NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

HIERARCHICAL TEXT TOPIC DETECTION FROM TEXT DATA

CHEN ZIAO
U1420681G

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
AY 2017/2018

NANYANG TECHNOLOGICAL UNIVERSITY

SCE17-0153
HIERARCHICAL TEXT TOPIC DETECTION FROM TEXT DATA

Submitted in Partial Fulfillment of the Requirements
for the Double Degree in Bachelor of Engineering (Computer Science)
and Bachelor of Business of the Nanyang Technological University

by

CHEN ZIAO

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
AY 2017/2018

Abstract

In an age of information boom, people get used to obtain information from online documents. However, useful information is not able to directly be extracted from most online documents due to the unstructured and unorganized nature of text. Therefore, to facilitate the searching and retrieval of information, documents are always labeled with tags, topics or categories. This paper presents with an efficient hierarchical topic detection algorithms which detect topics in a hierarchical manner from a large collection of documents. The algorithm starts with document processing which turns each document into a vector after a series of text preprocessing using some Natural Language Processing techniques. Since vectorized text data is normally very sparse, two dimensionality reduction techniques: *Autoencoder* and *Word2Vec* are applied respectively to reduce the data sparsity. Hierarchical Fuzzy Clustering is then performed on the two sets of text vectors respectively to generate two hierarchical topic trees. After that, two topic trees are evaluated based on topic quality metrics such as association index and duplication index to select the best algorithm. The best algorithm is named as Hierarchical Fuzzy Topic Detection (HFTD). Finally, HFTD is tested on the text data of computer science papers from 1993 to 2017 on *IEEE* to obtain insights of computer science research topic changing trend.

Acknowledgments

This Final Year Project (FYP) was made possible thanks to the support of the my FYP supervisor Assoc Prof Adams Wai Kin Kong and Asst Prof Ke Yiping Kelly, who have provided useful guidance on my project. I would like to express my deepest gratitude to them, for their valuable advice throughout the project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

We are living in an era of information explosion. IBM reported that every day, 2.5 quintillion bytes of data are created and 90% of the data in the world today have been created in the last two years alone [1]. Its a truism that 80 percent of business-relevant information originates in unstructured form, primarily text [2]. Searching and extracting information efficiently from large amount of unstructured text is never easy. Therefore, detecting text topics among large text document collections would be a crucial step in greatly enhancing the understanding of unstructured text before other text mining tasks such as text summarization and information retrieval. The algorithm to detect topics among a large collection of text documents is defined as text topic detection algorithm and topic model acts as the fundamental component of it.

According to Wikipedia, the definition of topic model is type of statistical model for discovering the abstract topics that occur in a collection of documents [3]. Another definition is that topic model is the model that find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection and similarity judgment [4]. Both definitions emphasize that the topic model is a statistical model, which in some extent, explains the reason why that some popular topic models such as Latent Dirichlet Allocation [4] and Hierarchical Latent Class Model [5]), use techniques like Bayesian Inference and Expectation-Maximization to detect topics.

However, it is surprising that there are very few researches on applying clustering in building topic models. Clustering is the process of grouping a set of objects in a way such that objects in the same cluster are more similar to each other than to those in other clusters [6]. Clustering can be categorized into soft and hard clustering where soft clustering allows one object to be clustered into more than one clusters and vice versa for hard clustering. Hence, soft clustering seems to be a perfect match for topic detection algorithm where each cluster can be regarded as a topic. One document can belong to different topics and one topic can include more than one document. This paper would try to understand the reason why currently very few text topic detection algorithms are built based on soft clustering and also to develop a complete hierarchical text topic detection method using clustering analysis as its core component.

1.2 Aims and Objectives

This project aims to study the problem of hierarchical text topic detection from text data, identify the key challenges, and develop effective techniques to solve it. In details, this project should leverage different existing text mining and topic modeling algorithms, improve or extend them into a hierarchical version and in the end, combine them into an integrated solution for hierarchical text topic detection. The main objectives for this project are:

1. Collect and pre-process large-scale text data into proper format to feed into the text topic detection algorithm;

2. Design and implement effective algorithms for hierarchical text topic detection;

3. Select the best algorithm through evaluating and comparing the performance of the different hierarchical text topic algorithms;

4. Use the best algorithm to detect computer science research topics changing trend through the years.

1.3 Scope

The domain for techniques used in building the hierarchical text topic detection algorithm is restricted to machine learning and natural language processing. However, the designed algorithm can be extended and improved by incorporating knowledge from other domains such as semantic network and ontology.

The hierarchical fuzzy topic detection (HFTD) algorithm developed in this project is an algorithm which can be implemented in different programming language. For simplicity reason, this project uses *Python* as the main programming language. There is no consolidated application or library built at the end of the project. However, readers may follow the logic described in later sections to implement this algorithm. A sample output of HFTD using 2017 data is shown in Appendix A.

1.4 Report Organization

This paper is divided into six chapters and an overview of each chapter is as follows:

- Chapter 1 describes background, aims and objectives, and scope for this project;
- Chapter 2 reviews some past literature on fuzzy clustering, text topic detection, text categorization and hierarchical unsupervised learning algorithms;
- Chapter 3 presents the methodology for text data collection, text data pre-processing, vector dimensionality reduction and hierarchical fuzzy clustering topic detection;
- Chapter 4 analyzes the topic tree detected by different algorithms with the help of various clustering metrics and selects the best algorithm;
- Chapter 5 analyzes topic changing trend by applying the best algorithm (HFTD);
- Chapter 6 concludes the paper and discusses future work that can be done on this topic.

Chapter 2

Literature Review

2.1 Text Categorization

Text topic detection and clustering both fall under unsupervised learning area. Although there are very few papers with regards to document clustering, there are many researches done on document classification which falls under supervised learning area. Text categorization is to categorize documents into predefined categories whereas for text clustering, there is no predefined categories. There are also very mature methodologies in hierarchical text categorization such as neural network approach [7] and probabilistic approach [8]. Although categorization is different from clustering, the text data pre-processing step which converts unstructured text data into vector space is similar. For example, tasks such as stop words removal, tokenization, stemming, lemmatization and document vectorization can be shared between text categorization and text topic detection.

2.2 Topic Model

Topic modeling is an essential component in hierarchical text topic detection. The first step of constructing a topic model is always to find out a concise representation of a document. A primitive way to convert unstructured text into structured form is through count vectorization where each document is represented as a vector with each dimension as a word in the vocabulary. The value of the vector is the number of appearance of the word in that document. When a collection of documents is presented, count vectors can be stacked into a count matrix. Taking a step further, the count value in count matrix can be changed to *Term-frequency Inverse-document-frequency(tf-idf)* value

[9]. Term-frequency refers to the number of times a word appearing in a document and inverse-document-frequency refers to the reciprocal of number of documents a word has appeared in.

However, *tf-idf* value provides a relatively small amount of reduction in term of document size and reveals little information of inter-document or intra-document statistical relationship. Another document vectorization called *Latent Semantic Indexing (LSI)* [10] uses singular value decomposition of the *tf-idf* matrix to identify a linear subspace in the space of *tf-idf* features that captures most of the variance in the collection, which is able to achieve significant compression in large collections. For *tf-idf* and *LSI*, they both just extract useful features from documents without consolidating all features based on their similarity. Hence, more advance techniques are required for further feature extraction.

A major step forward from text features to text topics was done by *probabilistic LSI (pLSI)* [11]. It models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as topics. Each word is generated from a single topic and hence, different words in a document can be generated from different topics. In the end, each document can be represented as a probability distribution on a fixed set of topics.

However, *pLSI* is incomplete in some extent because it provides no probabilistic model at document level. This issue is resolved by *Latent Dirichlet Allocation (LDA)* [4] which is one of the most popular topic detection algorithm nowadays. *LDA* is a generative probabilistic and three-level hierarchical Bayesian model where each document is modeled as a finite mixture over an underlying set of topics. Each topic is in turn modeled as an infinite mixture over an underlying set of topic probabilities. It is based on the *nested Chinese restaurant process (nCRP)* [12]. Moreover, there are many applications in using Bayesian Inference model in calculating word distribution [13] [14]. However, *LDA* only can detect a flat topic structure with number of topics being set beforehand. Therefore, it is still very far away from the hierarchical text topic detection method described in this project, which can detect number of topics automatically and build topics in a hierarchical form.

2.3 Hierarchical Clustering

In data mining and statistics, hierarchical clustering is an algorithm of cluster analysis which seeks to build a hierarchy of clusters. There are two popular types of hierarchical clustering: Agglomerative and Divisive Hierarchical Clustering. Agglomerative Hierarchical Clustering is a bottom-up algorithm which starts with individual objects and combine them into clusters one by one until only one cluster remains. The Divisive Hierarchical Clustering is a top-down algorithm which does the contrary of the agglomerative one. Generally speaking, the dendrogram generated by both algorithms resemble a hierarchical topic structure. However, traditional hierarchical clustering falls under hard clustering which means that one object can only belong to one cluster at certain level, which totally does not fit the text topic definition. On the other hand, due to large amount of text documents to be clustered, running traditional hierarchical clustering would be extremely time-consuming and inefficient.

When looking at soft clustering where one object can belong to several clusters simultaneously, *Fuzzy C-means Clustering algorithm (FCM)* [15] never fails to be a popular choice. It is an advanced version of *K-means Clustering* which recursively an algorithm over and over again until centroids converge, which assigns objects to cluster centroids and recomputes centroids based on newly assigned objects. *FCM* improves K-means Clustering by allowing objects being assigned to different clusters. However, *FCM* only provides flat clusters instead of giving out a cluster hierarchy, which means that only one level of clusters would be detected.

To apply *FCM* in a hierarchical manner, *Hierarchical Unsupervised Fuzzy Clustering (HUFC)* [16] algorithm can be used. It implements a new recursive algorithm which has features of hierarchical clustering, while maintaining *FCM* characteristics. The Hierarchical Fuzzy Topic Detection algorithm presented in this paper is mainly based on *HUFC*, with large modifications in model parameters and model logic because *HUFC* initially is designed for normal numeric data in low dimension while text data is always sparse and text data converted to low dimension would incur huge information loss.

Chapter 3

Methodology

3.1 Data Collection

3.1.1 Data Source

As mentioned in the project objectives, this project aims to identify computer science research topic changing trend through the years. Since *IEEE* is one of the world's largest technical professional platforms with numerous papers in different research topics, *IEEE* paper data is chosen as the primary data source of this project. For this project, only data of computer science related papers which were published between 1993 and 2017 are collected. Due to limited storage and physical memory, for each paper, only title, keywords and abstract are crawled by the web crawler.

3.1.2 Crawler

To facilitate text data collection, a *Python* crawler is implemented. It uses *Python* package *Selenium* and application *Chrome driver* as its crawling engine which is able to automate user actions such as mouse clicking, hovering and scrolling. For each year, only around first 2000 papers ranked by descending order of number of citations are crawled. All paper data is from *IEEE Xplore Digital Library* (`http://ieeexplore.ieee.org/Xplore/home.jsp`). The crawling speed is about 1 second per paper, and hence, it takes around 16 hours to collect all text data required for this project. A simple example of crawled data is shown at Table 3.1.

Table 3.1: Crawled Paper Data Examples

Title	Keyword	Abstract
Algorithm-Dependent Generalization Bounds for Multi-Task Learning	Algorithm design and analysis, Stability analysis . . .	Often, tasks are collected for multitask learning (MTL) because they share similar feature structures. Based on this observation, in this . . .
Bi-Level Semantic Representation Analysis for Multimedia Event Detection	Semantics, Detectors, Training, Multimedia communication . . .	Multimedia event detection has been one of the major endeavors in video event analysis. A variety of approaches . . .
Robust Joint Graph Sparse Coding for Unsupervised Spectral Feature Selection with Region Proposal Networks	Manifolds, Computational modeling, Learning system . . .	In this paper, we propose a new unsupervised spectral feature selection model by embedding a graph regularizer into the framework of joint sparse regression for . . .

3.2 Data Preprocessing

To feed the text data crawled from IEEE website, it needs to be cleaned and processed into a model readable format which in the end is in the form of a matrix. Each row of the output matrix is a high dimension vector representing each document. All data preprocessing tasks are executed using *Python* with the help of data analytics and natural language processing packages such as *Pandas*, *sklearn* and *nltk*. Figure 3.1 is a pipeline chart of the whole data preprocessing process which includes tokenization, stop words removal, lemmatization and *Tf-idf* vectorization.

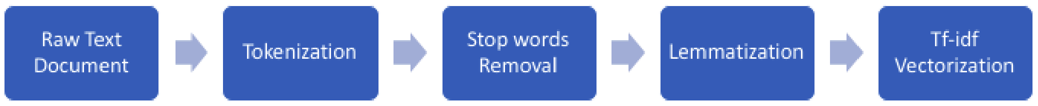


Figure 3.1: Data Preprocessing Pipeline Chart

3.2.1 Tokenization and Stop Words Removal

Tokenization can also be called as Lexical Analysis in computer science. It is a process of converting a sequence of characters into a sequence of tokens. In this project, since all collected documents are in English, the characters sequence consists of English alphabetical characters, digits from 0 to 1, punctuations and some special characters such as , # and . However, for generating text topics, characters other than English characters from a to z are not useful and thus they are removed from the character sequence prior to tokenization. After tokenization, each text document is split into a set of tokens which are all valid English words. Figure 3.2 shows the entire process flow of tokenization and Figure 3.3 shows an example of the input and output of the tokenization step.

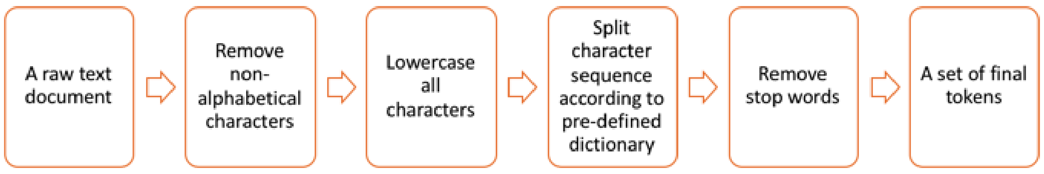

A raw text document
Remove non-alphabetical characters
Lowercase all characters
Split character sequence according to pre-defined dictionary
Remove stop words
A set of final tokens

Figure 3.2: Tokenization Process Flow

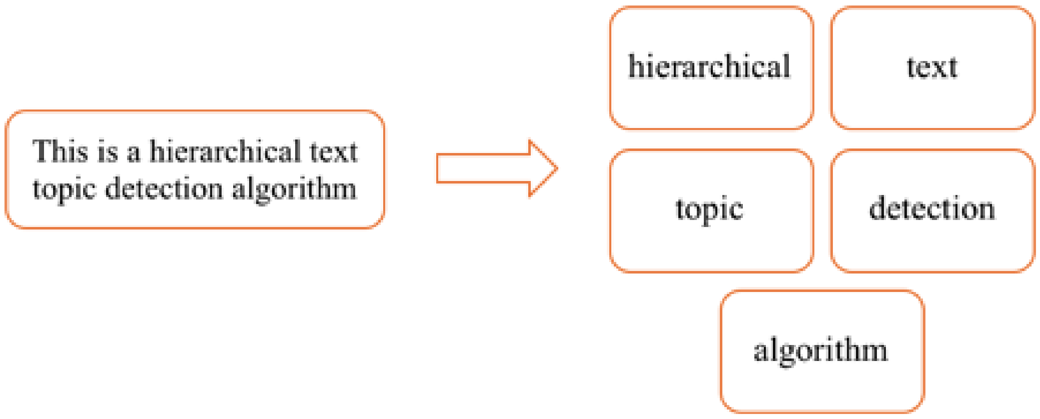


Figure 3.3: Tokenization Example

3.2.2 Lemmatization

In essence, text topic detection is to detect the commonality among a collection of documents. After tokenization, all documents are represented by sets of tokens so intuitively, if a token has been used many times by a document, then it is highly likely that this token can represent topics discussed in this document. However, in English, tokens in different forms may share the same meaning, but program is able to understand these variations. For example, nouns can be in its plural or singular form and verbs can be in different tenses. Hence, tokens need to be reduced to their base forms which are called stems in stemming analysis. There are a lot of stemming algorithms and Porter's rule-based algorithms [17] is one of the most popular choice for English text stemming. Nevertheless, stemming has major disadvantage that stemming usually goes through a crude heuristic process that chops off the ends of words in the hope of extracting stems correctly most of the time. Unfortunately, it often includes the removal of derivational affixes. Therefore, stemming sometimes ends up with stems that do not make sense or are hard to be interpreted. In the case of topic detection, if the stems are not easily understandable, the topic they represent becomes difficult to interpret. Therefore, stemming is not suitable for text preprocessing in topic detection.

Another approach to reduce tokens into their base forms is lemmatization. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. Using lemmatization instead of stemming makes the final text topics much easier to interpret. Table 3.2 shows examples on comparison of the effectiveness between stemming and lemmatization.

Table 3.2: Lemmatization and Stemming (Porter Stemmer) Comparison

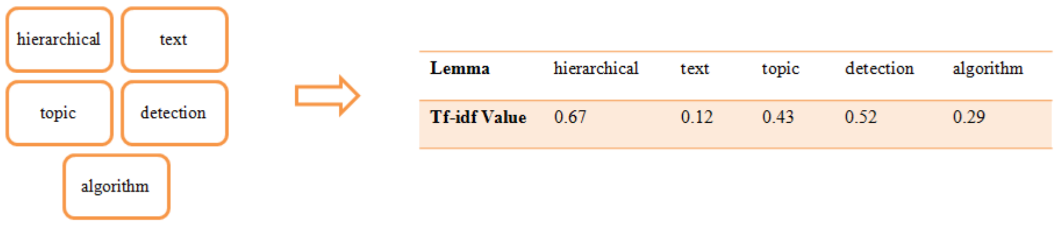
Token	**After Stemming**	**After Lemmatization**
is	is	be
are	ar	be
am	am	be
better	better	good
best	best	good

3.2.3 Tf-idf Vectorization

After lemmatization, all documents are converted into sets of lemmas, but they are still considered unstructured because lemmas also cannot be directly feed into any hierarchical text topic detection algorithms. Each set of lemmas needs to be transformed into vector form. There are a lot of methods in vectorizing documents. A simple one could be transferring documents into a *document-term matrix* where columns are lemmas used among all documents in the collection and each row represents one document. The values in the document-term matrix can be *Term-frequency (tf)* or *Term-frequency Times Inverse document-frequency (tf-idf)* [9]. Since *tf-idf* assigns more accurate weights to the lemmas that are more representative with respect to the document, *tf-idf* vectorization is performed on all sets of lemmas. Formula for *tf-idf* is

$$TFIDF(d,t) = TF(d,t) \times \log(\frac{n}{DF(d,t)}) + 1 \tag{3.1}$$

where *df* refers to the document-frequency of a lemma, *d* refers to a document and *t* refers to a term (lemma). The rationale of *tf-idf* can be explained in two aspects. For *tf*, it is straightforward that the higher number of times that a lemma is used in a document, the more important this lemma would be for this document. For *idf*, if a lemma appears in a large number of documents, then it indicates that this lemma may not be useful in differentiating those documents, so the weight for it should be lower. For example, lemmas with low *idf* can be some common terms in computer science such as 'computer', 'data' and 'program'. Furthermore, all *tf-idf* values need to be normalized using cosine normalization method row by row to ensure the length of each document vector equal to one. After normalization, each text document is converted into a unit vector whose distance to each other can be calculated using either *Cosine Similarity* or *Euclidean Distance*. If without normalization, the document vector can only be compared using *Cosine Similarity*. Figure 3.4 shows an example for the *tf-idf* vectorization.



Lemma	hierarchical	text	topic	detection	algorithm
Tf-idf Value	0.67	0.12	0.43	0.52	0.29

Figure 3.4: Tf-idf Vectorization Example

3.3 Dimensionality Reduction

After data preprocessing, all text documents in the collection are converted into a huge sparse document-term matrix with more than 50000 columns and more than 2000 rows. The sparsity of this matrix is normally very high because most entries are zero. This is due to the fact that each document uses only a subset of words from the dictionary and only those words being used have a non-zero *tf-idf* value entry. However, when using sparse data as input, clustering would not behave well because it would regard all data points as being very far apart and fail to detect any clusters by just converging all centroids into the center of the all data points. Since fuzzy clustering is chosen to be the foundation of the Hierarchical Fuzzy Topic Detection algorithm proposed in this project, the sparse document-term matrix cannot be fed into the it directly without performing dimensionality reduction. There are many mature dimensionality reduction techniques available. Given that the main focus of this project is not comparing the effectiveness of different dimensionality reduction, only two techniques which are *Autoencoder* and *Word Embedding* are selected and they would be compared in Model Comparison section to find the best dimensionality reduction technique based on the text topic detection result.

3.3.1 Autoencoder

An *Autoencoder* is a type of artificial neural network used for unsupervised learning of efficient codings. It is widely used in learning hidden data patterns, especially for image and text data. Learned data patterns can be used for fast model training. There are many variations of *Autoencoder* such as *Denoising Autoencoder* for data noise removal and *Sparse Autoencoder* for fast feature extraction. For this project, a simple *Autoencoder* is implemented, which is essentially a 3-layered Perceptron network. Figure 3.5 shows the basic structure of the 3-layered *Autoencoder* used in this project.

For the input layer, the number of input neurons is equal to the number of dimensions in the input document vector. The output layer has the same number of neurons as the input layer. However, the hidden layer should have much less number of neurons compared to other two to achieve dimensionality reduction purpose. The output from hidden layer's activation function can be regarded as the learned features that represent the document vector. The input and hidden layer constitutes the encoder while the hidden and output layer constitutes the decoder which amplifies the learned features back to the original document vector. Since *tf-idf* values are continuous value ranging from 0 to 1, *sigmoid* function whose output also ranging from 0 to 1 would be a good choice for all neurons' activation functions. The Formula for each layer's input and output is shown below.

Formula 3.2 shows the linear combination function which linearly combines the input from input layers and feeds the combination result to the activation function. X refers to the matrix of document vectors. V refers to the weight matrix of the connections between input neurons and hidden neurons. B_h

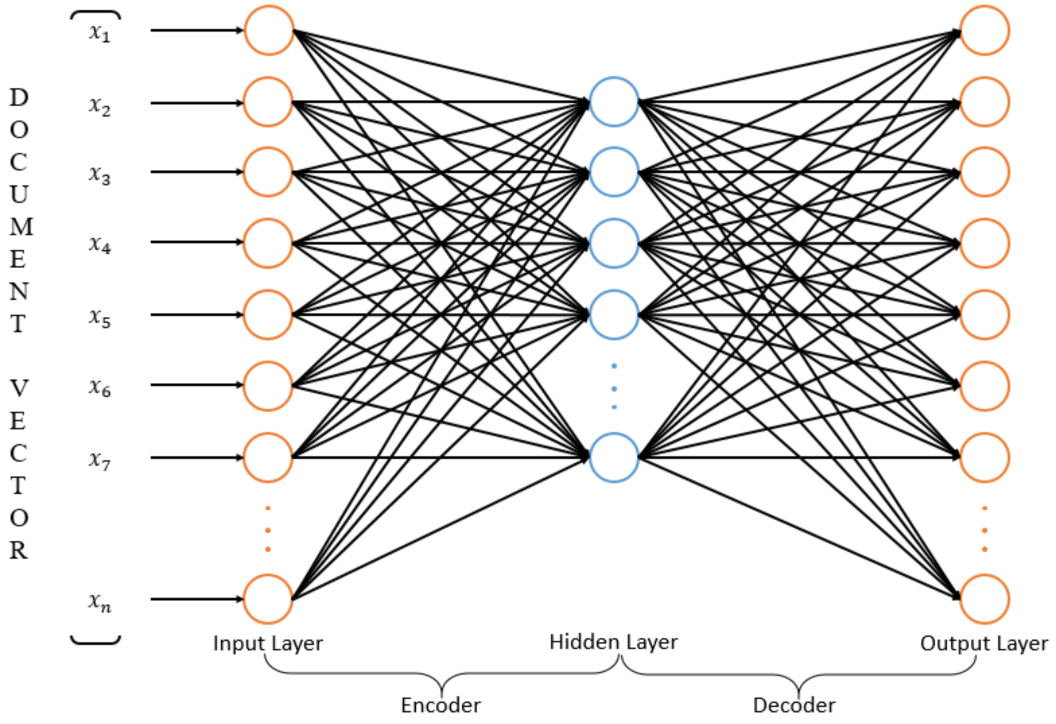


Figure 3.5: Autoencoder Structure

denotes that bias added to the combination result before it is passed to the activation function.

$$S = XV + B_h \tag{3.2}$$

Formula 3.3 shows the activation process of neurons in hidden layer. Z is the matrix of activation output to be passed to the next layer while g is the *sigmoid* activation function (Formula 3.4).

$$Z = g(S) \tag{3.3}$$

$$g(x) = f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

Formula 3.5 does the same thing as Formula 3.2 except that the input for it is from hidden layer's output instead of input layer's. Z refers to the matrix of activation outputs from hidden layer. W refers to the weight matrix of the connections between hidden neurons and output neurons. B_o denotes that bias added to the combination result before it is passed to the *sigmoid* activation function.

$$U = ZW + B_o \tag{3.5}$$

Formula 3.6 is the formula for activation process for the output layer. Y is final output of the *Autoencoder* while f is the sigmoid function (Formula 3.4)

$$Y = f(U) \tag{3.6}$$

The training algorithm use Gradient Descent to find the local minimum of the loss function J (Formula 3.7). In loss function J, P is the total number of document vectors in the document-term matrix while K is the number of dimensions (vocabulary size) of the document-term matrix. Back-propagation algorithm is used to update the weight of the connections based on the difference between the actual output and original input data.

$$J = \frac{1}{P}\sum_{p=1}^{P}\sum_{k=1}^{K}(x_pk - y_pk)^2 \tag{3.7}$$

Formula 3.8 shows the delta to change that equals to the different between actual matrix $X andoutputmatrix$Y) multiplied by the derivatives of the activation function Y.

$$\Delta = (X - Y)f'(U) \tag{3.8}$$

After calculating the Δ, weight and bias of the connection between hidden layer and output layer need to be updated correspondingly (see Formula 3.9 and Formula 3.10). α is the Gradient Descent learning rate pre-defined by the user.

$$W = W + \alpha Z^T\Delta \tag{3.9}$$

$$B = B + \alpha\Delta^T 1_P \tag{3.10}$$

Due to the fact that there are thousands of documents with around 50000 dimensions for each vector and also there is limited computing power when using local machines, it would be time-consuming to train the *Autoencoder* perfectly with zero training error. Furthermore, under same number of epochs, Stochastic Gradient Descent is too expensive to use while Batch Gradient Descent Learning sometimes fail to train the network properly. Hence, Mini-batch Gradient Descent algorithm is used to train the *Autoencoder*. The model parameters for training are shown in table 3.3

After the *Autoencoder* is well-trained, all document vectors are fed into the input layer once again, but this time, the output is taken directly from the hidden layer instead of output layer to obtain new document vectors with

Table 3.3: Mini-batch Gradient Descent Training Parameters

Parameter	**Value**
Number of Neurons in Hidden Layer	200
Epochs	1000
Batch Size	16
Learning Rate	0.1
Decay	0

much lower dimension. The new document vectors summarize the important features of the original document vector. The new document vectors would be the input to the next stage's Hierarchical Fuzzy Clustering (HFC).

3.3.2 Word Embedding

Although *Autoencoder* is able to reduce vector dimensions effectively by extracting useful information from raw document vectors, it is very time-consuming to train the neural network properly until it reaches low training error, especially when there is limited computing resources and amount of data is huge. Therefore, another dimensionality reduction technique called *Word2vec* which takes less time to complete is also tested.

Word2vec refers to a group of related models that are used to produce word embeddings [18]. Word embedding refers to language modeling and feature learning techniques in *NLP* where words from the vocabulary are mapped to vectors of real numbers. There are two famous architectures in *Word2vec*: Continuous-bag-of-words (CBOW) and Skip Grams [18]. In CBOW architecture, the model predicts the current word from a window of its surrounding context words. The order of context words does not influence prediction because of the bag-of-words assumption. On the contrary, for Skip Gram architecture, the model uses the current word to predict its surrounding context words. For this project, CBOW architecture is applied since the final output of the dimensionality reduction process is a single vector instead of a set of vectors. In essence, it is a two-layered neural network with first layer as input layer and second layer as output layer. The activation function of second layer is a *softmax* activation function. During training phase, the *Word2vec* model will create a sliding window which contains the context of the target token. In one iteration, each word would become target once and all neighboring words within the window would become the input to predict the target word. Please note that all words are represented as vectors whose values are randomly assigned before the training phase starts. A simple example of the *Word2vec* model is shown at Figure 3.6.

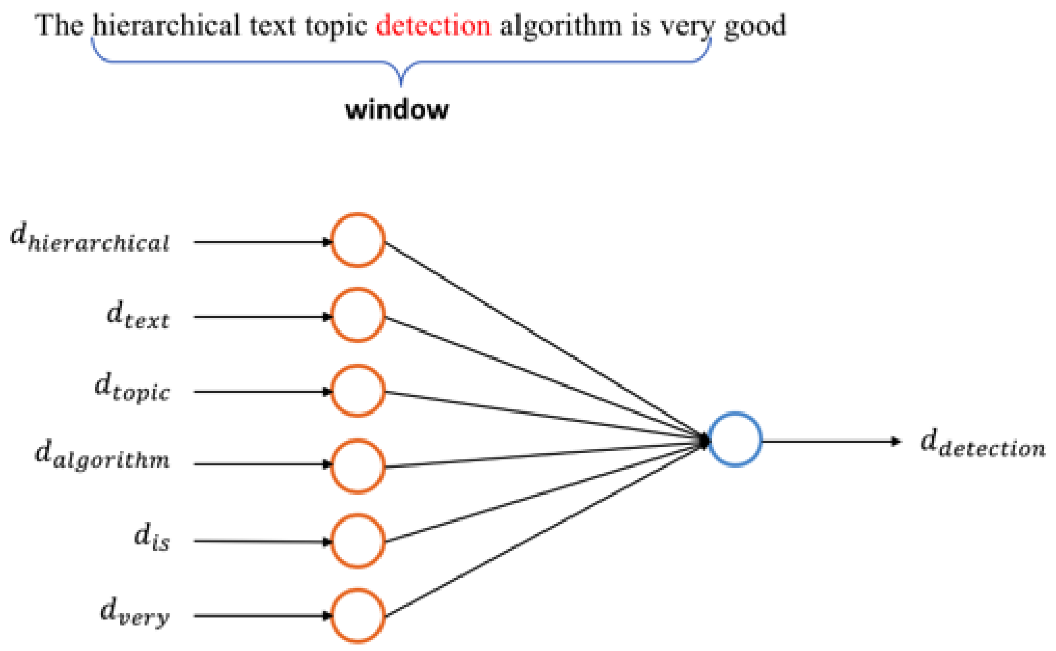


Figure 3.6: Word2vec Model Example

After the *Word2vec* model is trained properly, each word can be supplied to the trained model to obtain its vector representation. However, for each document, there are many different words being used, which means there is a need to aggregate all word vectors into a single document vector. A primitive way is to take the weighted average by frequency among all the words used within the document. Nevertheless, this method may be biased when some common words are used frequently in many documents, but those words are actually not meaningful or representative at all. Hence, *tf-idf* value of word with respect to the document is used as the weight of the word vector. The weighted average value of all word vectors using *tf-idf* values as weight is treated as the final vector representation of a document. Figure 3.7 shows a overview of the steps in *Word2vec* dimensionality reduction process. Some model parameters are present at Table 3.4.

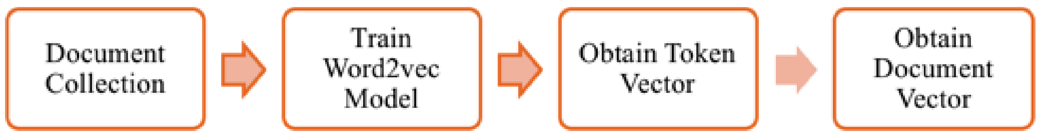


Figure 3.7: Word2vec Dimensionality Reduction Process

Table 3.4: Word2vec Model Parameters

Parameter	**Value**
Token Vector Dimension	200
Iterations	1000
Window Size	6

3.4 Hierarchical Fuzzy Clustering

After dimensionality reduction, all documents are converted into low dimension document vectors with values ranging from 0 to 1. The format of the documents vectors are same no matter which dimensionality reduction is used. Therefore, in this section, there is no distinction on the output from *Autoencoder* or *Word2vec*. The *Hierarchical Fuzzy Clustering (HFC)* algorithm as core component of the overall *Hierarchical Fuzzy Topic Detection (HFTD)* algorithm proposed in this project is a variation of the *Hierarchical Unsupervised Fuzzy Clustering (HUFC)* algorithm proposed in Amir B. Geva's paper [16]. *HUFC* algorithm has constraints on the total weights of all document vectors and uses both Euclidean Distance and Exponential Distance to represent the distance between document and centroids. However, the *(HFC)* algorithm proposed in this project has been designed in a way that it can accept relatively high dimension data as input with no constraints on vector weights. Furthermore, Euclidean Distance is the only distance metrics used and cluster validity metrics used is also different from *HUFC*. The basic building block of both *HUFC* and *HFC* is the *Fuzzy C-means Clustering (FCM)* algorithm [15].

3.4.1 Fuzzy C-means Clustering

Fuzzy C-means Clustering (FCM) algorithm is an extension of the popular *K-means Clustering* algorithm. *FCM* allows each document being clustered into different clusters (topics) at the same time. Therefore, each document vector has a weight vector which represents the weight of this document on different clusters. The sum of values in the weight vector should be equal to one whereas in *K-means Clustering* the weight vector should only has one value being either 1 or 0. Initially, the weight vector of each document is randomly initialized. During clustering, the cluster centroids and weight vectors of all documents are updated in turns until the centroids or weight vectors converge. The left flow chart of figure 3.8 shows a flow chart of *FCM*.

To update the weight vector, Formula 3.11 is used. w_{ij} is the weight value of *ith* weight vector on *jth* cluster. d is the document vector with reduced dimension. c is the vector of centroid. m is a fuzziness control parameter whose value is assigned based on the input data characteristics. For this project, a heuristic function $m = dimension(\mathbf{d})$ is applied.

$$w_{ij} = \frac{1}{\sum_{t=1}^{c} \left(\frac{\|\mathbf{d}_i - \mathbf{c}_j\|}{\|\mathbf{d}_i - \mathbf{c}_t\|} \right)^{\frac{2}{m-1}}} \tag{3.11}$$

Formula 3.12 shows the way to obtain new centroids after weight vectors are updated.

$$c_t = \frac{\sum_d w_t(d)^m d}{\sum_d w_t(d)^m} \tag{3.12}$$

The whole *FCM* algorithm aims to minimize the *Within Cluster Sum of Squares (WCSS)* as Formula 3.13 shows. It is a straightforward cost function because ideally, vectors in the each cluster should be close to the their centroids as much as possible.

$$WCSS = \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \|\mathbf{d}_i - \mathbf{c}_j\|^2 \tag{3.13}$$

3.4.2 Recursive FCM

Although *FCM* is able to perform soft clustering, it only generates a flat text topic structure. Hence, *FCM* needs to be continuously and recursively applied to generate a hierarchical text topic tree where each topic can be further divided into multiple children topics which should be more specific than their parent topic. The general idea is that, after 1 level of *FCM* is completed, for each topic in the result, another *FCM* is applied. This process is applied until the topic tree reaches the maximum tree level allowed predefined by the user or no more topics can be detected. The details of the algorithm is shown on the right flow chart of Figure 3.8

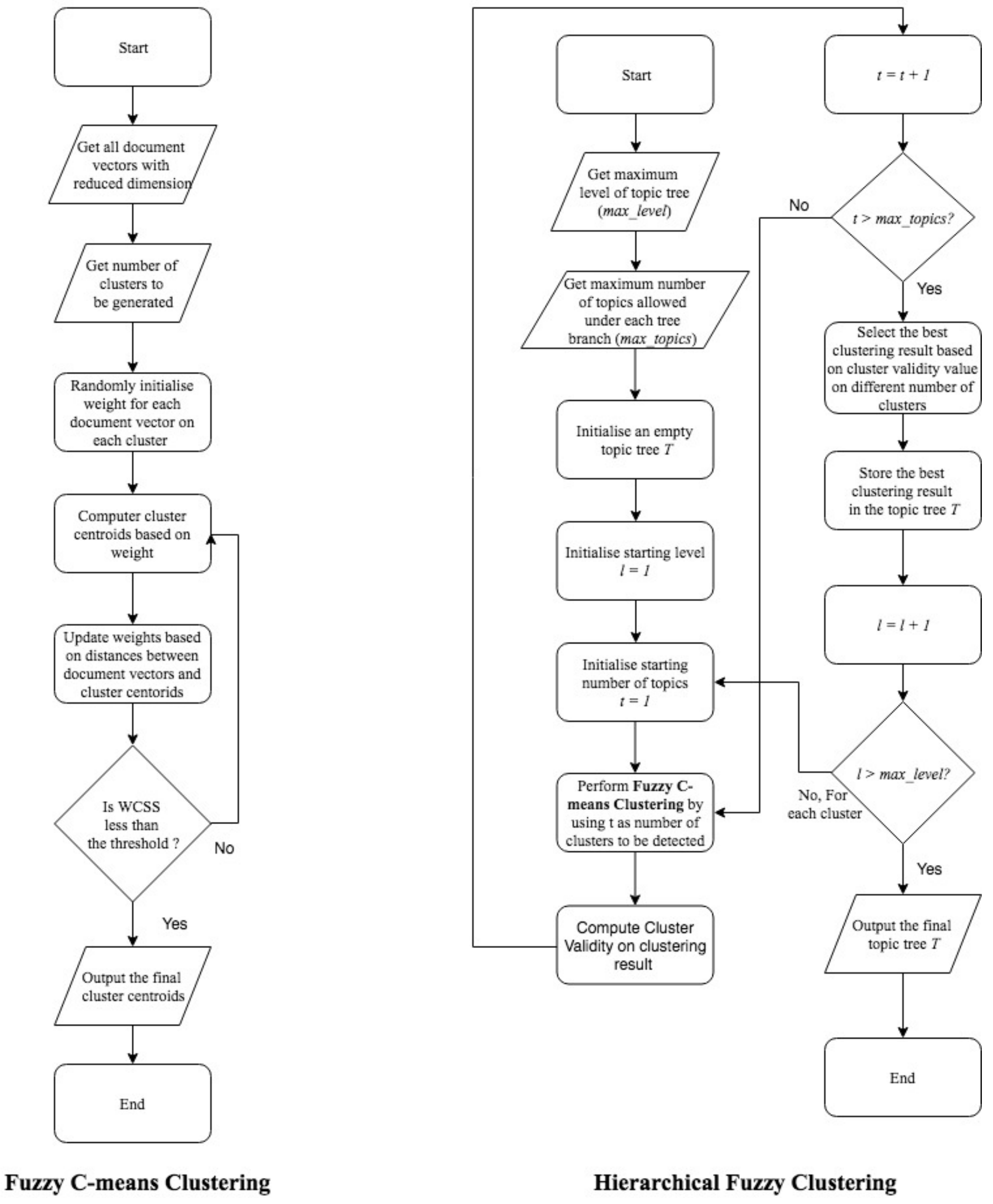

Start
Get all document vectors with reduced dimension
Get number of clusters to be generated
Randomly initialise weight for each document vector on each cluster
Computer cluster centroids based on weight
Update weights based on distances between document vectors and cluster centorids
Is WCSS less than the threshold ?
No
Yes
Output the final cluster centroids
End
Fuzzy C-means Clustering
Start
Get maximum level of topic tree (max_level)
Get maximum number of topics allowed under each tree branch (max_topics)
Initialise an empty topic tree T
Initialise starting level l = 1
Initialise starting number of topics t = 1
Perform Fuzzy C-means Clustering by using t as number of clusters to be detected
Compute Cluster Validity on clustering result
t = t + 1
t > max_topics?
No
Yes
Select the best clustering result based on cluster validity value on different number of clusters
Store the best clustering result in the topic tree T
l = l + 1
l > max_level?
No, For each cluster
Yes
Output the final topic tree T
End
Hierarchical Fuzzy Clustering

Figure 3.8: Hierarchical Fuzzy Clustering Flow Chart

3.4.3 Cluster Validity

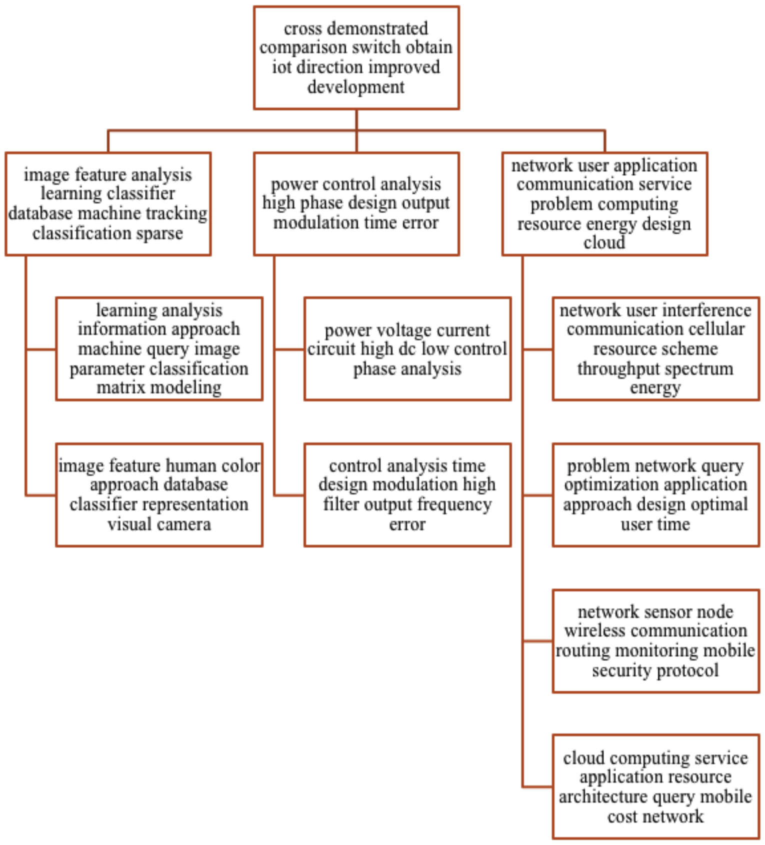
When running *FCM* at each level of the topic tree, it needs to know the optimal number of topics to be detected. To let *HCM* algorithms be able to smartly identify the optimal number of topics to be detected, *FCM* needs to be run in different number of clusters (topics) respectively and based on certain cluster validity metrics, select the optimal number of clusters. There are many existing cluster validity metrics such as *Partition Coefficient (PC)*, *Classification Entropy (CE)* and *Fuzzy shell-hypervolume (FHV)* [19]. For this project, *Modified Partition Coefficient (MPC)* is chosen to be the cluster validity metrics used during *HFC*. It is a normalized version of *PC*. *PC* ranges from $[\frac{1}{c}, 1]$ (c is the number of clusters) while *MPC* ranges from $[0, 1]$. Hence, the value range of *MPC* is not affected by the number of clusters. Normally, a good successful clustering can achieve *MPC* with value more than 0.9. The closer to 1 *MPC* is, the further the clusters are partitioned apart. The Formula 3.14 shows the calculation of *PC* and Formula 3.15 shows the calculation of *MPC*. w_{ij} is the weight of *ith* document vector on *jth* topic. n is the total number of documents. c is the total number of topics.

$$PC = \frac{\sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^2}{n} \tag{3.14}$$

$$MPC = 1 - \frac{c}{c-1}(1 - PC) \tag{3.15}$$

3.4.4 Topic Representation

At the end of *HFC*, a complete text topic tree is generated where each node on the tree is represented by a topic centroid. The centroid is a vector with reduced dimension and it is impossible to interpret the topic solely based on the values in the centroid vector. Therefore, the vector needs to be amplified back to the original dimension which is the dimension of the document vector right after *tf-idf vectorization*. Since there are two dimensionality reduction techniques are used, the amplification techniques for *Autoencoder* and *Word2vec* are also different. For *Autoencoder*, the centroid vector should be fed into the decoder part of the network and the output vector from output layer is the vector with amplified dimension. Using the amplified topic vector, words with highest *tf-idf* value can be selected, and they can be used to represent this topic. Figure 3.9 shows an example on a 3-level topic tree generated by *Autoencoder* using *IEEE* 2017 paper data. For *Word2vec*, it is much simpler because the trained *Word2vec* model is able to output words that are most similar to the centroid vector and those words can be used to represent this topic. Figure 3.10 shows an example on a 3-level topic tree generated by Word2vec model using *IEEE* 2017 paper data.



cross demonstrated comparison switch obtain iot direction improved development
image feature analysis learning classifier database machine tracking classification sparse
power control analysis high phase design output modulation time error
network user application communication service problem computing resource energy design cloud
learning analysis information approach machine query image parameter classification matrix modeling
power voltage current circuit high dc low control phase analysis
network user interference communication cellular resource scheme throughput spectrum energy
image feature human color approach database classifier representation visual camera
control analysis time design modulation high filter output frequency error
problem network query optimization application approach design optimal user time
network sensor node wireless communication routing monitoring mobile security protocol
cloud computing service application resource architecture query mobile cost network

Figure 3.9: Autoencoder Topic Tree Example

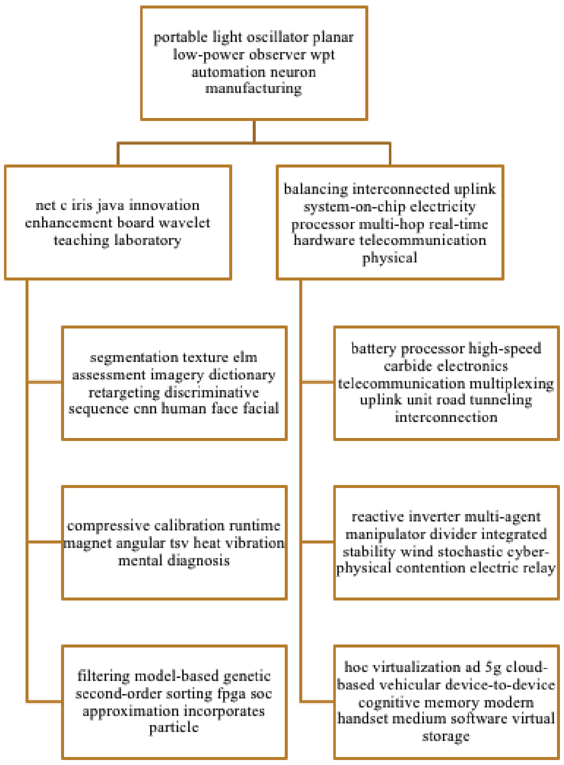

portable light oscillator planar low-power observer wpt automation neuron manufacturing
net c iris java innovation enhancement board wavelet teaching laboratory
balancing interconnected uplink system-on-chip electricity processor multi-hop real-time hardware telecommunication physical
segmentation texture elm assessment imagery dictionary retargeting discriminative sequence cnn human face facial
compressive calibration runtime magnet angular tsv heat vibration mental diagnosis
filtering model-based genetic second-order sorting fpga soc approximation incorporates particle
battery processor high-speed carbide electronics telecommunication multiplexing uplink unit road tunneling interconnection
reactive inverter multi-agent manipulator divider integrated stability wind stochastic cyber-physical contention electric relay
hoc virtualization ad 5g cloud-based vehicular device-to-device cognitive memory modem handset medium software virtual storage

Figure 3.10: Word2vec Topic Tree Example

Chapter 4

Algorithms Comparison

4.1 Two Dimensionality Reduction Technique

Before applying the *Hierarchical Fuzzy Topic Detection (HFTD)* algorithm to paper data across different years to get insights from the changing topic trend, model selection needs to be done between two dimensionality reduction techniques (*Autoencoder* and *Word2vec*). In this project, data preprocessing steps including the dimensionality reduction would also be considered as part of the whole *HFTD* algorithm, which means that the *HFTD* algorithm would take raw text documents as input and the output a complete hierarchical text topic tree. Hence, there are in total two algorithms that require comparison, which are *Hierarchical Fuzzy Clustering With Autoencoder (HFCWA)* and *Hierarchical Fuzzy Clustering with Word2vec (HFCWW)*. To better illustrate the effectiveness of the two algorithms designed in this project, a hierarchical text topic detection algorithms using Andrew Ng's *Latent Dirichlet Allocation (LDA)* [4] as its core component is also developed and used as part of the comparison.

Unfortunately, there are very few existing metrics to evaluate the text topics detected after any topics detection algorithm. Most researches evaluate the text topics detected subjectively by human interpretation, which is not quantitative and hard to be used to compare across different algorithms. In order to effectively evaluate the text topic tree quality generated by two different models, two metrics are designed specifically for the hierarchical text topic tree. The two metrics are duplication index and association index.

4.2 Duplication Index

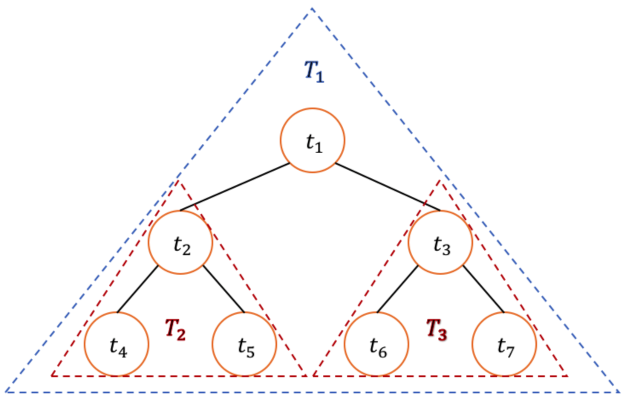
Since text topics are represented by set of tokens, it is very common that same tokens are used to describe more than one topic. For example, token 'computer' is highly likely to appear in the token sets of different topics because it is very commonly used in all *IEEE* computer science related papers. However, text topics with good quality should be different from each other as much as possible, which means that in ideal case, each topic should be described by a totally distinct set of tokens with different meanings. Hence a duplication index metric is designed to indicate the amount of meaning overlaps between topics at the same level under same parent topics. After the topic tree is generated, each topic on the tree has its own duplication index except for leaf topics. Formula 4.1 shows the calculation of duplication index for a specific topic.

$$DI_{Topic} = \frac{Count_{children}(Duplicate\ Tokens)}{Count_{children}(All\ Tokens)} \tag{4.1}$$

However, it is much more complicated to evaluate the quality of the whole hierarchical topic tree instead of just a topic. In this project, topics that reside on higher level would be given more importance because they are more general topics with less possibility to have duplicate tokens. The calculation of the tree duplication index starts from the bottom level until it reaches the root node. It is inherently a recursive algorithm because the whole topic tree can be divided into multiple small subtrees and subtrees can also be further divided. A topic tree's duplication index is calculated based on the root topic's duplication index and all its subtrees' duplication index. A simple example on the calculation of a 3-level topic tree's duplication index is shown at Figure 4.1. The duplication index is normalized into range between 0 and 1. 1 means there is no duplicate tokens used at all in the tree while 0 means that the whole tree uses one token to represent all topics.

4.3 Association Index

Just using duplication index is not enough for text topic tree quality evaluation because duplication index only shows the how bad topics in the tree have overlapping meaning with each other. Another metric needs to be designed in a way that it is able to indicate how closely related for tokens that belongs to same topics are. This new metric would try to resemble the degree of concen-



$$DI(T_2) = \frac{DI(t_2) + DI(t_4) + DI(t_5)}{3}$$

$$DI(T_3) = \frac{DI(t_3) + DI(t_6) + DI(t_7)}{3}$$

$$DI(T_1) = \frac{DI(t_1) + DI(T_2) + DI(T_3)}{3}$$

Figure 4.1: Duplication Index Calculation Example

tration for a text topic. To achieve this, database on relationships between all tokens needs to be pre-established before evaluation of topics. However, there is no existing well-established database that provide such functionality. Since the main focus of this project is not building a semantic network of tokens, Word Association API [20] is used, which provides a weighted association score between any two words. Although the relationship weight between different tokens may not be accurate, it is already good enough for this project due to resource and time constraints.

The calculation of association index is very similar to duplication index. Each topic in the topic tree has its own association index which depends on the closeness of the relationship between tokens that represent this topic. Formula 4.2 shows the calculation of association index of a specific topic where n is the total number of tokens used to represent the topic.

$$AI_{Topic} = \frac{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} Association(token_i, token_j)}{n} \tag{4.2}$$

The way to calculate the association index for the whole hierarchical text topic tree based on individual topic's association index is the same as the way to obtain duplication index. It is normalized into a range between 0 and 1 and the larger value is, the higher association between tokens have.

4.4 Comparison Result

To compare *HFCWA* and *HFCWW* algorithms, three metrics which are duplication index, association index and time are calculated. *HLDA* is also added into the comparison. Three algorithms are run on 5 sets of data which are *IEEE* computer science paper data from 2013 to 2017. The result is shown in Table 4.1, Table 4.2 and Table 4.3.

Table 4.1: HFCWA Model Result

Data	**Duplication Index**	**Association Index**	**Time**
2013	0.898	0.083	8094
2014	0.897	0.064	8178
2015	0.914	0.077	7625
2016	0.921	0.063	7611
2017	0.904	0.051	7722
Average	**0.907**	**0.068**	**7846**

Table 4.2: HFCWW Model Result

Data	**Duplication Index**	**Association Index**	**Time**
2013	0.953	0.113	405
2014	0.931	0.110	444
2015	0.930	0.101	423
2016	0.965	0.120	402
2017	0.954	0.111	471
Average	**0.947**	**0.111**	**429**

From the above three tables, it is obvious that in terms of time, *HFCWW* performs the best. In terms of duplication index, *HFCWW* has the highest average duplication index. For association index, *HFCWW* also has the highest average value. Generally speaking, *HFCWW* is relatively better than the other two because it is performs well in all three metrics. Therefore, it is chosen to be the best model for the topic trend analysis across years. Another observation is that *HFCWA* performs the worst in all 3 metrics among three algorithms and the time it takes is much larger than the other two because it takes long time to train an *Autoencoder* until it reaches very high training accuracy, especially when input data's dimension is large.

Table 4.3: HLDA Model Result

Data	**Duplication Index**	**Association Index**	**Time (s)**
2013	0.935	0.042	965
2014	0.937	0.039	838
2015	0.961	0.043	880
2016	0.946	0.043	1038
2017	0.953	0.032	1044
Average	**0.946**	**0.040**	**953**

Chapter 5

Research Topic Trend Analysis

5.1 Data Preparation

After the best dimensionality reduction algorithm is chosen as *Word2vec*, the whole hierarchical fuzzy topic detection algorithm is applied to identify potential research topic changing trend for Computer Science domain. Due to time and computing resource constraint, the data used for research topic trend analysis only includes *IEEE* computer science paper data that are published from 1993 to 2017. For papers in each year, only around first 2000 papers data ranked by number of citation is used. To reduce the complexity of analysis, data in different years is aggregated into groups and each group spans for 5 years. Therefore, there are in total 5 groups of data. Table 5.1 shows the grouping details. Due to limited computing power, the maximum depth of the text topic tree is set as 3 excluding the root level and the maximum number of topics allowed at each level is set as 5.

Table 5.1: Data Grouping

Group Number	**Start Year**	**End Year**
Group 1	1993	1997
Group 2	1998	2002
Group 3	2003	2007
Group 4	2008	2012
Group 5	2013	2017

5.2 Trend Analysis On Keyword

First and foremost, to analyze the research topic trend, the trend of words used in topics would be a good indicator. Using the keyword that describes the topic as a basic unit would be helpful in identifying the topic trend. However, since the proposed hierarchical fuzzy topic detection algorithm identifies topics in a hierarchical manner, it is not reasonable to compare topics that reside different level because they have different level of abstraction. Therefore, the trend analysis is performed with topics in different level separately. The first level topics are not included in keyword analysis because there are too few keywords in total for first level topic, which may make the keyword trend biased.

5.2.1 Level 2 Topics

Figure 5.1, 5.2, 5.3, 5.4 and 5.5 show the histogram of the topic 15 keywords used in describing level 2 topics in 5 different year groups.

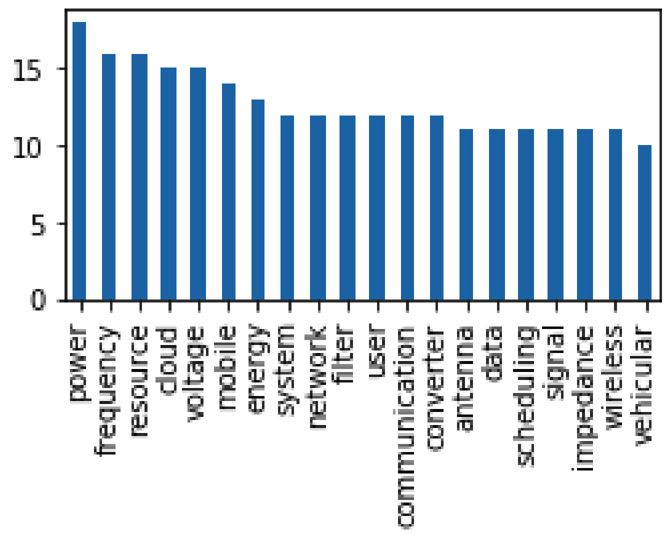


Figure 5.1: Level 2 Topics Top 15 Keywords for 2013-2017

From all histograms for level 2 topics, here are some key observations:

1. From 1993 to 2017, the research keywords do not deviate greatly, keywords such as as 'communication', 'resource', 'network' and 'data' almost appear in all five groups, which means that the 25 years time space may not be enough to identify research topics trend.

2. From 1993 to 2002, the research focus seems to concentrate on network communication, energy-saving and hardware (keyword: communication, network, energy and wireless)

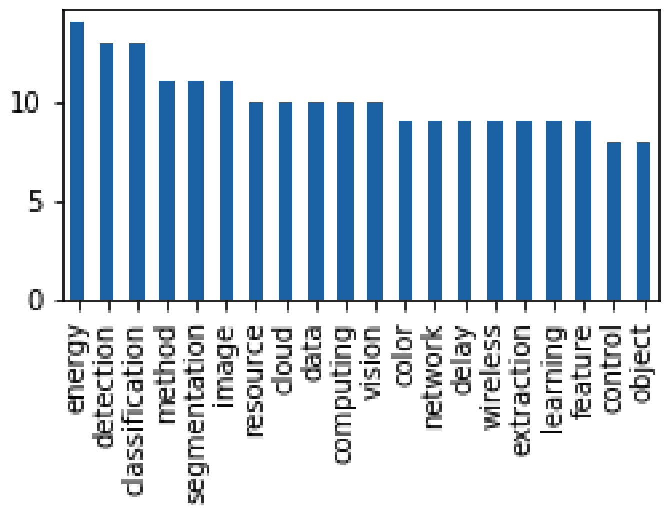


Figure 5.2: Level 2 Topics Top 15 Keywords for 2008-2012

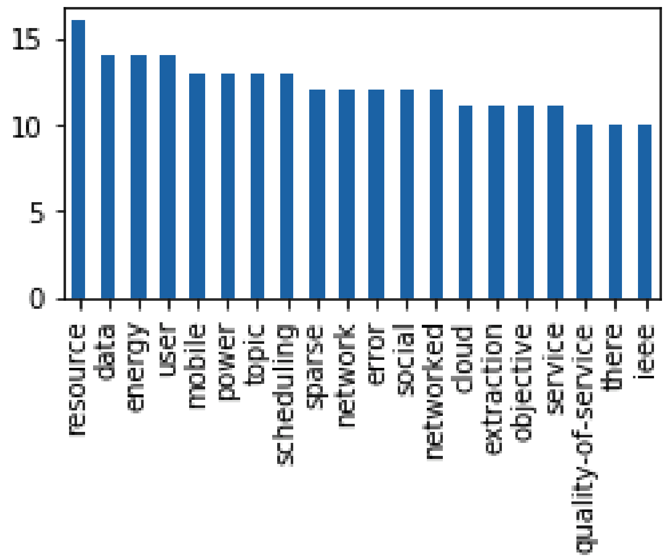


Figure 5.3: Level 2 Topics Top 15 Keywords for 2003-2007

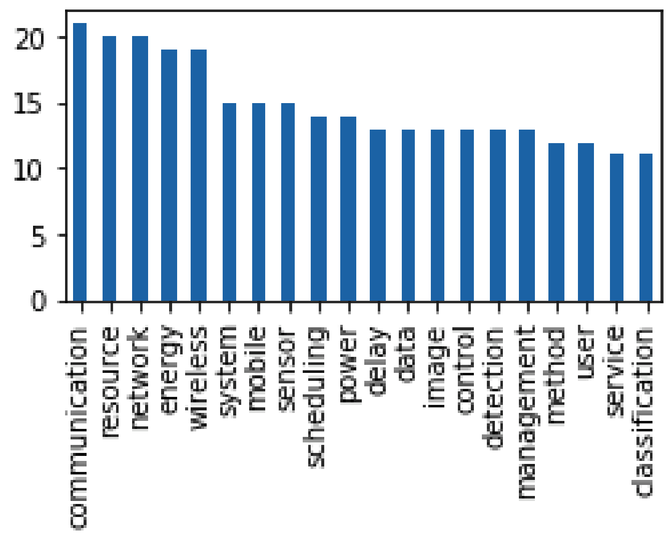


Figure 5.4: Level 2 Topics Top 15 Keywords for 1998-2002

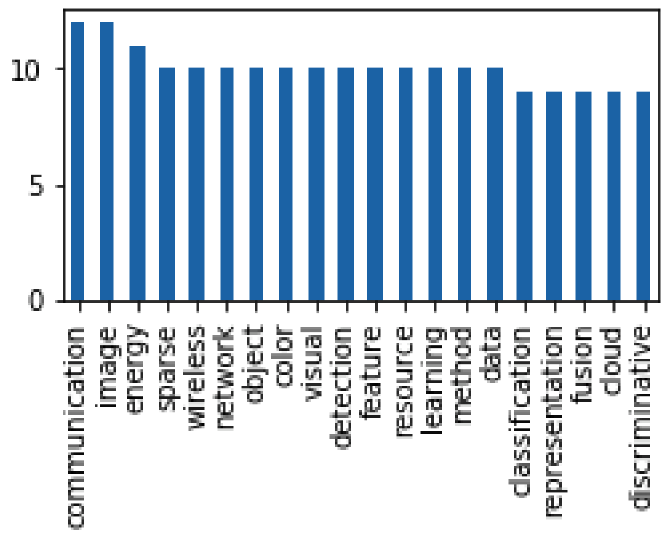


Figure 5.5: Level 2 Topics Top 15 Keywords for 1993-1997

3. From 1993 to 2002, some researches start to focus on artificial intelligence and image processing (keyword: classification, visual, image and discriminative).

4. From 2003 to 2017, the research focus seems to be concentrate on network and cloud computing (keyword: cloud, mobile, network). It is interesting that artificial intelligence related keywords disappear.

5. From hardware related keywords such as 'voltage', 'antenna' and 'converter' come back again. A new keyword 'vehicular' suddenly appear on the top 15 keyword. These observation may be explained by the current hot trend of driverless car.

5.2.2 Level 3 Topics

Figure 5.6, 5.7, 5.8, 5.9 and 5.10 show the histogram of the topic 15 keywords used in describing level 3 topics in 5 different year groups.

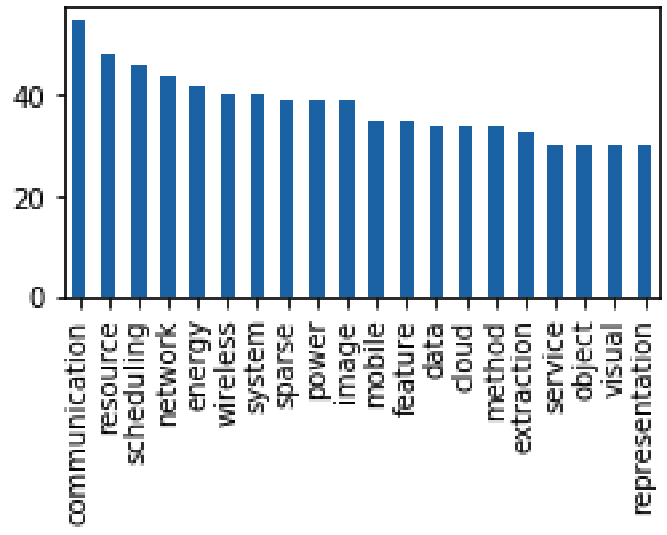


Figure 5.6: Level 3 Topics Top 15 Keywords for 2013-2017

From all histograms for level 3 topics, there are some insights can be observed:

1. The top 15 keywords are very inconsistent with keywords in the level 2 topic keywords analysis, which further prove the fact that the level 2 topics is a high level abstraction of level 3 topics.

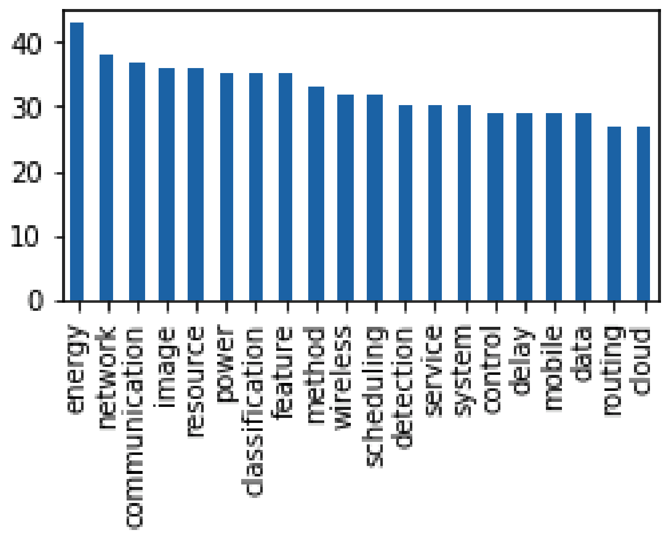


Figure 5.7: Level 3 Topics Top 15 Keywords for 2008-2012

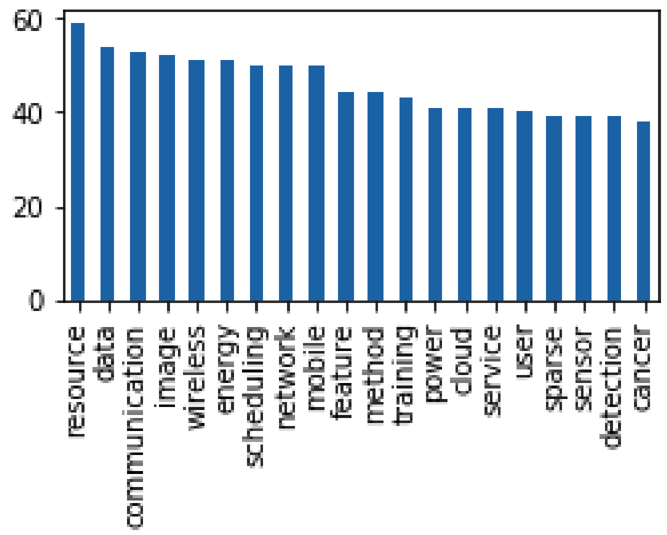


Figure 5.8: Level 3 Topics Top 15 Keywords for 2003-2007

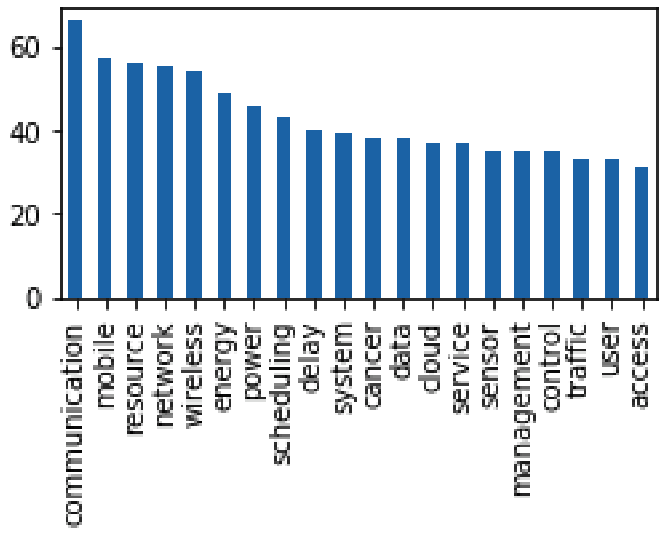


Figure 5.9: Level 3 Topics Top 15 Keywords for 1998-2002

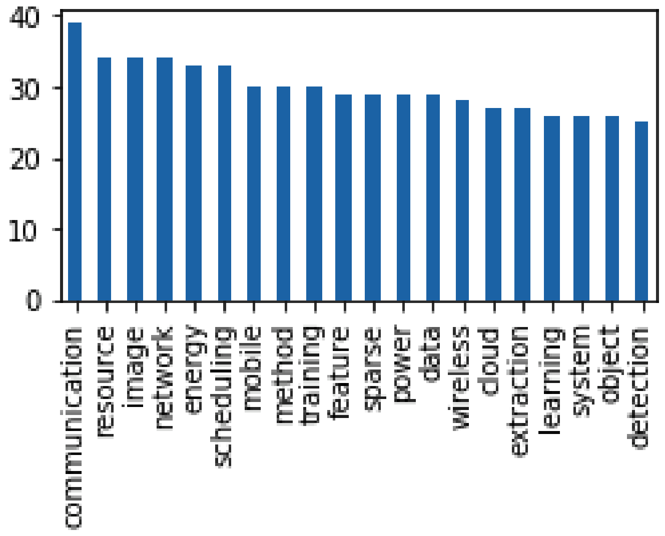


Figure 5.10: Level 3 Topics Top 15 Keywords for 1993-1997

2. One interesting observation is the keyword 'cancer' only appears from 1998 to 2007, which may indicate that during those years, computer science has been applied in health care, especially cancer treatment.

3. Keyword 'cloud' appears across all years, which may show that cloud computing has been hot through those years.

4. keyword 'communication' never falls out of top 3 keywords, which means that network communication issue always remains to be a hot topic.

5.3 Trend Analysis on Topics

Compared to trend analysis on keywords, trend analysis on topics is much more difficult because for two keywords, they are easy to tell whether they are same while for two topics, it is hard to tell whether they are talking about the same thing from a program's perspective. To overcome this issue, a simple topic aggregation algorithm is implemented, which compares two topics using their keywords. If two topics have a certain amount of overlapping in terms of keywords, they can be aggregated as the same topic. The amount of overlapping should exceed a certain threshold. The threshold for this project is set as $\frac{1}{3}$ of the total number of keywords describing a topic. Same as trend analysis on keywords, the analysis with topic as basic unit is performed level by level separately.

After analyzing the topics shown in Table 5.2, 5.3 and 5.4, some basic insights can be obtained:

1. Through the years from 1993 to 2017, the research topic focus does not seem to deviate much.

2. There are roughly in total 8 distinct topics can be identified. They are machine learning, image processing, feature extraction, cloud computing, network communication, computer hardware, operating system and health care (cancer).

3. Research topics such as computer hardware, cloud computing, operating system and health care seems to be more popular in 1990s, but they start to disappear in 21st century.

4. Research topics such as machine learning, image processing and feature extraction are always among the most popular research topics.

Table 5.2: Level 1 Topics

Time	Topic 1	Topic 2	Topic 3
1993 - 1997	set, classification, hyperspectral, color, image, detection, learning, recognition	force, 2-d, cancer, torque, surface, distortion, tissue, insertion, skin	control, three-phase, battery, harmonic, frequency, voltage, inverter, antenna, capacitance, diode
1998 - 2002	set, classification, color, image, deep, detection, learning, recognition, visualization	control, load, protocol, computing, communication, energy, bandwidth, reliability, radio	switched, neuron, integral, direct, incremental, asynchronous, voltage, proportional, practical
2003 - 2007	classification, color, image, detection, learning, recognition, visualization, saliency, feature	control, load, three-phase, generation, harmonic, stability, frequency, voltage, inverter, battery	communication, energy, downlink, spectrum, rate, fading, capacity, network, antenna, interference
2008 - 2012	protocol, computing, communication, energy, spectrum, streaming, cost, radio, scheduling, cloud	classification, color, image, detection, learning, recognition, visualization, saliency, feature	characterization, ambient, resistance, thermal, frequency, transforms, inductor, skin
2013 - 2017	set, classification, hyperspectral, color, image, deep, detection, learning, recognition, visual	control, three-phase, battery, generation, harmonic, stability, frequency, voltage, measurement	control, protocol, communication, energy, reliability, scheduling, cloud, information, network

Table 5.3: Level 2 Topics

Time	Topic 1	Topic 2	Topic 3
1993 - 1997	six, force, cancer, torque, tissue, insertion, skin, ct, selected, polarization	control, load, three-phase, battery, harmonic, stability, voltage, inverter, capacitance, system	protocol, cellular, communication, energy, spectrum, radio, scheduling, access, management, network
1998 - 2002	communication, energy, scheduling, cloud, management, network, service, access, user, data	three-phase, harmonic, frequency, voltage, inductor, inverter, capacitance, diode, reactive	set, classification, hyperspectral, color, image, deep, detection, learning, recognition, visualization
2003 - 2007	computing, communication, energy, scheduling, cloud, network, service, access, sensor, wireless	set, classification, hyperspectral, color, image, detection, learning, recognition, discrimination	evolutionary, cancer, scan, event, concern, accelerometer, needed, barrier, get, leveraging
2008 - 2012	protocol, computing, communication, energy, spectrum, streaming, cost, scheduling, cloud, access	transmitting, seizure, rat, incentive, skin, electrode, polarization, when, microwave, stimulation	control, three-phase, battery, harmonic, stability, frequency, voltage, measurement, inverter
2013 - 2017	frequency, impedance, wideband, port, antenna, radiation, polarization, resonator, wave, band	control, load, three-phase, frequency, voltage, inverter, capacitance, system, reactive, power	set, classification, color, image, detection, learning, recognition, visualization, discrimination

Table 5.4: Level 3 Topics

Time	Topic 1	Topic 2	Topic 3
1993 - 1997	communication, energy, scheduling, access, management, network, service, routing, wireless, user	coding, fault-tolerant, virtualization, scheduling, discus, cache, surveillance, content	force, cancer, surface, tissue, insertion, ct, calibration, selected, polarization, camera
1998 - 2002	projection, cancer, surface, tissue, insertion, ct, compact, rather, selected, camera	computing, communication, energy, scheduling, cloud, management, network, service, sensor, wireless	coding, fault-tolerant, virtualization, scheduling, relatively, discus, cache, surveillance
2003 - 2007	set, classification, color, image, learning, recognition, saliency, feature, extraction, method	protocol, communication, energy, scheduling, cloud, network, service, routing, sensor, wireless	coding, virtualization, scheduling, relatively, discus, cache, surveillance, content, backhaul
2008 - 2012	classification, color, image, detection, learning, recognition, saliency, feature, extraction	communication, energy, scheduling, cloud, access, management, network, service, interference, delay	certain, closed-form, virtualization, scheduling, relatively, additional, cache, content, backhaul
2013 - 2017	classification, color, image, detection, learning, feature, extraction, method, segmentation	set, classification, color, image, detection, learning, discriminative, saliency, feature, extraction	virtualization, scheduling, relatively, cache, surveillance, content, backhaul, optimal

5. Research topics like network communication remains hot throughout the years.

6. For level 1 topics, research topics in health care is hot only in 1993-1997 period and they never show up in top 3 topics again after that.

7. For level 1, level 2 and level 3 topics, Network and communication researches are becoming hot since 2003. They are the hottest topic from 2008 to 2012. However, they also start to cool down and no longer belong to top 3 topics from in 2013-2017 group.

8. For level 2 topics, it seems that cloud computing and network-related researches start to get hot in 1993-1997 period and they become the hottest topic from 1998 to 2007. However, they become less popular in 2008-2012 and 2013-2017 period.

9. For level 3 topics, it seems that machine learning and data analytics related papers dominate the research world from 2013 to 2017. And these topics start to get popular in 2003-2007 period.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

An complete hierarchical text topic detection algorithm called Hierarchical Fuzzy Topic Detection is developed. This algorithm solves the issue that common fuzzy clustering algorithm cannot be applied to text data or sparse data. The whole algorithm starts from data preprocessing, followed by dimensionality reduction and then ends with hierarchical fuzzy clustering. The input of this algorithm is a collection of raw text documents and the output of it is a complete hierarchical text topic tree with each topic on the tree labelled by set of keywords. For dimensionality reduction, two techniques *Autoencoder* and *Word Embedding*are implemented and compared using reliable metrics. The algorithm using *Word Embedding* seems to work better that *Autoencoder* and the corresponding topic detection algorithm words better than the hierarchical version of existing *Latent Dirichlet Allocation* algorithm. Some useful insights are obtained based on the hierarchical text topic tree detected using proposed algorithm.

6.2 Recommendations for Future Work

Due to the time and computing resource constraints, some areas are not fully examined and there are some limitations in the hierarchical text topic detection algorithm. Future work can explore the following directions:

1. Compare *HFC* with other existing hierarchical topic detection algorithms;
2. Fine-tune *Autoencoder* and *Word2vec* models;

3. Design more robust and effective cluster validity metrics;
4. Design more comprehensive metrics for hierarchical topics quality evaluation.
5. Use more data across different domains to do meta-analysis to do overall research topic trend analysis

6.2.1 Comparison Against Other Topic Detection Method

There are many other mature topic detection method like *Hierarchical Latent Tree Analysis* [5]. The clustering method used in the algorithm designed in this project is very different from the popular Bayesian Inference approach used in other topic detection algorithms. More comparison can be done between them so that pros and cons can be clearly identified for both types of algorithms, which would also help to understand what circumstances they fit in the most.

6.2.2 Fine-tune Dimensionality Reduction Models

For models used in dimensionality reduction like *Autoencoder* and *Word2vec*, they are both neural network models, so there are a lot of hyper-parameters which are available to fine-tune the model. Some hyper-parameters can be batch size, learning rate, decay parameter and cost function (e.g. L1 or L2 Norm). These hyper-parameters can be set in different combinations until the model that gives best accuracy is found. Due to time constraint and large amount of data to be trained, in this project, the range of hyper-parameters for tuning is set to be relatively slow. Given more computing resources, larger range can be used for tuning so that the resulting model would perform more accurate dimensionality reduction.

6.2.3 Design Effective Cluster Validity Metrics

There is only one cluster validity metric used in this project during *HFC*, which is *MPC*. Sometimes it could be biased because it is actually based on the sum of squared weight of all document on all topics. The fuzziness parameter m would in fact affect the weight and hence, different m value would result in different *MPC*. There are many other cluster validity metrics like silhouette coefficient. The hierarchical topic detection model would be more reliable and accurate if it uses a combination of different effective cluster validity to determine the optimal number of clusters (topics) at each level.

6.2.4 Design Effective Topic Quality Metrics

In this project, there are two topic quality metrics designed, which are duplication index and association index. However, these metrics evaluate topic quality solely based on the tokens in the topics detected without looking at the statistics behind. Perplexity in the *Latent Dirichlet Allocation* [4] would be a good example in using log-likelihood to evaluate topic quality. On the other hand, the Word Association API used in association index is not built for topic quality evaluation and therefore, all association index is relatively low in all models tested in the project. If there is a topic ontology based on topic keywords, it can be used to build a more precise association index.

6.2.5 Research Topic Trend Analysis On More Data

Due to limited computing power, this project only uses computer science paper data from *IEEE*, which may not be able to represent the whole research area. More analysis can be done in other area such as physics, material science and mathematics. Data should also be collected from other data source such as dblp and Google Scholar, so that more thorough meta-analysis can be performed on research topic trend.

Bibliography

[1] What is big data? https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html. [Online; accessed 5-Mar-2017].

[2] Seth Grimes. Unstructured data and the 80 percent rule. https://breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/, January 2008. [Online; accessed 16-Mar-2018].

[3] Topic model. https://en.wikipedia.org/wiki/Topic_model/, March 2018. [Online; accessed 16-Mar-2018].

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Hierarchical unsupervised fuzzy clustering. *Journal of Machine Learning Research*, 3: 993–1022, 2003.

[5] Nevin L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:97–723, June 2004.

[6] Cluster analysis. https://en.wikipedia.org/wiki/Cluster_analysis, March 2018. [Online; accessed 16-Mar-2018].

[7] MIGUEL E. RUIZ and PADMINI SRINIVASAN. Hierarchical text categorization using neural networks. *Information Retrieval*, 5:87–118, 2002.

[8] ANDREAS S. WEIGEND, ERIK D. WIENER, and JAN O. PEDERSEN. Exploiting hierarchy in text categorization. *Information Retrieval*, 1:193–216, 1999.

[9] Salton and M. McGill, editors. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[10] Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6), 1990.

[11] T. Hofmann. Probabilistic latent semantic indexing. *Proceedings of the Twenty-Second Annual International SIGIR Conference*, 1999.

[12] D. M. Blei, T. L. Grifths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *ACM 57*, 2(7), January 2010.

[13] Loulwah AlSumait, Daniel Barbara, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. *IEEE International Conference on Data Mining*, 2008.

[14] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. *ACL*, pages 183–190, 1993.

[15] JAMES C. BEZDEK, ROBERT EHRLICH, and WILLIAM FULL. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10 (3):191–203, 1984.

[16] Amir B. Geva. Hierarchical unsupervised fuzzy clustering. *IEEE*, 7(6), December 1999.

[17] Porter and Martin F. An algorithm for suffix stripping. *Program*, 14(3): 130–137, 1980.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *In Proceedings of Workshop at ICLR*, 2013.

[19] Rajesh N. Dave. Validating fuzzy partitions obtained through c-shells clustering. *Pattern Recognition Letters*, 17:613–623, 1996.

[20] Word association api. `https://wordassociations.net/en/api`, March 2018. [Online; accessed 16-Mar-2018].

Appendices

Appendix A

Hierarchical Text Topics Detected For 2017

level 0
Topic 1: system control network power communication energy resource wireless architecture estimation
level 1 topic 1
Detect 2 topics
Topic 1: system control network power communication energy resource wireless architecture estimation
Topic 2: forecasting wind predictive markov discharge establish nn output supply qos
level 2 topic 1
Detect 2 topics
Topic 1: system network control power communication energy resource wireless architecture estimation
Topic 2: fault diagnosis magnet power synchronous voltage leakage converter circuit defect
level 3 topic 1
Detect 4 topics
Topic 1: resource mobile service network cloud computing energy communication wireless management
Topic 2: image feature learning recognition method sparse color classification dictionary training
Topic 3: forecasting positioning wind stability square discharge predictive markov nn establish
Topic 4: power control voltage system circuit measurement stability switching

converter frequency
level 3 topic 2
Detect 5 topics
Topic 1: fingerprint generating extraction color consideration display vein activity finger level
Topic 2: malware nash fall o risk community networksin emergency interval open
Topic 3: diagnosis fault extended little patient verification outage anomaly assessment motor
Topic 4: fault magnet power voltage synchronous leakage converter switching transient circuit
Topic 5: causal changing whose exploiting basis first six ev efficient crowd
level 2 topic 2
Detect 1 topics
Topic 1: forecasting wind predictive markov discharge establish nn output supply qos