi)	ർ (r)
രിൽനിന്നുകൊണ്ട് (rilninnukondu)	ർ (r)	റിലേക്കായിരുന്നു (rilekkayirunnu)	ർ (r)
രിൽനിന്ന് (rilninnu)	ർ (r)	റിലേക്കുള്ള (rilekkulla)	റ് (ru)
രില്ല (rilla)	ര് (ru)	റിലേക്കുള്ളത് (rilekkullathu)	ർ (r)
രും (rum)	ർ (r)	റിലേക്ക് (rilekku)	ർ (r)
രുടെ (rude)	ർ (r)	റിൽ (ril)	റ് (ru)
രുടെ (rude)	ർ (r)	റിൽനിന്നാണ് (rilninnanu)	ർ (r)
രുണ്ടായിരുന്ന (rundayirunna)	ർ (r)	റിൽനിന്നുകൊണ്ട് (rilninnukondu)	ർ (r)
രുണ്ടായിരുന്നതായി (rundayirunnathayi)	ർ (r)	റിൽനിന്ന് (rilninnu)	ർ (r)
രുണ്ടായിരുന്നു (rundayirunnu)	ർ (r)	റുകളും (rukalum)	ർ (r)
രുണ്ടായിരുന്നു (rundayirunnu)	ർ (r)	റുകളുമായി (rikalumayi)	ർ (r)
രുമായിരുന്നു (rumayirunnu)	രുക (ruka)	റുണ്ടായിരുന്ന (rundayirunna)	ർ (r)
രുമായിരുന്ന (rumayirunna)	ർ (r)	റുണ്ടായിരുന്നതായി (rundayirunnathayi)	ർ (r)
രുമുണ്ടായിരുന്നു (rumundayirunnu)	ർ (r)	റുണ്ടായിരുന്നു (rundayirunnu)	ർ (r)
രുമുള്ള (rumulla)	ർ (r)	റുപയോഗിച്ച് (rupayogichu)	ർ (r)
രുള്ളത് (rullathu)	ർ (r)	റുമുണ്ടായിരുന്ന (rumundayirunna)	ർ (r)
രു് (ru)	ർ (r)	റുമുണ്ടായിരുന്നു (rumundayirunnu)	ർ (r)
രെ (re)	ർ (r)	റുമുള്ള (rumulla)	ർ (u)
രേ (rea)	ർ (r)	റുള്ള (rulla)	◌് (rootu)
രോ (ro)	ർ (r)	റൂട്ടിലെ (roottile)	റൂട്ട് (r)
രോടുകൂടി (rodukoodi)	ർ (r)	റെ (re)	ർ (r)
രോടുള്ളത് (rodullathu)	ർ (r)	റോ (ro)	ർ (r)
രോട്ട് (rodu)	ർ (r)	റോടെ (rode)	ർ (r)
രോടെ (rode)	ര് (r)	റോട് (rodu)	റ് (ru)
രോട് (rodu)	ർ (r)	റോളം (rolam)	ർ (r)
രോളം (rolam)	ർ (r)	ലത്തു (laththu)	ൽ (l)
ര് (r)	ർ (r)	ലത്തെ (laththe)	ലെ (le)
ർക്കായ് (rkkayi)	ർ (r)	ലത്ത് (laththu)	ൽ (l)
ർക്ക് (rkku)	ർ (r)	ലല്ല (lalla)	ൽ (l)
ർന്നതിന്നു (rnaathinnu)	രുക (ruka)	ലാകട്ടെ (lakatte)	ൽ (l)
ർന്നപ്പോൾ (rnnappol)	രുക (ruka)	ലാകാം (lakaam)	ൽ (l)
ർന്നില്ല (rnnilla)	രുക (ruka)	ലാക്കി (lakki)	ൽ (l)
ർന്നു (rnnu)	രുക (ruka)	ലാക്കിയ (lakkiya)	ൽ (l)
ർന്നുപോയി (rnnupoyi)	രുക (ruka)	ലാടി (ladi)	ലാടുക (laduka)
ർന്നുപോയ (rnnupoya)	രുക (ruka)	ലാണിത് (lanithu)	ൽ (l)
റത്തേക്ക് (raththekku)	റ് (r)	ലാണുണ്ടാവുക (lanundavuka)	ൽ (l)
റത്തേയ്ക്ക് (raththeykku)	റ് (r)	ലാണുള്ളത് (lanullathu)	ൽ (l)
റന്നത് (rannathu)	റക്കുക (rakkuka)	ലാണ് (lanu)	ൽ (l)
റന്നു (rannu)	റക്കുക (rakkuka)	ലായ (laya)	ൽ (l)
റവായ (ravaya)	റവ് (ravu)	ലായി (layi)	ൽ (l)
റാകട്ടെ (rakatte)	ർ (r)	ലായിരിക്കും (layirikkum)	ൽ (l)

FIGURE A.6: Suffix-replacement rules starting with 'ra' - 'la'



Suffix	Replacement	Suffix	Replacement
ലായിരിക്കാം (layirikkam)	ൽ (l)	ളൂടേ (lude)	ൾ (l)
ലായും (layum)	ൽ (l)	ളുണ്ടായിരുന്ന (lundayirunna)	ൾ (l)
ലിനും (linum)	ൽ (l)	ളുണ്ടായിരുന്നതായി (lundayirunnathayi)	ൾ (l)
ലിനെ (line)	ൽ (l)	ളുണ്ടായിരുന്നു (lundayirunnu)	ൾ (l)
ലിന്റെ (linte)	ൽ (l)	ളുപയോഗിച്ച് (lupayogichchu)	ൾ (l)
ലിലേക്കായി (lilekkayi)	ൽ (l)	ളുമായിരുന്നു (lumayirunnu)	ൾ (l)
ലിലേക്കായിരുന്നു (lilekkayirunnu)	ൽ (l)	ളുമുണ്ടായി (lumundayi)	ൾ (l)
ലിൽ (lil)	ൽ (l)	ളുമുണ്ടായിരുന്ന (lumundayirunna)	ൾ (l)
ലിൽനിന്നാണ് (lilninnanu)	ൽ (l)	ളുമുണ്ടായിരുന്ന	ൾ (l)
ലിൽനിന്നുകൊണ്ട് (lilninnukondu)	ൽ (l)	ളുമുള്ള (lumulla)	ൾ (l)
ലിൽനിന്ന് (lilninnu)	ൽ (l)	ളുള്ളത് (lullathu)	ൾ (l)
ലും (lum)	ൽ (l)	ളോ (lo)	ൾ (l)
ലൂടെ (loode)	ൽ (l)	ളോടുകൂടി (lodukoodi)	ൾ (l)
ലൂടേ (loode)	ൽ (l)	ളോടുള്ളത് (lodullathu)	ൾ (l)
ലുണ്ടായി (lundayi)	ൽ (l)	ളോട് (lodu)	ൾ (l)
ലുണ്ടായിരുന്ന (lundayirunna)	ൽ (l)	ളോളം (lolam)	ൾ (l)
ലുണ്ടായിരുന്നതായി (lundayirunnathayi)	ൽ (l)	ള്ളിന്റെ (llinte)	ള്ള് (llu)
ലുണ്ടായിരുന്നു (lundayirunnu)	ൽ (l)	ള്ളോട് (llodu)	ള്ള് (llu)
ലുപയോഗിച്ച് (lupayogichu)	ൽ (l)	ഴത്തെ (zhaththe)	ഴെ (zhe)
ലുമാണ് (lumanu)	ൽ (l)	ഴിൽ (zhil)	ഴ് (zhu)
ലുമായി (lumayi)	ൽ (l)	ഴുള്ള (zhulla)	ഴ് (zhu)
ലുമുണ്ടായിരുന്ന (lumundayirunna)	ൽ (l)	ഴ്ന്നു (zhnnu)	ഴുക (zhuka)
ലുമുണ്ടായിരുന്ന (lumundayirunna)	ൽ (l)	വന്ന (vanna)	വരുക (varuka)
ലുമുള്ള (lumulla)	ൽ (l)	വരില്ല (varilla)	വരുക (varuka)
ലുള്ളത് (lullathau)	ൽ (l)	വരും (varum)	വരുക (varuka)
ലേ (le)	് (u)	വലുതും (valuthum)	വലുത് (valuthu)
ലോ (lo)	ൽ (l)	വാണും (vanum)	വാഴുക (vazhuka)
ലോട് (lodu)	ൽ (l)	വാണുണ്ടാവുക (vanundavuka)	വ് (vu)
ല്ലിന്റെ (linte)	ല്ല് (llu)	വാണുള്ളത് (vanullathu)	വ് (vu)
ല്ലോട് (llodu)	ല്ല് (llu)	വായി (vayi)	വ് (vu)
ളഞ്ഞും (lanjnjum)	ളയുക (layuka)	വായിരിക്കും (vayirikkum)	വ് (vu)
ളത്തു (laththu)	ളും (lum)	വായോ (vayo)	വരുക (varuka)
ളത്തെ (laththe)	ളും (lum)	വിന്റെ (vinte)	വ് (vu)
ളത്ത് (laththu)	ളും (lum)	വീട്ടിലൂടെ (veettiloode)	വീട് (veedu)
ളല്ല (lalla)	ൾ (l)	വീട്ടിലേക്കാണ് (veettilekkanu)	വീട് (veedu)
ളാകട്ടെ (lakatte)	ൾ (l)	വീട്ടിലേക്ക് (veettilekku)	വീട് (veedu)
ളാകാം (lakam)	ൾ (l)	വും (vum)	വുക (vuka)
ളാക്കി (lakki)	ൾ (l)	വുകളും (vukalum)	വ് (vu)
ളാക്കിയ (lakkiya)	ൾ (l)	വുമായി (vumayi)	NULL
ളാണിത് (lanithu)	ൾ (l)	വുള്ള (vulla)	വ് (vu)
ളാണിവിടെ (lanivide)	ൾ (l)	വേ (ve)	വുക (vuka)
ളാണുണ്ടാവുക (lanundavuka)	ൾ (l)	വോടെ (vode)	വ് (vu)
ളാണുള്ളത് (lanullathu)	ൾ (l)	വോട് (vodu)	വ് (vu)
ളാണ് (lanu)	ൾ (l)	ശാകട്ടെ (sakatte)	ശ് (su)
ളായ (laya)	ൾ (l)	ശാണിത് (sakatte)	ശ് (su)
ളായി (layi)	ൾ (l)	ശാണുണ്ടാവുക (sanundavuka)	ശ് (su)
ളായിരിക്കാം (layirikkam)	ൾ (l)	ശാണുള്ളത് (sanullathu)	ശ് (su)
ളായിരിക്കും (layirikkum)	ൾ (l)	ശായി (sayi)	ശും (sum)
ളിനും (linum)	ൾ (l)	ശിനും (sinum)	ശ് (su)
ളിനെ (line)	ൾ (l)	ശിന്റെ (silekkayi)	ശ് (su)
ളിലൂടെ (liloode)	ൾ (l)	ശിലേക്കായി	ശ് (su)
ളിലെയ്ക്ക് (lileykku)	ൾ (l)	ശിലേക്കായിരുന്നു (silekkayirunnu)	ശ് (su)
ളിലേക്കാണ് (lilekkanu)	ൾ (l)	ശിൽ (sil)	ശ് (su)
ളിലേക്കായി (lilekkayi)	ൾ (l)	ശിൽനിന്നാണ് (silninnanu)	ശ് (su)
ളിലേക്കായിരുന്നു (lilekkayirunnu)	ൾ (l)	ശിൽനിന്നുകൊണ്ട് (silninnukondu)	ശ് (su)
ളിലേക്കുള്ള (lilekkulla)	ൾ (l)	ശിൽനിന്ന് (silninnu)	ശ് (su)
ളിലേക്ക് (lilekku)	ൾ (l)	ശുണ്ടായിരുന്ന (sundayirunna)	ശ് (su)
ളിലേക്ക് (lilekku)	ൾ (l)	ശുണ്ടായിരുന്നതായി (sundayirunnathayi)	ശ് (su)
ളിൽനിന്നാണ് (lilninnanu)	ൾ (l)	ശുണ്ടായിരുന്നു (sundayirunnu)	ശ് (su)
ളിൽനിന്നുകൊണ്ട് (lilninnukondu)	ൾ (l)	ശുണ്ടായിരുന്നു	ശ് (su)
ളിൽനിന്ന് (lilninnu)	ൾ (l)	ശുമുണ്ടായിരുന്ന (sundayirunna)	ശ് (su)
ളില്ല (lilla)	ൾ (l)	ശുമുണ്ടായിരുന്നു (sumundayirunnu)	ശ് (su)
ളുകളുമായി (lukalumayi)	ൾ (l)	ശുമുള്ള (sumulla)	(su)

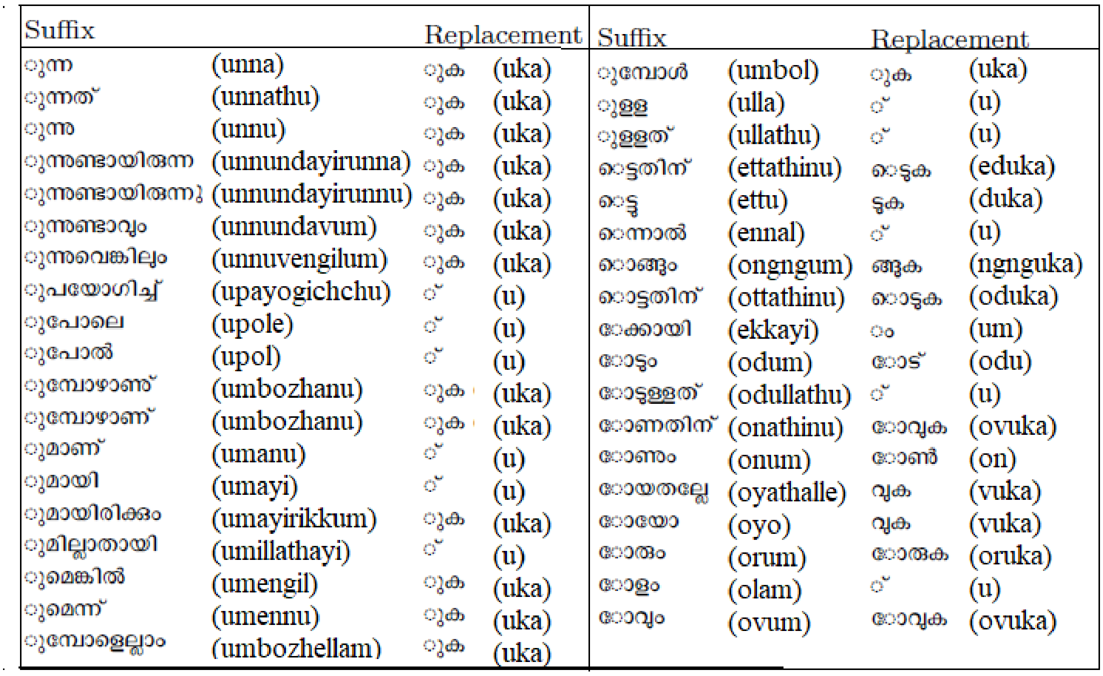
FIGURE A.7: Suffix-replacement rules starting with 'la' - 'sa'



Suffix		Replacement	
ശോ	(sso)	ശ്	(su)
ശോളം	(ssolam)	ശ്	(su)
ഷാണുണ്ടാവുക	(shanundavuka)	ഷ്	(shu)
ഷാണുള്ളത്	(shanullathu)	ഷ്	(shu)
ഷായി	(shayi)	ഷ്	(shu)
ഷായിരിക്കാം	(shayirikkam)	ഷ്	(shu)
ഷായിരിക്കും	(shayirikkum)	ഷ്	(shu)
ഷുപയോഗിച്ച്	(shupayogichu)	ഷ്	(shu)
സാകട്ടെ	(sakatte)	സ്	(su)
സാക്കി	(sakki)	സ്	(su)
സാണിത്	(sanithu)	സ്	(su)
സായ	(saya)	സ്	(su)
സായും	(sayum)	സ്	(su)
സിനും	(sinum)	സ്	(su)
സിന്റെ	(sinte)	സ്	(su)
സിലെ	(sile)	സ്	(su)
സിലേക്കായി	(silekkayi)	സ്	(su)
സിലേക്കായിരുന്നു	(silekkayirunnu)	സ്	(su)
സിൽ	(sil)	സ്	(su)
സിൽനിന്നാണ്	(silninnanu)	സ്	(su)
സിൽനിന്നുകൊണ്ട്	(silninnukondu)	സ്	(su)
സിൽനിന്ന്	(silninnu)	സ്	(su)
സുപയോഗിച്ച്	(supayogichchu)	സ്	(su)
സുമാണ്	(sumanu)	സ്	(su)
സുമായി	(sumayi)	സ്	(su)
സുള്ള	(sulla)	സ്	(su)
സോ	(so)	സ്	(su)
സോട്	(sodu)	സ്	(su)
സ്സാകട്ടെ	(ssakatte)	സ്സ്	(ssu)
സ്സാകാം	(ssakam)	സ്സ്	(ssu)
സ്സാക്കി	(ssakki)	സ്സ്	(ssu)
സ്സാക്കിയ	(ssakkiya)	സ്സ്	(ssu)
സ്സായി	(ssayi)	സ്സ്	(ssu)
സ്സായും	(ssayum)	സ്സ്	(ssu)
സ്സിനും	(ssinum)	സ്സ്	(ssu)
സ്സിന്റെ	(ssinte)	സ്സ്	(ssu)
സ്സിലേക്കായി	(ssilekkayi)	സ്സ്	(ssu)
സ്സിലേക്കായിരുന്നു	(ssilekkayirunnu)	സ്സ്	(ssu)
സ്സുമാണ്	(ssumanu)	സ്സ്	(ssu)
സ്സുള്ള	(ssulla)	സ്സ്	(ssu)
സ്സോളം	(ssolam)	സ്സ്	(ssu)
ഹവുമായി	(havumayi)	ഹം	(hum)
ഹോ	(ho)	ഹം	(hum)
◌ാകട്ടെ	(akatte)	◌്	(u)
◌ാകേ	(akea)	◌ാകുക	(akuka)
◌ാക്കിയ	(akkiya)	◌്	(u)
◌ാങ്ങില്ല	(angngilla)	ങ്ങുക	(ngnguka)
◌ാച്ചും	(achchum)	◌ാച്ച്	(achchu)
◌ാഞ്ഞും	(anjnjum)	യുക	(yuka)
◌ാടും	(adum)	◌ാട്	(adu)
◌ാണല്ലേ	(analle)	◌്	(u)
◌ാണിത്	(anithu)	◌്	(u)
◌ാണും	(anum)	ണുക	(nuka)
◌ാണുണ്ടാവുക	(anundavuka)	◌്	(u)
◌ാണുള്ളത്	(anullathu)	◌്	(u)
◌ാണ്	(anu)	◌്	(u)
◌ാനാണ്	(ananu)	◌ുക	(uka)
◌ാനായി	(anayi)	◌ുക	(uka)
◌ാനുള്ള	(anulla)	◌ുക	(uka)
◌ാനോളം	(anolam)	നം	(num)
◌ാൻ	(an)	◌ുക	(uka)
◌ാമ്പോളം	(ambolam)	മ്പ്	(mbu)
◌ായതിനാൽ	(ayathinal)	◌്	(u)
◌ായിരിക്കും	(ayirikkum)	◌ുക	(uka)

Suffix		Replacement	
◌ായിരിക്കും	(ayirikkum)	◌്	(u)
◌ായും	(ayum)	യുക	(yuka)
◌ായുണ്ട്	(ayundu)	◌്	(u)
◌ായോ	(ayo)	യ	(ya)
◌ാരും	(arum)	രുക	(ruka)
◌ാറായ	(araya)	◌ുക	(uka)
◌ാറായി	(arayi)	◌ുക	(uka)
◌ാറായിരിക്കും	(arayirikkum)	◌ുക	(uka)
◌ാവും	(avum)	◌ാവ്	(avu)
◌ികളും	(ikalum)	NULL	
◌ികൾ	(ikal)	◌ി	(i)
◌ിച്ചപ്പോഴും	(achchappozhum)	◌ിക്കുക	(ikkuka)
◌ിച്ചും	(ichchum)	◌ിക്കുക	(ikkuka)
◌ിച്ചുപോയി	(ichcupoyi)	◌ിക്കുക	(ikkuka)
◌ിച്ച്	(ichchu)	◌ിക്കുക	(ikkuka)
◌ിഞ്ഞും	(injnjum)	◌ിയുക	(iyuka)
◌ിനടുത്തുള്ള	(inaduththulla)	◌്	(u)
◌ിനായി	(inayi)	◌്	(u)
◌ിനാൽ	(inal)	◌്	(u)
◌ിനിടയിൽ	(inadiyil)	◌്	(u)
◌ിനും	(inum)	◌്	(u)
◌ിനോടുചേർന്ന്	(inoduchernnu)	◌്	(u)
◌ിനോടൊപ്പമായി	(inodoppamayi)	◌്	(u)
◌ിനോടൊപ്പമായ്	(inodoppamay)	◌്	(u)
◌ിനോടോപ്പം	(inodoppam)	◌്	(u)
◌ിൻ	(in)	◌്	(u)
◌ിൽനിന്നാണ്	(ilninnanu)	◌്	(u)
◌ിൽനിന്നുകൊണ്ട്	(ilninnukondu)	◌്	(u)
◌ിൽനിന്ന്	(ilninnu)	◌്	(u)
◌ില്ല	(illa)	◌്	(u)
◌ില്ലാതായി	(illathayi)	◌്	(u)
◌ില്ലാതെ	(illathe)	◌്	(u)
◌ില്ലാത്തതുകൊണ്ടുള്ള	(illaththathukondulla)	◌്	(u)
◌ീടും	(iidum)	◌ീട്	(eedu)
◌ീണും	(iinum)	◌ീഴുക	(eezhuka)
◌ീരും	(iirum)	രുക	(ruka)
◌ും	(um)	◌്	(u)
◌ു	(u)	◌്	(u)
◌ുകളിലൊന്നാണ്	(ukalilonnanu)	ർ	(r)
◌ുകളിലോട്ടല്ല	(ukalilottalla)	◌്	(u)
◌ുകളിലോട്ടല്ലാത്തതാണെന്നാണ്	(ukalilottallaththathanennanu)	◌്	(u)
◌ുകളിലോട്ടല്ലാത്തതാണ്	(ukalilottallaththathu)	◌്	(u)
◌ുകളിലോട്ടല്ലാത്തത്	(ukalilottallaththathu)	◌്	(u)
◌ുകളിൽ	(ukalil)	◌്	(u)
◌ുകളുമാണ്	(ukalumanu)	◌്	(u)
◌ുകളുമായി	(ukalumayi)	◌്	(u)
◌ുകളുമായ്	(ukalumay)	◌്	(u)
◌ുകളുള്ള	(ukallulla)	ർ	(r)
◌ുകളെ	(ukale)	◌്	(u)
◌ുകൾ	(ukalum)	◌്	(u)
◌ുകൾക്കല്ല	(ukalkkala)	◌്	(u)
◌ുകൾപോലെ	(ukalppole)	◌്	(u)
◌ുകൊണ്ടതിന്	(ukondathinu)	◌്	(u)
◌ുഞ്ഞും	(unjnjum)	◌ുഞ്ഞ്	(unjnju)
◌ുണ്ടായിരിക്കും	(undayirikkum)	◌്	(u)
◌ുണ്ടായിരുന്നപ്പോൾ	(undayirunnappol)	◌്	(u)
◌ുണ്ടായിരുന്നു		◌്	(u)
◌ുണ്ടായിരുന്ന		◌്	(u)
◌ുണ്ടെങ്കിൽ		◌്	(u)
◌ുണ്ട്		◌്	(u)
◌ുണ്ടോ		◌്	(u)
◌ുത്തിട്ട്		◌ുക്കുക	(ukkuka)
◌ുത്തു		◌ുക്കുക	(ukkuka)
◌ുത്ത്		◌ുക്കുക	(ukkuka)

Figure A.8: Suffix-replacement rules starting with 'sha' to 'sa'



Suffix		Replacement		Suffix		Replacement	
◌ുന്ന	(unna)	◌ുക	(uka)	◌ുമ്പോൾ	(umbol)	◌ുക	(uka)
◌ുന്നത്	(unnathu)	◌ുക	(uka)	◌ുള്ള	(ulla)	◌്	(u)
◌ുന്നു	(unnu)	◌ുക	(uka)	◌ുള്ളത്	(ullathu)	◌്	(u)
◌ുന്നുണ്ടായിരുന്ന	(unnundayirunna)	◌ുക	(uka)	◌െട്ടതിന്	(ettathinu)	◌െടുക	(eduka)
◌ുന്നുണ്ടായിരുന്നു	(unnundayirunnu)	◌ുക	(uka)	◌െട്ടു	(ettu)	ടുക	(duka)
◌ുന്നുണ്ടാവും	(unnundavum)	◌ുക	(uka)	◌െന്നാൽ	(ennal)	◌്	(u)
◌ുന്നുവെങ്കിലും	(unnuvengilum)	◌ുക	(uka)	◌ൊങ്ങും	(ongngum)	ങ്ങുക	(ngnguka)
◌ുപയോഗിച്ച്	(upayogichchu)	◌്	(u)	◌ൊട്ടതിന്	(ottathinu)	◌ൊടുക	(oduka)
◌ുപോലെ	(upole)	◌്	(u)	◌േക്കായി	(ekkayi)	◌ം	(um)
◌ുപോൽ	(upol)	◌്	(u)	◌ോടും	(odum)	◌ോട്	(odu)
◌ുമ്പോഴാണ്	(umbozhanu)	◌ുക	(uka)	◌ോടുള്ളത്	(odullathu)	◌്	(u)
◌ുമ്പോഴാണ്	(umbozhanu)	◌ുക	(uka)	◌ോണതിന്	(onathinu)	◌ോവുക	(ovuka)
◌ുമാണ്	(umanu)	◌്	(u)	◌ോണും	(onum)	◌ോൺ	(on)
◌ുമായി	(umayi)	◌്	(u)	◌ോയതല്ലേ	(oyathalle)	വുക	(vuka)
◌ുമായിരിക്കും	(umayirikkum)	◌ുക	(uka)	◌ോയോ	(oyo)	വുക	(vuka)
◌ുമില്ലാതായി	(umillathayi)	◌്	(u)	◌ോരും	(orum)	◌ോരുക	(oruka)
◌ുമെങ്കിൽ	(umengil)	◌ുക	(uka)	◌ോളം	(olam)	◌്	(u)
◌ുമെന്ന്	(umennu)	◌ുക	(uka)	◌ോവും	(ovum)	◌ോവുക	(ovuka)
◌ുമ്പോളെല്ലാം	(umbozhellam)	◌ുക	(uka)				

FIGURE A.9: Suffix-replacement rules starting with 'u' to 'oo'

References

[1] https://machinelearningmastery.com/gentle-introduction-text-summarization, Nov. 2017.

[2] https://hackernoon.com/how-to-run-text-summarization-with-tensorflow-d4472587602d., 2017

[3] Mehdi Allahyari, Seyedamin Pouriyeh et. al, *Text Summarization Techniques: A Brief Survey* , arXiv, pp. 1-9, July 2017.

[4] M. M. Kamal, K. Z. Sultana, *A Comprehensive Tool for Text Categorization and Text Summarization in Bioinformatics*, 15th International Conference on Computer and Information Technology (ICCIT), pp. 592-597, 2012

[5] H. N. Fejer, N. Omar, *Automatic Arabic Text Summarization Using Clustering and Key phrase Extraction*, International Conference on Information Technology and Multimedia (ICIMU), pp. 293-298, Nov. 2014.

[6] N. K. Nagwani, *Summarizing large text collection using topic modeling and clustering based on MapReduce framework*, Journal of Big Data, pp. 1-18, 2015.

[7] Nedunchelian, Ramanujam and Manivannan Kaliappan, *An Automatic Multidocument Summarization Approach based on Naive Bayesian Classifier Using Timestamp strategy*, The Scientific World Journal, pp. 1-10, 2016.

[8] K. Kaikhah, *Automatic Text Summarization with Neural Networks*, In Proceedings of second international Conference on intelligent systems, IEEE, pp. 40-44, Texas, USA, June 2004.

[9] Dharmendra Hingu, Deep Shah , Sandeep S. Udmale, *Automatic Text Summarization of Wikipedia Articles*, International Conference on

Communication, Information and Computing Technology(ICCICT), pp. 1-4, 2015.

[10] Alexander M. Rush, Sumit Chopra, Jason Weston, *A Neural Attention Model for Sentence Summarization*, In proceedings of Conference on Empirical methods in Natural language processing, pp. 379-389, Sep. 2015.

[11] Ramesh N, Bowen Z, Cicero dos S , alar Gulehre ,Bing Xian, , *Abstractive Text Summarization using Sequence- to-sequence RNNs and Beyond*, In proceedings of 20th SIGNLL conference on Computational Natural Language Learning, pp. 280-290, Aug. 2016.

[12] Abigail See, Peter J. L, Christopher D. Manning, Get To The Point: *Summarization with Pointer-Generator Networks*, arXiv:1704.04368v2, pp. 1-20, April 2017.

[13] F.E. Gunawan, A. V. J, B. Soewito, *An Automatic Text Summarization using Text features and Singular value Decomposition for Popular Articles in Indonesian Language*, International Seminar on Intelligent Technology and its applications, pp. 27-32, 2015.

[14] Erwin Yudi H, Fahri F, Khafiizh H, Ika N.D, Azhari, *Automatic Text Summarization Using LDA for Document Clustering*, International Journal of Advances in Intelligent Informatics, Vol. 1(3), pp. 132-139, 2015.

[15] Kuldeep Kaur, Anantdeep Kaur, *Email Summarization using LDA, International Journal of Advance Research in Science and Engineering*, Vol. 6(9), pp. 9-18, Sept. 2017.

[16] Oguejioforchibueze, *NLP for Topic Modeling and Summarization of Legal Documents*, https://towardsdatascience.com/nlp-for-topic-modeling-summarization-of-legal- documents-8c89393b1534, 2018.

[17] Seyedamin Pouriyeh, Mehdi Allahyari, Krys Kochut, Gong Cheng, and Hamid Reza Arabnia, *ES- LDA: Entity Summarization Using Knowledge based Topic Modeling*, Proceedings of the 8th International Joint Conference on Natural Language Processing, 2017, pp. 316-325.

[18] N. Alami, M. Meknassi, S. A. Ouatik, N. Ennahnahi, *Arabic Text Summarization based on Graph Theory*, IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), pp. 1-8, 2015.

[19] Xu Han, Tao Lv, Zhirui Hu, Xinyan Wang, and Cong Wang, *Text Summarization Using FrameNet- Based Semantic Graph Model*, Scientific programming, Hindawi Publishing corp, pp. 1-11, 2016.

[20] Yazan Alaya AL-Khassawneh1, Naomie Salim and Mutasem Jarrah, *Improving Triangle-Graph Based Text Summarization using Hybrid Similarity Function*, Indian Journal of Science and Technology, Vol 10(8), DOI: 10.17485/ijst/2017/v10i8/108907, pp. 1-15, February 2017.

[21] K. Sarkar, K. Saraf, A. Ghosh, *Improving Graph Based Multidocument Text Summarization Using an Enhanced Sentence Similarity Measure*, IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), pp. 359-365, 2015.

[22] H. Rashidghalam, M. Taherkhani, F. Mahmoudi, *Text summarization using concept graph and BabelNet knowledge base*, Artificial Intelligence and Robotics (IRANOPEN), pp. 115-119, 2016.

[23] Saif alZahir, Qandeel Fatima, Martin Cenek, *New graph-based text summarization method*, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 396-401, 2015.

[24] Yazan A. Jaradat, Ahmad T. Al-Taani, *Hybrid-based Arabic Single-Document Text Summarization Approach Using Genetic Algorithm*, 7th International Conference on Information and Communication Systems (ICICS), pp. 85-91, 2016.

[25] R. S. Baraka and S. N. Breem, *Automatic Arabic Text Summarization for Large Scale Multiple Documents Using Genetic Algorithm and MapReduce*, 2017 Palestinian International Conference on Information and Communication Technology (PICICT), Gaza, Palestine, pp. 40-45, 2017.

[26] Mehdi Jafari, Jing Wang, Yongrui Qin, Mehdi Gheisari, Amir Shahab Shahabi, Xiaohui Tao, *Automatic Text Summarization using Fuzzy Inference*, 22nd International Conference on Automation and Computing (ICAC), 2016.

[27] Muhammad Azhari, Yogan J.K, *Improving text summarization using neuro-fuzzy approach*, Journal of Information and telecommunication , Vol. 1(4), pp. 367-379, 2017.

[28] S. Santhana Megala, Dr. A. Kavitha , Dr. A. Marimuthu, *Text Summarization System using Fuzzy Logic and Conditional Random Field Algorithm*, International journal of Innovative Research in Computer Science and Engineering, Vol. 1(5), pp. 1-8, 2015.

[29] Rajina Kabeer, Sumam Mary Idicula, *Text Summarization for Malayalam Documents an Experience*, International Conference on Data science and Engineering, pp. 145-150, 2014.

[30] Renjith S.R, Sony P, *An automatic Text Summarization for Malayalam using Sentence Extraction*, International Journal of advanced computational engineering and networking, Vol. 3(8), pp. 46-50, Aug 2015.

[31] Krishna Prasad P, Sooryanarayanan A, Ajeesh R, *Malayalam Text Summarizer, An extractive approach*, International conference on Next generation Intelligent systems(ICNGIS), pp. 1-4, 2016.

[32] Manish Gupta, Naresh Kumar Garg, *Text Summarization of Hindi documents using Rule based approach*, International Conference on Micro-electronics and Telecommunication Engineering, pp. 366- 370, 2016.

[33] Syed Sabin M, S . Hariharan, *A summarizer for Tamil language using centroid approach*, International Journal of Information Retrieval Research, Vol. 6(1), pp. 1-15, 2016.

[34] Jikitha Sheth, Bankim Patel, *Saaraansh: Gujarathi Text Summarization System*, IRACST, Vol. 7(3), May-June, 2017.

[35] Prejisha K, Dr Reghuraj P.C, *Stemmer for Malayalam Using Three Pass Algorithm*, International Conference on Control Communication and Computing (ICCC), pp. 149-152, 2013.

[36] Maral Rusiol, Volkmar Frinken, Dimosthenis Karatzas, Andrew D. Bagdanov, Josep Llads, *Multimodal page classification in administrative document image streams*, International Journal of Document Analysis and Recognition, IJDAR 17, pp. 331-341, 2014.

[37] Hongxi Wei, Guanglai Gao, *A keyword retrieval system for historical Mongolian document images*, International Journal of Document Analysis and Recognition, IJDAR, 17, pp. 33-45, 2014.

[38] Arwa Alqudsi, Nazlia Omar, Khalid Shaker, *Arabic Machine Translation:A survey*, Artificial Intelligence Review, 42, pp. 549-572, 2014.

[39] Elaheh Asghari, MohammadReza KeyvanPour, *XML document clustering: techniques and challenges*, Artificial Intelligence Review, 43: pp. 417-436, 2015.

[40] Nagwani N. K, *Summarizing large text collection using topic modeling and clustering based on MapReduce framework*, Nagwani Journal of Big Data, 2:6, 2015.

[41] Gheith A, Abandah, Alex Graves, Balkees Al-Shagoor, *Automatic diacritization of Arabic text using recurrent neural networks*, International Journal on Document Analysis and Recognition, 18, pp. 183-197, 2015.

[42] Osama Mohamed Elrajubi, *An Improved Arabic Light Stemmer*, Third International Conference on Research and Innovation in Information Systems, pp. 33-38, 2013.

[43] Ayu Purwarianti, *A Non Deterministic Indonesian Stemmer*, International Conference on Electrical Engineering and Informatics, G3-4, 2011.

[44] Cristian Moral, Angelica de Antonio, Ricardo Imbert, *A survey of stemming algorithms in Information Retrieval*, Information Research, Vol. 19(1), 2014.

[45] Walid Cherif, Abdellah Madani, Mohamed Kissi, *Integrated effective rules to Improve Arabic Text Stemming*, pp. 1077-1081, 2014.

[46] Vishal Gupta, Gurpreet Singh Lehal, *A survey of Common Stemming Techniques and Existing Stemmers for Indian Languages*, Journal of Emerging Technologies in Web Intelligence, Vol. 5(2), pp. 157-161, 2013.

[47] Jikitsha Sheth, Bankim Patel, *Dhiya : A Stemmer for morphological level analysis of Gujarati language*, International Conference on Issues and Challenges in Intelligent Computing Techniques, pp. 151-154, 2014.

[48] Redowan Mahmud Md, Mahbuba Afrin, Md. Abdur Razzaque Ellis Miller, Joel Iwashige, *A rule based Bengali stemmer*, ICACCI, pp. 2750-2756, 2014.

[49] Vasudevan N, Pushpak Bhattacharya, *Little by little Semi Supervised Stemming through Stem Set Minimization*, IJCNLP, pp. 774-780, 2013.

[50] Vasudevan N, Pushpak Bhattacharyya, *Optimal Stem Identification in Presence of Suffix List, Computational linguistic and Intelligent Text Processing*, 13th International Conference CICLing, pp. 92-103, 2012.

[51] Rohit Kansal, Vishal Goyal and Lehal G.S, *Rule Based Urdu Stemmer*, Proceedings of COLING, pp. 267-276, 2012.

[52] Upendra Mishra et. al, *MAULIK: An Effective Stemmer for Hindi Language*, International Journal on Computer Science and Engineering, Vol. 4(5), pp. 711-717, 2012.

[53] Gupta V, Lehal G. S, *Punjabi Language Stemmer for Nouns and Proper Names*, Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing, pp. 35-39, 2011.

[54] Satyendr Singh and Tanveer J. Siddiqui, *Evaluating Effect of Context Window Size, Stemming and Stop Word Removal on Hindi Word Sense Disambiguation*, IEEE, pp. 1-5, 2012.

[55] Das S, Mitra. P, *A Rule-based Approach of Stemming for Inflectional and Derivational Words in Bengali*, In Proceeding of the 2011 IEEE Students Technology Symposium, IIT Kharagpur, pp. 134-136, 2011.

[56] Vishal Gupta, *Hindi Rule Based Stemmer for Nouns*, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4(1), pp. 62-65, 2014.

[57] Kasthuri M, Britto Ramesh Kumar S, *An Improved Rule based Iterative Affix Stripping Stemmer for Tamil Language using K-Mean Clustering*, International Journal of Computer Applications, Vol. 94(13), pp. 36-41, 2014.

[58] Ramachandran V. A, Illango Krishnamurthi, *An Iterative Stemmer for Tamil language*, ACIIDS, Springer, pp. 197-205, 2012.

[59] Vijay Sundar Ram and Sobha Lalitha Devi, *Malayalam Stemmer, Morphological Analysers and Generators*, LDC-IL, Mysore, pp. 105-113, 2010.

[60] Vinod P. M, Jayan V, Bhadran V. K, *Implementation of Malayalam Morphological Analyzer Based on Hybrid Approach*, Proceedings of the Twenty-Fourth Conference on Computational Linguistics and Speech Processing, ROCLING, pp. 307-317, 2012.

[61] Jisha P. Jayan, Rajeev R. R, Dr S. Rajendran, *Morphological Analyser and Morphological Generator for Malayalam-Tamil Machine Translation*, International Journal of Computer Applications, Vol. 13(8), pp. 15-18, 2011.

[62] Prejitha U, Sreejith C, Reghu Raj P. C, *Lalitha : A light weight Malayalam Stemmer using Suffix Stripping Method*, International Conference on Control Communication and Computing (ICCC), pp. 244-248, 2013 .

[63] Santhosh Thottingal, *www.silpa.org/stemmer*, SILPA project, 2014-2016.

[64] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schtze, *An Introduction to Information retrieval*, Cambridge University Press, April 1, 2009.

[65] Denyong Zhou, Jiayuan Huang, Bernhard Scholkopf, *Learning with Hypergraphs: Clustering, Classification and Embedding*, citeseer, 2007.

[66] Sepideh Seifzadeh, Ahmed K. Farahat, Mohamed S. Kamel, *Short-Text Clustering using Statistical Semantics*, International World Wide Web Conference Committee (IW3C2), WWW 2015 Companion, pp. 805-810, 2015.

[67] Tingting Weia, Yonghe Luc, Huiyou Changb, Qiang Zhoua, Xianyu Baod, *A semantic approach for text clustering using WordNet and lexical chains*, Expert Systems with Applications, Vol. 42(4), pp. 2264-2275, 2015.

[68] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, Hongwei Hao, *Short Text Clustering via Convolutional Neural Networks*, Proceedings of NAACL-HLT, pp. 62-69, 2015.

[69] Seyednaser Nourashrafeddin, Evangelos Milios, Drik V. Arnold, *An ensemble approach for text document clustering using Wikipedia concepts*, Proceedings of the ACM symposium on Document engineering, pp. 107-116, 2014.

[70] Benjamin C. M, Fung Ke Wang, Martin Ester, *Hierarchical Document Clustering Using Frequent Itemsets*, In Proceedings of the Third SIAM International Conference on Data Mining, pp. 1-12, 2003.

[71] Inderjit S. Dhillon, Subramanyam Mallela, Rahul Kumar, *A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification*, Journal of Machine Learning Research, pp. 1265-1287, 2003.

[72] Ilya Karpov, Alexandr Goroslavskiy, *Application of BIRCH to text clustering*, In Proceedings of the 14th All-Russian Conference on Digital Libraries: Advanced Methods and Technologies, Digital CollectionsRCDL-2012, Pereslavl Zalesskii, Russia, pp. 102-105, 2012.

[73] Diego A. Ingaramo, Marcelo L. Errecalde Paolo Rosso, *Density-based clustering of short-text corpora*, Procesamiento del lenguaje Natural, No. 41, pp. 81-87, 2008.

[74] Huaijun Qiu, Edwin R. Hancock, *Graph matching and clustering using spectral partitions*, Science Direct, Pattern Recognition 39, pp. 22-34, 2006.

[75] Debarghya Ghoshdastidar, Ambedkar Dukkipati, *Spectral Clustering Using Multilinear SVD: Analysis, Approximations and Applications*, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2610-2616, 2015.

[76] James P. Fairbanks, Geoffrey D. Sanders, David A. Bader, *Spectral Partitioning with Blends of Eigen vectors*, Computer Science-Numerical Analysis, Cornell University Library, pp. 1-31, 2016.

[77] Inderjit S. Dhillon, *Co-clustering documents and words using Bipartite Spectral Graph Partitioning*, KDD 2001 San Francisco, California, USA, pp. 2701-2709, 2001.

[78] Michael W. Berry, Malu Castellanos, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, Springer, pp. 3-87, 2008.

[79] Asma Khazaal Abdulsahib, Siti Sakira Kamaruddin, *Graph Based Text Representation for Document clustering*, Journal of Theoretical and Applied Information Technology, Vol. 76(1), pp. 1-13, 2015.

[80] M. Shahriar Hossain, Rafal A. Angryk, *GDClust: A Graph-Based Document Clustering Technique*, Seventh IEEE International Conference on Data Mining-Workshops, pp. 1-7, 2007.

[81] Alessandro Lulli, Thibault Debatty, Matteo DellAmico, Pietro Michiardi, Laura Ricci, *Scalable K-NN based text clustering*, IEEE International Conference on Big Data, pp. 1-6, 2015.

[82] S. S. Sonawane, Dr. P. A. Kulkarni, *Graph based Representation and Analysis of Text Document: A Survey of Techniques*, International Journal of Computer Applications (0975-8887) Vol. 96(19), pp. 1-8, 2014.

[83] Vikas Kumar Sihag , Subhash Kumar, *Graph based Text Document Clustering by Detecting Initial Centroids for K-means*, International Journal of Computer Applications (0975 - 8887) Vol. 62(19), pp. 1-4, 2013.

[84] Yanjun Li, Congnan Luo, Soon M. Chung, *A parallel text document clustering algorithm based on neighbors*, Cluster Computing, Springer, Vol. 18(2), pp. 933-948, 2015.

[85] Weizhong Zhao, Huifang Ma, Qing He, *Parallel K-Means Clustering Based on MapReduce*, In: Proceedings of the 2nd International Conference on Cloud and Green Computing, pp. 226-229, 2012.

[86] Aboutabl, A.E, Elsayed M.N, *A novel parallel algorithm for clustering documents based on the hierarchical agglomerative approach*, IJCSIT, Vol. 3(2), pp. 152-163, 2011.

[87] Jace A. Mogill, David J. Haglin, *Toward Parallel Document Clustering*, Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011.

[88] Isabelle Bloch, Alain Bretto, *Mathematical morphology on hypergraphs: Preliminary definitions and results.* In Discrete Geometry for Computer Imagery, pp. 429-440. Springer, 2011.

[89] Isabelle BLOCH, Alain BRETTO, *Mathematical morphology on hypergraphs, application to similarity and positive kernel*, Computer vision and image understanding, Vol. 117(4), pp. 342-354, 2013.

[90] Isabelle Bloch, Alain Bretto, Aur lie Leborgn, *Similarity between hypergraphs based on mathematical morphology*, In Mathematical Morphology and Its Applications to Signal and Image Processing, Springer, pp. 1-12, 2013.

[91] Jean Cousty, Laurent Najman, Fabio Dias, Jean Serra. *Morphological filtering on graphs*, Computer Vision and Image Understanding, Elsevier, pp. 1-20, 2012.

[92] R. Dharmarajan, K. Kannan, *A hypergraph-based algorithm for image restoration from salt and pepper noise*, AEU-International Journal of Electronics and Communications, Vol. 64(12), pp. 1114-1122, 2010.

[93] Krassimir T. Atanassov, *Intuitionistic Fuzzy Sets Past, Present and Future*, In proceedings of EUSFLAT, pp. 12-19, 2003.

[94] John N. Mordeson, Premchand S. N, *Fuzzy graphs and Fuzzy hypergraphs*, Studies in Fuzziness and Soft computing, Physica-Verlag, pp. 135-231, 2014.

[95] Parvathi R, Thilagavathi S and Karunambigai M. G, *Intuitionistic fuzzy hypergraphs, Cybernetics and Information Technologies*, Vol. 9, pp. 46-53, 2009.

[96] Parvathi R, Thilagavathi S, Karunambigai M. G, *Operations on Intuitionistic Fuzzy Hypergraphs*, International Journal of Computer Applications , Vol. 51(5), pp. 46-54, 2012.

[97] Parvathi R, Thilagavathi S, Atanassov K. T, *Isomorphism on Intuitionistic Fuzzy Directed Hypergraphs*, International Journal of Scientific and Research Publications, pp. 278-285, 2013.

[98] Muhammad Akram, Wieslaw A. Dudek, *Intuitionistic fuzzy hypergraphs with applications*, Information Sciences, Vol. 218, pp. 182-193, 2013.

[99] Ejegwa P. A, Akubo A. J, Joshua O. M, *Intuitionistic fuzzy set and its application in career determination via normalized Euclidean distance method*, European Scientific Journal, Vol. 10, pp. 529-536, 2014.

[100] Myithili K. K, Parvathi R, *Transversals of Intuitionistic Fuzzy Directed Hypergraphs*, Notes on IFS, Vol. 21, pp. 66-79, 2015.

[101] Samanta T. K, Mohinta S, *Generalized Strong Intuitionstic Fuzzy Hypergraph*, Mathematica Moravica, Vol. 18, pp. 55-65, 2014.

[102] Dhanya P. M, A. Sreekumar, M. Jathavedan, Ramkumar P.B, *Algebra of Morphological Dilation on Intuitionistic Fuzzy Hypergraphs*, International Journal of Scientific Research in Science, Engineering and Technology, Vol. 4(1), pp. 300-308, 2018.

[103] Giorgio Gallo, Giustino Longo, Stefano Pallottino, Sang Nguyen, *Directed Hypergraphs and Applications, Discrete Applied Mathematics*, Vol. 42, pp. 177-201, 1993.

[104] Bino S, Unnikrishnan A, Kannan B, Ramkumar P. B *Mathematical Morphology on hypergraphs using vertex-hyperedge correspondence*, ISRN Discrete Mathematics, Hindawi publishing corporation, Vol. 2014, pp. 1-6, 2014.

[105] Kannan B, Bino S, Unnikrishnan A, Ramkumar P. B, *Morphological Filtering on Hypergraphs*, Discrete Applied Mathematics, pp. 307-320, 2014.

[106] Dhanya P. M, A. Sreekumar, M. Jathavedan, *Vriksh: A Tree based Malayalam lemmatizer using Suffix Replacement Dictionary*, IJETER, Vol. 6(1), pp. 31-43, 2018.

[107] Dhanya P. M, A. Sreekumar, M. Jathavedan, Ramkumar P.B, *Document Modeling and Clustering using Hypergraphs*, IJAER, Vol. 12(10), pp. 2127-2135, 2017.

[108] Dhanya P. M, A. Sreekumar, M. Jathavedan, Ramkumar P. B, *Metric Induced Morphological Operations on Intuitionistic Fuzzy Hypergraphs*, International Journal of Mathematics and Mathematical Sciences, Hindawi publications, pp. 1-11, June 2018.

List of Publications

1. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *Text Summarization using Morphological filtering of Intuitionistic Fuzzy Hypergraphs*, Journal of Computer Science, Science publications, Vol. 14(6), pp. 837-853, 2018. DOI: 10.3844/jcssp.2018.837.853.

2. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *Metric Induced Morphological Operations on Intuitionistic Fuzzy Hypergraphs*, International Journal of Mathematics and Mathematical Sciences, Hindawi publications, pp. 1-11, June 2018. https://doi.org/10.1155/2018/6045358.

3. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *On Constructing Morphological erosion of Intuitionistic Fuzzy Hypergraphs*, Journal of Analysis, Springer, pp. 1-20, 2018. https://doi.org/10.1007/s41478-018-0096-3.

4. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *Algebra of Morphological Filter on Intuitionistic Fuzzy Hypergraphs with application in Summarization of Documents*, International Journal of Applied Engineering Research, Vol. 13(3), pp. 5-12, 2018.

5. Dhanya P. M, Sreekumar A, Jathavedan M, *Vriksh: A Tree based Malayalam lemmatizer using Suffix Replacement Dictionary*, International Journal of Emerging Technologies in Engineering Research, Vol. 6(1), pp. 31-43, 2018.

6. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *Algebra of Morphological Dilation on Intuitionistic fuzzy hypergraphs*, International Journal of Scientific Research in Science, Engineering and Technology, Vol. 4(1), pp. 300-308, 2018.

7. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *A Survey of Recent Techniques in Automatic Text Summarization*, International Journal of Computer Engineering and Technology, Vol. 9(2), pp. 74-85, 2018.

8. Dhanya P. M, Sreekumar A, Jathavedan M, Ramkumar P. B, *Document Modeling and Clustering using Hypergraphs*, International Journal of Applied Engineering Research, Vol. 12(10), pp. 2127-2135, 2017.

9. Dhanya P. M, Jathavedan M, *A Comparative study of Text Summarization in Indian Languages*, International Journal of Computer Applications, Vol. 75(6), pp. 17-21, 2013.

10. Dhanya P. M, Jathavedan M, *Application of Modified Spectral bisection for Segmenting Malayalam Documents*, 2013 Third International Conference on Advances in Computing and Communications (ICACC), pp. 29-33, 2013.

Alphabetical Index