Language Independent Text Summarizer and Deep Self-Organizing Cube

Ahmed Abdelfattah Saleh Sherif

Tese De Doutorado em Sistemas Mecatrônicos
Programa de Pós-Graduação em Sistemas Mecatrônicos (PPMEC)
Departamento de Engenharia Mecânica – ENM

Orientador

Prof. Dr. Li Weigang

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

Brasília
2022

Universidade de Brasília - UnB
Programa de Pós-Graduação em Sistemas Mecatrônicos - PPMEC
Departamento de Engenharia Mecânica – ENM

Language Independent Text Summarizer and Deep Self-Organizing Cube

Ahmed Abdelfattah Saleh Sherif

Tese de Doutorado Acadêmico submetida ao Programa de Pós-graduação (PPMEC) em Sistemas Mecatrônicos do Departamento de Engenharia Mecânica da Faculdade de Tecnologia da Universidade de Brasília, como Parte dos Requisitos Necessários para a Obtenção do Grau de Doutor em Sistemas Mecatrônicos.

Aprovada por:

Brasília, 1 de Dezembro de 2022

Acknowledgement

I owe my deepest gratitude to my supervisor Prof. Dr. Li Weigang for his continuous support and advice since the first Artificial Intelligence course I took with him. He inspired me greatly to work in the field of Artificial Intelligence and to continue pursuing my academic and knowledge advancement.

Deep and special thanks as well to Mr. ambassador Hossam Zaki, Deputy Secretary-General of the Arab League, for his great support and continuous encouragement for me to work on my Ph.D. during his mission as the ambassador of Egypt to Brazil.

At last and kindly, I would like to thank all my family members for their continuous support, persistent help, encouragement and sincere prayers.

Ahmed A. Saleh

2022

Resumo

O rápido desenvolvimento da Internet e o crescimento exponencial de dados em texto na web trouxe desafios consideráveis para tarefas relacionadas ao gerenciamento de texto, classificação e recuperação de informações. Nesta tese, propomos dois novos modelos independents de domínio, com o objetivo de melhorar o desempenho da generalização nas áreas de Processamento de Linguagem Natural (NLP) e Deep Learning (DL), para enfrentar os desafios impostos pelo grande crescimento de dados e a necessidade de extrair informação adequada e melhorar a inferência de conhecimento. Ambos os modelos adotam uma abordagem direta, porém eficiente, que depende da extração de características intrínsecas nos dados modelados, a fim de realizar sua tarefa pretendida de forma totalmente independente do domínio. A estratégia de avaliação de desempenho aplicada nesta tese visa testar o modelo em um conjunto de dados de referência e então comparar os resultados obtidos com os modelos padrão existentes**.** Além disso, os modelos propostos são testados contra modelos de última geração apresentados na literature, para o mesmo conjunto de dados de referência.

No domínio da NLP, a maioria das técnicas de resumo de texto na literatura dependem, de uma forma ou de outra, de léxicos pré-estruturados dependentes da linguagem, bancos de dados, marcadores (*taggers*) e/ou *parsers*. Tais técnicas requerem um conhecimento prévio da linguagem do texto que está sendo resumido. Nesta tese, propomos uma nova ferramenta de resumo, *UnB Language Independent Text Summarizer (UnB-LITS),* que é capaz de resumir um texto de maneira independente do idioma. O modelo proposto baseia-se em características intrínsecas do texto que está sendo resumido e não de seu idioma e, portanto, elimina a necessidade de léxicos, bancos de dados, e marcadores ou parsers que dependem do idioma. Dentro dessa ferramenta, desenvolvemos uma forma inovadora de codificar as formas dos elementos do texto (palavras, *n-grams*, frases e parágrafos), além de propor algoritmos independentes de linguagem, capazes de normalizar palavras e performar derivações relativas ou lematização. Os algoritmos propostos e sua rotina *Shape-Coding* permitem que a ferramenta *UnB-LITS* extraia características intrínsecas dos elementos do documento e os pontue estatisticamente para obter um resumo extrativo representativo independente da linguagem do documento. O modelo proposto foi aplicado em diferentes conjuntos de dados referência, em inglês e português, e os resultados foram comparados com doze abordagens consideradas de ponta pela literatura recente. Além disso, o modelo foi aplicado em conjuntos de dados de notícias em francês e espanhol, e os resultados foram comparados aos obtidos por ferramentas comerciais padrão. O *UnB-LITS* apresentou uma melhor performance do que todas as abordagens de última geração, bem como quando comparado às outras ferramentas comerciais nos quatro idiomas, mantendo a sua natureza independente à linguagem.

Por outro lado, a tarefa de classificação multidimensional (MDC) pode ser considerada a descrição mais abrangente de todas as tarefas de classificação, pois une vários espaços de classe e seus vários membros de classe em um único problema de classificação composta. Os desafios no MDC surgem das possíveis dependências de classe em diferentes espaços. E também do desequilíbrio de rótulos em conjuntos de dados de treinamento devido à falta de todas as combinações possíveis. Nesta tese, propomos um classificador de aprendizado profundo MDC que conta com uma natureza simples mas eficiente, chamado *"Deep Self-Organizing Cube"* ou "DSOC" que pode modelar dependências entre classes, enquanto consolida sua capacidade de classificar combinações raras de rótulos. O DSOC é formado por dois componentes n-dimensionais: o classificador de hipercubo (*hypercube*) e as múltiplas redes neurais DSOC conectadas ao hipercubo. O componente de múltiplas redes neurais é responsável pela seleção de recursos e segregação de classes, enquanto o classificador hipercubo é responsável por criar a semântica entre vários espaços de classe e acomodar o modelo para classificação de amostras raras. O DSOC é um algoritmo de aprendizado de várias saídas que classifica amostras com sucesso em todos os espaços de classe, de maneira simultanea. Para desafiar o modelo DSOC proposto, realizamos uma avaliação em dezessete conjuntos de dados de referência nos quatro tipos de tarefas de classificação: binário, multiclasse, multi-rótulo e multidimensional. Os resultados obtidos foram comparados com quatro classificadores padrão e oito abordagens competitivas de última geração relatadas na literatura. O DSOC alcançou desempenho superior em relação aos classificadores padrão, bem como as abordagens de última geração em todas as quatro tarefas de classificação. Além disso, em termos de métricas de precisão exata (*Exact Match*), o DSOC performou melhor do que todas as abordagens de última geração em **77,8% dos casos**, o que reflete a capacidade superior do DSOC de modelar dependências e classificar, com sucesso, as amostras raras em todas as dimensões de maneira simultanea.

Palavras-chave: *Summarization extractivas; Aprendizagem profunda; Classificador de Aprendizagem Profundo; Cubos Profundos de Auto-Organização; DSOC; Idioma Independente Summarization; UnB-LITS.*

Abstract

The rapid development of the Internet and the massive exponential growth in web textual data has brought considerable challenges to tasks related to text management, classification and information retrieval. In this thesis, we propose two novel domain agnostic models, aiming at improving the generalization performance in the fields of Natural Language Processing (NLP) and Deep Learning (DL), to address the challenges imposed by the massive growth in data and the need for proper information retrieval and knowledge inference. Both models adopt a straightforward, yet efficient, approaches that depend on extracting intrinsic features in the modeled data, in order to perform their intended task in a totally domain agnostic manner. The performance evaluation strategy applied in this thesis aims at testing the model on benchmark dataset and then compare the obtained results against those obtained by the standard models. Moreover, the proposed models are challenged against state-of-the-art models presented in literature for the same benchmark dataset.

In NLP domain, the majority of text summarization techniques in literature depend, in one way or another, on language dependent pre-structured lexicons, databases, taggers and/or parsers. Such techniques require a prior knowledge of the language of the text being summarized. In this thesis, we propose a novel extractive text summarization tool, *UnB Language Independent Text Summarizer* (UnB-LITS), which is capable of performing text summarization in a language agnostic manner. The proposed model depends on intrinsic characteristics of the text being summarized rather than its language and thus eliminates the need for language dependent lexicons, databases, taggers or parsers. Within this tool, we develop an innovative way of coding the shapes of text elements (words, n-grams, sentences and paragraphs), in addition to proposing language independent algorithms that are capable of normalizing words and performing relative stemming or lemmatization. The proposed algorithms and its *Shape-Coding* routine enable the *UnB-LITS* tool to extract intrinsic features of document elements and score them statistically to extract a representative summary independent of the document language. The proposed model was applied on an English and Portuguese benchmark datasets, and the results were compared to twelve state-of-the-art approaches presented in recent literature. Moreover, the model was applied on French and Spanish news datasets, and the results were compared to those obtained by standard commercial summarization tools. *UnB-LITS* has outperformed all the state-of-the-art approaches as well as the commercial tools in all four languages while maintaining its language agnostic nature.

On the other hand, Multi-dimensional classification (MDC) task can be considered the most comprehensive description of all classifications tasks, as it joins multiple class spaces and their multiple class members into a single compound classification problem. The challenges in MDC arise from the possible class dependencies across different class spaces, as well as the imbalance of labels in training datasets due to lack of all possible combinations. In this thesis, we propose a straightforward, yet efficient, MDC deep learning classifier, named "*Deep Self-Organizing Cube*" or "*DSOC*" that can model dependencies among classes in multiple class spaces, while consolidating its ability to classify rare combinations of labels. *DSOC* is formed of two *n*-dimensional components, namely the *Hypercube Classifier* and the multiple *DSOC Neural Networks* connected to the hypercube. The multiple neural networks component is responsible for feature selection and segregation of classes, while the Hypercube classifier is responsible for creating the semantics among multiple class spaces and accommodate the model for rare sample classification. DSOC is a multiple-output learning algorithm that successfully classify samples across all class spaces simultaneously. To challenge the proposed DSOC model, we conducted an assessment on seventeen benchmark datasets in the four types of classification tasks, binary, multi-class, multi-label and multi-dimensional. The obtained results were compared to four standard classifiers and eight competitive state-of-the-art approaches reported in literature. The DSOC has achieved superior performance over standard classifiers as well as the state-of-the-art approaches in all the four classification tasks. Moreover, in terms of *Exact Match* accuracy metrics, DSOC has outperformed all state-of-the-art approaches in 77.8% of the cases, which reflects the superior ability of DSOC to model dependencies and successfully classify rare samples across all dimensions simultaneously.

Keywords: *Extractive Summarization*; *Deep Learning, Deep Learning Classifier*; *Deep Self-Organizing Cube*; *DSOC*; *Language Independent Summarization; UnB-LITS.*

Contents

List of Figures

List of Tables

List of Abbreviations

ATS	Automatic Text Summarization
BC	Binary Classification
BR	Binary Relevance
CC	Concrete Concepts
CNN	Convolutional Neural Network
CPS	Class Powerset
DiC	Expected Degree of Interactions between Classes
DL	Deep Learning
DNN	Deep Neural Network
DoC	Degree of Compression of the Summary
DoS	Degree of Similarity
DSOC	Deep Self-Organizing Cube
ESC	Ensembles of Super Classes
GNN	Graph Neural Network
LITS	Language Independent Text Summarizer
LSTM	Long-short Term Memory
MCC	Multi-Class Classification
MDC	Multi-Dimensional Classification
MLC	Multi-Label Classification
nD	Deep Self-Organizing Cube‘s Neighbor Distance
NE	Named Entity
NER	Named Entity Recognition
nGfW	N-gram Form Weight. It can be 2*GfW*, 3*GfW*, 4*GfW* or 5*GfW* for bigrams, trigrams, 4-grams and 5-grams respectively.
nGS	N-gram score. It can be 2GS, 3GS, 4GS or 5GS for bigrams, trigrams, 4-grams and 5-grams respectively.
nGsW	N-gram Shape Weight. It can be 2*GsW*, 3*GsW*, 4*GsW* or 5*GsW* for bigrams, trigrams, 4-grams and 5-grams respectively.
NLP	Natural Language Processing
OOV	Out-of-Vocabulary
PS	Paragraph Score
PsW	Paragraph Shape Weight

RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SC	Sentence Score
SLU	Sentence Language Uniformity
SsW	Sentence Shape Weight
UnB-LITS	UnB - Language Independent Text Summarizer
WfW	Word Form Weight
WS	Word Score
WsW	Word Shape Weight

Chapter 1: Introduction

This study focuses on developing domain agnostic models, aiming at improving their generalization performance in fields of Natural Language Processing (NLP) and Deep Learning (DL). With the recent advances in computation, NLP field is gaining great advantage from adopting models and methodologies from Artificial Intelligence, while DL is benefiting from the fast pace developments in computational power.

On the other hand, the rapid development of the Internet and the massive exponential growth in web textual data has brought considerable challenges to tasks related to text management, classification and information retrieval. As such, Automatic text summarization (ATS) is becoming an extremely important means to solve this problem. ATS tends to mine the gist of the original text and then automatically generate a concise and readable summary that reflects the core important information in that text. Therefore, developing an efficient text summarization model is essential for information retrieval, knowledge inference, text processing, and dimensionality reduction for subsequent classification and understanding.

As will be discussed in details in the following chapter, text summarization task can be performed with a wide range of techniques. The vast majority of which depend on pre-structured lexicons, databases, parsers and taggers, which are language dependent. In other words, such techniques require a prior knowledge of the language of the text to be summarized and in certain situations requires knowledge of the contextual domain. Such prerequisite might affect the generalization performance of the model in case it faces new language, therefore arises the need for an efficient, straightforward language agnostic model.

Moreover, with this exponential growth in data, another challenge arises in the field of classification tasks. Where, observations, data and knowledge components exist in multidimensional spaces with complex interactions and dependencies. This complexity is considered a major challenge that requires a multi-dimensional classifier capable of modeling dependencies and infer knowledge at various abstract level. It requires an efficient multi-dimensional classifier that is capable of performing seamlessly in a domain agnostic manner while modeling complex semantics in data, and be prepared for unseen observation in domains that suffer from profound data imbalance, and hence maximize its generalization performance.

In this study, we propose two novel models, a language agnostic text summarization model in the field of NLP, and a domain agnostic deep learning classifier for multi-dimensional classification tasks. We call the first model ***UnB-LITS***, a short for "***UnB-Language Independent Text Summarizer***", while the deep learning classifier is called "***DSOC***", stands for "***Deep Self-Organizing Cube***".

Due to the dual nature of this study, the thesis is divided between the language agnostic summarization model and the deep multi-dimensional classifier. As such, wherever relevant, a section will be divided into two subsections; one dedicated to each of the research interlinked domains.

1.1 Overview

1.1.1 Language Agnostic Text Summarizer

The present description of sentence processing in human cortex differentiates three linguistic processing phases [1]. In the first sentence-level processing phase, the local phrase structure is built based on word category information. In the second phase, syntactic and semantic relations in the sentence are computed. These involve the computation of the relations between the verb and its arguments, thereby leading to the assignment of thematic roles (i.e., the analysis of who is doing what to whom). Once both semantic and syntactic information lead to the compatible interpretation, comprehension can easily take place. [2]

This means, simply, that in order to understand a written text first we have to get the words, then sentences and then relate them to topic comprehension or context to achieve a human like understanding. Words themselves should be categorized into *Named Entities* (nouns, concrete concepts, etc.) as well as verbs, prepositions, etc.

In this research, we aim to achieve similar level of textual understanding in a language agnostic manner, avoiding the need to extract verbs, nouns or other syntactical relations that require a prior knowledge of language and/or its context. Rather, we extract prominent phrases to form an extractive summary and then use these features and phrases to represent the document for further multi-dimensional classification, if needed.

1.1.1.1 Natural Language Processing (NLP)

Natural language processing (NLP) applications have spread widely and gained increased attention and focus in the recent years, for analyzing human language computationally [3]. Many fields have benefited from the recent advancement of NLP models and algorithms as machine translation, information retrieval, summarization and question answering, etc.

Natural language processing is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human natural languages [4].

The most important models used in the field of *NLP* are: a) *State Machines*, b) *Rule systems*, c) *Logic*, d) *Probabilistic models*, and e) *Vector-space* models. Other algorithms have been derived from these core models, among the most important of which are: a) *State Space Search Algorithms,* such as *Dynamic Programming,* and b) *Machine Learning Algorithms*, such as *Classifiers* and other learning algorithms [5].

In general, Natural Language Processing tasks can be grouped in many classes, the most common of which are:

i. *Text Summarization*: to extract descriptive information from a text. Summaries can be *abstractive* (text rephrasing), *extractive* (extract important elements) summarization, or *hybrid* [6].
ii. *Machine Translation:* translate between different languages [7].
iii. *Named Entity and Concrete Concepts Extraction*: extract locations, persons, drugs, addresses, titles, etc. [8]
iv. *Information Retrieval and Extraction*: extracting important information from text, like a meeting, location, time, etc. [9]
v. *Question Answering*: to answer questions formulated using natural language in a correct and efficient way. This field also includes retrieving information from databases based on asked questions [10].
vi. *Sentiment Analysis*: extract the sentiment people have over a specific topic, which has a wide range of applications in marketing and political campaigning [11].
vii. *Part-of-Speech (POS) tagging:* to tag part of texts defining which part is noun, verb, adjective, etc. [12]

In this research, we focus on Automatic Text Summarization (ATS) as a base for more complex tasks, like information retrieval and concrete concepts extractions and other, with no need of language dependent tools like parsers, taggers, lexicons or databases. Detailed discussion on ATS as well as the recent work in literature is presented in Chapter 2: Literature Review and Related Work.

1.1.2 Deep Learning Classifier

As mentioned in section 1.1.1.1, human brain performs at several levels of processing in order to achieve the ultimate understanding or knowledge capturing. It is believed that each level is learning features or representations at increasing levels of abstraction. This kind of observations has inspired a fast growing trend in machine learning known as ***Deep Learning*** [13], which attempts to replicate this kind of architecture in computer [14].

Many algorithms have been used in the literature to achieve Deep Learning Architecture, the classical and most widely used of which are: *Deep Convolutional Networks* [15], *Deep Boltzmann Machine* [16], *Deep Belief Network* [17] and *Deep Neural Network* [18].

Deep Learning though, has couple of disadvantages or limitations [19]. One of which is the curse of dimensionality, where deep models often have millions of parameters especially in the natural language processing ***NLP*** field. However, in this study we are proposing a Deep Learning multi-dimensional classifier that has an integrated dimensionality reduction layers to address the curse of dimensionality problem.

1.1.2.1 Multi-dimensional Classification

Multi-dimensional classification (MDC) is a generalization of the multi-label classification (MLC) problem where the classification output is a vector in multiple heterogeneous class spaces. The output (*Y*) is a vector of multiple class variables, where each member (Y_i) specifies its class membership in one particular class space. Ma & Chen [20] have displayed the different types of classification problems in terms of the number of output class variables (*m*) and their possible values (*k*) as shown in **Figure 1**. Both binary classification (BC) and multi-class classification (MCC) have a single output class variable, whose value is binary [0, 1] in case of BC and multiple [0, 1, ..*K*] in MCC. On the other hand, both MLC and MDC have multiple output class variables ($m \geq 2$), whose values are binary in case of MLC and multiple in MDC.

From another perspective, BC and MCC are considered single-output learning problems, while MLC and MDC are categorized under multiple-output learning. Multiple-output classification aims at simultaneously predicting multiple outputs given a single set of input variables. Xu et al. [21] have pointed out the lack of sufficient studies that generalize different forms of multiple-output learning despite of its growing importance in complex decision-making problems and the attention it has got in the recent years. As such, they conducted a thorough survey, depicting all aspects of multi-output learning including challenges, inputs and outputs structure, evaluation metrics as well as applications (e.g. Multi-label Document Categorization [22], Multi-label Semantic Scene Classification [23], Multi-label Video Annotation [24], etc.).

Due to the multi-variate nature of the output space of multi-output classification, classes across various dimensions might exhibit dependencies, correlations and/or complex interactions that should be considered when designing the classification model. Such dependencies and interactions means that the prediction of one label influences predictions of other labels. MDC models that do not take into consideration such dependencies will suffer from propagation of error and lack of classification accuracy across the entire class spaces. Independent Classifiers (IC) or Binary Relevance (BR) that involve building single dimension multi-class classifier for each class variable [25] suffer from lack of accuracy, while Classifiers Chain (CC) that involves chain of Bayesian classifiers [26] or even the Monte Carlo scheme for chain sequencing and inference [27] suffer from error propagation due to the difficulties in finding optimal chain order. [28]

Dembczynski et al. [29] have categorized label dependencies among classes in MLC into two categories, *conditional* and *unconditional* label dependencies. Using the standard statistical notations of multivariate regression models ($Y_i =$

$h_i(\boldsymbol{X}) + \epsilon_i(\boldsymbol{X})$), *unconditional* dependence represents the deterministic part (i.e. the function $h_i(\boldsymbol{X})$), while the *conditional* dependence represents the stochastic part (i.e. the error $\epsilon_i(\boldsymbol{X})$). As such, *unconditional* dependence refers to the expected dependencies between classes in Y, and can be measured using Pearson correlation or any other correlation coefficients.

To overcome the aforementioned drawbacks and limitations as well as to model the underlying class dependencies in MLC, MCC and MDC, several models and techniques were introduced in literature. Some of which involve models that consider all possible class combinations as *label powerset* (LP) or modified versions of it [30], others create *Super-Classes* based on the *conditional dependencies* among all or part of the class spaces [31], as well as models based on pairwise interactions among those classes [32]. Others used ensembles to tackle the problem of label-combinations, as for example EPS that discard the less frequent label combinations, RAKEL that creates ensemble of random label subsets, etc. [33]. However, those methods still suffer from the *imbalance problem*, where the output class variables combinations might not be uniformly distributed over the data space in the training set. This imbalance causes the classifier to lean (skew) towards the majority classes and fail to model the characteristic of minority classes, with the risk of overfitting which will affect the accuracy and robustness of the overall model. [34]

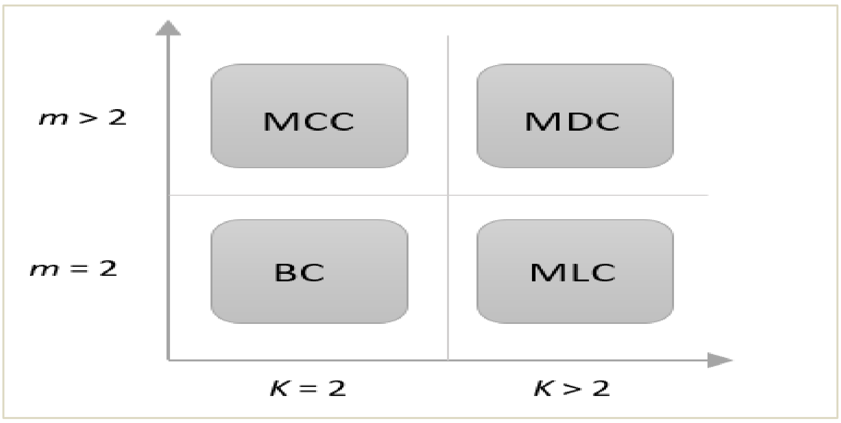


Figure 1. Different classification tasks, where *m* is the number of class variables and *k* is the number of possible values per class variable. [20]

1.2 The Research Objective

For each of the two studies conducted in this thesis, we can summarize the research objective as follows:

1.2.1 Language Agnostic Text Summarizer

1.2.1.1 Problem Statement

The main objective of this research is to obtain a model capable of performing efficient extractive text summarization in a language independent manner. However, this goal is challenged by the heavy dependence of text pre-processing steps on language related preprocessing tools (parsers, stemmers, taggers, stop words lexicons, etc.).

Moreover, efficient tagger or parsers are not always available for a particular language. In addition, knowing that lexicons are mostly contextual, therefore, preparing and refining domain specific lexicons for all languages is considered a big challenge among linguistic researchers.

Apart from the language dependence of preprocessing tools, obtaining an efficient representative summary also requires extracting or identifying Named Entities (NE's) and Concrete Concepts (CC's) due to their influence on the summarization quality. Such task is heavily dependent on prior knowledge of the language of the text to be summarized.

1.2.1.2 Proposed Solution

In order to fulfill the objective of the study and to address the underlying challenges, we propose a novel model that tackles the aforementioned problems while achieving superior summarization performance in a language agnostic manner. The proposed model should be characterized by the following, in order to achieve its intended goal:

a) The model should depend on intrinsic features of the elements of the document to be summarized rather than its language.

b) The model should be completely independent of any language-related tool throughout the entire process, starting from the pre-processing step until the post-processing step and the subsequent summary generation.

c) The model should be, as well, capable of removing or neutralizing the effect of unimportant words (like stop words) without the need of stop words lexicons that are by definition language dependent.

d) The model should preserve the weight of potential words and sentences, with the ability to identify strong and important CC's and NE's without any dependence on external databases or corpora.

e) Moreover, the model should be capable of determining sentences containing multiple languages and determine the importance of those sentences in accordance.

As such, we propose a novel extractive text summarization tool, *UnB Language Independent Text Summarizer* (UnB-LITS), which is capable of performing text summarization in a language agnostic manner. The proposed model depends on intrinsic characteristics of the text being summarized rather than its language and thus eliminates the need for language dependent lexicons, databases, taggers or parsers.

Within this tool, we develop an innovative way of coding the shapes of text elements (words, n-grams, sentences and paragraphs), in addition to proposing language independent algorithms that is capable of normalizing words and performing relative stemming or lemmatization. The proposed algorithms and its *Shape-Coding* routine enable the *UnB-LITS* tool to extract intrinsic features of document elements and score them statistically, and identify influential tokens (NE's and CC's) to extract a representative summary independent of the document language.

1.2.2 Deep Learning Classifier

1.2.2.1 Problem Statement

Building a domain agnostic classifier, that achieve an efficient multiple-output classification in a multi-dimensional class spaces, faces many challenges, the most important of which are:

a) Dependencies among classes across the multiple class spaces.

b) Data imbalance, due to the enormous possible combinations of classes in the model's multi-dimensional spaces, which hinders the ability to provide a training dataset that, contains all possible combinations in considerable density.

c) Curse of dimensionality, where in certain domains (especially in text datasets that use bag-of-words), the high dimensionality of data might adversely affect the performance and/or the computational complexity of MDC models.

1.2.2.2 Proposed Solution

For the proposed model to address the aforementioned challenges, it should be capable of modeling class dependencies and manage the problem of data imbalance, especially when classifying future samples with combination of classes seen by the model for the first time.

Moreover, the model should be capable of performing adequate dimensionality reduction, in order to reduce the computational complexity as well as to improve its classification performance especially in highly sparse datasets.

As such, we propose a novel Deep Learning classifier, the *Deep Self-*

organizing Cube, (*DSOC*) that address the aforementioned challenges, and can be used for MDC, MLC, MCC as well as BC.

DSOC can model dependencies among classes in multiple class spaces, while consolidating its ability to classify rare combinations of labels. *DSOC* is formed of two *n*-dimensional components, namely the *Hypercube Classifier* and the multiple *DSOC Neural Networks* connected to the hypercube. The multiple neural networks component is responsible for feature selection, dimensionality reduction and segregation of classes, while the Hypercube classifier is responsible for creating the semantics among multiple class spaces and accommodate the model for rare sample classification. DSOC is a multiple-output learning algorithm that successfully classify samples across all class spaces simultaneously.

1.3 Software

In this thesis, a set of software tools and libraries were used for model building, analysis and data visualization. These tools are: *Jupytar Notebooks* [35] for Python, *Scikit-Learn* machine learning library [36], *MATLAB* R2015a [37], *Voyant* visualization tools [38], and *VBA* of Microsoft Excel [39].

1.4 Contributions of the Thesis

1.4.1 Language Agnostic Text Summarizer

The main contributions of this study are as follows: a) we propose an efficient, language independent text summarizer, named "*UnB-LITS*". b) *UnB-LITS* is an entirely unsupervised model, in terms of extracting influential tokens as NE's and CC's. c) We developed, as well, language agnostic algorithms for stemming and stop words removal. Moreover, d) we proposed an algorithm we called, *Sentence Language Uniformity*, "*SLU*", that can identify sentences, which contain multiple languages, as in the case of scientific papers with Latin expressions, or in case of foreign names, etc.

1.4.2 Deep Learning Classifier

The main contributions of this study are as follows: a) we propose an efficient, yet straightforward multidimensional deep learning classifier, the "DSOC". b) The model we designed has an embedded variable selection layers to achieve dimensionality reduction while increasing the discriminatory power of the model and achieve appropriate class segregation. c) This work experimentally demonstrates the effectiveness of DSOC in all types of classification tasks, regardless the severity of data imbalance or the strength of class dependencies, due to its design that models semantics among classes along different classification spaces even in case of imbalanced datasets.

1.5 Thesis Structure

The rest of the thesis is structured in six more chapters covering all aspects of the research, starting by literature review and related work, followed by presenting the two proposed models, *UnB-LITS* and *DSOC*, and their performance evaluation approach, and finally the study is concluded in chapter 7.

Chapter 2 presents a detailed literature review for text summarization, as well as multi-dimensional classification. The first section, presents an in-depth review for the recent work done in text summarization as well as the challenges facing the summarization tasks. It then focuses on the extractive summarization tasks in particular, presenting the recent state-of-the-art models in each of the five categories of extractive summarization. The second section presents challenges and state-of-the-art models and approaches used to solve MDC tasks.

Chapter 3 presents the language independent text summarization model, *UnB-LITS*, where the chapter is organized in sections and subsection presenting all aspects of the proposed model. Our novel, shape-coding approach as well as our proposed language agnostic stemmer and token normalization algorithm are discussed in details with a working example. The document's elements scoring algorithms and equations are presented in details followed by the summary extraction strategy with hands-on working example of an English news article obtained from DUC02 benchmark dataset.

Chapter 4 presents the performance evaluation of the proposed model. The first section presents the conducted experiment, datasets, evaluation metrics and the compared approaches. The following section shows a detailed case study, as the proposed *UnB-LITS* model was applied on an English news article, where the outcomes of the model are analyzed in details to discuss the effect of shape-coding on summary extraction as well as named entities and concrete concepts identification. The following sections apply *UnB-LITS* on two benchmark news datasets, English and Portuguese, where the obtained results are compared to the state-of-the-art methods reported in literature for the same benchmark datasets. The last section applies the proposed model on Spanish and French datasets to prove the language agnostic nature of the model, the results were compared to those obtained by commercial summarizers.

Moreover, Chapter 5 presents the proposed deep multi-dimensional classifier, *DSOC*. The chapter is divided into sections and subsections that explain in details the model components, namely the n-dimensional *Hypercube Classifier* and the multiple novel *DSOC Neural Networks* connected to the hypercube. The chapter discusses the role of each layer of those neural networks, specifically the rule of the *VSC double layer* (based on our *Variable Strength Coefficient*) in achieving class segregation and strengthening the discriminatory power of the model, and the role of the *Pooling* layers in representing class spaces. The chapter shows how the *Hypercube* can model dependencies among classes in different class spaces. Detailed discussions of all model parameters, as well as the training and classifying processes are presented as well.

On the other hand, Chapter 6 presents the empirical assessment to evaluate the *DSOC* performance on seventeen benchmark datasets. The first subsection in the experiment section presents all features of those datasets, which were selected to evaluate the model performance in all four classification tasks. The following subsections present the evaluation strategy, four standard classification algorithms beside eleven state-of-the-art approaches selected to evaluate the model's performance. The results are then presented and discussed along several aspects in the last section of the chapter.

Finally, chapter 7 concludes the thesis for both proposed models, highlighting the findings and conclusions of each individual study, as well as then future work and research possibilities to extend the applications of both models.

Chapter 2: Literature Review and Related Work

As stated in the introduction chapter that this study introduces two generalized domain agnostic algorithms, the first one is a Language Independent Text Summarizer, "*UnB-LITS*", in the field of NLP, and a domain independent multi-dimensional deep learning classifier, the Deep Self-Organizing Cube, "*DOSC*". As such, this literature review chapter is divided into two sections each of which covers a specific domain.

The first section covers the recent work and state-of-the-art models applied in the field of text summarization with respect to the proposed language agnostic approach. While, the following section presents literature review for different classification approaches used in the field of MDC and the underlying challenges with respect to the proposed straight forward deep learning classifier.

2.1 Automatic Text Summarization

Automatic Text Summarization (ATS) can be divided into three main approaches, *Extractive*, acts on extracting the most influential sentences of the text to be summarized [40]; *Abstractive* depends on semantics to create new representative sentences made of new set of words [41]; and a *Hybrid* approach [42].

Another way to look at the ATS is by considering the dimensionality of the text to be summarized. ATS could be applied for single document summarization, or multiple document summarization, which typically involves summarizing a set of documents belonging to the same topic while maintaining the relevancy and avoiding redundancy [43].

From the architecture viewpoint, El-Kassas et al. [6] has divided ATS into three distinct steps, *Pre-processing*, *Processing* and *Post-processing* as per **Figure 2**. Where, pre-processing step [44] includes segmentation of sentences, tokenization, stemming, lemmatization [45], tagging [46], stop words removal [47], etc. while the processing step means applying the summarization technique itself, finally, the post-processing step focuses on refining the summary by solving problems and facing challenges. On the other hand, **Figure 3** shows a generalized framework for abstractive ATS based on neural networks.

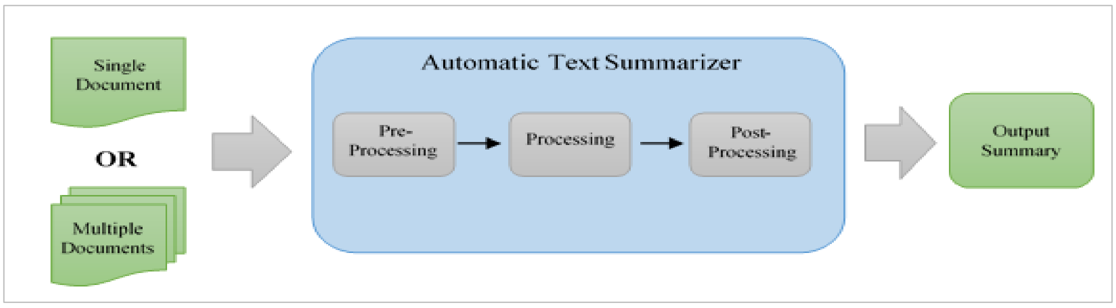


Figure 2. Generalized architecture for automatic text summarizer for a single document or multiple documents [6].

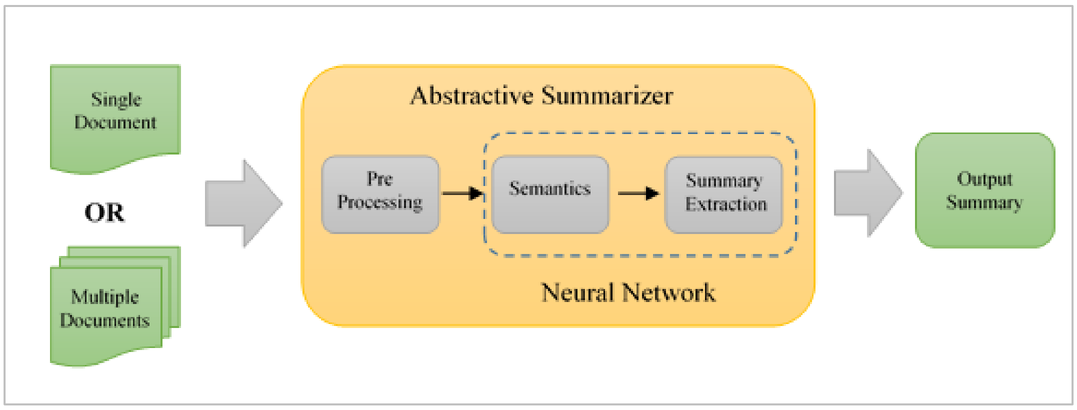


Figure 3. Abstractive summarization framework for Single document or multiple documents.

2.1.1 ATS Preprocessing Tools

Language summarization algorithms typically depends on feature extraction techniques, as stop words removal, stemming, lemmatization, POS tagging, etc. Such techniques are language dependent in nature, which requires the presence of lexicons, parsers and other language specific tools.

Stop words, for instance, are common words that are neither indexed nor searchable in search engine [48], as in English languages, words like “is”, “the”, “in”, and others, also, the words “في”, “و”, “كل”, etc. in Arabic language [49]. Stop words impose noise to NLP models as such, their removal enhance the performance of NLP models significantly.

On the other hand, stemming was introduced by [50], then it was developed through the years, and many algorithms have been developed for specific languages, as Nazief & Adriani stemmer for Indonesian language [51], improved Arabic light-based stemmer [52], in addition to various specialized language dependent lemmatizers [53].

In addition, Part-of-Speech (POS) Tagging [54], which is the process of annotation of tokens in a text, where a word is assigned to a speech class (noun, verb, subject, etc.), has gained growing attention and were implemented in various languages across the globe. As a language-dependent process, recent literature shows that intense work has been done for POS tagging of different languages using wide range of machine learning and deep learning models. For example, Bidirectional Encoder Representations from Transformers (BERT) model was used to build POS taggers for Arabic [55], Croatian [56] and even for ancient languages as Ancient and Byzantine Greek [57]. Moreover, POS taggers were built for indigenous languages as Khasi language spoken by indigenous people of the state of Meghalaya in India, where Conditional Random Field (CRF) method was used to build a Khasi POS Tagger [58].

As deep learning techniques advances, many recent studies in literature have used supervised deep learning models to build language dependent POS taggers [59]. For example, [60] has used deep learning networks, recurrent (RNN) and long-short term memory (LSTM) neural networks, to build a POS tagger for Turkish language, while

[61] has used RNN and LSTM as well to build POS tagger for one of the south Indian languages (Kannada). Moreover, many POS taggers were built based on deep learning models for local languages, as Malayalam (south Indian language) [62], Maithili [63] in addition to national languages as Kazakh [64], Persian [65], Thai [66], and Mongolian [67].

Semi-unsupervised approaches, as well, have benefited from deep learning and its deep neural networks to build POS taggers that can handle rare words, and out-of-vocabulary tokens [68]. For example, [69] has used semi-unsupervised deep learning based on word embedding representation to build POS taggers for Italian and English language.

2.1.2 Challenges in ATS

In general, ATS frameworks, whether extractive, abstractive or hybrid, are less biased and faster in processing than manually generated summarizes due to human bias. However, ATS has its own set of challenges as: a) Minimizing Redundancy, b) Maintaining diversity of topics in hybrid texts, as well as c) generating human readable summary (especially in abstractive ATS), and d) the challenge of Out-of-Vocabulary words (OOV) and repetition.

Many attempts were made to tackle those challenges. Kouris et al. [70] have proposed a framework for human readable abstractive summarization using knowledge-based content generalization and deep learning networks. Moreover, many deep learning approaches especially Long-Short Term Memory (LSTM) have been used to reduce redundancy while maintaining a readable human summary [71], even in complex languages as Arabic language due to its high semantics, syntactical complexity and enormous word derivatives [72]. In general, common deep neural networks (DNNs) as recurrent neural networks (RNN), convolutional neural network (CNN), and graph neural network (GNN) are widely used in abstractive summarization to tackle some of the challenges mentioned above [73].

Many frameworks, in the deep learning domain of abstractive text summarization, are used to tackle the challenge of understanding the text and generate human readable summaries, as sequence-to-sequence framework [74] [75], as well as other encoder-decoder models, as encoder-decoder with basic attention mechanism [76] [77], Hierarchical Encoder-Decoder Models [78], and CNN-Based Encoder-Decoder Models [79].

On the other hand, OOV words and repetition problems are handled in the abstractive summarization tasks through mixed approaches in the deep learning domain. Xu et al. [80] have integrated core word information of the original vocabulary with the traditional attention mechanism to create FCWAM model, stands for Fusion Core Word Attention Mechanism Model, to tackle that problem. Others created datasets specific for particular languages as [81] did for Turkish and Hungarian languages.

The aforementioned techniques has somehow tackled the challenges related to structure and readability, however there is an important challenge that has not been addressed appropriately, which is the challenge of *topic bias*. Where the summary could be biased towards specific subtopics within a document (especially long ones) or in a set of grouped documents (multi-document summarization).

This tradeoff between readability and bias is more prominent in abstractive text summarization and to a lesser extend in extractive summarization. Recent work in literature has faced this problem thorough introducing an unsupervised component in the summarization model. In extractive summarization task, [82], [83] and [84] have used topic modeling, while [85], [86] and [87] have used unsupervised clustering, and [88] has used a combination of both approaches to achieve a proper unbiased summaries.

2.1.3 Categories of Extractive ATS

Various extractive summaries approaches were introduced in literature in the recent years, Gambhir and Gupta [89] have divided extractive summarization approaches into five main categories according to the approach used in achieving the ATS task. Those categories are, a) statistical based, b) Topic based, c) Graph based, d) discourse based and e) machine learning based.

It is worth mentioning that most of these approaches are language dependent as they depend in one or more steps on a language dependent tool (taggers, lemmatizer, stemmers, etc.). Hereafter, we present the recent models introduced in literature in each of these five categories of extractive ATS.

2.1.3.1 Statistical based Approaches

The extracted sentence depends on statistical features of the sentence itself and its containing document rather than its linguistic properties. However, those statistical methods might depend on one or more language dependent tool in the preprocessing steps, as taggers, parsers, lexicons, etc. Many statistical methods have been used for document element scoring, and the subsequent sentence/element selection and extraction [90].

Zhou et al. [91] have integrated the sentence selection and scoring routines into a single end-to-end neural network framework for extractive document summarization using hierarchical encoder.

Some methods utilize single word statistics while others utilize n-grams and other complex combinations of tokens. [92] used word frequency algorithms to extract the main features from paragraphs to achieve summarization on the paragraph level.

However, the work done by [93] is considered a strong base for statistical-based extractive text summarization, where they have applied multiple statistics, optimization and neural networks techniques to score and extract sentence-level features such as sentence position, positive keywords, negative keywords, and more. Their work is extended in recent literature, [94] has introduced Ranksum, an approach based on the rank fusion of sentence features that fused together using weighted scores of topic information, semantic content, significant keywords, and their positions in an unsupervised manner. While [95] has combined the statistical and semantic features with topic modeling for Arabic text summarization.

2.1.3.2 Topic-based Approaches

This approach was first introduced by Lin and Hovey。 They proposed to extract automatically sets of topic signatures of related words, and compute their associated weights as related to the head topics [96]. This approach becomes later a base for a category of extractive text summarization task.

Belwal et al. [97] used a mixed approach of topic-based modeling and the semantic measure within the vector space model to address the challenge of redundancy mentioned earlier. They aimed at extracting the strong sentences that represent the maximum of the embedded topics in the text to be summarized.

Srivastava et al. [98] has combined *Latent Dirichlet Allocation* (LDA) and *K-Medoids* clustering, the first is used to cluster sentences according to topics and the second to choose the most important sentences that form the summary in all subtopics. This model is language dependent as it depends on spaCy’s POS lemmatizer [99]. It is worth mentioning, that LDA was also used by [100] for topic based approach text summarization but in the abstractive ATS tasks.

Moreover, [101] has proposed a topic modeling approach that is applied on lower level entities inside a document, they modeled subtopics at clusters level in a single document, and then they addressed the limitations that might arise using an incorporated statement selection technique.

2.1.3.3 Graph-based Approaches

Since the graph based approach LexRank was introduced by [102], many methods have been presented in literature using the graph-based approach with different document elements graph representation.

Mallick et al. [103] have proposed a graph-based text summarization method using modified TextRank algorithm to constructs a graph with sentences as the nodes and compute their similarities to define the weights of the edges connecting them. It is worth mentioning that TextRank is a graph-based word-ranking model for keyword extraction, and widely used in text processing and summarization in particular [104].

Mohammed and Oussalah [105] have used a modified version of TextRank to build the graph-based text summarizer, where they computed the modified inverse sentence frequency-cosine similarity and used it to assign the weights for graph edges. Their approach differ from the typically used cosine similarity in that it gives different weightage to different words in the sentence, rather than the equal weights assigned by the traditional cosine similarity.

El-Kassas et al. have also introduced an extractive graph-based framework, named EdgeSumm [106], that combines a set of four extractive algorithms, a) graph-based, b) statistical-based, c) semantic-based, and d) centrality-based methods) to benefit from their advantages and overcome their specific drawbacks.

Moreover, [107] has proposed a graph-based method integrated with a text processing tool that maintains semantic relation between sentences. On the other hand, [108] has introduced a mixed approach that integrates graphed-based approach with topic-based one, to create a model that uses the similarity between sentences and the document topic to assign the weight for the edges connecting individual sentences.

2.1.3.4 Machine Learning based Approaches

The machine learning based approach is the one that uses common machine learning algorithm, mostly classifiers, to achieve the summarization task through clustering or classifying the document elements into "*includeInSummary*" or "not(*includeInSummary*)". The used machine-learning algorithms in this approach include Support Vector Machines [109], Naïve Bayes [110], Decision Trees [111], logistic regression [112], etc.

Moreover, deep learning networks have been applied under this approach to achieve the summarization task. For example, Bae et al. [113] has used reinforcement learning through combining BERT based extractor and LSTM pointer network to achieve a hybrid extractive/abstractive summarization.

Ma et al. [114] have incorporated BERT and LSTM with word embedding to build a hybrid model, *T-BERTSum*, which utilizes the topic-based and machine-learning-based approaches to generate a topic-aware extractive and abstractive summary. While, Grail et al. [115] have proposed a hierarchical propagation layer to overcome the limitations of BERT on summarizing long documents.

2.1.3.5 Discourse-based Approaches

On the other hand, since the introduction of Rhetorical Structure Theory in the domain of computational linguistics in by Mann and Thompson in 1988 [116], many Discourse-based applications in the field of computational linguistics have been introduced [117].

In the field of text summarization, Discourse-based summarizations models have been introduced in literature; such approaches represent the discourse in a document as a tree and focuses on the rhetorical connections between the text elements as in [118], [119] and [120] for extractive summarization tasks, and [121], [122] in case of abstractive summarization.

2.2 Multi-dimensional classification

Bogatinovski et al. [123] pointed out the increasing interest in MLC task from the machine learning community, where they displayed a graph that showed an exponential growth in the number of scientific papers published throughout the last decade. This section presents the recent work in literature that tackles the classification problem from a multi-way perspective and its recent applications.

As mentioned in the introduction section, MDC is a generalized form of multi-label classification problem. Therefore, we are going to use the term MDC throughout the thesis when referring to all types of non-binary classification problems, unless otherwise specified.

2.2.1 Binary Relevance Approach

Despite the inability of Binary Relevance (BR) approach to model class correlations and dependencies, nevertheless BR is one of the most widely used approaches for MDC due to its simplicity and intuitiveness. BR techniques are built in one of two structures, *Chaining Structure* and *Stacking Structure*, or a mix of them. In *Chaining Structure*, independent binary classifiers are arranged in a chain order based on the results of the previous classifiers [124]. While in *Stacking Structure*, a set of meta-level BR models are stacked over another set of base-level BR models, where each meta-level binary classifier is built upon the predictions of all base-level ones [125].

Many variations of BR were introduced lately in an attempt to overcome the BR core limitations, some of which are: a) BR Stacking based on Pareto Optimum, a modified version of the staking method that takes into consideration the inherit nature of class labels and their own related subsets [126]. b) Dependent binary relevance (DBR), which is a mixed approach of chaining and stacking [127]. c) Stacking Model with Label Selection (SMLS), a two layer stacking based approach that overcomes limitations of DBR, through intensifying the class correlations in subsets to augment the features space [128].

2.2.2 Feature Manipulation approaches

While, most MDC approaches tends to model multiple class dependencies in the output class spaces, Jia and Zhang [129] tried a different approach by manipulating the input features space by enriching the existing original feature space with a set of new features. Their augmented feature approach, called *KRAM*, uses simple counting statistics and weighted *k*NN techniques (with extra bias terms) depending on the class membership of k nearest neighbors in the training set. *KRAM* showed good results when compared to other models that use original input features space. Both Jia and Zhang further extended their feature-augmentation strategy in [130] by introducing SFAM, an abbreviation for Selective Features Augmentation for Multi-dimensional classification. SFAM synergizes multiple kind of augmented features (standard *k*NN, weighted *k*NN and maximum margin techniques) to achieve classification along different dimensions. Furthermore, Wang et al. [131] proposed a deep neural network based model that integrates the Feature Augmentation and Label Embedding techniques to model the inter-class correlations and the intra-class exclusiveness in MDC problems.

In a related context, and to shed the light on the importance of features extraction, De Handschutter et al. [132] published a thorough survey on deep matrix factorization (deep MF) in comparison with constrained low-rank matrix approximations (CLRMA) to deal with the extraction of several layers of features as a principal step before conducting further machine learning tasks.

2.2.3 Challenges in MDC

As mentioned earlier in Chapter 1, class imbalance problem might affect the classification accuracy of a MDC model as well as its performance evaluation metrics [133]. Feature space manipulation can tackle the imbalance problem of MDC. Mishra and Singh [134] proposed a method called Feature Construction and Smote-based Imbalance handling (FCSMI), to tackle the problem of class combinations imbalances in MLD. FCSMI uses the distance between minority classes and others to alter the feature space and achieve the balance between minority and majority classes.

In addition, [135] proposed a Multi-Kernel Multi-Label (MKML) method to address, simultaneously, both class dependencies and class imbalance problems in MLC. While, [136] presented Partial Label Masking (PLM) method to tackle imbalanced datasets by partially masking major and minor classes and then continually adapts the target ratio based on the output probabilities, aiming at improving precision on frequent classes and recall on less frequent ones. In image classification, [137] proposed a new loss function that reduces the risk of misclassification of less frequent classes, by neutralizing the probability distribution of incorrect classes leading to a more robust classification of class-imbalanced scenarios. It is worth mentioning that the degree of class imbalance and its impact on MDC can be assessed by using either the *imbalance-ratio* or the *imbalance-degree* coefficient presented by [138], which is more sensitive in reflecting the skewness in MDC distributions.

2.2.4 MDC Applications

In recent years, many applications in different domains have benefited from MDC techniques. In the field of medicine, Yang [139] constructed a MDC model based on Support Vector Machine (SVM) for the diagnosis of schizophrenia and bipolar disorder. While in text MDC, Xie et al. in [140] proposed a multi-dimensional relation model to incorporate relations between dimensions for dimension score prediction in multi-dimensional sentiment analysis (valence-arousal-irony, VAI) of a Chinese corpus. Other applications in the field of text classification include Fake news classification using LSTM and BERT [141], emotions classifications using LSTM and Transformer Networks (RoBERTa and DistilBERT) [142].

Chapter 3: A Language Agnostic Text Summarization Model (*UnB-LITS*)

The idea of the research is to build a language agnostic summarizer, that we call "*UnB-Language Independent Text Summarizer*" or "*UnB-LITS*" for short. A summarizer that is capable of efficiently summarize text documents without any prior knowledge of the language(s) of the text being summarized. The *UnB-LITS* performs Extractive Summarization task that depend on extracting strong sentences that best represent the document being summarized.

In order to achieve such a goal, the core model of the *UnB-LITS* depends on extracting sets of features related to the document's hierarchal elements namely paragraphs, sentences and words. These sets of features are combined together to obtain the *overall sentence score* (*SC*) for sentences used to build the extractive summaries.

As such, the core model depends on extracting strong sentences that, in turn, consists of strong words and expressions (*n-grams*). In order to achieve this without any prior knowledge of the language of the text to be summarized, we propose a novel technique that converts words into a standard set of codes, we called "*Shape Codes*", which can then be quantified and used to assess the strength of a sentence. Shape Codes are applied for both: a) the main document elements (paragraphs, sentences and words), as well as, b) the extracted *n-grams*. The proposed technique then computes the relative weights of those "*encoded shapes*" and combine them with other features to determine the overall score of each element in the text.

In order to achieve an efficient language independent text summarizer, it is required as well to construct a robust algorithm that is totally language agnostic along all its sequential steps. In other words, basic NLP steps of stop word removal, stemming or lemmatization, named entities extractions, etc. should all be carried out using language agnostic procedures. As such, in this study we propose a chain of language agnostic routines to achieve an efficient meaningful language independent extractive summary.

In the following sections, the algorithms behind Shape-Coding as well as weights computation are explained in details with working example form a benchmark dataset. Section 3.1 explains the hierarchal document elements and how they can be linked together to achieve efficient summarization. The following sections explain the proposed language agnostic pre-processing techniques, followed by the shape-coding techniques for all the four document elements, and finally the algorithms for computing the overall score for each element in order to extract the automatic summary.

3.1 Document Elements

The Semantically Annotated LaTeX project (*SALT*) [143] has divided the semantic organization of a document, while preparing their sets of ontologies, into three layers: structural layer, rhetorical layer and finally the annotation layer that links the rhetorical characterizations with the structural components of the other two layers. While the rhetorical layer is based on the meaningful parts of a document [144], the structural layer is the one containing sentences, paragraphs, and other elements of a text document [145].

In this study, we focused on the structural of document (or a piece of text), where a document can be seen as a hierarchal structure that consists of four elements: *paragraphs*, *sentences*, *n-grams* and *words* as seen in see **Figure 4.**

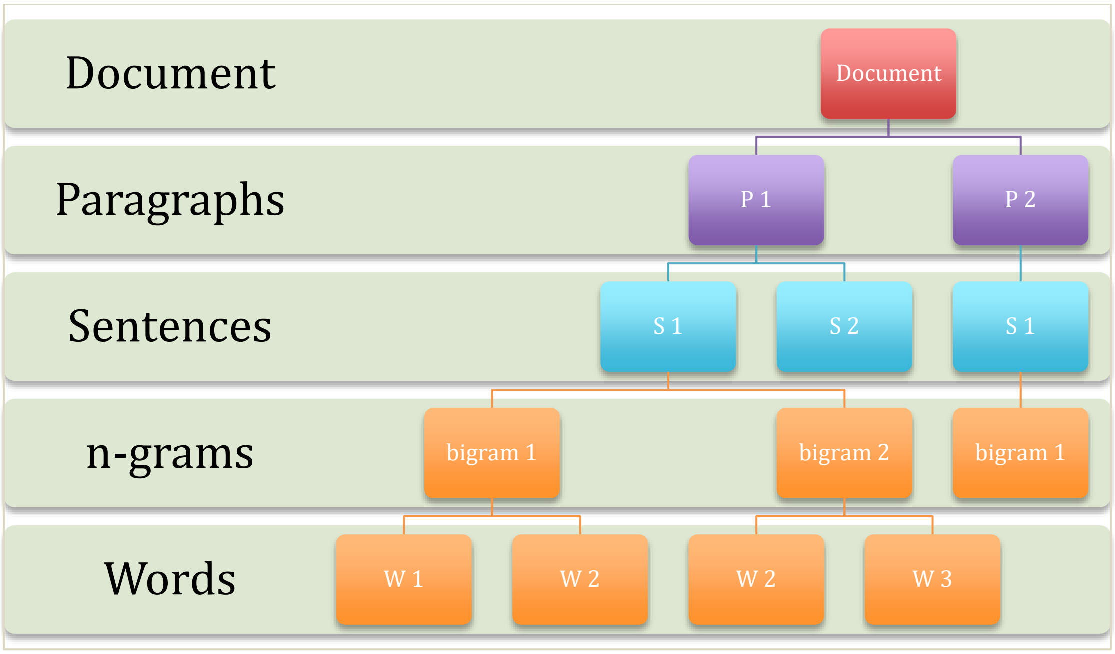


Figure 4. The hierarchal relation between document elements

Throughout the study, the following notation is being used. Where, A document D^j is considered as a set of paragraphs $D^j = \{P_1^j, P_2^j, P_3^j, \ldots, P_k^j\}$, where k is the total number of paragraphs in the document. On the other hand a single paragraph P_k^j consists of a set of sentences $P_k^j = \{S_1^{j,k}, S_2^{j,k}, S_3^{j,k}, \ldots, S_m^{j,k}\}$, where m is the total number of sentences in the paragraph. As the hierarchal relation goes deeper, a sentence $S_m^{j,k}$, in turn, consists of a set of words $S_m^{j,k} = \{W_1^{j,k,m}, W_2^{j,k,m}, \ldots, W_n^{j,k,m}\}$, where n is the total number of words in a sentence.

A group of consecutive words in a sentence can be grouped together to form what is called n-grams, where, a bigram is formed of two words $W_1^{j,k,m}, W_2^{j,k,m}$ or $W_2^{j,k,m}, W_3^{j,k,m}$, in addition a tri-gram consists of a sequence of three consecutive words $W_1^{j,k,m}, W_2^{j,k,m}, W_3^{j,k,m}$ or $W_2^{j,k,m}, W_3^{j,k,m}, W_4^{j,k,m}$. As such, a single sentence can also be seen as a collection of different size n-grams. Definitions of the four document elements, top down, used in this study are listed below with respect to the aforementioned notations.

3.1.1 *Paragraphs*

A Paragraph is a brief piece of text that consists of at least one sentence that is usually describing a single topic. A typical paragraph is formed of three main parts: *i) Topic sentence,* that reflects the main idea (usually at the beginning of the paragraph); *ii) Supporting sentence*, to explain or support the topic sentence, *iii) Concluding sentence,* a brief summary for the main idea. [146]

However, some paragraphs may consist of a single sentence that, in certain cases, might reflect its importance, as in document titles, section headings, numbered items, etc. In this study, the relative length of the sentence is one of the factors affecting its overall score.

In addition, the proposed shape coding technique depends on other paragraph formatting characteristics as, initial marks (numbered, bulleted), font size, etc. as well as other features related to the shapes of its individual words.

3.1.2 *Sentences*

Being the building unit of paragraphs, a sentence is considered the grammatical unit that is formed of one or more words that expresses an independent statement, question, request, command, exclamation, etc. A single sentence typically has a subject as well as a predicate, and starts with a capital letter and ends with the appropriate punctuation. [147].

As for shape coding of paragraphs, sentence shape coding and its subsequent scoring will depend its formatting characteristics as size, font, punctuation, as well as on its position in the paragraph and the scores of its individual words.

3.1.3 *N-grams*

An *n-gram* is a sequence of *n* words from a given sequence of text, where *n* is the size of the window that forms the sequence of words [148]. To create *n-grams,* the *n* sized window starts from the word at position one (creating the first n-gram from a sequence of *n* words) then moves one word forward to create the new *n-gram*, and so on. This process of moving *n*-sized window results in creating a set of *n-grams* for each sentence. When $n = 2$, then the resultant sequences of words are called *bigrams,* while $n = 3$ produces *trigrams*, subsequently $n > 3$ produces a list four-grams, five-grams and so on. It is worth mentioning that a single word, i.e. $n = 1$, can be referred to as *unigram*.

For example, for the sentence "***Egypt is a country linking northeast Africa with the Middle East and it dates to the time of the pharaohs***"

If $n=2$ then the list of *bi-grams* would be: {*Egypt is, is a, a country, country* *linking, linking northeast, northeast Africa,.., the pharaohs*}, while if *n=3* then the list of *tri-grams* would be: {*Egypt is a, is a country, a country linking, country linking northeast, .., of the pharaohs*}.

The total number of *n-grams* that can be extracted from a sentence *m* that consists of *N* words can be determined using equation 3.1 below:

$$nGrams\ Count = N - (n - 1) \tag{3.1}$$

As such, using the example mentioned above the total number of bigrams that can be extracted from that sentence is: $bigrams\ Count = 20 - (2 - 1) = 19$.

3.1.4 Words

A word is a single distinct meaningful element of speech or writing, used with others (or sometimes alone) to form a sentence and typically shown with a space on either side when written or printed. [149]. A single word can be considered as any segment of written or printed discourse ordinarily appearing between spaces or between a space and a punctuation mark [150].

In the proposed technique, the feature weights of a word depend on its *shape code* as well as its position in the sentence and its frequency in the document to be summarized.

3.2 Shape Coding

Shape-Coding is the main feature used in the proposed model of the *UnB-LITS* tool. The idea of *shape coding* is to extract the main features of a document element and code them in a simple way to reflect these features using a compact and intuitive sequence of letters. *Shape-coding* takes into consideration both the nature of the characters forming a document element (numbers or letters), the case of that element (capital, lower case or mixed) as well as its format (font size, bold, italic, etc.).

This proposed technique of *shape-coding* is the main part of the algorithm responsible for the language independent nature of the entire model.

With reference to the document elements mentioned earlier, three *shape-coding* techniques are proposed in this thesis: a) *shape-coding* of words and n-grams, b) *shape-coding* of sentences, and c) *shape-coding* of paragraphs.

These *shape-coding* techniques give the model the ability to extract the most powerful and influential words, n-grams, Named Entities (NE's), Concrete Concepts (CC's), key phrases and sentences. Where, combing the n-gram shapes with their frequency of occurrence can lead to the identification of NE's and CC's in language agnostic way. Those techniques are explained in details in the following sub-sections:

3.2.1 *Shape-Coding of Words and N-grams*

Words are the building blocks of a document or text. As such, coding the shape of a word will be reflected, directly, in coding its parent *n-grams* and, indirectly, in coding the containing sentences and paragraphs. *Shape-coding* of a word means transferring each character in the word to its corresponding code in a process that results in a compact word in a limited set of codes, which in turn, reflects the word's important features.

In case of *shape-coding* of words, the *code set* consists of 5 elements (or letters) that are used to encode the word features. This elements of the *code set* are {X, x, C, c, N, n}. The indication of each code is explained in **Table 1** below:

Table 1. Set of codes used in shape-coding of words and n-grams.

Code Element	*Indication*
X	Indicates a single capital letter.
x	Indicates a single lower case letter.
C	Indicates 1 or more capital letters.
c	Indicates 1 or more lower case letters.
N	Indicates a single numeric character.
n	Indicates 1 or more numeric characters.

Shape-coding of a word is done in four main steps:

i. Remove all non-alphanumeric characters as: {. , " / & ; : @ etc.}.

ii. Change all numeric characters to "N".

iii. Change all letters to "X" and "x" for capital and lower case letters respectively.

iv. Group sequential repeated codes using "C", "c" and "n" for repeated "X", "x" and "N" respectively. In other words, "C", "c" or "n" are used to replace a sequence of similar characters of size ≥ 2. Where, the first character of the identical sequence is kept unchanged while all the following similar characters are replaced with one of three continuity codes, "C", "c" or "n", to encode the continuity of the same shape code.

For example, "XXX" is grouped using C to be "XC", while "xxxx" and "NNNNNN" are grouped using "c" and "n" resulting in "xc" and "Nn" respectively.

Using the aforementioned steps for *shape-coding* of words, below is a couple of examples showing the shape-coding technique in step by step manner.

Example 1: The word "Egypt" can be coded as follows:

a) Encode the five letters into "X" and "x" according to the letter case, "E" → "X", "gypt" → "xxxx".
Therefore, the first step results in "Egypt" → "Xxxxx".

b) Group the sequence of similar codes using the continuity codes "C" and/or "c". Since "X" is not followed by any other capital "X" thus, no grouping is performed. While, "gypt" resulted in a sequence of 4 lower case x's, "xxxx", that can be grouped by maintaining the first x and replacing the other 3 x's with "c", i.e. "xxxx" → "xc".

c) Thus the final result of shape-coding of "Egypt" is "Xxc".

Example 2: The word "mRNA" can be coded as follows:

a) Encode the four letters into "x" an "X" according to the letter case, "m" → "x", "RNA" → "XXX". By the end of this step, the temporary code for "mRNA" is "xXXX".

b) Use the continuity codes to group the sequence of similar codes using "C" and/or "c".

   In the first part of the word, since there is no other lower case x's following the first "x" thus, no grouping is required. While, for the second part, "RNA", there is a sequence of 3 capital X's, "XXX", that can be grouped by maintaining the first X and replacing the other 2 X's with "C", i.e. "XXX" → "XC".

c) Thus the final shape-coding of "mRNA" is "xXC".

Example 3: The word "UnB" can be coded as follows:

a) Change letters into "x" an "X" according to the letter case, "U" → "X" , "n" → "x" and "B" → "X" Therefore the first step results in "UnB" → "XxX"

b) Since there is no sequence of consecutive X's or x's, as such, no grouping can be performed.

c) The final code of "UnB" is "XxX".

Example 4: The number "1,027,708" can be coded as follows:

a) Remove the non-alphanumeric characters. Thus, "1,027,708" → "1027708".

b) Change all numbers into "N". Therefore, "1027708" → "NNNNNNN"

c) Group sequence of similar codes using the continuity code for numbers, "n", so that "NNNNNNN" is converted into "Nn".

d) As such, the result of *number shape-coding* of "1,027,708" is simply "Nn".

Other examples:

a) Coding the word "game":
game → xxxx → xc

b) Coding abbreviations, like "USA" or "U.S.A."
USA → XXX → XC
U.S.A. → USA → XXX → XC

c) Coding the proposed tool UnB-LITS
UnB-LITS → UnBLITS → XxXXXXX → XcXC

d) Coding words containing alphanumeric characters, like "2-way"
2-way → 2way → Nxxx → Nxc

e) Coding decimals as "25.44"
25.44 → 2544 → NNNN → Nn

f) Coding single digit number, like "9",
9 → N → Nn

This is the only exception in grouping routines, where a word consisting of a single character is mapped to a 2 characters word. For example, a single digit number is mapped to "Nn" instead of "N" and a single letter word like "a" is mapped as well to "xc". This exception is allowed in order to prevent giving higher weights for single character words as discussed later in this chapter.

As seen from the examples above, *shape-coding* of words results in encoding those words into a small set of equivalent classes that represent their shapes. For example, all numbers are normalized to "Nn", while the majority of words in the texts are converted to "xc". On the other hand, names of persons, cities are converted to "Xxc" that is less common as compared to "xc".

In addition, the step of removing non-alphanumeric characters from a word, adds more power to the proposed model as it helps in normalizing similar words especially in case of abbreviations. For example, words like "USA" or "U.S.A." referring to the United States of America will be treated in our model in the same way, as both words are encoded into the same shape-code "XC".

The rareness of a word shape in a document reflects its importance. Using the same examples stated above, a capitalized word as "USA" has a word shape "XC". Moreover, a rare word like the name of our proposed tool "UnB-LITS" has even more rare shape-code, "XcXC".

Both shapes "XC" and "XxXC" are rarer in a document than all other common words (verbs, nouns, adjectives, etc.) that will typically be encoded into the most abundant shape-code, "xc".

In addition, words encoded into rare shape-codes, are most likely to be a Named-Entity (name of a person, city, country, tool, abbreviations, etc.).

As a conclusion, *shape-coding* of words results in:

i. Mapping words into a small set of equivalent classes.

ii. Normalizing numbers and similar words into similar shape-codes.

iii. Identifies important words and NE's.

On the other hand, *Shape-coding* of n-grams is performed simply by concatenating the shapes of the n-gram's individual words. Where, a *bigram* can be shape-coded by encoding its two individual words and then concatenate them together using a space delimiter. See **Table 2** for some examples of shape-coding of n-grams**.**

In addition, words or n-grams format can also be encoded easily using the same philosophy of the proposed shape-coding technique. Where different word formats can be encoded simply by applying the same format of the word (bold, italic, etc.) on the encoded shape.

For example, the word "***Brazil***" with bold and italic formatting will have a shape-code "***Xxc***", i.e. the shape-code is formatted in bold and italic as its parent word.

The rareness of formatted shape-codes reflects the importance or the influence of the original words, as the in the case of: "*UnB-LITS*" → *XcXC*.

Table 2. Example for *shape-coding* different n-grams.

n-gram Class	*n-grams*	*n-grams Coded Shape*
Bigram	Middle East	Xxc Xxc
	school bus	xc xc
	100 mph	Nn xc
Trigrams	Republic of Ireland	Xxc xc Xxc
	189 square feet	Nn xc xc
	He plays football	Xx xc xc
4-grams	linking northeast Africa with	xc xc Xxc xc
	Arab Republic of Egypt	Xxc Xxc xc Xxc
	United States of America	Xxc Xxc xc Xxc
5-grams	the Federative Republic of Brazil	xc Xxc Xxc xc Xxc
	Egyptian Minister of Foreign Affairs	Xxc Xxc xc Xxc Xxc
	BFC bought 88% of BankAtlantic	XC xc Nn xc XxcXxc

3.2.2 *Shape-Coding of Sentences*

Shape-coding of a sentence means transferring its main features into a representative code in a process that results in a single compact code reflecting the sentence's important features.

Sentence *shape-coding* is much simpler than word's *shape-coding*, where the *code set* consists of only 2 elements (or letters) that are used to reflect the sentence features. This *code set's* elements are {Z, z}. The indication of those codes is explained in **Table 3** below:

Table 3. Sentence Code Set.

Code Element	Indication
Z	Indicates a sentence with a capital initial letter.
z	Indicates a sentence with an initial lower case letter.

Shape-coding of a sentence is done in the following main steps:

i. Before start encoding a sentence, it is worth mentioning that, in contrast to the word's shape-code, the sentence shape-code has a fixed length of three codes whatever the length of the sentence or the shapes of its individual words are.

ii. If the sentence starts in a word with a capital first letter then add "**Z**" to the beginning of the code, while if it starts with a lower case letter (which is very rare) then add "**z**" to the beginning of the shape-code.

iii. Count the number of words that starts in lower case letter (*L*) and the number of other words that starts with an upper case letter (*U*).

iv. Calculate the ratio of words with initial capital letter in the sentence to its total number of words and then decide the second part of the *shape-code* as seen in equation 3.2 below.

$$ShapeCode = \begin{cases} ZZ & if \ \frac{U}{L+U} \geq \omega_1 \\ Zz & if \ \frac{U}{L+U} < \omega_1 \ and \ \geq \ \omega_2 \\ zz & if \ \frac{U}{L+U} < \omega_2 \end{cases} \qquad 3.2$$

Where, ω_1 and ω_2 are the code-shaping thresholds that are decided by the model designer. In this study, the default values of these thresholds are $\omega_1 = 0.7$, while $\omega_2 = 0.4$.

v. Suppose we have a sentence that starts with a word with initial capital letter, therefore the first code is "Z", then coding the rest of the sentence will be performed as follows:

a) If the ratio of words that starts in capital letter is more than 0.7 then the code part "**ZZ**" will be added to the shape-code to be: "**ZZZ**" which indicates that the sentence is almost fully capitalized (usually in document titles).

b) While, if the ratio is more than 0.4 and less than 0.7 then the code part "**Zz**" will be added to the shape-code to be: "**ZZz**" which indicates that the sentence has mixed case words in considerable amount.

c) On the other hand, if the ratio of words that starts in capital letter is less than 0.4, then the code part "**zz**" will be added to the shape-code to be "**Zzz**", that implies that the sentence is mostly lower case, similar to the majority of sentences in English text.

vi. If the sentence starts with a word whose first letter is lower case, then the sentence can be encoded as follows: "**zzz**", "**zZz**" or "**zZZ**" which have the same meaning discussed above with the exception that the sentence starts in a lower case letter.

vii. In cases where the sentence starts with a number, then consider the number as an upper case letter, as such, the *code-shape* of the sentence will start with "***Z***".

Using the aforementioned steps for *shape-coding* of sentences, we present below a couple of examples showing the shape-coding technique in a step by step manner.

Example 1: The sentence "***Egypt is a country linking northeast Africa with the Middle East and it dates to the time of the pharaohs***", will be encoded as follows:

a) It starts with the word "***Egypt***" that has a capital letter in the beginning. Thus, the shape-code starts with "**Z**"

b) The total number of words that starts with a capital letter (*U*) = 4, while the count of words that starts with lower case letters (*L*) = 16.

c) Ratio of *U* to the total number of words (*U* + *L*) = 0.2, thus the sentence is considered to be mostly lower case, so we add "**zz**" to the coded shape.

d) As such, the final *shape-coding* of the sentence is "**Zzz**".

Example 2: The title sentence of this study: "***Language Independent Text Summarizer and Deep Self Organizing Cubes"***, is encoded as follows:

a) It starts with the word "***Language***" that has a capital letter in the beginning. Thus, the shape-code starts with "**Z**"

b) The total number of words that starts with a capital letter (*U*) = 8, while the count of words that starts with lower case letters (*L*) = 1.

c) Ratio of *U* to the total number of words (*U* + *L*) = 0.88, thus the sentence is considered to be upper case sentence, thus add "**ZZ**" to the encoded shape.

d) As such, the final *shape-coding* of the sentence is "**ZZZ**".

Example 3: The sentence "***In this thesis we propose UnB Language Independent Text Summarizer for language agnostic modeling"***, is encoded as follows:

a) It starts with the word "***In***" that has a capital letter in the beginning. Thus, the shape-code starts with "***Z***"

b) The total number of words that starts with a capital letter (*U*) = 6, while the count of words that starts with small letters (*L*) = 8.

c) Ratio of *U* to the total number of words (*U* + *L*) = 0.43, thus the sentence is considered to be a mixed case sentence, thus add "***Zz***" to the coded shape.

d) As such, the final *shape-coding* of the sentence is "**ZZz**".

As seen from the examples above, *shape-coding* of sentences results in encoding those sentences into a small set of equivalent classes that represent their shapes. The rareness of a sentence shape in a document reflects the importance of that sentence. For example a sentence like those of examples 2 and 3, with coded shapes "**ZZz**" and "**ZZZ**" respectively, are much more rare than sentences with coded shape "**Zzz**" which reflects their importance and therefore, should receive higher weights.

3.2.3 *Shape-Coding of Paragraphs*

Shape-coding of a paragraph means encoding its main features into a representative code in a process that results in a single compact code reflecting the paragraph's important features.

The paragraph's *code set* consists of 7 elements (or letters) that are used to reflect the paragraph features. The *code set's* elements are {B, N, O, S, M, P, p}. Indications of those codes are listed in **Table 4** below:

Table 4. Paragraph Code Set.

Code Element	Indication
P	Indicates a paragraph with a capital initial letter.
p	Indicates a paragraph with an initial lower case letter.
N	Indicates Numbered Paragraphs.
B	Indicates Bulleted Paragraph.
O	Indicates Ordinary Paragraph (neither numbered nor bulleted).
S	Indicates a Single Sentence Paragraph.
M	Indicates Multiple Sentences Paragraph.

Before start encoding a paragraph, it is worth mentioning that, alike the word's shape-code, the paragraph shape-code has a fixed length of five codes whatever the shape or the size of the paragraph is. The five-length encoded shape has a pre-defined fixed position for every main feature in the paragraph. Where, the first letter in the code is reserved for bullets and numbering, the second letter is for the number of sentences forming the paragraph, while the last three letters are for the spread of words that starts with upper and lower case letters throughout the encoded paragraph.

Shape coding of a paragraph is performed as per the following steps:

i. The first letter in the paragraph's coded-shape indicates whether the paragraph is *numbered*, *bulleted* or *ordinary* paragraph. Therefore, the first letter in the code is either "***N***", "***B***" or "***O***" respectively. Where:

a) If the paragraph starts with a numbered list formed from real number (1, 2, 3, etc.), letters (a, b, c, etc.) or roman numerals (i, ii, iii, etc.) then the first letter in the coded shape is encoded as "***N***".

b) While, if the paragraph starts with bulleted list (−, •, ▯, ■, etc.) then the first letter in the coded shape will be "***B***".

c) However, if the paragraph is an ordinary paragraph that does not start with numbered or bulleted system, then the first letter in the coded shape is set to "***O***".

ii. The second letter in the paragraph's coded-shape indicates how many sentences form the paragraph. Then, the second letter in the code is either "***S***" or "***M***". Where:

a) If the paragraph consists of a single sentence (like document titles, subsection titles, etc.), then the second letter in the coded shape is set to "***S***".

b) On the other hand, if the paragraph consists of multiple sentences (like the body of typical text paragraphs), then the second letter in the coded shape is encoded to "***M***".

iii. Moreover, if the paragraph starts in a word with capital first letter then add "***P***" to the third position of the code, otherwise add "***p***" in case it starts with a lower case letter.

iv. Count the number of words that start in lower case letter (*L*) and the number of others that start with an upper case letter (*U*) in the entire paragraph.

v. Calculate the ratio of words with initial capital letter in the paragraph to its total number of words and then decide the last two letters of the *shape-code* as per equation 3.3 below.

$$ShapeCode = \begin{cases} PP & if \ \frac{U}{L+U} \geq \omega_1 \\ Pp & if \ \frac{U}{L+U} < \omega_1 \ and \ \geq \omega_2 \\ pp & if \ \frac{U}{L+U} < \omega_2 \end{cases} \qquad 3.3$$

Where, ω_1 and ω_2 are the code-shaping thresholds similar to those used in sentence *shape-coding*. Such thresholds are decided by the model designer. In this study, the default values of these thresholds were set to 0.7 and 0.4 for ω_1 and ω_2 respectively, similar to the threshold values used for sentence encoding.

vi. Defining the last two letters in the paragraph coded-shape can be done as follows:

a) If the ratio of words that starts in capital letter is more than 0.7 then the code part "***PP***" is added to the shape-code. Therefore, the last three letters in the code are "***PPP***", which indicate that the paragraph is almost fully capitalized.

b) While, if the ratio is more than 0.4 and less than 0.7 then the code part "***Pp***" is added to the shape-code. Therefore, the last three letters in the paragraph code are set to "***PPp***", which indicate that the paragraph has mixed case words in considerable amount.

c) On the other hand, if the ratio of words that starts in capital letter is less than 0.4, then the code part "***pp***" is added to the shape-code. Therefore, the last three letters in the code will be "***Ppp***" showing that the paragraph is mostly lower case, similar to the majority of paragraphs in an English text.

vii. Moreover, if the paragraph starts with a word whose first letter is lower case, then the last three letters in the paragraph shape code can be: "***ppp***", "***pPp***" or "***pPP***" which have the same meaning discussed above with the exception that the paragraph starts in a lower case letter.

viii. In cases where the paragraph starts with a number that is not a numbered list, then consider that number as an upper case letter. As such, the third letter of the *code-shape* of the paragraph will start with "***P***".

Using the aforementioned steps for *shape-coding* of sentences, below is a couple of examples showing the shape-coding technique in a step by step manner.

Example 1: The paragraph "***4. Results and Discussions***", can be coded as follows:

a) It starts with a numbered list "***4.***". Thus, the shape-code starts with "***N***".

b) It is a single sentence paragraph. Therefore, the second coded letter is set to "***S***".

c) Following the numbered list, the paragraph starts with the word "***Results***" that has a capital initial letter. As such, the third letter of the shape-code will be "***P***" in upper case.

d) The total number of words that starts with a capital letter (*U*) = 2, while the count of words that starts with lower case letters (*L*) = 1.

e) Ratio of *U* to the total number of words (*U* + *L*) = 0.75, thus the paragraph is considered to be mostly upper case, thus add "***PP***" to the coded shape.

f) As such, the final shape-coding of the paragraph is "***NSPPP***".

Example 2: The paragraph "***In this thesis we propose UnB Language Independent Text Summarizer for language independent modeling. Also, Deep Self Organizing Cubes is introduced for multidimensional classification.***",is encoded as follows:

a) It starts with neither a numbered nor a bulleted list. Thus, the shape-code starts with "***O***", i.e. ordinary paragraph.

b) It is a two-sentence paragraph. Thus the second coded letter is "***M***"

c) It starts with the word "***In***" that has a capital letter in the beginning. Thus, the third letter in the shape-code is set to "***P***".

d) The total number of words that starts with a capital letter (*U*) = 11, while the count of words that starts with lower case letters (*L*) = 13.

e) Ratio of *U* to the total number of words (*U* + *L*) = 0.46, thus the paragraph is considered to be a mixed case paragraph, thus add "***Pp***" to the coded shape.

f) As such, the final *shape-coding* of the paragraph is "***OMPPp***".

As seen from the examples above, *shape-coding* of paragraphs results in encoding those paragraphs into a small set of equivalent classes that represent their shapes.

As in the case of words and sentences, the rareness of a paragraph shape in a document reflects its importance. For example, a single sentence paragraph with mostly capitalized word initials "***OSPPP***" is more probably represents a title, where normal paragraphs in the text will have a shape like "***OMPpp***". In other words, the rareness of "***OSPPP***" reflects its importance, which indicates that it should receive higher weight while computing the final score.

3.3 Features Extraction

Features extraction is the process of identifying the main properties of a specific element in a document. The extracted features will affect the overall score of that element. Since the proposed model is based on a *language independent* approach, therefore the properties or features to be extracted should not depend on any prior knowledge of the language of the text to be summarized with all of its etymology, grammatical, semantic or syntactic relations.

As presented in the literature review and related work in Chapter 2:, most of text summarization techniques are language dependent especially in the preprocessing routines. Those language dependent preprocessing steps include stop words removal [151], lemmatization or stemming [152], Part-of-Speech (POS) Tagging [153], and other language dependent preprocessing routines. These techniques require prior knowledge of the language of the text, to utilize the appropriate databases or lexicons (stop words, dictionary, etc.), knowledge bases and hand coded rules for stemming and lemmatization [154], as well as parsers and taggers (POS taggers, etc.) [155].

The proposed model is an unsupervised text summarizer that is totally language agnostic and hence independent of any external databases or parsers (that by definition are language dependent). As such, one of the main contributions of this study is proposing a language agnostic stop words removal algorithm as well as a totally language independent stemmer.

Below we explain the process of extracting intrinsic properties of document elements, which are related to the frequency of occurrence of their forms as well as their coded shapes.

3.3.1 *Words Scoring and Features Extraction*

Words are the main building blocks of all other document elements in the proposed model starting from *n-grams* up to sentences and paragraphs. Extraction of word features is a process of extracting statistical properties that depend on the word form (spelling) as well as its coded shape, and give them a weighted score. A word score is a combination of its extracted feature weights.

The score given to higher document elements depends heavily on the scores of individual words, which form that element. As such, word feature extraction and its subsequent scoring is the most crucial step in the entire model. Those features are:

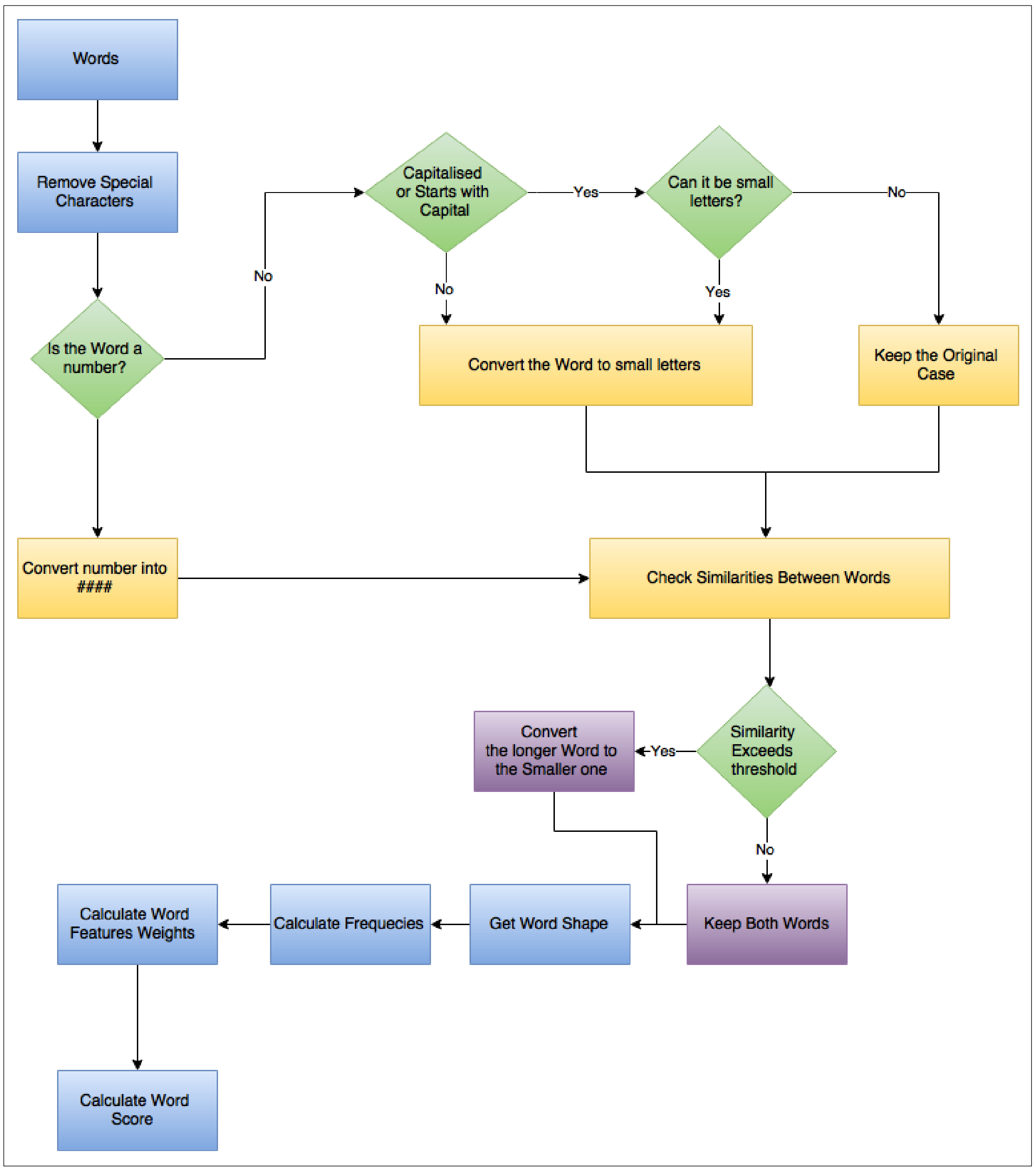
i. Word Shape,

ii. Word Shape frequency,

iii. Word Form frequency,

iv. Overall Word Score.

Word feature extraction and scoring depends on two assumptions: a) rare words in a text have more influence on the meaning of the text and should get heavier weights; and b) rare or less frequent word shapes may indicate a NE that implies more influence in the text, hence, receiving heavier weights. In conclusion, the rarer the word form and shape are, the higher the word score and thus the more important it is for summarizing the text.



Words
Remove Special Characters
Is the Word a number?
No
Capitalised or Starts with Capital
Yes
Can it be small letters?
No
No
Yes
Convert the Word to small letters
Keep the Original Case
Convert number into ####
Check Similarities Between Words
Convert the longer Word to the Smaller one
Yes
Similarity Exceeds threshold
No
Calculate Word Features Weights
Calculate Frequecies
Get Word Shape
Keep Both Words
Calculate Word Score

Figure 5. Process of word features extraction and Word Score (*WS*) computation.

3.3.1.1 Process of Word Features Extraction

The process of word features extraction involves many steps that begins with language independent pre-processing; *shape-coding;* through counting frequencies of shapes and forms; and ending with computing the final word score. **Figure 5** shows the process of word features extraction from the beginning of the preprocessing algorithm till computing the word score. The steps of word features extraction are:

i. *Extract Words (Tokenization)*: A sentence is segmented into its individual tokens (words).

ii. *Removal of special characters*: All none alphanumeric characters, as commas, hyphens, points, semicolons, etc., are removed from each individual token.

iii. *Normalize Numbers*:

If the token is a number, then it is converted into “####”. This normalization step of numbers is crucial to avoid assigning false high weights for each number. Since numbers tend to be different in a document (i.e. it is rare that a single number is repeated many times in a single document especially in scientific texts), thus, without normalization, the model will consider every number as a rare influential word leading to an over estimated importance of the number and its containing sentence.

iv. *Normalize Word Forms:*

Encoded shapes depend on the word form and its case (upper case or lower case letters or mixed). And since the rareness of a word shape reflects its importance (usually Named Entities are capitalized or start with upper case letter), thus it’s crucial to normalize word case such that upper case words are those of NE’s and not for words that accidently appear at the beginning of a sentence.

To achieve this normalization, each word that starts with a capital letter is searched in the whole document. If the word was found in any other position in the document with lower case letters, then, the word case is changed to lower case.

v. *Combining Similar Words*:

Some words are similar and should be treated as one when counting the frequency of occurrence of words. For example, in English, words like “*year*” and “*years*”; “*automatic*” and “*automatically*”; or “*allow*” and “*allowed*”; and in Portuguese, “*embaixada*” and “*embaixadas*”; or “*quero*” and “*quer*”.

Supervised text summarization models use *stemming* or *lemmatization* to revert a word back to its root/stem and hence combine similar words. This step, in case of supervised text summarization, is language dependent, due to the need of lexicons or language dependent hand coded rules. Since our prosed model is language agnostic, so it is important that we develop an algorithm that combines similar words depending on the word forms and the degree of similarities they have in common regardless the language of the original text and its words.

The proposed algorithm tends to calculate the number of similar letters between two words starting from the letter at position 1. In other words, the algorithm computes the length of sequence of common letters between two words, which is called *Degree of Similarity* or *DoS* for short.

The *Degree of Similarity* is computed using equation 3.4, and if that *DoS* value exceeds a predefined threshold, then the extra letters of the longer word are discarded, so that it is converted into the smaller one.

$$DoS = \frac{length\ of\ common\ letters\ sequence}{length\ of\ the\ longer\ word} \qquad 3.4$$

The pseudocode for similarity check between two tokens is explained in **Algorithm 1**. If the algorithm returns True, then extra letters are removed from the longer word in an action close to language dependent stemming of words.

Moreover, **Table 5** present examples for combining similar words using a threshold of 0.68.

Algorithm 1. The similarity check algorithm.

```
Pseudocode: Two words similarity check
    Inputs: strToken1, strToken2, dblThreshold
    Result: isDoStemLongerWord

/* Check letters similarities */
intCount = 0;
dblDoS = 0;

If (Length(strToken1) >= strToken2) Then
    strLongToken = strToken1;
    strShortToken = strToken2;
Else
    strLongToken = strToken2;
    strShortToken = strToken1;
End if

For i = 1 To Length(strShortToken) do
    If (strShortToken[i] = strLongToken[i]) Then
        intCount = intCount + 1;
    Else
        Exit do;
    End if
Loop

/* Calculate Degree of Similarity */
dblDoS = intCount/Length(strLongToken);

/* Check if DoS exceeds the preset threshold */
If (dblDoS >= dblThreshold) Then
```

```
        isDoStemLongerWord = True;
Else
        isDoStemLongerWord = False;
End if

Return isDoStemLongerWord;
```

Table 5. Examples for combining similar words using a threshold of 0.68. Words with *DoS* < the threshold (colored in red) are maintained unchanged.

Word 1	Word 2	Number of Similar letters	*DoS*	Action Performed
Year	Years	4	4/5 = 0.8	"*Years*" is changed to "*Year*"
Automatic	Automatically	9	9/13 = 0.69	"*Automatically*" is changed to "*Automatic*"
An	And	2	2/3 = 0.66	Both words are maintained
Form	From	1	1/4 = 0.25	Both words are maintained
University	Universities	9	9/12= 0.75	"*Universities*" is changed to "*University*"
Allow	Allowed	5	5/7 = 0.71	"*Allowed*" is changed to "*Allow*"
Quer	Quero	4	4/5 = 0.8	"*Quero*" is changed to "*Quer*"

vi. Get Word Coded-Shape:

After normalizing tokens by converting numbers to "####", change the word case to the appropriate one and combine similar words. Each character in the token is mapped to its appropriate shape code as mentioned under section 3.2.1. **Algorithm 2** shows the pseudocode for applying *shape-coding* for extracted normalized tokens.

Algorithm 2. Shape-coding of normalized Tokens.

```
Pseudocode: Shape-Coding
    Inputs: strToken
    Result: strShape

/* Map each character to its appropriate code */
strTemp = “”;
For i = 1 To Length(strToken) do
    Switch Case (strToken[i])
           Case “#”
                 strTemp = strTemp & “N”;
           Case Else
                If (isUpperCase(strToken[i])) Then
                    strTemp = strTemp & “X”;
                Else
                    strTemp = strTemp & “x”;
                End if
    End Switch
Loop

/* Group Similar Codes */
strShape = “”;
For i = Length(strTemp) To 2 Step -1 do
    If (strTemp[i] = strTemp[i-1]) Then
        If (isUpperCase(strTemp[i-1])) Then
            If  (strTemp[i-1] = “X”) Then
                 strShape = “C” & strShape;
            ElseIf  (strTemp[i-1] = “N”) Then
                 strShape = “n” & strShape;
            End if
        Else
            strShape = “c” & strShape;
        End if
    Else
        strShape = strTemp[i] & strShape;
    End If
Loop
strShape = strTemp[1] & strShape;

/* Return the coded shape */
Return strShape;
```

vii. *Get Word Frequency and Compute Word Form Weight (WfW):*

Get the count of each word in the text to be summarized. This word frequency is used to compute the weight of the word form (*WfW)*. Since, the proposed model assumes that a rare word has more influence on the text than a more common word, thus as seen in equation 3.5 below, the WfW is computed by taking the reciprocal of the natural logarithm (log of base e) of the word count.

$$WfW = \frac{1}{\ln(count(word) + 1)} \qquad 3.5$$

The natural logarithm is inverted in order to give non-linear higher weights to rare words compared to abundant ones. While, the add 1 normalization is done to avoid dividing by zero in case of a word that was mentioned only once, as *ln(1) = 0*.

The impact of the multiplicative inverse of the natural logarithm weight and the assumption of the higher influence of rare words are better understood using an example. Where, a word like "the" may be mentioned in a piece of text around 40 times, while a word like "*UnB*" is mentioned twice. Thus the *WfW* of "the" in this case is equal to $\frac{1}{\ln(40+1)}$ = 0.2693, while the rare word "*UnB*" will get a *WfW* = 0.9102. This weight reflects clearly the difference in importance between these two words and the subsequent effect on evaluating the importance of particular sentence.

This way the language agnostic proposed model, *UnB-LITS*, can get rid of the effect of stop words, which are usually abundant in the text, without any need of language dependent lists and lexicons.

viii. Get Coded Shapes Frequency and Compute Word Shape Weight (WsW*):*

Get the count of each word shape in the text to be summarized. This shape frequency is used to compute the weight of the word shape (*WsW)*. Since, the proposed model assumes that a rare coded shape has more influence on the text than a more common shape, thus as seen in equation 3.6 below, the WsW is computed by taking the reciprocal of the logarithm (log of base 10) of the coded shape count.

$$WsW = \frac{1}{\log(count(shape) + 1)} \qquad 3.6$$

The logarithm is inverted in order to give non-linear higher weights to rare shapes compared to abundant ones. While, the add 1 normalization is used to avoid dividing by zero in case of a word shape that existed only once, since, *log(1) = 0*.

For example: the coded shape of a word like "the" is "*xc*", which is the most abundant shape in the entire text, since all lower case words has that encoded shape. If this shape exists "300 times" in the text, thus its WsW in this case is equal to $\frac{1}{\log(300+1)}$ = 0.4035. While a rare word like "*UnB*" which has a relatively rare word shape "*XcX*" that might exist only three times in the entire text. Therefore, the WsW of "*XcX*" in this case is equal to $\frac{1}{\log(3+1)}$ = 1.6610. This weight reflects the difference in importance between these two encoded shapes, which in turn affect their overall scores.

It is worth mentioning that logarithm of base 10 is used to compute WsW rather than the natural logarithm used for computing WfW since this technique will give much higher weights for shapes in comparison to word forms. Where, if two words have equal frequency of occurrence, then the word with more rare coded shape will have higher *Word Score* (*WS*). A detailed discussion of the relative weights of WfW and WsW is presented under item *ix computing Word Score*.

ix. *Computing Word Score (WS):*

Computing the *Word Score* (*WS*) of a token is simply done by combining its *Word form Weight* (*WfW*) and its *Word shape Weight* (*WsW*) as seen in equation 3.7 below.

$$WS = (2 - \alpha)\, WfW \; + \; \alpha\, WsW \qquad 3.7$$

Where α is constant greater than or equal to 0 and less than or equal to 2. The α constant is used to adjust the relative weights of each term of the *WS* computing equation. Usually α is set to 1 in order to maintain the relative weights of both terms of the equation imposed by the difference between *log* and *ln*. However, in certain situations where short texts are to be summarized, it is recommended to give larger weight for the shape term *WsW,* as in case of short text unimportant words might, accidently, occur in unusual low frequency. In such situation, higher α is used to give more weights for named entities.

As discussed previously rare words with rare encoded shapes have higher *WS* than the most abundant ones. Where, a common words like "*the*" with frequency of occurrence = 40, and its encoded shape "*xc*" has frequency = 300, will get a word score $WS = 1 \times 0.2693 + 1 \times 0.4035$ = 0.6728. While, the word "*UnB*" that exists 2 times and its coded shape "*XcX*" exits three times, its $WS = 1 \times 0.9102 + 1 \times 1.6610$ = 2.5712. Accordingly, the *WS* for both tokens reflect the importance of the token “*UnB*” when compared to the token “*the*”.

As discussed in the previous section, the word shape is given a higher weight than the word form by using the natural logarithm for computing *WfW* and logarithm of base 10 for computing *WsW*. This technique of computing weights is useful in giving a relatively lower score for relatively unimportant words with very common shape “*xc*” that might appear, in rare situations, few times in the text to be summarized. For example, a word like “*for*” may exists only 3 times in a text, thus its common coded shape weight will pull it lower as compared to other words that has equal frequency but more rare coded shape.

3.3.1.2 A Working Example

This working example will implement all the techniques described above for tokenization of a piece of text and its subsequent normalization, features extraction and computing the overall word scores. This example uses a piece of text from a document (ID: AP880911-0016) of the benchmark dataset (DUC2002) provided by the American National Institute of Standards (NIST) during the Document Understanding Conference in 2002 - DUC 2002 [156].

The document was subjected to a text segmentation process that resulted in 16 sentences listed in **Table 6**.

Table 6. Document AP880911-0016 from DUC 2002 dataset.

Sentence ID	Sentence
S1	Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas.
S2	The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
S3	``There is no need for alarm, "Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.
S4	Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
S5	An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
S6	Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.
S7	The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

S8	The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a ``broad area of cloudiness and heavy weather'' rotating around the center of the storm.
S9	The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
S10	Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet feet to Puerto Rico's south coast.
S11	There were no reports of casualties.
S12	San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
S13	On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast.
S14	Residents returned home, happy to find little damage from 80 mph winds and sheets of rain.
S15	Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
S16	The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

By applying all the steps of the word scoring technique described above on this piece of text, we got the following:

i. *Tokenization*: resulted in 317 tokens, {*Hurricane, Gilbert, swept, toward, the, Dominican, Republic, Sunday, .., coast, .., ''Civil, 100,000, people, .., 67.5, .., mph, .., On, .., U.S., .., The, .., coastal, .., month*}.

ii. *Removal of special characters*: All special characters as commas, decimal points, punctuations, etc. were removed, as follows:

100,000 → 100000, 67.5 → 675, ''Civil → Civil, U.S. → US, etc.

iii. *Normalize Numbers:* all numbers were changed to ####. For example, *100000 → ####, 675 → ####.*

iv. *Normalize Word Forms*: Different words were normalized to its appropriate case. Where, some words that have initial capital letters were changed to lower case, while others are maintained in their original case. For example: *Hurricane → hurricane, Gilbert → Gilbert, On → on, The → the, etc.* Therefore the set of normalized tokens become {*hurricane, Gilbert, swept, toward, the, Dominican, Republic, Sunday, .., coast,.., civil, ####, people, .., ####, .., mph, .., on, .., US, .., the, .., coastal, .., month*}.

v. *Combining Similar Words*: this step was performed using *DoS Threshold* = 0.6 resulted in 151 unique words. Where some words were combined together, for example: *coastal → coast, Gilbert's → Gilbert, reported → report, etc.* Thus the set of tokens become {*hurricane, Gilbert, swept, toward, the,* *Dominican, Republic, Sunday, .., coast,.., civil, ####, people, .., ####, .., mph, .., on, .., US, .., the, .., coast, .., month*}.

vi. *Get Word Shapes: shape-coding* of the normalized tokens resulted in 4 unique word shapes {Nn, xc, Xxc, XC}. Where *hurricane* → *xc, Gilbert* → *Xxc, US* → *XC.* As such, the set of encoded shape tokens is {*xc, Xxc, xc, xc, xc, Xxc, Xxc, Xxc, .., xc,.., xc, Nn, xc, .., Nn, .., xc, .., xc, .., XC, .., xc, .., xc, .., xc*}.

vii. *Get Word Frequency and Compute Word Form Weight (WfW)*: **Table** 7 shows the frequency and the subsequently computed *WfW* of some tokens from the sample text. As shown in the table, abundant unimportant words like "*the*" or "*####*" have the least weights. While important words (NE), like "*US"* and "*Dominican",* have the highest weights reflecting their importance. However some unimportant words like "*from*" has higher *WfW* than an obviously important word in that context like "*Gilbert*", this is due to the fact that the word "*from*" occurred only 3 times in the whole text compared to "*Gilbert*" that appeared 5 times. This pitfall will be corrected when the coded shape weight (*WsW*) is included in computing the overall word score.

Table 7. Example of Computing *WfW* for some normalized tokens.

Token	*Count*	*WfW*
####	*15*	*0.361*
the	*23*	*0.131*
Gilbert	*5*	*0.558*
US	*1*	*1.443*
Dominican	*1*	*1.443*
from	*3*	*0.721*

viii. *Get Coded Shapes Frequency and Compute Word Shape Weight (WsW):* As mentioned above, the shape-coding technique results in 4 shapes only. **Table 8** shows the frequency and the *WsW* of these shapes.

Table 8. Frequencies of the extracted encoded shapes and their *WsW*.

Coded Shape	*Count*	*WsW*
Nn	*15*	*0.830*
xc	*254*	*0.416*
Xxc	*47*	*0.595*
XC	*1*	*3.322*

ix. *Computing Word Score (WS):* **Table 9** shows the computed *WS* for the selected tokens using α set to 1. The results shown below proves that using the reciprocal logarithm of base 10 for encoded shapes has added more influence to rare words rather than the unimportant ones. As such, important words like "*Gilbert*" got a higher overall *WS* than the unimportant word "*from*".

Table 9. The overall word score (*WS*) for extracted tokens.

Token	*Shape*	*WfW*	*WcW*	*WS*
####	*Nn*	*0.361*	*0.830*	*1.191*
the	*xc*	*0.131*	*0.416*	*0.547*
Gilbert	*Xxc*	*0.558*	*0.595*	*1.153*
US	*XC*	*1.443*	*3.322*	*4.765*
Dominican	*Xxc*	*1.443*	*0.595*	*2.038*
from	*xc*	*0.721*	*0.416*	*1.137*

3.3.2 *N-grams scoring and Features Extraction*

As defined earlier, an *n-gram* is a sequence of *n* words from a given text, where *n* is the size of the window that forms the sequence of words. The extraction of *n-gram* features is the process of extracting statistical properties based on the n-gram's form (spelling), and encoded shapes as well as the *Word Score*s of their constituent words. The *n-gram score* (*nGS*) is a combination of the extracted feature weights. In this model, we are going to extract features for *bigrams*, *trigrams*, *4-grams* and *5-grams*. The extracted features are:

i. N-gram Shape,
ii. N-gram Shape frequency,
iii. N-gram Form frequency,
iv. Overall N-gram Score (*nGS*; Also it can be denoted as *2GS*, *3GS*, *4GS* or *5GS* for bigrams, trigrams, 4-grams and 5-grams respectively).

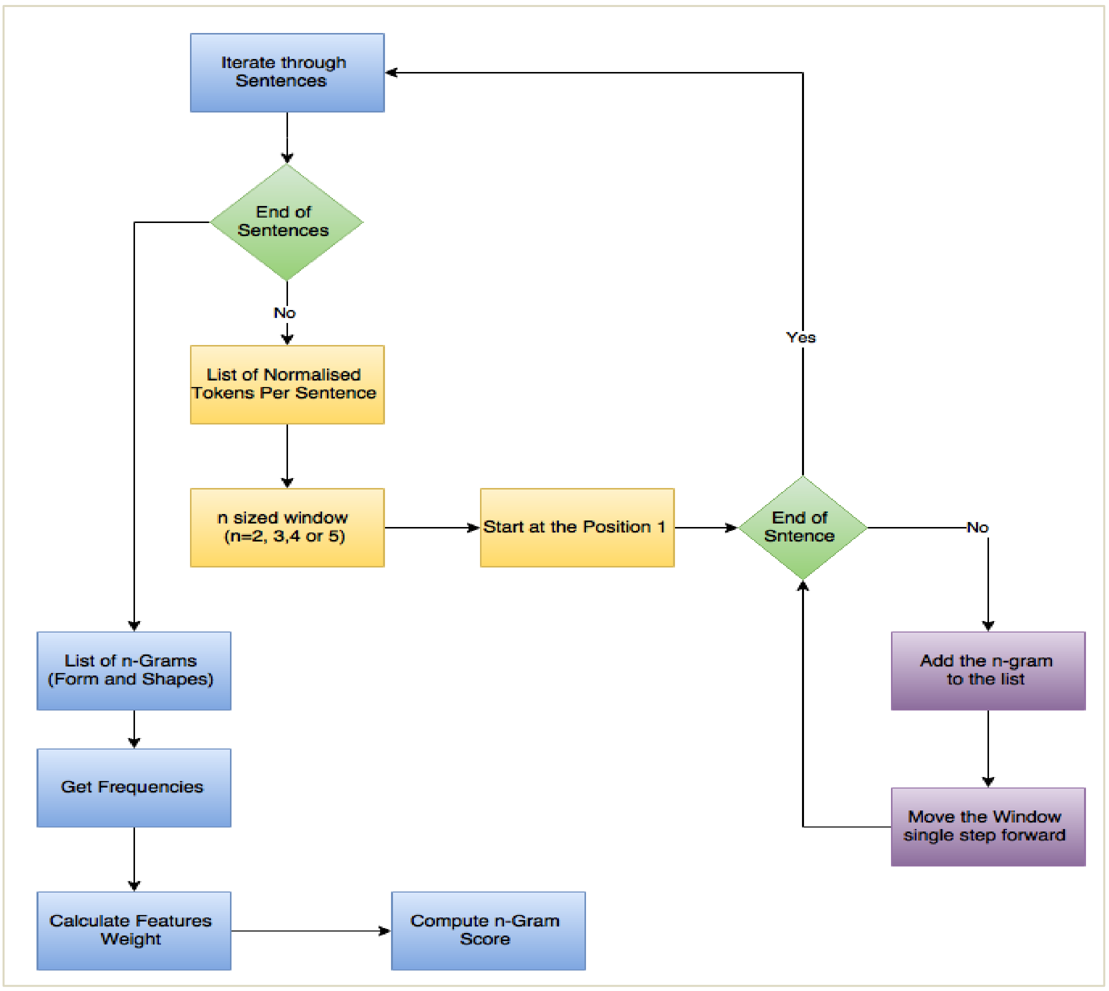

Iterate through Sentences
End of Sentences
No
List of Normalised Tokens Per Sentence
n sized window (n=2, 3,4 or 5)
Start at the Position 1
End of Sntence
Yes
No
Add the n-gram to the list
Move the Window single step forward
List of n-Grams (Form and Shapes)
Get Frequencies
Calculate Features Weight
Compute n-Gram Score

Figure 6. Extracting n-Grams and computing their scores (*nGS*).

In contrast to the word scoring assumption of rare words have more importance than the more abundant ones, in our proposed model the n-gram feature extraction is based on the exact opposite assumption. N-gram scoring approach assumes that probable *n-gram's*¸ in terms of form and shape, have more influence on the meaning of the text than less probable ones, and accordingly should get heavier weights.

This assumption aids in identifying and extracting Named Entities and Concrete Concepts in a language agnostic manner.

Moreover, this assumption and its rule in identifying NE's and CC's can be explained in view of the fact that the overall score of an *n-gram* (*nGS*) is influenced by the scores of its constituent words. As such, in case of bigrams for instance, the more probable two rare words occur together the higher the weight the bigram should have. For example, two rare words like "*United*" and "*States*" if they occur together in relatively high frequency this indicates that the bigram "*United States*" (with shape "*Xxc Xxc*") is more likely to be a *Named Entity* or a *Concrete Concept*.

In conclusion, the more abundant the *n-gram* form and the more rare its individual words are, the higher its *nGS* and thus the more important it is for summarizing the text.

3.3.2.1 Process of N-grams Features Extraction

The process of *n-gram* features extraction involves many steps that depend mainly on the previous step of *Word Features Extraction*. It starts with concatenating normalized tokens and their shapes, and ends up with computing the final n-Gram Score (*nGS*). No preprocessing or normalization techniques are applied in this step as the tokens are already normalized and pre-processed before. **Figure 6** shows the process of extracting n-grams and computing their *nGS*. The steps of *n-Gram* features extraction are:

i. *Extract N-Grams:*

Four different types of n-grams are extracted in this step; bigrams, trigrams, 4-grams and 5-grams. As such and as discussed earlier, extracting n-grams involves setting a window of size *n* (2, 3, 4 and 5) that moves forward along each sentence extracting the appropriate n-grams and add them to a list.

ii. *Get N-Grams Encoded Shapes*:

This is simply done by concatenating the encoded shapes of the tokens forming the n-gram and separated by a single space character. For example, if a bigram is formed of the words "*United*" and "*States*" thus the bigram form is "*United States*" and its encoded shape is "*Xxc Xxc*", while for the trigram "*the United States*" its coded shape will be "*xc Xxc Xxc*".

iii. Get N-gram Frequency and Compute N-gram Form Weight (nGfW):

Get the count of each *n-gram* in the text to be summarized. The *n-gram* frequency is used to compute the Maximum Likelihood Estimate (MLE) of that n-gram form. Since, the proposed model assumes that the more likely (probable) the *n-gram* is the more influential it is, thus the *nGS* is the product of the MLE of the *n-gram* form (*nGfW*) and shape (*nGsW*), as per the equations below.

$$nGfW = \left(\prod_{i=1}^{n} P(w_i|w_1,..,w_{i-1})\right) \times \sum_{i=1}^{n} WS(word_i) \qquad 3.8$$

$$Where, \qquad P(w_i|w_1,..,w_{i-1}) = \frac{count\ (w_1,..,w_i)}{count\ (w_1,..,w_{i-1})} \qquad 3.9$$

Where, w_i is the word at position i of an *n-gram*. While, $\sum_{i=1}^{n} WS(word_i)$ is the sum of words scores of all the words that form the n-gram. This term is included to lower the value of *nGfW* for probable n-grams that are made from weak words, and hence, give higher weights for n-grams made from stronger words.

As seen in equation 3.9, $P(w_i|w_1,..,w_{i-1})$ is the maximum likelihood estimate of an n-gram formed from i words, which is calculated as the probability of the word w_i given the sequence of words $\{w_1,..,w_{i-1}\}$, which is equal to ratio of the frequency of occurrence of the whole sequence of words $\{w_1,..,w_i\}$ to the frequency of occurrence of the sequence formed from the words $\{w_1,..,w_{i-1}\}$. [5]

Using the same working example stated in 3.3.1.2, computing the *nGfW* (*4GfW*) for the *4-gram* "***Director Eugenio Cabral said***" is done as follows:

a) Calculate the *MLE* for the 4-gram word form:

$$MLE = \frac{C\left(\textit{Director Eugenio Cabral said}\right)}{C\left(\textit{Director Eugenio Cabral}\right)} \times \frac{C\left(\textit{Director Eugenio Cabral}\right)}{C\left(\textit{Director Eugenio}\right)} \times \frac{C\left(\textit{Director Eugenio}\right)}{C\left(\textit{Director}\right)}$$

b) Compute the sum of word scores (*WS*) of the 4-gram words (from "***Director***" to "***said***"):

$$Sum\ of\ WS_{Director}^{Said} = WS(\textit{Director}) + WS(\textit{Eugenio}) + WS(\textit{cabral}) + WS(\textit{said})$$

c) Multiply the $Sum\ of\ WS \times MLE$:

$$4GfW = MLE \times Sum\ of\ WS_{Director}^{Said}$$

iv. Get N-gram Shape Frequency and Compute N-gram Shape Weight (nGsW):

Get the count of coded shapes of each *n-gram* in the text to be summarized. The *n-gram* shape frequency is used to compute the Maximum Likelihood Estimate (MLE) of that n-gram shape that will be referred to as the *n-Gram shape Weight* (*nGsW)* as per equation 3.10 below.

$$nGsW = \left(\prod_{i=1}^{n} P(s_i|s_1,..,s_{i-1})\right) \qquad 3.10$$

$$Where, \qquad P(s_i|s_1,..,s_{i-1}) = \frac{count\ (s_1,..,s_i)}{count\ (s_1,..,s_{i-1})} \qquad 3.11$$

Where, s_i is the shape at position *i* of that *n-gram*.

As per Equation 3.11, $P(s_i|s_1,..,s_{i-1})$ is the maximum likelihood estimate of an n-gram formed from *i* encoded shapes. The MLE in this case is calculated as the probability of the shape s_i given the sequence of encoded shapes $\{s_1,..,s_{i-1}\}$, which is equal to ratio of the frequency of occurrence of the whole sequence of shapes $\{s_1,..,s_i\}$ to the frequency of occurrence of the sequence formed from the shapes $\{s_1,..,s_{i-1}\}$.

Using the same 4-gram example of "***Director Eugenio Cabral said***", this 4-gram has encoded shape of "***Xxc Xxc xc xc***", as such, for computing its *nGsW* (*4GsW*), the *MLE* is computed as follows:

$$MLE = \frac{c\ (\boldsymbol{Xxc\ Xxc\ xc\ xc})}{c\ (\boldsymbol{Xxc\ Xxc\ xc})} \times \frac{c\ (\boldsymbol{Xxc\ Xxc\ xc}\ )}{c\ (\boldsymbol{Xxc\ Xxc})} \times \frac{c\ (\boldsymbol{Xxc\ Xxc})}{c\ (\boldsymbol{Xxc}\ )} \qquad 3.12$$

v. Computing N-gram Score (nGS):

Computing the *n-gram score* (*nGS*) for the extracted bigrams, trigrams, 4-grams and 5-grams is simply done by combining their *n-gram form Weight* (*nGfW*) and their *n-gram shape Weight* (*nGsW*) as per equation 3.13 below.

$$nGS = nGfW\ \times\ nGsW \qquad 3.13$$

As discussed previously, the high probable n-grams form and shapes that are composed of rare words will get higher *nGS* than the less probable and/or weaker ones. For example in a text about astronomy, a relatively abundant bigram like "*Solar System*" will get higher *nGS* than less abundant one like "*the flare*".

However, in certain circumstances, a highly probable n-gram can receive lower *nGS* than less probable ones. This happens when the n-gram is composed of weak words with weak *WS*. For example, a probable and abundant bigram like "*in the*" with coded shape "*xc xc*" will have lower overall *nGS* than other less probable ones like "*United States*" due to the fact that "*in the*" is composed of weak words, while, "*United States*" is composed of strong words that intensify its overall *nGS*.

As such, *nGS* reflects the actual importance of an n-gram based on its probability of occurrence as well as the strength of its constituent words.

3.3.2.2 A Working Example

Continuing with the working example of section 3.3.1.2. Applying the techniques described above for scoring and extracting n-gram features on the selected text results in the following:

i. *Extract N-Grams:* The technique extracted 268 bigrams, 281 trigrams, 269 4-grams and 253 5-grams. Some of the extracted n-grams are listed in **Table 10** below.

Table 10. Extracted n-grams.

N-gram	N-gram Form
Bigrams	US Gulf Director Eugenio Dominican Republic Virgin Islands in the on the
Trigrams	US Gulf coast Virgin Islands until Debby reached minimal Director Eugenio cabral in the city on the north
4-grams	Virgin Islands until at Debby reached minimal hurricane Defense Director Eugenio cabral Director Eugenio cabral said in the city of on the north coast
5-grams	Virgin Islands until at least Debby reached minimal hurricane strength Islands until at least #### Director Eugenio cabral said in in the city of Barahona on the north coast had

ii. *Get N-Grams Shape: shape-coding* of the extracted n-grams resulted in 9 unique bigrams shapes, 18 unique trigrams shapes, 31 unique 4-grams shapes and 48 unique 5-grams shapes. Some of the extracted n-grams shapes are listed in **Table 11** below.

Table 11: Extracted n-grams unique shapes with their frequency of occurrence.

N-gram	N-gram Shape	Frequency
Bigrams	xc xc	196
	xc Xxc	30
	Xxc xc	28
	xc Nn	15
	Xxc Xxc	15
	Nn xc	14
Trigrams	xc xc xc	152
	xc xc Xxc	22
	Xxc xc xc	22
	xc Xxc xc	18
	xc Nn xc	14
	xc xc Nn	13
4-grams	xc xc xc xc	117
	xc xc xc Xxc	18
	Xxc xc xc xc	18
	xc Xxc xc xc	15
	xc xc Nn xc	12
	xc Nn xc xc	11
5-grams	xc xc xc xc xc	92
	xc Xxc xc xc xc	14
	Xxc xc xc xc xc	13
	xc xc xc xc Xxc	11
	xc Nn xc xc xc	9
	xc xc Nn xc xc	9

iii. *Get N-gram Frequency and Compute N-gram Form Weight (nGfW):* **Table 12** shows the computed *nGfW* of some of the extracted *n-gram* from the sample text. As shown in the table abundant unimportant n-grams like "*in the*" or "*on the*" have the least weights. While, important n-grams (NE's), like "*Director Eugenio cabral"* and "*Dominican Republic",* have the highest weights reflecting their importance.

Table 12. Example of computing *nGfW* for some extracted n-grams.

N-gram	**N-gram Form**	**ML**	**Sum of Words**	***nGfW***
Bigrams	US Gulf	*1*	*6.802*	***6.802***
	Director Eugenio	*1*	*4.075*	***4.075***
	Dominican Republic	*1*	*4.075*	***4.075***
	Virgin Islands	*1*	*4.075*	***4.075***
	in the	*0.5*	*1.66*	***0.830***

	on the	*0.5*	*2.056*	***1.028***
Trigrams	US Gulf coast	*1*	*7.731*	***7.731***
	Virgin Islands until	*1*	*5.933*	***5.933***
	Debby reached minimal	*1*	*5.753*	***5.753***
	Director Eugenio cabral	*1*	*5.401*	***5.401***
	in the city	*0.167*	*3.518*	***0.588***
	on the north	*0.5*	*3.382*	***1.691***
4-grams	Virgin Islands until at	*1*	*6.97*	***6.970***
	Debby reached minimal hurricane	*1*	*6.683*	***6.683***
	Defense Director Eugenio cabral	*0.5*	*6.906*	***3.453***
	Director Eugenio cabral said	*1*	*6.653*	***6.653***
	in the city of	*0.167*	*4.323*	***0.722***
	on the north coast	*0.5*	*4.311*	***2.156***
5-grams	Virgin Islands until at least	*1*	*8.828*	***8.828***
	Debby reached minimal hurricane strength	*1*	*8.009*	***8.009***
	Islands until at least ####	*1*	*7.982*	***7.982***
	Director Eugenio cabral said in	*1*	*7.467*	***7.467***
	in the city of Barahona	*0.167*	*5.828*	***0.973***
	on the north coast had	*0.5*	*6.169*	***3.085***

iv. *Get N-gram Shape Frequency and Compute N-gram Shape Weight (nGsW)*: **Table 13** shows computed *nGsW* of some of the extracted *n-gram* shapes from the sample text.

Table 13. Example of computing *nGsW* for some extracted n-grams shapes.

N-gram	**N-gram Form**	***nGsW***
Bigrams	XC Xxc	1
	Xxc Xxc	0.0213
	Xxc Xxc	0.0213
	Xxc Xxc	0.0213
	xc xc	0.0039
	xc xc	0.0039
Trigrams	XC Xxc xc	1
	Xxc Xxc xc	0.0005
	Xxc xc xc	0.0005
	Xxc Xxc xc	0.0005
	xc xc xc	1.5E-05
	xc xc xc	1.5E-05
4-grams	Xxc Xxc xc xc	9.6E-06
	Xxc xc xc xc	9.6E-06
	Xxc Xxc Xxc xc	9.6E-06
	Xxc Xxc xc xc	9.6E-06
	xc xc xc xc	6.1E-08
	xc xc xc xc	6.1E-08
5-grams	Xxc Xxc xc xc xc	2.1E-07
	Xxc xc xc xc xc	2.1E-07
	Xxc xc xc xc Nn	2.1E-07
	Xxc Xxc xc xc xc	2.1E-07
	xc xc xc xc xc	2.4E-10
	xc xc xc xc xc	2.4E-10

v. *Computing N-gram Score (nGS):* **Table 14** shows the computed *nGS* for the extracted n-gram. The results shown below prove that important n-grams get heavier weights than less important and less abundant ones.

Table 14. Example of computing the overall *nGW* for some extracted n-grams

N-gram	**N-gram Form**	***nGfW***	***nGsW***	***nGW***
Bigrams	US Gulf	6.802	1	*6.802*
	Director Eugenio	4.075	0.0213	*0.0868*
	Dominican Republic	4.075	0.0213	*0.0868*
	Virgin Islands	4.075	0.0213	*0.0868*
	in the	0.830	0.0039	*0.0033*
	on the	1.028	0.0039	*0.004*
Trigrams	US Gulf coast	7.731	1	*7.731*
	Virgin Islands until	5.933	0.0005	*0.003*

	Debby reached minimal	5.753	0.0005	*0.0029*
	Director Eugenio cabral	5.401	0.0005	*0.0027*
	in the city	0.588	1.5E-05	*0.000009*
	on the north	1.691	1.5E-05	*0.000025*
4-grams	Virgin Islands until at	6.970	9.6E-06	*0.00007*
	Debby reached minimal hurricane	6.683	9.6E-06	*6.4E-05*
	Defense Director Eugenio cabral	3.453	9.6E-06	*3.3E-05*
	Director Eugenio cabral said	6.653	9.6E-06	*6.4E-05*
	in the city of	0.722	6.1E-08	*4.4E-08*
	on the north coast	2.156	6.1E-08	*1.3E-07*
5-grams	Virgin Islands until at least	8.828	2.1E-07	*1.9E-06*
	Debby reached minimal hurricane strength	8.009	2.1E-07	*1.7E-06*
	Islands until at least ####	7.982	2.1E-07	*1.7E-06*
	Director Eugenio cabral said in	7.467	2.1E-07	*1.6E-06*
	in the city of Barahona	0.973	2.4E-10	*2.3E-10*
	on the north coast had	3.085	2.4E-10	*7.4E-10*

3.3.3 *Sentences Scoring and Features Extraction*

In extractive summarization task, sentences are the only output of any extractive summarization tool. As such, sentence scoring is the ultimate goal of the entire automatic summarization algorithm. After all the sentences in a text are scored, then, they are arranged in descending order according to their overall score *Sentence Score* (*SC*).

In contrast to word scoring, sentence scoring depends on both intrinsic properties within the sentence itself, like the sentence shape and language (i.e. whether a sentence has the same language like the rest of the text, which is done by unsupervised clustering with no need to identify the language itself), as well as the score of all other documents elements. As such, the sentence score is the combination of words, n-grams and paragraph scores in addition to the score of the sentence's intrinsic features.

Sentence features Extraction is the process of extracting statistical intrinsic properties that depends on the sentence encoded shape, language uniformity as well as its words and n-grams. The extracted features are:

i. Sentence encoded shape,
ii. Sentence's Language Uniformity (*SLU*),
iii. Sum of scores of all of its Words,
iv. Sum of scores of all of its n-grams (bigrams, trigrams, 4-grams and 5-grams),
v. Overall Sentence Score.

Sentence feature extraction and scoring depends on three assumptions, a) rare sentence shape has more importance and should receive heavier weight. b) If the language of the sentence is not uniform with respect to the language of the entire text, then it should receive heavier weight. In addition, c) the higher the scores of the sentence components (words and n-grams) the more important it is.

In conclusion, the rarer the sentence is and the more important (heavier) its components the higher the sentence score and thus the higher the probability of its appearance in the final summary of the text.

The process of sentence features extraction involves many steps that begins with *shape-coding*, and the subsequent counting of shape frequencies; through detection of language uniformity, up to summing the scores of its components and calculating the final sentence score (*SC*).

The steps for sentence features extraction are as follows:

i. *Get Sentence encoded shape*:

A sentence *shape-coding* is done as mentioned under the shape-coding section in this chapter. A sentence shape can be one of the following six shapes: "***Zzz***, ***ZZZ***, ***ZZz***, ***zzz***, ***zZZ*** or ***zZz***".

ii. *Get encoded shape frequency and Compute Sentence shape Weight (SsW)*:

Get the count of each sentence shape in the text to be summarized. This shape frequency is used to compute the weight of the sentence shape (*SsW)*. Since, the proposed model assumes that a rare encoded shape has more influence on the text than a more common one, thus as seen below in equation 3.14, the *SsW* is computed by taking the multiplication inverse of the logarithm (log of base 10) of the encoded shape count.

$$SsW = \frac{1}{\log(count(shape) + 1)} \qquad 3.14$$

The logarithm is inverted in order to give non-linear higher weights to rare shapes compared to the abundant ones. While, the add 1 normalization is used to avoid dividing by zero in case of a sentence shape that existed only once, since *log(1) = 0*.

For example, the coded shape of a typical sentence is "*Zzz*", which is the most abundant shape in the entire text. If this shape exists "30 times" in the text, thus its SsW is equal to $\frac{1}{\log(30+1)}$ = 0.6705. While a rare sentence like a title sentence which has a rare shape "*ZZZ*" that might exist only once in the entire text. Therefore, the *SsW* of "*ZZZ*" in this case is equal to $\frac{1}{\log(1+1)}$ = 3.32.

This result reflects clearly the difference in importance between these two sentences and their encoded shapes, and the subsequent effect on computing the overall score of the sentence.

iii. *Get the Sentence Language Uniformity (SLU)*:

In this study, we propose a new term called *Sentence Language Uniformity* or *SLU*. *SLU* is a term that we will use to indicate one of three cases: a) if the sentence is written in the same language as the rest of the entire text. b) It is written in a completely different language (this might happen if the sentence is a quote of speech). Alternatively, c) the sentence contains a mixture of languages, a case that might happen if the sentence contains some Latin scientific terms or foreign names, etc.

Sentence Language Uniformity can be done in an entirely unsupervised manner using any unsupervised clustering technique, in this study we use *K-Means clustering*. *SLU* is calculated as follows:

a) First create a matrix of size (83 × 83), where both rows and columns represent 83 different letters (A, B, C, Õ, Ç, Ñ, À, Á, Œ, etc.), then the Maximum Likelihood Estimate of a letter being preceded by another letter is calculated as seen in the equation 3.15 below. This extracts the language pattern in the sentence.

$$MLE_{G,N} = \frac{count\ (NG) + 1}{count\ (N) + 83} \qquad 3.15$$

Equation 3.15 shows the MLE of the letter "***G***" being preceded by the letter "***N***", which is calculated by counting all the times that "***NG***" appears together in the sentence and divide them by the total number that "***N***" appears in that sentence. This equation is smoothed by *1-smoothing*, where 1 is added to the numerator while 83 (the total number of letters in the model) to account for letters that doesn't appear in that sentence and it should not take a probability of zero.

The Tables below show examples for the frequencies and the maximum likelihood estimate for the co-occurrences of two characters, i.e. the probability of occurrence of a character pair giving the first character. **Table 15** and **Table 16** are for an English text, while **Table 17** and **Table 18** are for a Portuguese text, and finally **Table 19** and **Table 20** are for a German text.

Table 15. Frequency of character pairs from an English text.

	A	B	C	D	E	F	G
A	0	15	39	25	2	3	23
B	14	0	0	0	33	0	0
C	24	12	10	0	51	0	3
D	20	0	1	1	72	0	0
E	47	11	36	66	25	11	5
F	17	1	0	0	14	11	0
G	8	1	0	0	37	0	1
H	71	1	0	0	160	0	0
I	27	11	46	22	34	12	33
J	1	0	0	0	6	0	0
K	1	1	0	0	15	0	0
L	43	0	0	28	54	3	0
M	27	1	3	0	44	0	0
N	29	0	21	78	81	2	67

Table 16. Smoothed MLE of character pairs from an English text (using the frequencies in the previous table).

	A	B	C	D	E	F	G
A	0.001	0.022	0.055	0.036	0.004	0.006	0.033
B	0.064	0.004	0.004	0.004	0.144	0.004	0.004
C	0.072	0.037	0.032	0.003	0.149	0.003	0.011
D	0.086	0.004	0.008	0.008	0.300	0.004	0.004
E	0.057	0.014	0.044	0.079	0.031	0.014	0.007
F	0.079	0.009	0.004	0.004	0.066	0.053	0.004
G	0.041	0.009	0.005	0.005	0.174	0.005	0.009
H	0.173	0.005	0.002	0.002	0.386	0.002	0.002
I	0.040	0.017	0.067	0.033	0.050	0.019	0.049
J	0.021	0.011	0.011	0.011	0.074	0.011	0.011
K	0.015	0.015	0.008	0.008	0.122	0.008	0.008
L	0.108	0.002	0.002	0.071	0.135	0.010	0.002
M	0.115	0.008	0.016	0.004	0.184	0.004	0.004
N	0.057	0.002	0.042	0.150	0.155	0.006	0.129

Table 17. Frequency of character pairs from a Portuguese text.

	A	B	C	D	E	F	G
A	0	11	36	82	0	3	29
B	17	0	1	0	12	0	0
C	42	12	3	0	44	0	3
D	85	0	1	0	185	0	0
E	10	13	40	27	3	8	13
F	16	1	0	0	13	1	0
G	11	1	0	0	30	0	0
H	24	0	0	0	10	0	0
I	76	4	47	42	16	8	27
J	5	0	0	0	13	0	0
K	0	1	0	0	1	0	0
L	27	0	0	0	40	0	2
M	79	6	3	0	86	0	0
N	54	0	27	63	22	4	9

Table 18. Smoothed MLE of character pairs from a Portuguese text (using the frequencies in the previous table).

	A	B	C	D	E	F	G
A	0.001	0.015	0.046	0.103	0.001	0.005	0.037
B	0.094	0.005	0.010	0.005	0.068	0.005	0.005
C	0.105	0.032	0.010	0.002	0.109	0.002	0.010
D	0.154	0.002	0.004	0.002	0.333	0.002	0.002
E	0.013	0.016	0.048	0.032	0.005	0.010	0.016
F	0.099	0.012	0.006	0.006	0.081	0.012	0.006
G	0.060	0.010	0.005	0.005	0.155	0.005	0.005
H	0.166	0.007	0.007	0.007	0.073	0.007	0.007
I	0.112	0.007	0.070	0.063	0.025	0.013	0.041
J	0.054	0.009	0.009	0.009	0.125	0.009	0.009
K	0.010	0.020	0.010	0.010	0.020	0.010	0.010
L	0.095	0.003	0.003	0.003	0.139	0.003	0.010
M	0.209	0.018	0.010	0.003	0.227	0.003	0.003
N	0.099	0.002	0.050	0.115	0.041	0.009	0.018

Table 19. Frequency of character pairs from a German text.

	A	B	C	D	E	F	G
A	1	30	50	9	0	14	34
B	15	0	0	0	84	0	1
C	1	12	4	0	11	0	3
D	61	1	1	0	221	0	1
E	9	21	16	14	11	5	20
F	21	1	0	0	16	12	0
G	12	1	0	0	128	0	2
H	36	1	0	1	70	1	0
I	12	4	83	12	172	5	65
J	5	0	0	0	12	0	0
K	8	1	0	0	21	0	2
L	18	7	1	2	76	2	5
M	17	0	3	2	56	0	1
N	44	4	2	115	142	9	72

Table 20. Smoothed MLE of character pairs from German text (using the frequencies in the previous table).

	A	B	C	D	E	F	G
A	0.003	0.050	0.083	0.016	0.002	0.024	0.057
B	0.059	0.004	0.004	0.004	0.316	0.004	0.007
C	0.005	0.035	0.014	0.003	0.033	0.003	0.011
D	0.124	0.004	0.004	0.002	0.442	0.002	0.004
E	0.007	0.016	0.012	0.011	0.008	0.004	0.015
F	0.086	0.008	0.004	0.004	0.066	0.051	0.004
G	0.039	0.006	0.003	0.003	0.384	0.003	0.009
H	0.099	0.005	0.003	0.005	0.189	0.005	0.003
I	0.015	0.006	0.098	0.015	0.202	0.007	0.077
J	0.056	0.009	0.009	0.009	0.120	0.009	0.009
K	0.047	0.010	0.005	0.005	0.114	0.005	0.016
L	0.048	0.020	0.005	0.008	0.194	0.008	0.015
M	0.074	0.004	0.017	0.012	0.236	0.004	0.008
N	0.066	0.007	0.004	0.169	0.208	0.015	0.106

b) Use the chosen unsupervised clustering technique, *k*-Means for example, to cluster the sentences.
c) Calculate the frequency of occurrence for each class.
d) Then SLU for a sentence j is calculated as per equation 3.16 below.

$$SLU_j = \frac{Count(Sentences)}{Count(U^j\ )} \qquad 3.16$$

Where, $\text{Count}(Sentences)$ is the total numbers of sentences in the text to be summarized, while $\text{Count}\ (U^j)$ is the count of sentences that belong to the same cluster as the sentence j. This equation gives more weight for sentences represented by a rare cluster, which indicates the presence of unusual rare character pattern that might be an indication of the presence of words from different languages, foreign names or expressions.

iv. Calculate the Overall Sentence Score (SC):

Computing the *Sentence Score* (*SC*) of a sentence is done by combining sentence shape weight (*SsW*) and *SLU* together with the sum of scores of its words (*WS*) and *n-grams (nGS)*, in addition to the score of its containing paragraph (*PS*). *SC* is computed using equation 3.17 below.

$$SC^j = \eta . \left(\lambda_1 \sum_{i=1}^{N_w^j} \frac{WS_i}{\max(WS)} + \lambda_2 \sum_{i=1}^{N_{2g}^j} \frac{2GS_i}{\max(2GS)} + \lambda_3 \sum_{i=1}^{N_{3g}^j} \frac{3GS_i}{\max(3GS)} + \lambda_4 \sum_{i=1}^{N_{4g}^j} \frac{4GS_i}{\max(4GS)} + \lambda_5 \sum_{i=1}^{N_{5g}^j} \frac{5GS_i}{\max(5GS)} \right) + \beta . \frac{SsW}{max(SsW)} + \gamma . \frac{SLU}{max(SLU)} + \delta . \frac{PS}{max(PS)} \qquad 3.17$$

Where, $\mathrm{N_w^j}$, $\mathrm{N_{2g}^j}$, $\mathrm{N_{3g}^j}$, $\mathrm{N_{4g}^j}$ and $\mathrm{N_{5g}^j}$ are the total number of words, bigrams, trigrams, 4grams and 5grams in sentence *j* respectively. While, $\max(..)$ means the maximum value of that score in the entire text (not only the sentence).

The weights λ_{1-5} are weights given to word and n-grams scores respectively, to help tweaking the sentence-scoring performance. It is worth mentioning that $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1$.

In addition, $\eta, \beta, \gamma\ and\ \delta$ are weights ranging between 0 and 1 to adjust the relative importance of the sum of words and n-grams scores (*WS* and *nGS*), the Sentence Shape Weight (*SsW*), the Sentence Language Uniformity (*SLU*) and the Paragraph Score (*PS*) respectively.

It is worth mentioning that all scores are normalized by dividing them with the maximum value of the score noticed in the entire text. This is done in order to keep the scores of all elements between 0 and 1 and prevent any unwanted effect of off-scale scores for some terms.

3.3.4 *Paragraphs Scoring and Features Extraction*

The process of paragraph features extraction involves only two steps, a) *shape-coding,* followed by b) counting the shape frequencies, in order to get the final *Paragraph Score* (*PS*).

As such, the steps for paragraph features extraction are as follows:

i. *Get the paragraph's encoded shape*:

Shape-coding of a paragraph is done as mentioned under the shape-coding section earlier this chapter. A paragraph shape can be one of many shapes, the most common of which is "***OMPpp***", which is the typical multi-sentence paragraph that starts with capital letter. Other encoded shapes, which are relatively rare, include, "***NSPPP***" for numbered section titles, where the paragraph is a part of a numbered list with single sentence and majority of words start with capital letter. Alternatively, "***OSPPP***" for document titles, where the paragraph is neither numbered nor bulleted, with single sentence and majority of words start with capital letter.

ii. *Get Encoded Shapes Frequency and Compute Paragraph Score*:

Get the count of each paragraph shape in the text to be summarized. This shape frequency is used to compute the weight of the paragraph shape (*PsW).* In this case, since the paragraph score (*PS)* depends only on its shape, thus *PS* is the same as *PsW*. Equation 3.18 is used to calculate the *PsW* by taking the reciprocal of the logarithm (log of base 10) of the encoded shape count.

$$PS = PsW = \frac{1}{\log(count(shape) + 1)} \tag{3.18}$$

The logarithm is inverted in order to give non-linear higher weights to rare paragraph shapes compared to the abundant ones. While, the add 1 normalization is used to avoid dividing by zero in case of a paragraph shape that existed only once, since *log(1) = 0*.

It is worth mentioning that the *Paragraph Score* is one of the terms used to calculate the overall sentence score as seen in equation 3.17.

3.4 Extract the Summary

After the document elements are scored properly, all sentences in the text are then arranged in a descending order of their score. Then, the top *N* sentences are selected, where the number of sentences to be selected (*N*) depends on the *Degree of Compression (DoC)* of the summary.

Degree of Compression (DoC) can be set as: a) a percentage of sentences or words to be extracted from the original text, or b) the number of words that should appear in the summary.

For example: *DoC* = 150e means that the extractive summary should contain around 150 words, while *DoC* = 50% means that the summary should be compressed to have a size that is 50% from the original text.
As such, the top sentences (with the highest scores) are extracted and ordered according to their order in the original text to produce the required summary.

3.4.1 A Working Example

Continuing with the working example of section 3.3.1.2., applying feature extraction and document element scoring routine resulted in the overall sentence score depicted in **Table 21**.

Table 21. The overall Sentence Score for the all 16 sentences of Document AP880911-0016 from DUC 2002 benchmark dataset. Scores embossed in bold are the top 6 sentences in terms of the SC.

Sentence ID	*Sentence*	*SC*
S1	Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas.	**0.252367896**
S2	The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.	0.155781083
S3	``There is no need for alarm, "Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.	**0.220551897**
S4	Cabral said residents of the province of Barahona should closely follow Gilbert's movement.	0.095416106
S5	An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.	**0.333612589**
S6	Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.	0.117738646
S7	The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.	**0.615467358**
S8	The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a ``broad area of cloudiness and heavy weather" rotating around the center of the storm.	**0.286516823**
S9	The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.	0.208073321
S10	Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet feet to Puerto Rico's south coast.	0.200377636
S11	There were no reports of casualties.	0.038714746

S12	San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.	0.131175308
S13	On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast.	**0.533937014**
S14	Residents returned home, happy to find little damage from 80 mph winds and sheets of rain.	0.161095457
S15	Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.	0.175060774
S16	The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.	0.15586512

By applying *DoC* = 150e, only six sentences were extracted and ordered to form an automatic summary made of approximately 150 words, representing 49.7% compression rate.

The final generated automatic summary of the document using UnB-LITS tool is as follows:

> **“*Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. “There is no need for alarm,” Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo. On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast.*”**

Chapter 4: Performance Evaluation of the language agnostic summarizer (UnB –LITS)

This chapter aims to evaluate the performance of the proposed language independent text summarization tool, *UnB-LITS*. The chapter is divided into three sections, the first section presents the conducted experiment and the datasets used as well as the evaluation metrics. The results and discussions are presented in the second section, and then we conclude the study in the third section.

4.1 The Experiment

4.1.1 Datasets

In order to assess the performance of the proposed tool against state-of-the-art summarization models listed in literature, a text summarization benchmark dataset was selected. *DUC2002*, a benchmark dataset provided by the American National Institute of Standards NIST for the Document Understanding Conference [156].

DUC2002[1] is considered one of the most widely used benchmark dataset for English document summarization task. It consists of 59 sets of document, each of which contains around 5-10 English news article, i.e. a total of 567 news articles.

Two assessors have manually summarized each article or set of articles, to be used as the evaluation standard. A Single article is summarized to a level of 100 words per summary. This rate of compression ranges from 40%, in case of documents with 250 words, down to 10% for documents with around 1000 words (which is considered a high compression rate).

Moreover, and for testing the ability of UnB-LITS to undergo an efficient text summarization in a language agnostic manner, three other datasets were used to assess the language agnostic capabilities of the proposed tool. Datasets of text documents in Portuguese, Spanish and French languages were selected.

For Portuguese language, *TeMario*[2], a Brazilian Portuguese text collection, is used. It consists of a collection of 100 news articles. The manually generated summaries are done with a degree of compression (*DoC*), ranging from 25% to 30% of the original documents.

While, in case of French and Spanish languages, two datasets 40 documents each, were obtained from French and Spanish news websites respectively. Each article has two manually generated summaries of 100 words length (i.e. *DoC* = 100e).

[1] http://www-nlpir.nist.gov/projects/duc/past_duc/duc2002/test.html
[2] http://www.linguateca.pt/Repositorio/TeMario/

4.1.2 Parameters Selection

The model parameters could have been adjusted by tenfold cross validation technique, using randomly selected articles. However, and in order to maintain the unsupervised nature of the model, the parameters were initially set in a way that gives equal weights to all terms of the sentence scoring equation as per **Table 22** below.

Table 22. Value of the parameters used for document element scoring in UnB-LITS.

Parameter	Value	Applied to
DoS Threshold	0.65	*Threshold for the degree of similarity between words*
α	1	*Apply weight to WsW and WfW to calculate WS*
λ_1	0.2	*The weight of WS used in calculating the SC*
λ_2	0.2	*The weight of 2GS used in calculating the SC*
λ_3	0.2	*The weight of 3GS used in calculating the SC*
λ_4	0.2	*The weight of 4GS used in calculating the SC*
λ_5	0.2	*The weight of 5GS used in calculating the SC*
η	1	*Weight applied to the sum of WS and nGS in a sentence*
β	1	*Weight applied to the SsW to compute the SC*
γ	1	*Weight applied to the SLU to compute the SC*
δ	1	*Weight applied to the PS to compute the SC*

4.1.3 Evaluation Metrics

The summarization results obtained are evaluated against the human generated summaries using, standard text summarization evaluation metrics known as ROUGE-1 and ROUGE-2 techniques [157].

The metric ROUGE stands for "*Recall-Oriented Understudy for Gisting Evaluation*". It is an evaluation metric for automatic text summarization task that does not require human annotation. ROUGE is considered the most commonly used intrinsic summarization evaluation metric, and was developed by Lin et al. [158], [159].

ROUGE is inspired by the BLEU metric, "*bilingual evaluation understudy*", used for evaluating machine translation output [160]. ROUGE evaluates the candidate of computer-generated summaries by measuring the amount of *n*-grams overlap between the candidate and human-generated summaries. ROUGE can be applied to measure the overlap of any type of *n-grams*. As such, *ROUGE-1* is used to measure the overlapping *unigrams*, while *ROUGE-2* and *ROUGE-3* are used to measure the overlapping *bigrams* and *trigrams* respectively.

To calculate the ROUGE metric for a computer-generated summary for a document *D*, first, one or more human candidates should summarize that document. Then the amount of overlapping (matching) n-grams between the computer-generated summary and each of the human generated ones is calculated. As such, ROUGE-n metric is equal to the overall sum of the aforementioned overlaps. equation 4.1 shows the calculation of ROUGE-2 metric using *bigrams* overlap.

$$ROUGE_2 = \frac{\sum_{S \in \{ref\ summaries\}} \sum_{bigrams \in S} Count_{match}(Bigrams)}{\sum_{S \in \{ref\ summaries\}} \sum_{bigrams \in S} Count(Bigrams)} \qquad 4.1$$

4.1.4 Compared Approaches

In this study, the language agnostic summarization performance of the proposed UnB-LITS model was assessed by comparing the obtained results to that obtained by commercial extractive summarization tools, as well as state-of-the-art extractive summarization approaches.

For the English and Portuguese benchmark datasets, the classification performance was assessed against state-of-the-art approaches that spans different categories of the extractive summarization tasks.

However, for the Spanish and French datasets, two commercial summarization tool were used, Apple’s integrated summarizer within macOS 12 “***Monterey***” and Autosummarizer.com.

4.2 Results and Discussions

In this section, we present in details the summarization of a single English document from DUC 2002 benchmark dataset. This document summarization case is used as a reference case study to be implemented on the entire 567 news articles, as well as the other three languages datasets in the following subsection.

4.2.1 Reference Case Study

In this section, we applied *UnB-LITS* to summarize English news article from the DUC 2002 datasets, article WSJ880912-0064 was used in this case study (sentences are listed in **Table 23**). In addition, the same article was summarized using the Text Summarizer integrated within Apple’s macOS 12 “***Monterey***”. Both summaries were compared against human generated summaries using ROUGE-1 and ROUGE-2.

The summarization were done at *DoC = 100e*, meaning that the summary should contain sentences with around 100 extracted words. The summaries are evaluated using ROUGE-1 and ROUGE-2 using the two human generated summaries, provided by DUC.

Table 23. Sentences of the article WSJ880912-0064

Sentence ID	Sentence
S1	Hurricane Gilbert swept toward Jamaica yesterday with 100-mile-an-hour winds, and officials issued warnings to residents on the southern coasts of the Dominican Republic, Haiti and Cuba.
S2	The storm ripped the roofs off houses and caused coastal flooding in Puerto Rico.
S3	In the Dominican Republic, all domestic flights and flights to and from Puerto Rico and Miami were canceled.
S4	Forecasters said the hurricane was gaining strength as it passed over the ocean and would dump heavy rain on the Dominican Republic and Haiti as it moved south of Hispaniola, the Caribbean island they share, and headed west.
S5	"It's still gaining strength.
S6	It's certainly one of the larger systems we've seen in the Caribbean for a long time," said Hal Gerrish, forecaster at the National Hurricane Center in Coral Gables, Fla.
S7	At 3 p.m. EDT, the center of the hurricane was about 100 miles south of the Dominican Republic and 425 miles east of Kingston, Jamaica.
S8	The hurricane was moving west at about 15 mph and was expected to continue this motion for the next 24 hours.
S9	Forecasters said the hurricane's track would take it about 50 miles south of southwestern Haiti.
S10	The hurricane center said small craft in the Virgin Islands and Puerto Rico should remain in port until conditions improve.
S11	The forecasters said the Dominican Republic would get as much as 10 inches of rain yesterday, with similar amounts falling in Haiti last night and tonight.
S12	Hurricane warnings were issued for the south coast of Haiti and Cuba by their respective governments.
S13	In Jamaica, the government issued a hurricane watch for the entire island.
S14	Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.
S15	In Puerto Rico, besides tearing off several roofs, the storm caused coastal flooding and brought down power lines and trees along roads and highways in the west and southwestern regions.
S16	Three people were injured in Guayanilla, Puerto Rico, when a tree fell on their vehicle as they traveled along Route 97, police reported.
S17	Four policemen stationed on Mona Island, between Puerto Rico and the Dominican Republic, were stranded as a result of the weather.

As seen in **Table 24**, by running the *UnB-LITS* on the selected article, the model has extracted 17 sentences, 354 words, 159 tokens (unique words) and 4 distinct word encoded shapes. In addition, by implementing *SLU* algorithm using *k*-means clustering (elbow method was used to determine the optimum number of *k*), a single cluster was detected, which returns to fact that the entire article is only in English language.

The four distinct encoded shapes are = {Xxc, xc, Nn, Nnxc}, where:

a) "Xxc" was repeated 51 times, as this code represents 34 named entities as Dominican, Republic, Gilbert and 17 words that start each of the 17 sentences in the article.

b) "xc" was repeated 293 times, as this code represents the majority of words in the article, as such it gets less weight.

c) "Nn" was repeated 8 times, as this code represents the numbers that were stated in the article as {3, 425, 15, ...}. This shape receives relatively higher weight when compared to the previous two shapes.

d) "Nnxc" a very rare shape code that occurred only once, encoding the word "100-mile-an-hour" that appeared in the first sentence, this shape code receives the highest shape score.

Figure 7 shows the word frequency graph for the unigrams of the original article before summarization.

Table 24. Document WSJ880912-0064 Result Analysis

Item	Value
Number of Sentences	17
Tokens	354
Unique Words	159
Unique word encoded shapes	4
Bigrams	278
Bigrams encoded shapes	12
Trigrams	301
Trigrams encoded shapes	20
4-grams	297
4-grams encoded shapes	32
5-grams	284
5-grams encoded shapes	47
Unique Sentence encoded shapes	1
Unique Paragraph encoded shapes	1
SLU *k* clusters detected	1

Moreover, hundreds of n-grams forms and encoded shapes were extracted. In case of bigrams, the most abundant shape was "xc xc" as it represents the majority of the text. However, Xxc Xxc was significant in identifying NE's as "Dominican Republic", "Hurricane Gilbert", "Coral Gables", etc.

While in the case of trigrams, the shape “Xxc Xxc Xxc” was capable of identifying an important NE in this context, “National Hurricane Center”.

It is worth mentioning that the number of unique shapes for sentences and paragraphs is only 1 for both, as all sentences start with capital letter and dominated by lower case words, thus the only sentence shape in the text was “***Zzz***”.

While, all paragraphs have the same shape “***OSPpp***”, i.e. ordinary single sentence dominated by lower case letters despite a capital initial letter. The reason for obtaining this single sentence paragraph encoded-shape, is because the article was originally provided, from DUC, as segmented text with every sentence form its own paragraph.

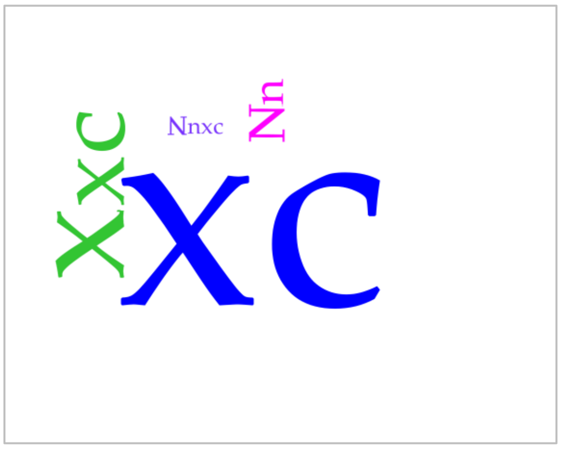


Figure 7. Unigram frequency graph for encoded word shapes.

In addition, **Figure 8** shows the unigram influence graph. It is clear that the most influential words are Hurricane (repeated 9 times with XC shape), Dominican (6), Republic (6), Puerto (6), Rico (6). This high influence is reflected when computing their individual scores, in terms of their word and encoded shape frequency weights.

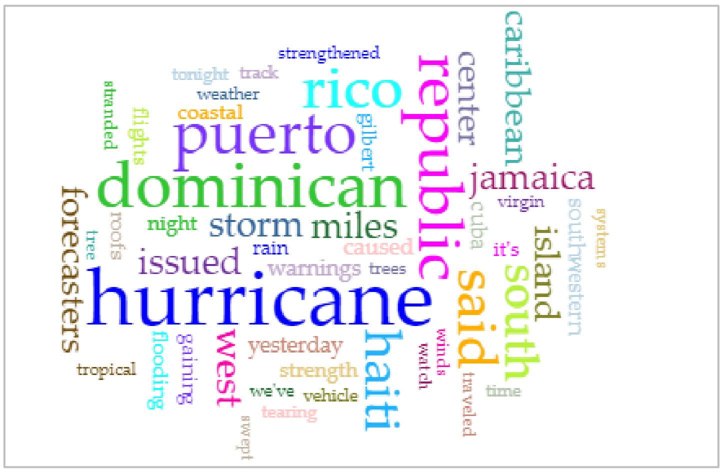


Figure 8. Word frequency graph for article WSJ880912-0064

As such, and after computing the overall sentence score for each of the 17 sentences, the top scored sentences are listed in **Table 25** (in descending order of their score), where it is clear that UnB-LITS was capable of identifying the most influential NE's and CC's.

Since the summary is limited to 100e words only, thus only the top four sentences were selected and then ordered according to their original order in the article, to generate the summary.

Table 25. The top five Sentences of the Document

ID	Score	Sentence
S1	**0.959**	Hurricane Gilbert swept toward Jamaica yesterday with 100-mile-an-hour winds, and officials issued warnings to residents on the southern coasts of the Dominican Republic, Haiti and Cuba.
S7	**0.814**	At 3 p.m. EDT, the center of the hurricane was about 100 miles south of the Dominican Republic and 425 miles east of Kingston, Jamaica.
S6	**0.189**	It's certainly one of the larger systems we've seen in the Caribbean for a long time," said Hal Gerrish, forecaster at the National Hurricane Center in Coral Gables, Fla.
S16	**0.159**	Three people were injured in Guayanilla, Puerto Rico, when a tree fell on their vehicle as they traveled along Route 97, police reported.
S10	**0.130**	The hurricane center said small craft in the Virgin Islands and Puerto Rico should remain in port until conditions improve.

As discussed above and shown in **Figure 8**, the model was capable of identifying influential terms due to the rareness of their encoded shapes and forms. Among the top important terms/entities that the model has successfully identified during the scoring process, are: Hurricane Gilbert, Dominican Republic, Hal Gerrish, Jamaica, Caribbean, Puerto Rico, Virgin Islands, etc.

The generated summary by UnB-LITS is as follows:

"***Hurricane Gilbert swept toward Jamaica yesterday with 100-mile-an-hour winds, and officials issued warnings to residents on the southern coasts of the Dominican Republic, Haiti and Cuba. It's certainly one of the larger systems we've seen in the Caribbean for a long time," said Hal Gerrish, forecaster at the National Hurricane Center in Coral Gables, Fla. At 3 p.m. EDT, the center of the hurricane was about 100 miles south of the Dominican Republic and 425 miles east of Kingston, Jamaica. Three people were injured in Guayanilla, Puerto Rico, when a tree fell on their vehicle as they traveled along Route 97, police reported.***"

The extracted summary was then evaluated using ROUGE-1 and ROUGE-2 against human-generated summaries and compared to the summaries done by Apple macOS 12 integrated summarizer. The results are shown in **Table 26** below:

Table 26. Evaluating the results of single English document summarization

Standard File	Tool	ROUGE 1	ROUGE-2
A	UnB-LITS	***61.2***	***36.3***
	macOS Summarizer	49.0	18.6
B	UnB-LITS	***51.0***	***18.0***
	macOS Summarizer	46.8	14.6
Total	UnB-LITS	***56.3***	***27.7***
	macOS Summarizer	47.9	16.8

As seen from the results above, UnB-LITS was able to summarize the news article successfully, achieving 61% success when compared to human-generated summaries. UnB-LITS has outperformed Apple's macOS summarizer in both evaluation metrics, ROUGE-1 and ROUGE-2.

4.2.2 Comparing to State-of-the-Art approaches

UnB-LITS was applied on two benchmark datasets, *DUC2002* for English text summarization, and *TeMario* for Portuguese text summarization. The obtained summaries were evaluated against human generated summaries using ROUGE-1 and ROUGE-2.

The results are then compared to state-of-the-art approaches reported in literature. The state-of-the-art methods were chosen to span the different categories of extractive summarization approaches discussed in chapter 2.

Table 27 and **Table 28** compare the results of UnB-LITS to those reported in literature for DUC2002 and *TeMario* datasets respectively.

With respect to DUC 2002 benchmark dataset, and as seen from the **Table 27**, **Figure 9** and **Figure 10**, the proposed UnB-LITS has outperformed all state-of-the-art models in terms of ROUGE-1 metric, and scored better performance than 69% of those models in terms of ROUGE-2 metric.

Table 27. Comparing UnB-LITS against state-of-the-art methods applied on the entire 567 articles in the DUC2002 benchmark dataset for single document summarization task.

Category	*Tool*	*ROUGE-1*	*ROUGE-2*	*REF*
Statistical-Based	***UnB-LITS*** (Language agnostic elements scoring)	***53.54***	*25.97*	-
Topic based	Topic modeled unsupervised clustering	49.35	**31.53**	[98]
	DeepSum (topic modeling and word embedding)	53.2	28.7	[161]
Graph based	Topic Modeling based on weighted graph representation	48.10	23.3	[162]
	CoRank (word–sentence relationship and graph-based ranking model)	52.6	25.8	[163]
Machine Learning based	BERT based extractor and LSTM pointer network	43.39	19.38	[61]
	Word2vector embedding	38.25	22.56	[164]
	SummCoder (deep auto-encoders)	51.7	27.5	[165]
	SummaRuNNer(RNN-based sequence classifier)	47.4	24.0	[166]
	HSSAS (Neural Network Classifier)	52.1	24.5	[167]
	TCNN (combines NN and LexRank)	44.3	19.68	[168]
Discourse-Based	FNARS (hierarchical Narrative Summaries – fully structured)	48.3	28.3	[169]
	SNARS (hierarchical Narrative Summaries – semi structured)	52.9	24.8	[169]

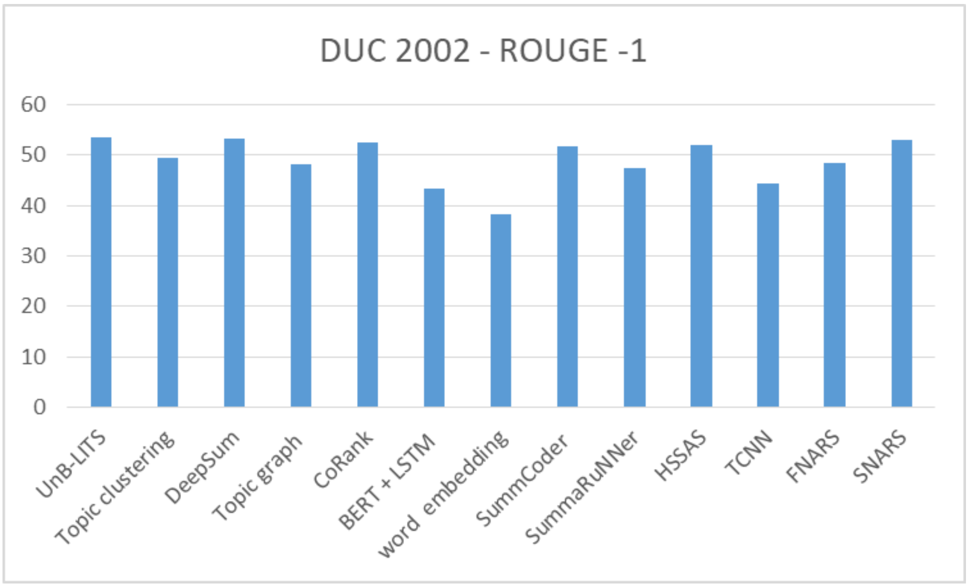


Figure 9. ROUGE-1 results for UnB-LITS against state-of-the-art approaches applied on DUC 2002 benchmark dataset.

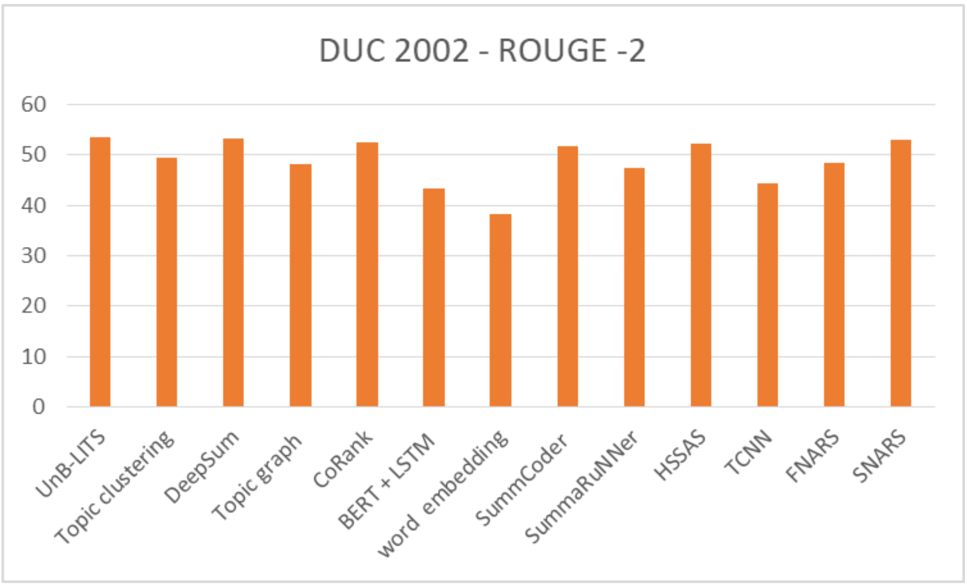


Figure 10. ROUGE-2 results for UnB-LITS against state-of-the-art approaches applied on DUC 2002 benchmark dataset.

On the other hand, and with respect to Portuguese Language, UnB-LITS was applied to the *TeMario* benchmark dataset. The generated summaries were 25-30% of the size of the original documents.

The obtained results were then compared to that reported in literature [170], **Table 28**, **Figure 11** and **Figure 12** shows that the proposed UnB-LITS has scored better results than all state-of-the-art approaches reported in literature on the *TeMario* dataset.

Table 28. Comparing UnB-LITS to state-of-the-art methods applied to *Temario* Portuguese benchmark dataset for single document summarization task.

Tool	*ROUGE-1*	*ROUGE-2*
UnB-LITS	***0.57***	***0.22***
MMR (λ =0.5)	0.43	0.15
Support Sets (Manhattan Distance and Support set cardinality = 2)	0.52	0.19
KP-Centrality (10 Key Phrases)	0.54	0.20
LSA	0.56	0.20
GRASSHOPPER	0.54	0.19
LexRank	0.55	0.20

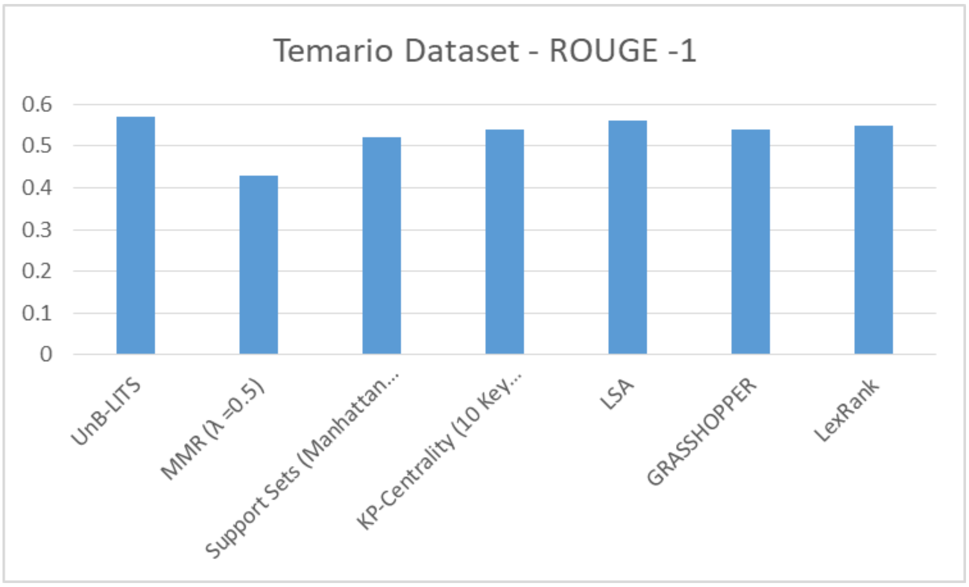


Figure 11. ROUGE-1 results for UnB-LITS against state-of-the-art approaches applied on *Temario* Portuguese benchmark dataset.

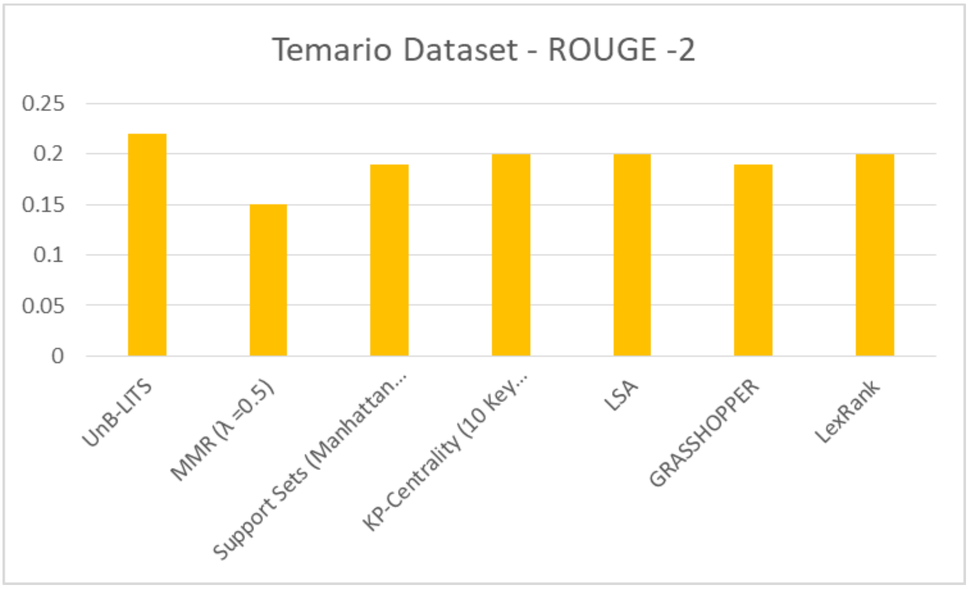


Figure 12. ROUGE-2 results for UnB-LITS against state-of-the-art approaches applied on *Temario* Portuguese benchmark dataset.

4.2.3 Applying UnB-LITS on Spanish and French datasets

Regarding, Spanish and French Languages, UnB-LITS was applied to generate a summary of 100 words. **Table 29** compares the summarization performance of UnB-LITS to commercial tools applied on the same datasets.

The depicted results show that our proposed model has achieved superior results when compared to those obtained by commercial tools, proofing the efficiency of the language agnostic nature of the model.

Table 29. UnB-LITS summarization performance for Spanish and French datasets.

Dataset	*Tool*	*ROUGE-1*	*ROUGE-2*
Spanish News Dataset	***UnB-LITS***	***0.6778***	***0.4652***
	Apple macOS 12	0.5310	0.2451
	Autosummarizer	0.5377	0.2640
French News Dataset	***UnB-LITS***	***0.5762***	***0.3389***
	Apple macOS 12	0.5075	0.2760
	Autosummarizer	0.4577	0.2221

Chapter 5: Deep Self-Organizing Cube (DSOC)

In this study, we propose a new Deep Learning classifier, the *Deep Self-organizing Cube*, (*DSOC*) that can be used for MDC, MLC, MCC as well as BC. DSOC takes into consideration class dependencies across class spaces while implementing multi-output learning, where classification across all dimensions is performed simultaneously. Due to the self-organizing nature of DSOC's *Pooling* layer as well as its probabilistic *Hypercube*, the model overcomes the problem of class imbalance among training dataset instances.

As such, the main contribution of this study is as follows: a) we propose an efficient, yet straightforward multidimensional deep learning classifier, the "DSOC". b) The model we designed has an embedded variable selection layers to achieve dimensionality reduction while increasing the discriminatory power of the model and achieve appropriate class segregation. c) This work experimentally demonstrates the effectiveness of DSOC in all types of classification tasks, regardless the severity of data imbalance or the strength of class dependencies, due to its design that models semantics among classes along different classification spaces even in case of imbalanced datasets.

This chapter and the following one are structured as follows; we introduce the proposed DSOC classifier in section 1, followed by the applied experiment in section 2, the results and discussions in section 3. Finally, we conclude the model in section 4.

5.1 Deep Self-Organizing Cube (DSOC)

In this section, we propose a new Deep Learning classifier that can be used in all four types of supervised classification tasks (BC, MCC, MLC and MDC). The proposed model is called "*DSOC*" a short for "*Deep Self Organizing Cube*". The model is based on the well-established concept of Self-Organizing Maps (SOM) introduced by [171] and the Once Learning approach introduced by [172] that adapts the SOM training routine into all-at-once learning, imitating the human brain activity of learning just once, "Once Seen Never Forgotten" approach.

In "*Deep Self-Organizing Cube*", the word "Deep" refers to the nature of the model that involves a multi-layer system of connected neurons, arranged in multiple abstraction levels (dimensions). While, the term "*Self-Organizing*" refers to the ability of the neurons within the model to organize themselves into various arrangement, enabling the "*DSOC*" to model inter and intra class dependencies. Finally, the word "*Cube*" refers to the multi-dimensional hypercube that lies at the center of the multi-layer neuron system.

In short, “*DSOC*” is a deep learning classifier, which consists of multi-layer neuron system connected to a central hypercube. The hypercube is an *n*-way cube formed of output neurons arranged in orthogonal plans, where each plan represents a classification dimension of a class apace.

5.1.1 *DSOC components*

Despite the multi-dimensional and multi-layer nature of the DSOC model, its structure can be divided into two major components, the *Hypercube Classifier* component, and the *Deep Neural Network* component.

5.1.1.1 The DSOC Hypercube Classifier

The *DSOC's* hypercube component is a multi-way cube responsible for the multi-output classification task across all class spaces. The hypercube is formed of *n* dimensions, each of which represents a classification space (*C*) with its own set of classes. Where, an *n-way* hypercube consists of *n* class spaces, i.e. ($Hypercube = C^1 \times C^2 \times \cdots \times C^n$), with each class space formed of *m* classes ($C^n = \{C_1^n, C_2^n, \dots, C_m^n\}$). As such, a 3-dimensional hypercube is formed of *N* output neurons equal to the product of class members of all class spaces, i.e. ($N = C^1 \times C^2 \times C^3$). Each neuron $c_{1,2,\dots,n}$ represents a combination of classes in the *n*-dimensional cube. The classification concept of the entire DSOC model is to find the winning neuron in the *Hypercube classifier*. Therefore, the final classification output of the model is a class vector (*Y*) of size *n* whose members represent the assigned classes of the winning neuron across the *n* dimensions, i.e. ($Y = [y^1, y^2, \dots, y^n]$).

From a different viewpoint, in case of a 3-dimensional *DSOC*, the hypercube classifier can be viewed as a stack of slices/plans of size ($C^1 \times C^2$) each of which represent a two-dimensional classification plan for a single class space (C^3) along the third dimension. For example, in a problem where text scientific papers are to be classified according to their *Topics*, *Languages* and scientific *Fields*. *DSOC* algorithm considers this example a 3-way multidimensional classification problem along three linked but discrete modes, of size $C^1, C^2$and C^3 respectively. As seen in **Figure 13**, the three-dimensional *hypercube classifier* could be seen as: a) a stack of C^2 two-dimensional plans of size ($C^1 \times C^3$) along the *Languages* dimension; b) a stack of C^3 two-dimensional plans of size ($C^1 \times C^2$) along the *Fields* dimension; or c) a stack of C^1 two-dimensional plans of size ($C^2 \times C^3$) along the *Topics* dimension.

Using the previous example, if the DSOC algorithm has selected a winning neuron ($c_{2,4,3}$) whose coordinates in the hypercube classifier are (2, 4, 3), then this neuron represents a document whose *Topic* belongs to class 2, written in *Language* 4 and under *Scientific Field* 3.

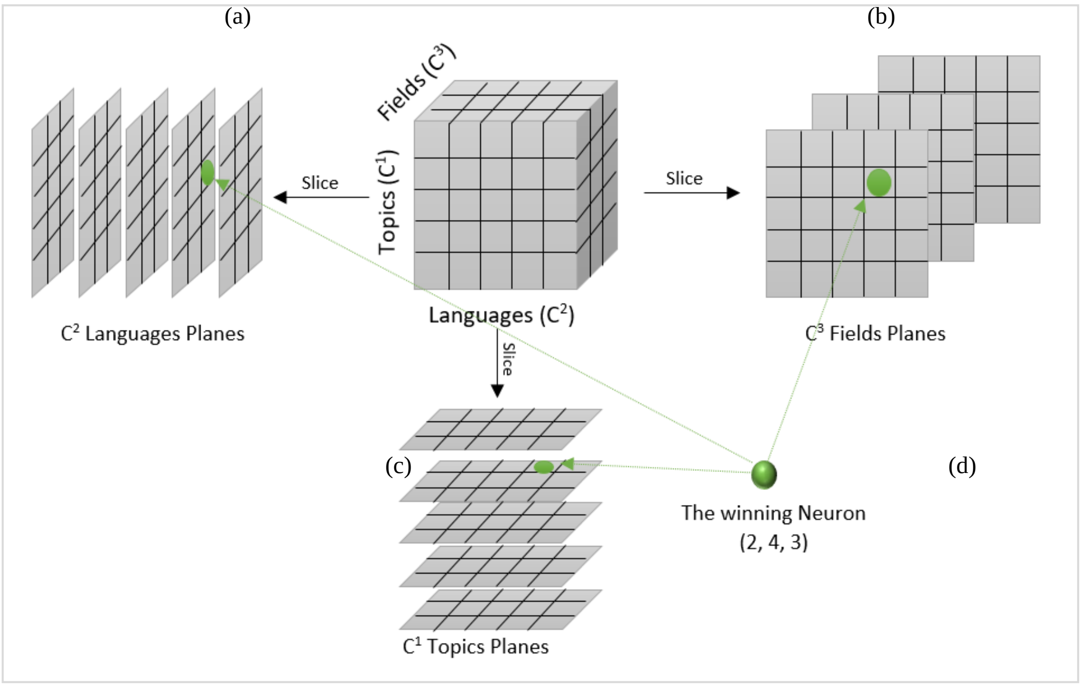


Figure 13. A three-dimensional hypercube classifier of size ($\boldsymbol{C^1} \times \boldsymbol{C^2} \times \boldsymbol{C^3}$). a) $\boldsymbol{C^2}$ Slices along the Language dimension. b) $\boldsymbol{C^3}$ slices along the *Fields* dimension; and c) $\boldsymbol{C^1}$ slices along the *Topics* dimension. d) The winning neuron at hypercube coordinates (2,4,3).

5.1.1.2 The Deep Neural Network Component

In order for the *DSOC's Hypercube classifier* to emit the classification vector (*Y*) across all dimensions, one of its output neuron has to be activated. The winning neuron activation process occur through a stack of deep neural networks, that we call "*DSOC Neural Network*", with one network for each dimension of the model. Therefore, in an *n*-dimensional DSOC model there are *n* "*DSOC Neural Networks*" that are connected to all *N* output neurons of the DSOC's *Hypercube*.

As seen in **Figure 14**, a single "*DSOC Neural Network*" consists of three main layers of neurons:

a) *Input layer*, consists of *P* neurons corresponding to sample input features (*p*).

b) *VSC double layer*, which consists of two connected layers, *VSC Selector* layer and *VSC Scaling* layer, this double layer depends on the *Variables Selection Coefficient* (VSC) algorithm introduced by [173].

c) *Pooling layer*, a Gaussian Probability function layer, whose neurons are directly connected to the DSOC hypercube classifier, and responsible for activating the hypercube winning neuron.

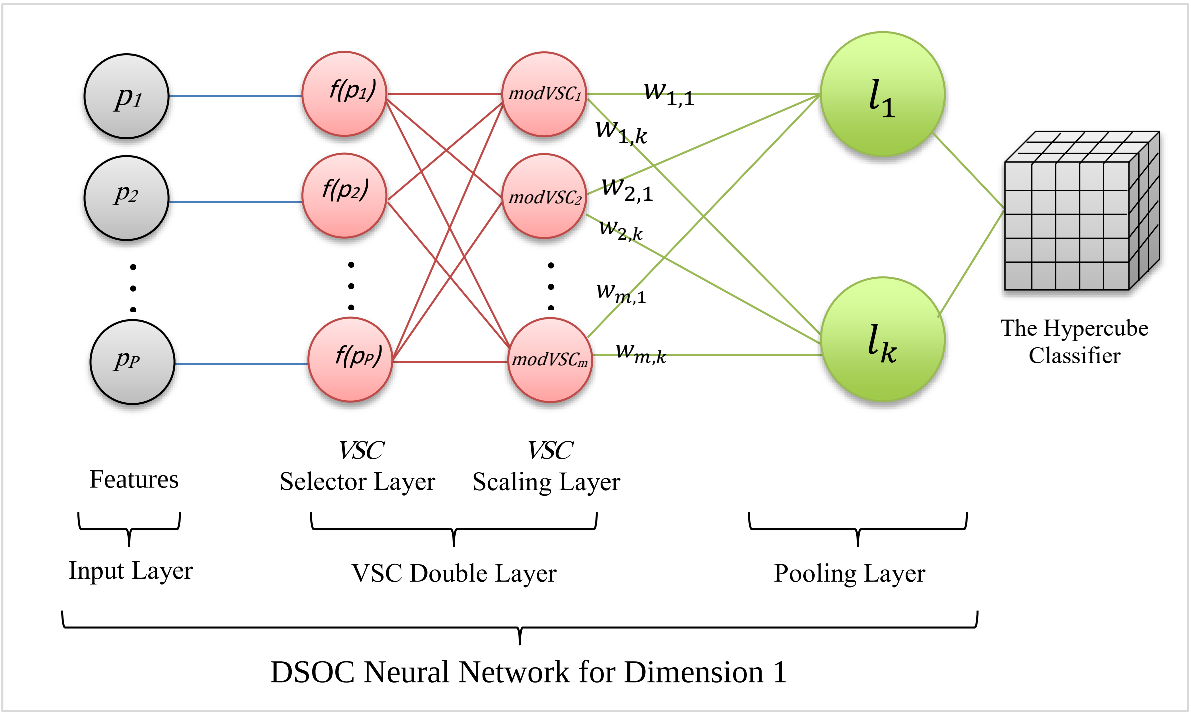


Figure 14. The DSOC Neural Network for Dimension 1.

5.1.1.2.1 The VSC Double Layer

Variable Strength Coefficient (VSC) is the overall measure of strength of a variable (feature) in fitting the data, as well as discriminating the classes within a class space. VSC combines, linearly, both *Modeling Power* of a variable for all classes in the model with the *Discriminatory Power* of the same variable for the same class space. The *Modeling Power* is the assessment of the ability of a variable to model data of a specific class, while *Discriminatory Power* is the assessment of its ability to discriminate between two classes. For example, for a variable *i* in class *j* the *Modeling Power* $\boldsymbol{M}_i^j$, and its *Discriminatory Power* $\boldsymbol{D}_i^{j,j+1}$ for classes *j* and *j+1*, are calculated as follows [174]:

$$\boldsymbol{M}_i^j = 1 - \frac{\boldsymbol{S}_{iraw}^j}{\boldsymbol{S}_{iresid}^j} \qquad 5.1$$

$$\boldsymbol{D}_i^{j,j+1} = \sqrt{\frac{j\ model\ (j+1)_{S_{iresid}}{}^2 + (j+1)\ model\ j_{S_{iresid}}{}^2}{j\ model\ j_{S_{iresid}}{}^2 + (j+1)\ model\ (j+1)_{S_{iresid}}{}^2}} \qquad 5.2$$

Where $\boldsymbol{S}_{iraw}^j$ is the standard deviation of variable *i* in the raw data and $\boldsymbol{S}_{iresid}^j$ is the standard deviation of the residuals for the same variable *i* in the model.

The VSC for a variable i ($\boldsymbol{VSC}_i$) is then calculated by linearly combining the overall modeling power of that variable ($\boldsymbol{M}_i$) along all classes in the class space, with its overall discriminatory power ($\boldsymbol{D}_i$) as shown in the equations below [175].

$$\boldsymbol{M}_i = \sum_{j=1}^{l} \boldsymbol{M}_i^j \cdot \frac{n^j}{N} \tag{5.3}$$

$$\boldsymbol{D}_i = \sqrt{\frac{\sum_j^l \sum_c^l j\ model\ c\ {s_{\boldsymbol{iresid}}}^{\boldsymbol{2}}}{\sum_j^l j\ model\ j\ {s_{\boldsymbol{iresid}}}^{\boldsymbol{2}}}} \tag{5.4}$$

$$\boldsymbol{D}_i = \frac{\boldsymbol{D}_i - \min\{\boldsymbol{D}\}}{\max\{\boldsymbol{D}\} - \min\{\boldsymbol{D}\}} \tag{5.5}$$

$$\boldsymbol{VSC}_i = \frac{\boldsymbol{w} \cdot \boldsymbol{M}_i + (2 - \boldsymbol{w})\, \boldsymbol{D}_i}{2} \tag{5.6}$$

In the *DSOC Neural Network* component, the purpose of the *VSC double layer* is strengthening the discriminatory power of the model, as well as inducing proper segregation of adjacent classes, especially in models with dense class spaces, and overlapping class boundaries. This purpose is achieved through the following:

a) *VSC Selector Layer*, a VSC strength threshold is determined, where an input feature with a stronger VSC (i.e. the feature's VSC is higher than the predetermined threshold) will pass to the following layer, while weaker features are discarded and excluded from the model.

b) *VSC Scaling Layer,* this layer contains a modified version the VSC equation, where the VSC weight variable $\boldsymbol{w}$ is not restricted to a value between [0,2] as in the original coefficient, but could take any value greater than zero. As such, a strong variable *i* that has passed successfully through the previous *VSC Selector layer* is then scaled up or down by that dynamic modified coefficient (*modVSC$_i$*).

The *VSC double layer* results in: a) dimensionality reduction of input features, by discarding variables with weak modeling and discriminatory powers, and b) independently scale each of the remaining features. As such, adequate segregation between classes is achieved, which is then reflected in the overall classification performance of the DSOC model.

The *VSC Selector* threshold as well as the weight variable of the *VSC Scaling* layer are both determined through cross-validation procedure during the training process of the DSOC model.

The effect of *VSC double layer* on the performance of the model is discussed in the following chapter.

5.1.1.2.2 The Pooling Layer

A single pooling layer consists a set of neurons (L) of size $C^n \times \alpha^n$, where C^n is the number of class members in this dimension and α^n is a scaler indicating the number of neurons that models each class member in a class space. The default value of α^n is 1, however it could be set to any value greater than or equal to 1. The optimum value of α^n is decided through cross validation routine. For example, in case of a binary class space, the number of neurons constructing this pooling layer is $2 \times \alpha^1$, i.e. the pooling layer is formed of at least two neurons ($L^1 = \{l_1, l_2\}$). Moreover, if $\alpha^1 = 3$, this means that each class is modeled by 3 neurons, and the total number of neurons forming the layer is 6 (i.e. $L^1 = \{l_1, \dots, l_6\}$).

Each neuron (l) in the pooling layer has a weight vector (w) that connects it to the *VSC Scaling* layer. The size of w is equal to the number of neurons (M) in the *VSC Scaling* layer.

5.1.2 *Designing the DSOC model*

In order to design a *DSOC* model, certain parameters should be specified. Those parameters include; the number of classification dimensions (n), the size of each dimension (C) which is the number of classes modeled by that dimension, in addition to the value of the alpha parameter (α) per dimension.

5.1.2.1 Number of Dimensions (n)

The number of dimensions in a DSOC model refers to the number of class spaces in the model. Where, in case of BC and MCC the number of dimensions (n) is equal to1 since those tasks have single output class space, while in case of MLC and MDC tasks, n is equal to the number of class variables (spaces) in the dataset.

Both DSOC components, the *Hypercube Classifier* and the *DSOC Neural Network*, have the same number of dimensions; therefore, by setting the value of n, we construct an *n-dimensional hypercube* as well as *n DSOC Neural Networks* connected to it.

5.1.2.2 Dimension Size (C)

The size of dimension (C) is simply the number of class members in a specific class space (dimension), or the number of possible values a class variable can have. In binary class spaces, where the class members could be {0, 1} or {-1, 1}, then the dimension size is 2 ($C = 2$). The number of classes per dimension determines the size of the *Hypercube Classifier* as well as the minimum number of neurons in the *Pooling* layer in each of the *n DSOC Neural Networks*.

For example, in case of the 3-dimenstional *DSOC* model mentioned in section 5.1.1.1, the model has three classification dimensions representing (topics, languages and fields), therefore the size of every dimension depends on the number of possible classes C in that class space. As such, if the topics class space has 5 possible values, while fields and languages have 3 and 5 possible values respectively, (i.e. $C^{Topic} = 5$, $C^{Field} = 3$ and $C^{Lang} = 5$), then the *DSOC* model is formed of a 3-dimensional *Hypercube Classifier* of size (5 × 3 × 5 = 75 neuron), and 3 *DSOC Neural Networks* whose *Pooling* layers have $5\alpha^1$, $3\alpha^2$ and $5\alpha^3$ neurons respectively.

5.1.2.3 The Alpha parameter (α)

The alpha parameter α^n, is the parameter that decides the number of neurons that models each class in a particular dimension and span the variance among its members . For example, if α^2 = 10, thus each class will be modeled by 10 neurons in the *Pooling* layer of the second *DSOC Neural Network* component. Moreover, if $C^2 = 5$ then the corresponding *Pooling* layer will have 50 neurons. As mentioned earlier, the default value of α^n is 1, but other values could be used after performing the cross validation routine during the training step.

Every neuron in the *n Pooling layers* of the model has a weights vector (w^n) of size M^n which is equal to the number of neurons in the previous *VSC Scaling* Layer. The weights of this vector are determined through the training process.

It is worth mentioning that the number of neurons in the *VSC Selector* layer is equal to the number of input neurons or features P^n in that dimension, while the number of neurons in the *VSC Scaling* Layer M^n is less than or equal to the number of neurons in the *VSC Selector* layer, i.e. $M^n \leq P^n$.

5.1.3 Deciding the Training Parameters of the DSOC Model

DSOC model training parameters are those determining the rate by which the Pooling neurons weights vectors (w^n) are being updated, as well as the range of neurons to be updated in each step of training. There are three main training parameters that should be decided before training the *DSOC* model, these parameters are: the number of *iterations* or *epochs* (*h*), the *learning rate (R)* and the *neighborhood distance* (*nD*).

5.1.3.1 Number of Iterations or Epochs (h)

The number of iterations, which is also called (epochs), is the number of times by which the entire training set is presented to the *DSOC* model for training and weights update. In this research, the default value is 100.

5.1.3.2 Learning Rate (R)

Learning rate is the rate by which the neurons weights vectors are updated at each iteration *h* as a function of time *t*. The learning rate should have a value greater than 0 and less than or equal to 1. In this study, the learning rate is initiated at ≈ 0.7-0.8 then decreases exponentially with time. It is worth mentioning that the word “time” with its notation *t* in this context refers to each time a training sample is presented to the cube. As such, the total “time” *T* of the training process equals to the product of the number of training samples and the number of iterations or epochs, as seen in equation 5.7.

$$T = h \times count(samples) \tag{5.7}$$

In this study we propose that the value of the learning rate as a function of time *t* is calculated as shown in equation 5.8 5.8below.

$$R(t) = e^{\left(-\frac{dt}{T}\right)} + 0.01 \tag{5.8}$$

Where, $R(t)$ is the learning rate as a function of time *t* and *T* is the total training time as calculated in equation 5.8 above while $e^{(\dots)}$ is the natural exponential function (*exp*) and *d* is the decay parameter. **Figure 15 (a)** demonstrates the effect of the decay parameter *d* on the learning rate. In this research we used *d* = 6 and the initial learning rate = 0.83.

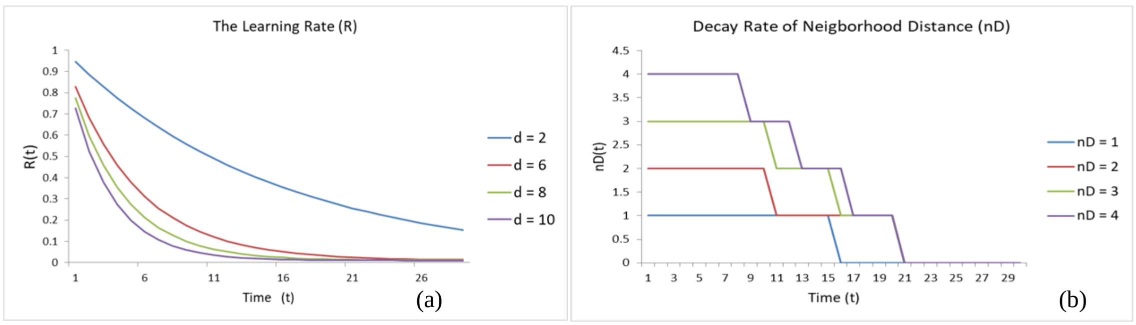


Figure 15. DSOC training parameters. a) the learning rate *R(t)* as a function of time t, using different values for the decay parameter *d*. b) The rate of decay of the neighborhood distance *nD(t)* as a function of time.

5.1.3.3 The Neighborhood Distance (nD)

The neighborhood distance *nD* determines the number of neurons in the *Pooling Layers* whose weights are to be updated at each time *t*. This process aims at rearranging the *Pooling* neurons in a manner that reflects their class similarities in terms of features and properties. For example, suppose we construct a language classifier, it is expected that Portuguese and Spanish languages be represented by very close neurons while oriental languages are represented by distant neurons due to the intrinsic dissimilarities among their language families.

A decay function is applied to the *nD* as well so that the neighborhood of neurons decreases with time *t*. Based on experience, the model designer decides the neighborhood rate of decay as a function of time *nD(t)*. In this study, the default neighborhood distance was set to 1 (i.e. *nD = 1)* across all dimensions.

Figure 15 (b) shows the rate of decay of *nD(t)* as a function of time *t*. As noticed from the graph, when the initial value of *nD* is set to 1, then it reaches zero at half the total time of training (*T*). On the other hand, higher *nD* keeps its initial value during the first one third of the training time, and then decreases linearly until it reaches 0 at the beginning of the last third of training.

5.1.4 *Model Training*

DOSC model training is the process of updating the weights of the weight vectors (w^n) of all neurons in the *n-Pooling* layers of the model, and then build the probability model of the *n*-dimensional *Hypercube Classifier.* The training process aims to make every neuron of the *hypercube* capable of representing a set of dependent classes across all class spaces simultaneously. As such, the training process of the *DSOC* model is performed in two stages:

5.1.4.1 Updating the weights vectors (w^n)

Hereafter we present the weights update process in a specific dimension (*j*). The same weights update process is then repeated in all *n* dimensions of the model. The process of weights update reorganizes the *Pooling* neurons and bring them closer to the class they represent. As such, for any particular dimension in the *DSOC* model, the weights update process is performed as follows:

i. *Initialize the weights vectors:* For all neurons (L^j) in the *Pooling* layer, randomly initialize a single weights vector (w^j) of size M^j, which is equal to the number of neurons in the previous *VSC Scaling* layer in dimension *j*.

ii. *Start the first Epoch*: for each sample (*i*) in the training set, compute its *Euclidian distance* (ED_i^j) from the pooling weights vector. For each samples in the model, ED_i^j is computed as the distance between the *VSC Scaling neurons* (v^j) and (w^j) in the *DSOC Neural Network* of dimension *j*.

$$ED_i^j = SQRT(\sum_{q=1}^{|M^j|} \left(v_q^i - w_q^j\right)^2) \qquad 5.9$$

iii. *Get the winning neurons*: the neuron with the least *ED* is the winning neuron (u).

iv. *Get the sub-set of neighboring neurons*: For each winning neuron get a subset of neighboring neurons according to the neighborhood distance *nD*.

v. *Update the weights of the winning neuron and its neighbors*: The weight update is done using equation 5.10 below for both the wining neuron and its neighbors. The magnitude of weights update depends on the learning rate of the model.

$$w^j\,(t+1) = w^j\,(t) - R(t)(\,v^j - w^j\,(t)) \qquad 5.10$$

vi. *Repeat until end of Epochs*: A single epoch means going through the entire training set and applying the steps of finding the winning neuron and weights update for all the samples in the training set. After finishing the first epoch, repeat the steps from *ii* to *v* on the entire training set until the maximum number of epochs *h* is reached.

5.1.4.2 *Build the probabilistic model and construct the Hypercube Classifier*

The last few steps of training the model are performed based on the Gaussian Naïve Bayes concept. The remaining steps aim at constructing the *Hypercube* classifier and populate it with the classification prior probabilities. These steps are carried out as follows:

5.1.4.2.1 *Construct the hypercube*

i. Create *n* dimensional hypercube of size $\prod_{j=1}^{n} C^j$ neurons where each neuron of the cube represent a combination of classes across all dimensions of the model.

ii. For each neuron (*r*) in the hypercube, set its initial value to 1 (to account for class imbalance and rare combinations with no training samples).

iii. Iterate through the training samples, where the value of the hypercube neuron (*r*) is incremented by 1 if, and only if, the class vector (Y_i) of that sample (*i*) exactly matches the combination of classes of that neuron.

iv. Calculate the probability of each neuron (*r*) in the cube. This is considered the prior probability of that neuron given the combinations of class members of all *n*-class spaces.

v. Each of the hypercube neurons is connected to the *Pooling* layer of each dimension through a weight vector (b^j) of size equals to the number of neurons L^j in that *Pooling* layer. For example, in a 3-dimensional DSOC model, each neuron in the *hypercube classifier* has three weights vector (b^1, b^2 and b^3) of size L^1, L^2 and L^3 respectively (see **Figure 16**). The members of (b^j) vector are the prior probabilities of neurons L^j given a class member $\{c \in C^j\}$.

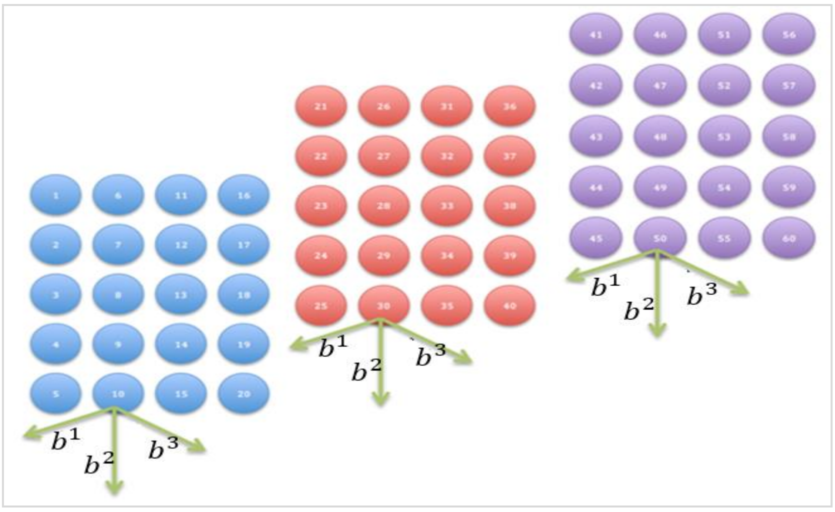


Figure 16. A DSOC *Hypercube classifier* of size 5 x 4 x 3 sliced along the C^3 dimension into 3 plans of size 5 x 4. Every neuron has three weights vectors (b^1, b^2 and b^3) connected to the *Pooling* layers of the three DSOC Neural Networks. Each neuron of the *Hypercube* represent a combination of classes from the three class spaces of the model.

5.1.4.2.2 Build the probabilistic model

For every neuron (l) of the L^j neurons of the *Pooling Layer* of dimension j, do the following:

i. Compute the standard deviation and mean *ED* for all samples belonging to each class of the dimension class space (C^j).

ii. Compute the probability when this neuron was activated, given each class in (C^j) as per equation 5.11. This is considered the prior probability of that neuron given a specific class.

$$p(l_k|C_q^j) = \frac{\sum active(l_k|C_q^j)}{\sum samples|\, C_q^j} \qquad 5.11$$

iii. These three values are used during the classification process of new samples.

5.1.5 *Classify Future samples*

When a new sample (i) is introduced to the DSOC, the classification process is performed in the following steps in each dimension j of the n dimensional model:

i. Sample features are presented to the *VSC double layer* of the DSOC Neural Network j, where sample features that have registered weak VSC during the training process are discarded by the *VSC Selector* layer and only strong features pass through to the *VSC Scaling* layer for feature independent scaling.

ii. The surviving neurons are then presented to the *Pooling layer* of dimension j, where the ED_i^j is computed between the (v^j) and (w^j) as per equation 5.9.

iii. For each neuron (l^j) of the L^j neurons, use the computed ED_i^j as well the mean and standard deviations computed in 5.1.4.2.2 to compute the likelihood of that neuron, $Likelihhod\left(l_k^j \mid c_q^j\right)$, given each class c_q where $\left\{c_q \in C^j\right\}$.

iv. The previous steps (*i* to *iii*) are repeated for all *n* dimensions in the model.

v. As per equation 5.12, for every neuron (*r*) in the *Hypercube Classifier*, Compute the sum of the natural log of its vector (b^j), which is the likelihoods computed for the *Pooling layer* in the previous step as well as the prior probabilities computed in equation 5.12, given the class represented by that neuron in each dimension (c_q^j). Add the result to the natural log of the neuron's prior probability computed in 5.1.4.2.1 to calculate the probability $p(r)$ that this neuron is activated.

$$p(r) = \sum_{i=1}^{j}\sum_{k=1}^{L} \ln(Likelihhod\left(l_k^j \mid c_q^j\right)) + \sum_{i=1}^{j}\sum_{k=1}^{L} \ln(p\left(l_k|C_q^j\right)) + \ln\left(prior(r)\right) \qquad 5.12$$

vi. Compare the computed $p(r)$ for all neurons in the Hypercube Classifier, where the neuron with the highest $p(r)$ is the winning neuron.

vii. Emit the classification vector Y_r related to the winning neuron *r*, whose members are the classes represented by the winning neuron, $Y_r = [y^1, y^2, \dots, y^n]$

As such, a full classification across the *n* dimensions of the MDC task was performed simultaneously. In this setup, the *Hypercube classifier* is capable of classifying class combinations that were not presented to the model during the training process, due to the bond it creates among various class spaces in the model through discrete yet connected probabilities.

Chapter 6: Performance Evaluation of Deep Self-Organizing Cube (DSOC)

6.1 Experiment

The aim of the study is to propose a deep learning classifier that can be used for multi-output learning problems as well as for the single-output learning ones. The experiment was designed to evaluate the performance of the proposed DSOC model in the four types of classification problems discussed earlier: BC, MCC, MLC and MDC.

As such, we conducted an empirical assessment on a variety of benchmark datasets, in order to compare the proposed model to standard classifiers. The proposed DSOC was then challenged with competitive state-of-the-art techniques reported in literature.

6.1.1 *Benchmark Datasets*

To evaluate the performance of the proposed model, DSOC was tested on seventeen benchmark datasets frequently used in literature to evaluate the performance of different classification models. The datasets were chosen to cover different aspects of classification tasks that are tackled by the proposed DSOC model. Those aspects include multiple dimensions, multi-class variables and labels per dimension, imbalance of labels, etc. The datasets are divided into four categories according to the type of the classification problem, where four datasets were chosen for BC problems, three for MCC, four for MLC and six for MDC.

Table 30 summarizes the main characteristics of the chosen datasets, as the number of observations and features, number of class spaces (i.e. number of dimensions), number of class labels per dimension, as well as their classification task category.

6.1.2 *Parameter Selection*

For the datasets that were not initially divided into *Training* and *Test* sets, a random selection technique was applied to divide the observations of the original dataset set into *Training* and *Test* sets in 75:25 ratio. Then a tenfold cross validation (CV) technique was employed on the training set to decide the DSOC model parameters. For the sake of simplicity, each of the general DSOC parameters (neighborhood, epochs and the decay parameter) was set to a constant value throughout the experiment (*nD=1, h = 100* and *d* = 6).

On the other hand, for the dimension specific parameters (*VSC Scaling* parameter, *VSC Selector* and α) the aforementioned tenfold CV was applied to decide those three parameters for each of the model multi-dimensional space.

Table 30. The characteristics of the benchmark datasets.

Classification Category	*Dataset*	*Number of Observations*	*Number of Features*	*Number of Dimensions*	*Number of Classes per Dimension*	*Domain and Reference*
BC	Diabetes[3]	768	8	1	2	Medical [176]
	Sonar[4]	208	60	1	2	Signal [177]
	Banknote[5]	1372	4	1	2	Forensic [178]
	Ionosphere[2]	351	34	1	2	Signal [179]
MCC	Iris[3]	150	4	1	3	Plant [180]
	Seeds[2]	210	7	1	3	Plant [181]
	Abalone[3]	4177	8	1	28	Marine [182]
MLC	Scene[2]	2407	294	6	2	Image [183]
	Emotions[2]	593	72	6	2	Music [184]
	Yeast[2]	2417	103	14	2	Genes [185]
	Image[2]	2000	135	5	2	Image [186]
MDC	Solar Flare[3]	315	10	3	3,4,2	Astronomy [187]
	Edm[2]	154	16	2	3	Machinery [188]
	WaterQuality[2]	1060	16	14	4	Life form [189]
	WQplants[2]*	1060	16	7	4	Plants [189]
	WQanimals[2]*	1060	16	7	4	Animals [189]
	Enb[2]	768	6	2	2,4	Energy [190]

* WQplants and WQanimals are subsets of the parent Water Quality dataset used to predict the relative representation of plant and animal species in Slovenian rivers.

[3] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
[4] https://www.openml.org/
[5] https://archive.ics.uci.edu/ml/index.php

6.1.3 *Evaluation Metrics*

In this paper, three widely used classification evaluation metrics are used to evaluate the performance of the proposed model. In BC problem, the *F1* measure is used as shown in equation 6.1. Where TP, FP and FN are True Positive, False Positive and False Negative respectively and are used to calculate the recall and precision as in [191].

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Or,

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6.1}$$

While in MLC and MDC tasks, *Hamming Accuracy* (HAccuracy) and *Exact Match* (EM) are used for evaluation. The *Hamming Accuracy* is the average of classification accuracy of all class variables in all dimensions, and it is used as well in MCC task. On the other hand, the *Exact Match* is the accuracy of predicting the entire classification vector across all class spaces simultaneously. [192], [131]

$$HAccuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Y_i^*|}{L} \tag{6.2}$$

$$EM = \frac{1}{N}\sum_{i=1}^{N} f(Y_i, Y_i^*) \tag{6.3}$$

Where, *N* is the number of observations, and *n* is the number of dimensions in the model. Y_i is the actual classification vector of size L while Y_i^* is the predicted classification vector of the same size and $|Y_i \cap Y_i^*|$ indicates the number of intersections between both vectors. On the other hand, $f(Y_i, Y_i^*)$ is a function that returns 1 when both classification vectors exactly match, and 0 otherwise.

6.1.4 *Performance Assessment*

In this study, the classification performance of the proposed DSOC model was assessed by comparing the obtained results to that obtained by standard classification algorithms, as well as state-of-the-art MDC approaches. All techniques were applied on the same benchmarks datasets.

For the single-output learning tasks, BC and MCC, the DSOC classification performance was assessed against four standard classification algorithms, namely k-Nearest Neighbor (*k*NN), Support Vector Machine (SVM), Naïve Bayes (NB) and Decision Tree (DT).

However, for the multi-output learning tasks, MLD and MDC, the DSOC performance was compared to eleven different approaches, divided into five categories that span the full classification challenges:

a) Standard Classification Algorithms: the same standard methods mentioned above.

b) Approaches based on manipulating the class spaces: *Binary Relevance* (BR), which divides the task into independent binary classifiers. *Ensembles of Class Powersets* (ECP) creates a combination of labels and treat the entire set of combinations as a single-dimension single-label class space. *Ensembles of Classifier Chains* (ECC), which divides the task into a chain of multi-class independent classifiers. *Ensembles of Super Class Classifiers* (ESC), which divides the classification space into sections of super-classes, formed of combination of subsets of class spaces. [31]

c) Approaches based on feature augmentation: those approaches tend to manipulate the input feature spaces rather than the output ones. This category includes, KRAM and SFAM (mentioned earlier in the literature review section).

d) Dependency modelling approach, the *Stacked Dependency Exploitation* (SEEM), which uses deterministic strategy to model class dependencies instead of the widely used probabilistic models. [28]

e) Regression based approach, gMML that creates independent regression models for each class variable and then, uses the Mahalanobis distance in the final classification step. [20]

The results obtained by applying those approaches on the selected benchmark datasets were extracted from literature, and used to assess the performance of the proposed DSOC model.

6.2 Results and Discussions

As mentioned earlier, the aim of the study is to construct a deep learning classifier that fits both single and multi-output learning tasks, with no structural alteration in the core design of the model other than simple parameters tweaking. In addition, the proposed design should be capable of successfully classifying observations in MLC and MDC tasks while modeling the class dependencies across all dimensions. Moreover, the model was challenged with datasets suffering from class imbalance in its multi-class spaces to assess its ability to classify, successfully, future rare observations.

A separate DSOC model was built for each of the 17 datasets. The DSOC dimensional parameters were determined using 10 fold CV applied on the training set. **Table 31** presents the selected parameters per dataset as well as the number of features that passed through the *VSC Selector* layer. The detailed parameters per dimension for one of the datasets, "*Image*" dataset, is presented in **Table 32**.

As seen in the last column of **Table 31**, the *VSC Selector* layer conducted a feature reduction routine that resulted in a substantial reduction in number of features in most of the models. The *Ionosphere* dataset was subjected to a sever dimensionality reduction, where only 35% of the original features have passed through the *VSC Selector* layer, while in case of *Image* and *Seeds* datasets only 52% and 57% have passed through that layer respectively. The other datasets were subjected to the same routine resulting in various levels of dimensionality reduction. On the other hand, six datasets have retained their full number of input features along all dimensions.

Table 31. DSOC Model parameters for each of the benchmark dataset obtained by applying 10 fold CV.

Classification Category	*Dataset*	*VSC Selector*	*Features Neurons*	*VSC Scaling Parameter*	α	*Selected Features after VSC*
BC	Diabetes	0	8	12.5	2	100%
	Sonar	0.6123	46	5.0	3	77%
	Banknote	0.735	3	5.5	3	75%
	Ionosphere	0.891	12	5.0	1	35%
MCC	Iris	0	4	10.5	1	100%
	Seeds	0.5383	4	2.5	1	57%
	Abalone	0	8	1.0	1	100%
MLC*	Scene	[0 – 0.7717]	[271 – 294]	[1 – 12]	[1 – 10]	98%
	Emotions	0	72	[1 – 5.5]	[1 – 3]	100%
	Yeast	[0 - 0.928]	[7-103]	[1 – 10.5]	[1 – 10]	76%
	Image	[0 - 0.9571]	[10 -135]	[2 - 7.5]	[1 – 4]	52%
MDC	Solar Flare 1	0	10	1,1.5,1	1	100%
	Edm	0.3398,0	9,16	1,1.5	3,4	78%
	WaterQuality	0	16	[1-3.5]	[1-3]	100%
	WQplants*	[0-0.7375]	[4-16]	[1-2]	[1-2]	70%
	WQanimals*	[0-0.8059]	[10-16]	[1-4.5]	[1-2]	82%
	Enb*	0	6	1,3.5	1,3	100%

* Values between brackets are the range of values of the DSOC parameter along the model's dimensions. For example, in the *Scene* dataset, the VSC Scaling parameter has a range of values of [1-12], which represents the range of values along the model's 6 dimensions whose values were $modVSC_1$ = 1, $modVSC_2$ = 1, $modVSC_3$ = 12, $modVSC_4$ = 10.5, $modVSC_5$ = 8, $modVSC_6$ = 1.

Table 32. Values of the DSOC parameters for the "*Image*" dataset along the model 5 dimensions (*L* = *5*).

Parameter	L_1	L_2	L_3	L_4	L_5
VSC Selector	0.6526	0.6383	0.9571	0.8554	0
modVSC	2.5	2	5	2	7.5
α	1	4	4	4	2
W	86	88	10	32	135

The *VSC Selector* layer serves two purposes, a) maximizing the DSOC's modeling and discriminatory power among classes in each of the class spaces, as well as b) reducing the size of input spaces by discarding the features that will inversely affect the classification power of the model. Reducing the input space is reflected in less computational power and time needed to train and implement the model as well as in maximizing the DSOC's discriminatory and modeling powers.

The effect of the *VSC Selector* layer is further augmented by the next layer, the *VSC Scaling layer*, which aims at deepening the discriminatory differences among classes along the remaining features.

In datasets with dense class spaces, classes are not clearly separated, but might suffer from sever overlap of their class spheres. Such density and overlap could be noticed in the tight Euclidian distances and the high similarity/correlation between classes that might, in some cases, reaches full correlation. DSOC model tackles this problem through its VSC double layer, *Selector* and *Scaling* layers, where both layers tend to intensify the discriminatory power of variables that, in turn, results in widening the distances between classes and relatively break the correlation among them.

Figure 17 demonstrates the effect of applying the VSC double layer on the "*Image"* dataset. The substantial reduction in dimensionality and the noticeable strengthening of the discriminatory power of the remaining input variables have resulted in widening the distance between classes from 0.0060 to 0.0191, and breaking the tight correlation from 0.9643 down to 0.7242, i.e. 313% and 25% improvement in class separation respectively.

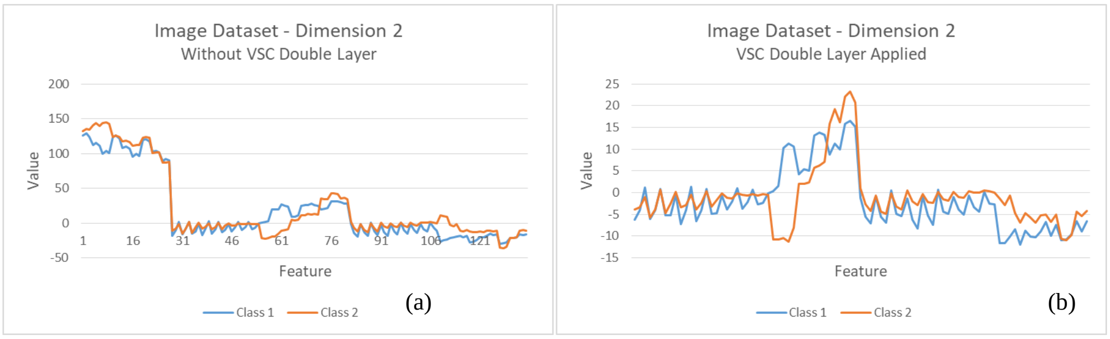


Figure 17. The effect of applying the VSC double layers on both classes of the second dimension of the Image dataset, a) before applying the VSC double layer (135 features and 0.9643 correlation coefficient), b) after applying both the VSC Selector and VSC Scaling layers (88 remaining features and 0.7242 correlation coefficient).

A closer look at **Figure 17 (a)** shows that the input features within the range [18-30] have the worst adverse effect on the model's discriminatory power. Such an effect was detected by the *VSC Selector* layer, which applied a threshold of **0.6383**, where all features with VSC weaker than that threshold were discarded. As such, and as seen in subfigure **(b)**, only 88 features have passed the *VSC Selector* layer, causing a significant dimensionality reduction, where only 65% of the original input features have survived the *Selector*. In order to deepen the difference between classes, the next layer, the *VSC Scaling* layer, has applied a dynamic scaling factor (based on its $modVSC_2$ = 2) ranging from 0.2654 to 0.9578 on the remaining 88 features, which results in expanding the ED between both classes by 313%.

Moreover, applying the VSC double layer on the "*Image"* dataset has increased the classification *Hamming* accuracy from 0.7100 to 0.8325, while the *Exact Match* metric has improved from 0.3198 to 0.3750. These results reflect the importance of the VSC double layer in improving the overall model accuracy as well as its ability to predict the exact classification vectors. **Table 33** presents the effect of applying the VSC double layer on the seventeen DSOC models constructed in this study, in terms of *HAccuracy* and *EM*. In addition, **Figure 18** visualizes the results presented in the table, demonstrating that the VSC double layer has enhanced the classification performance of the DSOC model in all of the four classification tasks.

Table 33. Effect of the VSC double layers on the performance of the DSOC Model applied on all benchmark datasets in the four classification tasks.

Classification Category	*Datasets*	*Hamming Accuracy*		*Exact Match*	
		VSC double layer Applied	*Without VSC double layer*	*VSC double layer Applied*	*Without VSC double layer*
BC	Diabetes	0.7708	0.6471	-	-
	Sonar	0.7692	0.5952	-	-
	Banknote	0.9854	0.9489	-	-
	Ionosphere	0.8740	0.7571	-	-
MCC	Iris	0.9867	0.9000	-	-
	Seeds	0.9143	0.8810	-	-
	Abalone	0.3599	0.2491	-	-
MLC	Scene	0.9398	0.8312	0.5334	0.4749
	Emotions	0.7463	0.5998	0.2178	0.0941
	Yeast	0.7719	0.7501	0.1538	0.1298
	Image	0.8325	0.7100	0.3750	0.3198
MDC	Solar Flare 1	0.9418	0.9101	0.8846	0.7460
	Edm	0.8000	0.7667	0.6333	0.6333
	WaterQuality	0.6405	0.5893	0.0047	0.0000
	WQplants	0.6604	0.6220	0.1038	0.0660
	WQanimals	0.6462	0.4993	0.0849	0.0142
	Enb	0.9019	0.8497	0.8823	0.7516

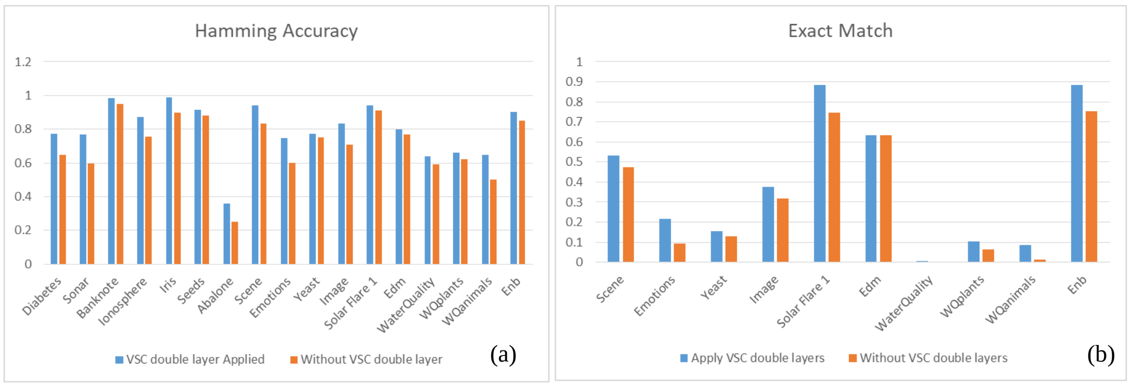


Figure 18. The effect of the VSC double layer on the performance of the DSOC Model applied on all benchmark datasets, a) Hamming Accuracy on all 17 datasets, b) the Exact Match metric applied on the datasets of MLC and MDC tasks only.

For evaluating of the proposed DSOC, comparisons have been conducted to the DSOC classification performance against that of standard classifiers as well as state-of-the-art MDC approaches on the benchmark datasets. Those comparisons are presented in the following three subsections.

6.2.1 *Comparison to Standard Classifiers*

The experimental results are reported in details in four tables, one for each of the classification tasks. **Table 34** for BC, **Table 35** for MCC, **Table 36** for MLC and **Table 37** for MDC. Based on the results obtained in these tables, we can point out the following observations:

- With regards to the standard classification techniques (kNN, NB, SVM and DT), a total of 112 experiments were carried out. Those experiments are composed of 5 standard algorithms, in addition to the proposed DSOC, 2 evaluation metrics and 17 datasets.

- Those experiments are divided across the four classifications tasks as follows: 20 experiments in the BC task, 16 in MCC, 32 in MLC and 48 in MDC.

- As seen in the aforementioned tables (with the winning results embossed in bold), the proposed DSOC model has outperformed the standard methods in 50% of the cases in the BC task, 100% in MCC, 75% in MLC and 83% in case of MDC. Moreover, **Figure 19 (a)** shows the accuracy results of the proposed DSOC in comparison with the other standard classifiers.

- With the exception of the “*EnB*” and “*Yeast*” datasets, the DSOC has outperformed all other standard algorithms in MLC and MDC, in terms of the *Exact Match* metrics. Since the EM metric reflects the ability of the model to successfully classify the instance along all dimensions, therefore the proposed DSOC is more capable of classifying the observations across all class spaces simultaneously. A comparison of the EM accuracy of the proposed DSOC and the other standard classifiers are shown in **Figure 19 (b)**.

- The last mentioned observation points out the intrinsic ability of the DSOC to span and to model dependencies among class variables across different dimensions.

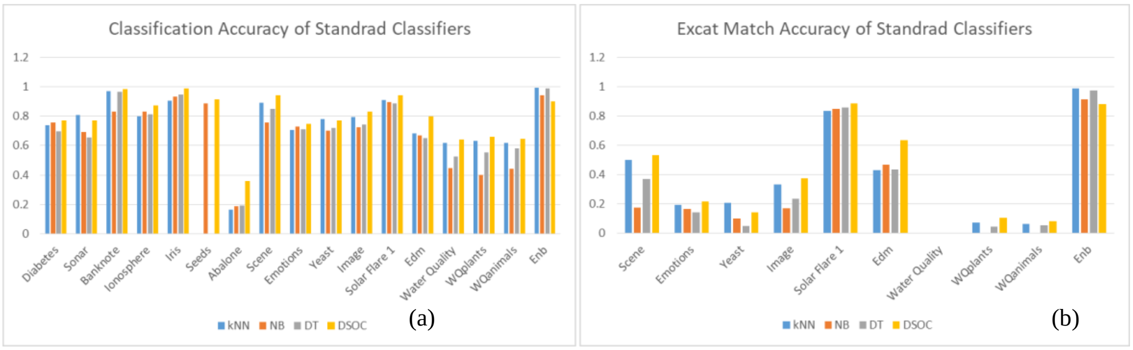


Figure 19. The classification performance of DSOC model compared to standard classifiers. a) Classification accuracy for BC, MCC, MLC and MDC datasets. b) Exact Match accuracy for MLC and MDC datasets.

Table 34. Comparing the classification results of DSOC against standard classification algorithms applied on BC benchmark datasets.

Dataset	***F1 Measure***				
	kNN	SVM	NB	DT	***DSOC***
Diabetes	0.7396	**0.7760**	0.7552	0.6979	0.7708
Sonar	**0.8077**	0.8077	0.6923	0.6538	0.7692
Banknote	0.9708	0.9854	0.8321	0.9635	**0.9854**
Ionosphere	0.8000	0.8429	0.8286	0.8143	**0.8740**

Table 35. Comparing the classification results of DSOC against standard classification algorithms applied on MCC benchmark datasets in terms of Hamming Accuracy.

Dataset	*Hamming Accuracy*			
	kNN	NB	DT	***DSOC***
Iris	0.9067	0.9333	0.9467	**0.9867**
Seeds	0. 8952	0.8857	0. 8952	**0.9143**
Abalone	0.1657	0.1897	0.1933	**0.3599**

Table 36. Comparing the classification results of DSOC against standard classification algorithms applied on MLC benchmark datasets in terms of Hamming Accuracy and Exact Match metrics.

Dataset	*Hamming Accuracy*				*Exact Match*			
	kNN	NB	DT	***DSOC***	kNN	NB	DT	***DSOC***
Scene	0.8913	0.7565	0.8478	**0.9398**	0.5008	0.1756	0.3687	**0.5334**
Emotions	0.7046	0.7302	0.7112	**0.7463**	0.1931	0.1683	0.1436	**0.2178**
Yeast	**0.7803**	0.6991	0.7184	0.7719	**0.2061**	0.1036	0.0523	0.1450
Image	0.7925	0.7235	0.7435	**0.8325**	0.3329	0.1725	0.2350	**0.3750**

Table 37. Comparing the classification results of DSOC against standard classification algorithms applied on MDC benchmark datasets, in terms of Hamming Accuracy and Exact Match metrics.

Dataset	*Hamming Accuracy*				*Exact Match*			
	kNN	NB	DT	***DSOC***	kNN	NB	DT	***DSOC***
Solar Flare 1	0.9103	0.8974	0.8846	**0.9418**	0.8333	0.8462	0.859	**0.8846**
Edm	0.6833	0.6667	0.650	**0.80**	0.433	0.4667	0.4333	**0.6333**
Water Quality	0.6179	0.4481	0.5236	**0.6405**	0	0	0	**0.0047**
WQplants	0.6301	0.4023	0.5553	**0.6604**	0.0755	0	0.0472	**0.1038**
WQanimals	0.6186	0.4434	0.5802	**0.6462**	0.0660	0	0.0556	**0.0849**
Enb	**0.9935**	0.9412	0.9869	0.9019	**0.9869**	0.9150	0.9739	0.8823

6.2.2 *Comparisons to the state-of-the-art methods*

In order to assess the ability of the proposed DSOC algorithm to model complex dependencies as well as imbalanced class spaces, the obtained results were challenged with state-of-the-art approaches from literature. The selected approaches span different approaches for solving classification tasks, these include approaches that depends on manipulating the class spaces during training (BR, ECC, ECP and ESC), approaches that uses feature augmentation to manipulate the input space (KRAM and SFAM), in addition to SEEM approach that model dependencies and gMML that uses regression approach to solve the classification task.

The results of the selected approaches were collected from recent literature ([28], [129] and [130]) to reflect the most recent work applied on the MLC and MDC benchmark datasets. It is worth mentioning that KRAM and the ensembles approaches are all based on Naïve Bayes classifiers.

Table 38 and **Figure 20** depicts the comparison between the proposed DSOC model and eight state-of-the-art approaches in terms of the *HAccuracy*. On the other hand, **Table 39** and **Figure 21** present the same comparison in terms of *EM* accuracy to assess the model ability to model dependencies. From those tables and figures, one can observe the following:

- In terms of *Hamming Accuracy*, DSOC has outperformed all state-of-the-art approaches in 5 datasets (55.6%), and achieved equivalent results in 3 datasets (33.3%).
- In terms of *Exact Match*, DSOC has outperformed all state-of-the-art approaches over 7 datasets (77.8%), which reflects the superior ability of DSOC to model dependencies and successfully classify samples across all dimensions simultaneously.

Table 38. Comparing the classification results of DSOC against state-of-the-art techniques applied on the same MLC and MDC benchmark datasets in terms of the Hamming Accuracy

Dataset	*Hamming Accuracy*								
	Classes Based Approaches				*Feature Augmentation*		*Dependency Modelling*	*Regression Based*	
	BR	*ECC*	*ECP*	*ESC*	*KRAM*	*SFAM*	*SEEM*	*gMML*	***DSOC***
Solar Flare1	0.886	0.883	0.908	0.896	0.872	0.925	0.925	0.923	**0.9418**
Edm	0.677	0.690	0.731	0.674	0.680	-	-	0.714	**0.8000**
Water Quality	0.389	0.360	0.599	0.609	0.488	0.641	**0.646**	0.643	*0.6405*
WQplants	0.397	0.353	0.607	0.442	0.506	**0.666**	0.661	0.655	0.6604
WQanimals	0.381	0.377	0.590	0.577	0.419	0.641	0.635	0.630	**0.6462**
Enb	0.774	0.773	0.764	0.754	-	0.865	0.777	0.742	**0.9019**
Scene	0.763	0.767	0.867	0.866	0.777	-	-	**0.893**	0.8913
Yeast	0.699	0.696	0.773	0.716	0.695	-	-	**0.800**	0.7719
Image	0.573	0.576	0.746	0.593	0.586	-	-	0.811	**0.8325**

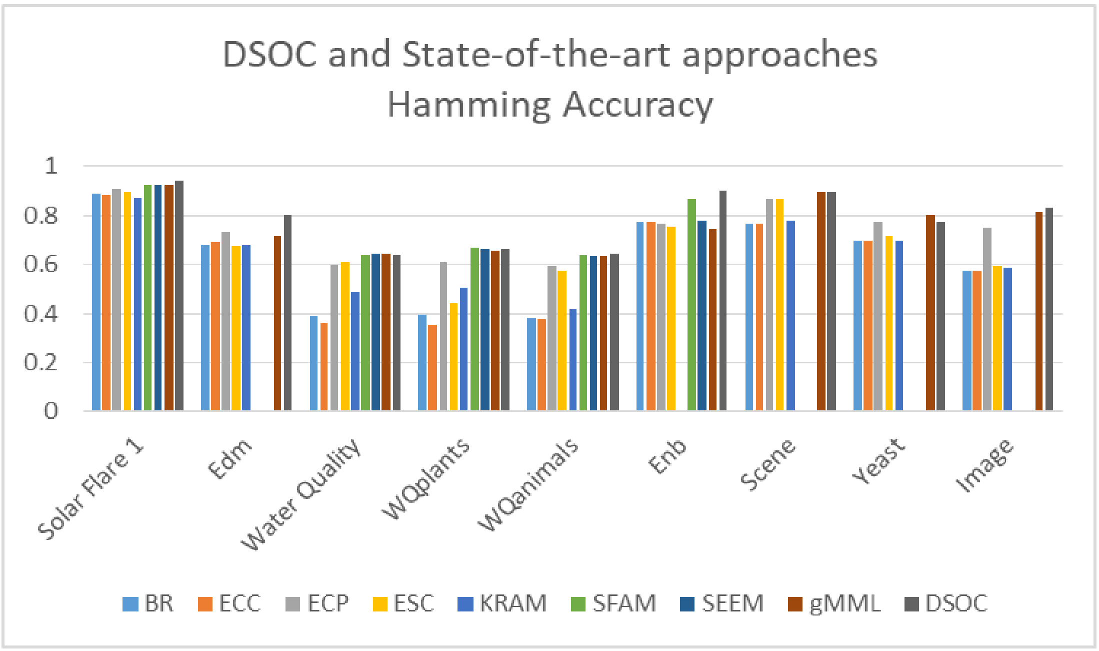


Figure 20. Comparison of the Hamming Accuracy of DSOC vs state-of-the-art approaches applied on MLC and MDC benchmark datasets.

Table 39. Comparing the classification results of DSOC against state-of-the-art approaches applied on the same MLC and MDC benchmark datasets in terms of the Exact Match.

Dataset	*Exact Match*								
	Classes Based Approaches				*Feature Augmentation*		*Dependency Modelling*	*Regression Based*	
	BR	*ECC*	*ECP*	*ESC*	*KRAM*	*SFAM*	*SEEM*	*gMML*	***DSOC***
Solar Flare 1	0.774	0.774	0.790	0.780	0.756	0.824	0.818	0.821	**0.8846**
Edm	0.432	0.451	0.554	0.432	0.445	-	-	0.487	**0.6333**
Water Quality	0.000	0.000	0.008	0.002	0.000	0.009	0.009	**0.006**	0.0047
WQplants	0.001	0.001	0.034	0.001	0.036	0.100	0.096	0.092	**0.1038**
WQanimals	0.004	0.007	0.020	0.024	0.008	0.058	0.049	0.062	**0.0849**
Enb	0.548	0.546	0.529	0.508	-	0.729	0.554	0.554	**0.8823**
Scene	0.177	0.181	0.550	**0.541**	0.198	-	-	0.457	0.5334
Yeast	0.095	0.102	0.203	0.110	0.115	-	-	0.134	**0.1450**
Image	0.069	0.069	0.285	0.069	0.074	-	-	0.289	**0.3750**

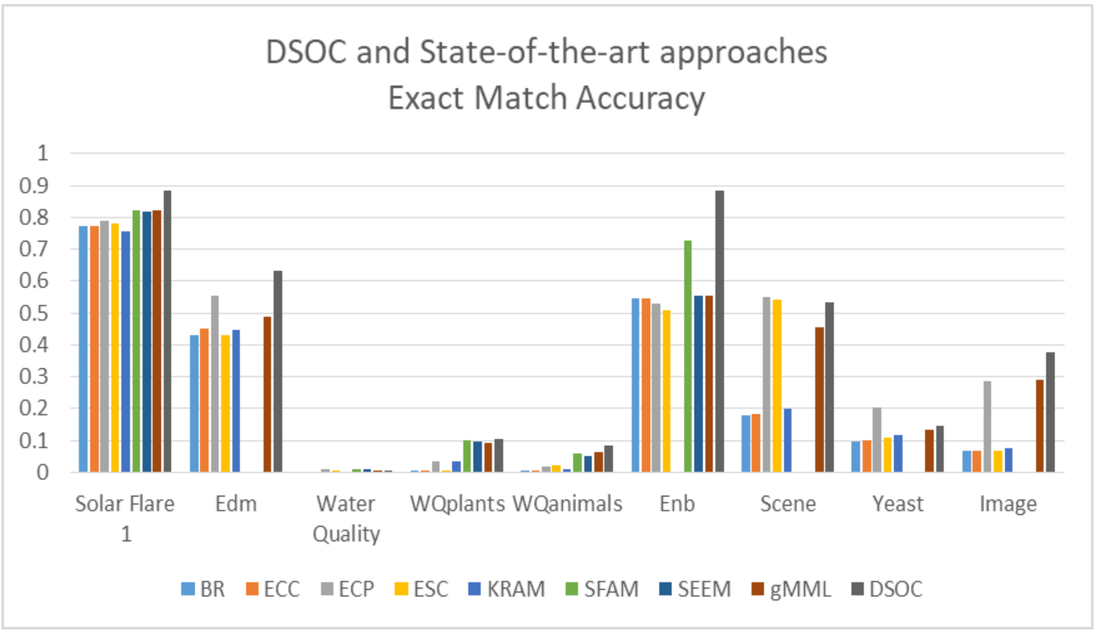


Figure 21. Comparison of the Exact Match Accuracy of DSOC vs state-of-the-art techniques applied on MLC and MDC benchmark datasets.

6.2.3 *Collective Performance Remarks*

By analyzing the overall results obtained under sections 6.2.1 and 6.2.2, we can conclude the following:

a. Despite the imbalance of class labels in MLC and MDC datasets (especially those with high dimensionality and variety of class labels), DSOC was capable of achieving better results than most standard classifiers as well as state-of-the-art approaches included in the study.

b. The better classification performance achieved by the DSOC implies its ability to segregate different classes within all dimensions simultaneously.

c. The superior performance of DSOC over other approaches in terms of EM indicates the built-in ability of different layers within the DSOC to model dependencies, which in turn is reflected in better classification performance.

d. In contrast to other approaches, which require specific steps to model decencies or handle class imbalance, DSOC uses a single straightforward approach that tackles those problems simultaneously in a single learning routine without prior treatment or algorithms.

From those four conclusion remarks, the superior performance of DSOC could be attributed to the following design characteristics of the proposed model:

a. The VSC double layer (*VSC Selector* and *VSC Scaling* layers) is responsible for segregation of classes and deepening the differences between them, which is then reflected in a better discriminatory power and superior classification performance.

b. The DSOC’s final classifier hypercube and its self-organizing structure, is responsible for modeling class dependencies and classify unseen class combinations in future samples of imbalanced datasets.

c. Moreover, the straightforward learning approach of DSOC provides it with the ability to solve tasks in different domains regardless the intensity of class dependencies or the severity of data imbalance.

Chapter 7: Conclusion and Future Work

In this chapter, we conclude the work done in both studies, the language independent summarization model and the deep learning multi-dimensional classifier.

The chapter is divided into three sections; the first one concludes the work done in the text summarization field and the proposed language agnostic model, *UnB-LITS*. The second section concludes the work performed in the deep learning domain, to solve multi-dimensional classification tasks using the proposed deep learning classifier, *DSOC*. Finally, the third section presents our vision for future work using these two domain agnostic models.

7.1 Conclusion on Language Agnostic Text Summarization Model (*UnB-LITS*)

In this study, a new automatic text summarization model is introduced, which is capable of performing text summarization in a language independent manner. The proposed tool, ***UnB-LITS*** or "***UnB-Language Independent Text Summarizer***", generates efficient language independent extractive summary when evaluated against human generated summaries. The tool extracts intrinsic features of text elements (words, n-grams, sentences and paragraphs) using an innovative way of *Shape-Coding* of those elements. *Shape-Coding* is performed by transferring the text element to a normalized shape that fits into relatively small number of encoded classes, which in turn, reflects the importance of elements with rare shapes.

The proposed tool is totally free of any particular language tools dependency due to the fact that it is capable of removing or neutralizing the effect of unimportant words (stop words) without the need of stop words lexicons that are by definition language dependent. The proposed model is capable, as well, of grouping similar words together in a way similar to stemming without the need of language dictionaries or hand coded stemmer tools.

As such, the proposed model preserves the weight of potential words and sentences, with the ability to extract strong and important key phrases with no dependency on external language databases or corpora.

The proposed tool, *UnB-LITS*, was tested on news datasets written in English, Portuguese, French and Spanish. The obtained results were evaluated against human-generated summaries using ROUGE-1 and ROUGE-2 metrics. In case of English and Portuguese the results were compared to state-of-the-art models and systems listed in literature. While, for French and Spanish results were compared to those obtained by Apple macOS 12 integrated summarizer as well as the online Automatic summarizer. *UnB-LITS* achieved better performance over other tools in all of the four languages. These results were achieved with no dependency on any domain specific or language related lexicons, parsers or corpora, which proves the quality of the proposed contributions.

7.2 Conclusion on Deep Self-Organizing Cube (DSOC)

Multi-dimensional classification (MDC) task can be considered the most inclusive description of all classifications tasks, as it joins multiple class spaces and their multiple class membership into a single compound classification problem. The challenges in MDC arise from the possible dependencies that classes in different class spaces could have, as well as the imbalance of labels in training datasets due to the enormous number of combinations needed for all labels across multiple class spaces. In this study, we proposed an MDC deep learning classifier, named "*Deep Self-Organizing Cube*" or "*DSOC*", which could model dependencies among classes in multiple class spaces, while consolidating its ability to classify combinations of labels that were not presented to the model during the training process. *DSOC* achieved its intended purpose through its two connected components, the *n*-dimensional *Hypercube classifier*, and *n DSOC Neural Networks* connected to the cube. Each of the DSOC's multiple neural networks, one per dimension, consists of three main hidden layers, namely the *VSC Selector*, *VSC Scaling* layers and the *Pooling* layer. The VSC double layer is responsible for feature selection and segregation of classes while the *Pooling* layer is responsible for developing the probability model per dimension. In addition, in the central core of the DSOC model, lies the *Hypercube* classifier, responsible for creating the semantics among multiple class spaces and accommodate the model for rare samples classification.

DSOC model belongs to multiple-output learning, where it emits a classification vector formed of a class label from each class space in the model. As such, DSOC is capable of classifying a sample along multiple class spaces simultaneously, achieving the core purpose of multiple-output learning. Considering this feature, DSOC can be used in multiple output classification tasks, as multi-dimensional classification and multi-label classification, as well as in single-output learning, as in binary classification and multi-class classification problems.

For challenging the proposed DSOC model, an experiment was designed to evaluate the performance of the proposed DSOC model in the four types of classification problems BC, MCC, MLC and MDC. We conducted an empirical assessment of the model on seventeen benchmark datasets, and the obtained classification results were compared to those obtained by four standard classifiers as well as by eight competitive state-of-the-art approaches reported in literature. The selected state-of-the-art approaches were chosen to challenge the model in solving the aforementioned MDC problems. These include approaches that manipulate the input feature space through feature augmentation, and approaches that manipulate the output class spaces to tackle the problem of dependencies, in addition to others based on regression or pure dependency modelling.

The DSOC has achieved superior performance over both standard classifiers as well as the state-of-the-art approaches in all the four classification tasks. In case of standard classifiers, DSOC has outperformed the standard methods in 50% of the cases in the BC task, 100% in MCC, 75% in MLC and 83% in case of MDC. Moreover, with the exception of only two datasets, DSOC has outperformed the standard algorithms in terms of the *Exact Match* accuracy metric, which measures the ability of the model to, successfully, classify a sample across all dimensions simultaneously.

In the same context, in terms of *Exact Match*, DSOC has outperformed all state-of-the-art approaches in 77.8% of the cases, which reflects the superior ability of DSOC to model dependencies and successfully classify samples across all dimensions simultaneously.

In contrast to other state-of-the-art approaches, the DSOC's structure, components and design do not change from one classification task to the other. DSOC maintains its straightforward training and classification technique regardless of the type of classification task.

As such, the main contribution of this study is proposing a straightforward yet efficient deep learning classifier of superior performance, which does not require structural modifications regardless the task in hand. A model that manipulates the input features space, via its VSC layers, to increase the discriminatory power of the model and segregate classes through augmenting the differences between different input variables. A model capable of modeling semantics among classes along different classification spaces even in case of data imbalance.

7.3 Future Work

As seen in the conclusion sections above, both models achieved significantly superior results, and outperformed the state-of-the-art algorithms in both fields. Both models have proven their ability to perform efficiently in a domain agnostic manner; *UnB-LITS* as a language independent summarizer, and *DSOC* as an MDC domain agnostic classifier.

Both of these findings encouraged us to explore the possibilities of extending the boundaries of this research, and seek the opportunities for more applications. Such motivation is reflected below in our plan for future wok.

In case of the language independent text summarization model, *UnB-LITS*, future research could tackle the following points:

i. Extending the boundaries of the model to solve multiple documents summarization tasks.

ii. The multiple-document summarization could be extended and applied on long texts or sets of documents that contain mixed languages or context, as in the case of scientific papers and language books.

iii. Expand the model's application domain to include oriental languages as Arabic, Persian, etc.

iv. Conduct experiments to study the effect of tweaking the model parameters for extracting NE's and CC's while maintaining the language agnostic nature of the model.

On the other hand, in case of the deep learning multi-dimensional classification model, *DSOC*, future research could tackle the following points:

i. Increase the number of *Pooling* layers of the DSOC neural networks in order to achieve features extraction at various abstract level, and study its effect on the overall classification performance of the model.

ii. Study the applicability and impact of adding additional *VSC double layer* after the Pooling layers, in order to achieve more class segregation as well as better modeling of dependencies across different class spaces.

iii. Tweak the parameters to expand the model boundaries in order to solve regression problems with discrete outputs.

iv. Study the ability of achieving unsupervised multi-dimensional clustering by tweaking the model parameters as well as the weight vectors of the pooling layer.

REFERENCES

[1] A. D. Friederici, "Towards a neural basis of auditory sentence processing," *Trends in Cognitive Sciences,* vol. 6, pp. 78-84, 2002.

[2] A. D. Friederici, "The Brain Basis Of Language Processing: From Structure To Function," *Physiol,* vol. 91, p. 1357–1392, 2011.

[3] D. Khurana, A. Koli, K. Khatter and S. Singh, "Natural language processing: State of the art, current trends and challenges.," *Multimedia Tools and Applications,* pp. 1-32, 2022.

[4] A. Clark, C. Fox and S. Lappin, The Handbook of Computational Linguistics and Natural Language Processing, 2010.

[5] D. Jurafsky and J. Martin, Speech and Language Processing, 3rd Edition, 2nd ed., New Jersy: Pearson, 2021.

[6] W. S. El-Kassas, C. R. Salama, A. A. Rafea and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications,* vol. 165, 2021.

[7] I. Rivera-Trigueros, "Machine translation systems and quality assessment: a systematic review.," *Language Resources and Evaluation,* pp. 1-27, 2021.

[8] L. B. W. L. &. S. A. A. Monteiro, "An approach of vector space model to link concrete concepts with Wiki entities.," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing.,* IEEE, 2015.

[9] J. Mothe, "Analytics Methods to Understand Information Retrieval Effectiveness—A Survey.," *Mathematics,* vol. 10, no. 12, 2022.

[10] M. A. C. Soares and F. S. Parreiras, "A literature review on question answering techniques, paradigms and systems.," *Journal of King Saud University-Computer and Information Sciences.,* vol. 32, no. 6, pp. 635-646, 2020.

[11] M. Birjali, M. Kasri and A. Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends.," *Knowledge-Based Systems,* vol. 226, 2021.

[12] S. G. Kanakaraddi and S. S. Nandyal, "Survey on parts of speech tagger techniques.," in *International Conference on Current Trends towards Converging Technologies (ICCTCT),* IEEE, 2018.

[13] Y. Bengio, "Learning Deep Architecture for AI," *Foundations and Trends in Machine Learning,* vol. 2, no. 1, pp. 1-127, 2009.

[14] M. A. Wani, F. A. Bhat, S. Afzal and A. I. Khan, Advances in deep learning, Springer, 2020.

[15] S. Mallat, "Understanding deep convolutional networks.," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 374, no. 2065, 2016.

[16] R. Salakhutdinov and G. Hinton, "Deep Boltzman Machine," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics,* 2009.

[17] Y. Hua, J. Guo and H. Zhao, "Deep belief networks and deep learning.," in *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, IEEE, 2015.

[18] R. M. Cichy and D. Kaiser, "Deep neural networks as scientific models.," *Trends in cognitive sciences*, vol. 23, no. 4, pp. 305-317, 2019.

[19] B. Zohuri and M. Moghaddam, "Deep learning limitations and flaws.," *Modern Approaches on Material Science*, vol. 2, pp. 241-250, 2020.

[20] Z. Ma and S. Chen, "Multi-dimensional classification via a metric approach.," *Neurocomputing*, vol. 275, pp. 1121-1131, 2018.

[21] D. Xu, Y. Shi, I. W. Tsang, Y. S. Ong, C. Gong and X. Shen, "Survey on multi-output learning.," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 7, pp. 2409-2429, 2020.

[22] D. Song, A. Vold, K. Madan and F. Schilder, "Multi-label legal document classification: A deep learning-based approach with label-attention and domain-specific pre-training.," *Song, D., Vold, A., Madan, K., & Schilder, F. (2022). Multi-label legal document classificatInformation Systems*, vol. 106, 2022.

[23] D. Senthilkumar and C. Akshayaa, "Efficient Deep Learning Approach for Multi-label Semantic Scene Classification.," in *Proceedings of the International Conference on Image Processing and Capsule Networks*, Bangkok, 2020.

[24] X. S. Xu, Y. Jiang, X. Xue and Z. H. Zhou, "Semi-supervised multi-instance multi-label learning for video annotation task.," in *Proceedings of the 20th ACM international conference on Multimedia*, Nara, 2012.

[25] M. L. Zhang, Y. K. Li, X. Y. Liu and X. Geng, "Binary relevance for multi-label learning: an overview.," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191-202, 2018.

[26] J. Read, B. Pfahringer, G. Holmes and E. Frank, "Classifier chains for multi-label classification.," *Machine learning*, vol. 85, no. 3, pp. 333-359, 2011.

[27] J. Read, L. Martino and D. Luengo, "Efficient monte carlo methods for multi-dimensional learning with classifier chains.," *Pattern Recognition*, vol. 47, no. 3, pp. 1535-1546, 2014.

[28] B. B. Jia and M. L. Zhang, "Multi-dimensional classification via stacked dependency exploitation.," *Science China Information Sciences*, vol. 63, no. 12, pp. 1-14, 2020.

[29] K. Dembszynski, W. Waegeman, W. Cheng and E. Hüllermeier, "On label dependence in multilabel classification.," in *In Workshop Proceedings of Learning from Multi-Label Data*, Haifa, 2010.

[30] J. C. Junior, E. R. Faria, J. A. Silva and R. Cerri, "Label powerset for multi-label data streams classification with concept drift.," in *Proceedings of the 5th Symposium on Knowledge Discovery, Mining Learn*, Uberlandia, 2017.

[31] J. Read, C. Bielza and P. Larrañaga, "Multi-dimensional classification with super-classes.," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1720-1733, 2013.

[32] J. Arias, J. A. Gamez, T. D. Nielsen and J. M. Puerta, "A scalable pairwise class interaction framework for multidimensional classification.," *International Journal of Approximate Reasoning*, vol. 68, pp. 194-210, 2016.

[33] F. Charte, A. J. Rivera, M. J. del Jesus and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms.," *Neurocomputing,* vol. 163, pp. 3-16, 2015.

[34] A. N. Tarekegn, M. Giacobini and K. Michalak, "A review of methods for imbalanced multi-label classification.," *Pattern Recognition,* vol. 118, pp. 1-12, 2021.

[35] T. Kluyver, B. Ragan-Kelley and F. P., "Jupyter Notebooks – a publishing format for reproducible computational workflows.," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas,* F. Loizides and B. Schmidt, 2016, pp. 87-90.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825--2830, 2011.

[37] I. The MathWorks, MATLAB R2015a, Massachusetts: Natick, 2015.

[38] S. Sinclair and G. Rockwell, Voyant Tool v 2.6.1, https://voyant-tools.org/, 2022.

[39] C. Microsoft, Microsoft Excel, Seattle, 2016.

[40] S. R. Rahimi, A. T. Mozhdehi and M. Abdolahi, "An overview on extractive text summarization.," in *IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2017.

[41] A. Alomari, N. Idris, A. Q. M. Sabri and I. Alsmadi, "Deep reinforcement and transfer learning for abstractive text summarization: A review.," *Computer Speech & Language,* vol. 71, 2022.

[42] W. T. Hsu, C. K. Lin, M. Y. Lee, K. Min, J. Tang and M. Sun, "A unified model for extractive and abstractive summarization using inconsistency loss.," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.*, Melbourne, Australia, 2018.

[43] M. Tomer and M. Kumar, "Multi-document extractive text summarization based on firefly algorithm.," *Journal of King Saud University-Computer and Information Sciences,* vol. 34, no. 8, pp. 6057-6065, 2022.

[44] K. Smelyakov, D. Karachevtsev, D. Kulemza, Y. Samoilenko, O. Patlan and A. Chupryna, "Effectiveness of preprocessing algorithms for natural language processing applications.," in *IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, IEEE, 2020.

[45] T. Bergmanis and S. Goldwater, "Context sensitive neural lemmatization with lematus.," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[46] S. Warjri, P. Pakray, S. A. Lyngdoh and A. K. Maji, "Part-of-speech (pos) tagging using conditional random field (crf) model for khasi corpora.," *International Journal of Speech Technology,* vol. 24, no. 4, pp. 853-864, 2021.

[47] J. Kaur and P. K. Buttar, "A systematic review on stopword removal algorithms.," *International Journal on Future Revolution in Computer Science & Communication Engineering,* vol. 4, no. 4, pp. 207-210, 2018.

[48] D. J. Ladani and N. P. Desai, "Stopword identification and removal techniques on tc and ir applications: A survey.," in *6th International Conference on Advanced Computing and Communication Systems (ICACCS)* , IEEE, 2020.

[49] D. Namly, K. Bouzoubaa and A. Yousfi, "A bi-technical analysis for arabic stop-words detection.," *Compusoft,* vol. 8, no. 5, pp. 3126-3134, 2019.

[50] J. Lovins, "Development of a stemming algorithm," *Mech. Trans. Comput. Linguist.,* vol. 11, no. 21, pp. 22-31, 1968.

[51] J. Jumadi, D. S. Maylawati, L. D. Pratiwi and M. A. Ramdhani, "Comparison of Nazief-Adriani and Paice-Husk algorithm for Indonesian text stemming process.," in *IOP Conference Series: Materials Science and Engineering,* 2021.

[52] H. Alshalabi, S. Tiun, N. Omar, F. N. AL-Aswadi and K. A. Alezabi, "Arabic light-based stemmer using new rules.," *Journal of King Saud University-Computer and Information Sciences.,* vol. 34, no. 9, pp. 6635-6642, 2022.

[53] R. Gupta and A. G. Jivani, "LemmaQuest Lemmatizer: A Morphological Analyzer Handling Nominalization.," *IETE Journal of Research,* pp. 1-9, 2022.

[54] A. Voutilainen, "Part-of-Speech Tagging," in *The Oxford Handbook of Computational Linguistics,* 2003, pp. 219-231.

[55] R. Saidi, F. Jarray and M. Mansour, "A BERT based approach for Arabic POS tagging.," in *International Work-Conference on Artificial Neural Networks*, Cham, Springer, 2021, pp. 311-321.

[56] D. Vasić, B. Žitko, A. Grubišić, S. Stankov, A. Gašpar, I. Šarić-Grgić and M. Markić-Vučić, "Croatian POS Tagger as a Prerequisite for Knowledge Extraction in Intelligent Tutoring Systems.," in *International Conference on Human-Computer Interaction,* Cham, 2021.

[57] P. Singh, G. Rutten and E. Lefever, "A Pilot Study for BERT Language Modelling and Morphological Analysis for Ancient and Medieval Greek.," in *The 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, , Humanities and Literature.,* 2021.

[58] S. Warjri, P. Pakray, S. A. Lyngdoh and A. K. Maji, "Part-of-speech (pos) tagging using conditional random field (crf) model for khasi corpora.," *International Journal of Speech Technology,* vol. 24, no. 4, pp. 853-864, 2021.

[59] A. Chiche and B. Yitagesu, "Part of speech tagging: a systematic review of deep learning and machine learning approaches.," *Journal of Big Data,* vol. 9, no. 1, pp. 1-25, 2022.

[60] C. A. Bahcevan, E. Kutlu and T. Yildiz, "Deep neural network architecture for part-of-speech tagging for turkish language.," in *3rd International Conference on Computer Science and Engineering (UBMK),* Bosnia and Herzegovina, 2018.

[61] M. Rajani Shree and B. R. Shambhavi, "POS Tagger Model for South Indian Language Using a Deep Learning Approach.," in *ICCCE,* Singapore, Springer, 2021, pp. 155-168.

[62] M. K. Junaida and A. P. Babu, "A Deep Learning Approach to Malayalam Parts of Speech Tagging.," in *Second International Conference on Networks and Advances in Computational Technologies,* Cham, 2021.

[63] A. &. S. S. K. Priyadarshi, "A study on the performance of Recurrent Neural Network based models in Maithili Part of Speech Tagging.," *Transactions on Asian and Low-Resource Language Information Processing.,* 2022.

[64] A. Serek, A. Issabek, A. Akhmetov and A. Sattarbek, "Part-of-speech tagging of Kazakh text via LSTM network with a bidirectional modifier.," in *16th International Conference on Electronics Computer and Computation (ICECCO),* 2021.

[65] S. Besharati, H. Veisi, A. Darzi and S. H. H. Saravani, "A hybrid statistical and deep learning based technique for Persian part of speech tagging.," *Iran Journal of Computer Science,* vol. 4, no. 1, pp. 35-43, 2021.

[66] S. Chotirat and P. Meesad, "Part-of-Speech tagging enhancement to natural language processing for Thai wh-question classification with deep learning.," *Heliyon,* vol. 7, no. 10, 2021.

[67] G. Lkhagvasuren, J. Rentsendorj and O. E. Namsrai, "Mongolian Part-of-Speech Tagging with Neural Networks.," in *Advances in Intelligent Information Hiding and Multimedia Signal Processing,* Singapore, Springer, 2021, pp. 109-115.

[68] B. Alshemali and J. Kalita, "Improving the reliability of deep neural networks in NLP: A review.," *Knowledge-Based Systems,* vol. 191, 2020.

[69] M. Pota, F. Marulli, M. Esposito, G. De Pietro and H. Fujita, "Multilingual POS tagging by a composite deep architecture based on character-level features and on-the-fly enriched Word Embeddings.," *Knowledge-Based Systems,* vol. 164, pp. 309-323, 2019.

[70] P. Kouris, G. Alexandridis and A. Stafylopatis, "Abstractive text summarization: Enhancing sequence-to-sequence models using word sense disambiguation and semantic content generalization.," *Computational Linguistics,* vol. 47, no. 4, pp. 813-859, 2022.

[71] D. Suleiman and A. Awajan, "Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges.," *Mathematical problems in engineering,* 2020.

[72] Y. M. Wazery, M. E. Saleh, A. Alharbi and A. A. Ali, "Abstractive Arabic Text Summarization Based on Deep Learning.," *Computational Intelligence and Neuroscience,* 2022.

[73] M. Zhang, G. Zhou, W. Yu, N. Huang and W. Liu, " A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning.," *Computational Intelligence and Neuroscience,* 2022.

[74] T. Cai, M. Shen, H. Peng, L. Jiang and Q. Dai, "Improving transformer with sequential context representations for abstractive text summarization.," in *CCF International Conference on Natural Language Processing and Chinese Computing,* Cham, Springer, 2019, pp. 512-524.

[75] T. Dong, S. Shan, Y. Liu, Y. Qian and A. Ma, "A Pointer-Generator Based Abstractive Summarization Model with Knowledge Distillation.," in *International Conference on Neural Information Processing,* Cham, Springer, 2021, pp. 168-177.

[76] S. Chopra, M. Auli and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks.," in *Proceeding of 2016 conference of the North American Chapter of the Association for Computational Linguistics,* California, 2016.

[77] C. Qu, L. Lu, A. Wang, W. Yang and Y. Chen, "Novel multi-domain attention for abstractive summarisation.," *CAAI Transactions on Intelligence Technology,* 2022.

[78] M. Qi, H. Liu, Y. Fu and T. Liu, "Improving Abstractive Dialogue Summarization with Hierarchical Pretraining and Topic Segment.," in *Findings of the Association for Computational Linguistics*, 2021.

[79] A. Kumar and M. K. Gupta, "Abstractive Summarization System.," *Journal of Electronics,* vol. 3, no. 4, pp. 309-319, 2021.

[80] W. Xu, C. Xiong and H. Cheng, "Research on Chinese Text Summarization Based on Core Word Attention Mechanism. In 2021," in *16th International Conference on Computer Science & Education (ICCSE)* , 2021.

[81] B. Baykara and T. Güngör, "Abstractive text summarization and new large-scale datasets for agglutinative languages Turkish and Hungarian.," *Language Resources and Evaluation,* pp. 1-35, 2022.

[82] R. &. L. D. K. Rani, "An extractive text summarization approach using tagged-LDA based topic modeling.," *Multimedia tools and applications,* vol. 80, no. 3, pp. 3275-3305, 2021.

[83] Y. Zou, L. Zhao, Y. Kang, J. Lin, M. Peng, Z. Jiang and X. Liu, "Topic-oriented spoken dialogue summarization for customer service with saliency-aware topic modeling," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2021.

[84] K. Issam and S. Patel, "Topic modeling based extractive text summarization.," *The International Journal of Innovative Technology and Exploring Engineering,* vol. 9, 2020.

[85] C. Ma, W. E. Zhang, M. Guo, H. Wang and Q. Z. Sheng, "Multi-document summarization via deep learning techniques: A survey.," *ACM Computing Surveys (CSUR).,* 2020.

[86] R. M. Alguliyev, R. M. Aliguliyev, N. R. Isazade, A. Abdi and N. Idris, "COSUM: Text summarization based on clustering and optimization.," *Expert Systems,* vol. 36, no. 1, 2019.

[87] S. Akter, A. S. Asa, M. P. Uddin, M. D. Hossain, S. K. Roy and M. I. Afjal, "An extractive text summarization technique for Bengali document (s) using K-means clustering algorithm.," in *IEEE International Conference on Imaging,* 2017.

[88] N. Alami, M. Meknassi, N. En-nahnahi, Y. El Adlouni and O. Ammor, "Unsupervised neural networks for automatic Arabic text summarization using document clustering and topic modeling.," *Expert Systems with Applications,* vol. 172, 2021.

[89] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey.," *Artificial Intelligence Review,* vol. 47, no. 1, pp. 1-66, 2017.

[90] A. A. Deshpande and V. G. Kottawar, "Survey of Sentence Scoring Techniques for Extractive Text Summarization.," in *Proceeding of International Conference on Computational Science and Applications*, Singapore, 2021.

[91] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou and T. Zhao, "Neural document summarization by jointly learning to score and select sentences.," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[92] G. K. Kumar and D. M. Rani, "Paragraph summarization based on word frequency using NLP techniques.," in *AIP conference proceedings*.

[93] M. AbdelFattah and F. Ren, "GA, MR, FFNN, PNN and GMM based models for automatic text summarization," *Computer Speech & Language,* vol. 23, no. 1, pp. 126-144, 2009.

[94] A. Joshi, E. Fidalgo, E. Alegre and R. Alaiz-Rodriguez, "RankSum—An unsupervised extractive text summarization based on rank fusion," *Expert Systems with Applications,* vol. 200, 2022.

[95] A. Qaroush, I. A. Farha, W. Ghanem, M. Washaha and E. Maali, "An efficient single document Arabic text summarization using a combination of statistical and semantic features.," *Journal of King Saud University-Computer and Information Sciences,* vol. 33, no. 6, pp. 677-692, 2021.

[96] C.-Y. Lin and E. Hovy, "The automated acquisition of topic signatures for text summarization.," in *Proceedings of the 18th conference on Computational linguistics*, 2000.

[97] R. C. Belwal, S. Rai and A. Gupta, "Text summarization using topic-based vector space model and semantic measure.," *Information Processing & Management,* vol. 58, no. 3, 2021.

[98] R. Srivastava, P. Singh, Rana, K. P. S. and V. Kumar, "A topic modeled unsupervised approach to single document extractive text summarization.," *Knowledge-Based Systems,* vol. 246, 2022.

[99] s. Lemmatizer, "spaCy API Lemmatizer," 2022. [Online]. Available: https://spacy.io/api/. [Accessed 25 10 2022].

[100] M. Ailem, B. Zhang and F. Sha, "Topic augmented generator for abstractive summarization.," *arXiv preprint* , 2019.

[101] R. C. Belwal, S. Rai and A. Gupta, "Extractive text summarization using clustering-based topic modeling.," *Soft Computing,* pp. 1-18, 2022.

[102] G. Erkan and D. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research,* vol. 22, p. 457–479, 2004.

[103] C. Mallick, A. K. Das, M. Dutta, A. K. Das and A. Sarkar, " Graph-based text summarization using modified TextRank.," in *Soft computing in data analytics*, Singapore., Springer, 2019, pp. 137-146.

[104] M. Zhang, X. Li, S. Yue and L. Yang, "An empirical study of TextRank for keyword extraction.," *IEEE Access,* vol. 8, pp. 178849-178858, 2020.

[105] M. Mohamed and M. Oussalah, "SRL-ESA-TextSum: A text summarization approach based on semantic role labeling and explicit semantic analysis.," *Information Processing & Management,* vol. 56, no. 4, pp. 1356-1372, 2019.

[106] W. S. El-Kassas, C. R. Salama, A. A. Rafea and H. K. Mohamed, "EdgeSumm: Graph-based framework for automatic text summarization.," *Information Processing & Management,* vol. 57, no. 6, 2020.

[107] T. Uçkan and A. Karcı, "Extractive multi-document text summarization based on graph independent sets.," *Egyptian Informatics Journal,* vol. 21, no. 3, pp. 145-157, 2020.

[108] R. C. Belwal, S. Rai and A. Gupta, "A new graph-based extractive text summarization using keywords or topic modeling.," *Journal of Ambient Intelligence and Humanized Computing,* vol. 12, no. 10, pp. 8975-8990, 2021.

[109] X. Mao, H. Yang, S. Huang, Y. Liu and R. Li, "Extractive summarization using supervised and unsupervised learning.," *Expert systems with applications,* vol. 133, pp. 173-181, 2019.

[110] S. Adhikari, "Nlp based machine learning approaches for text summarization.," in *Fourth International Conference on Computing Methodologies and Communication (ICCMC),* IEEE, 2020.

[111] Z. Nasar, S. W. Jaffry and M. K. Malik, "Textual keyword extraction and summarization: State-of-the-art.," *Information Processing & Management,* vol. 56, no. 6, 2019.

[112] X. Mao, H. Yang, S. Huang, Y. Liu and R. Li, "Extractive summarization using supervised and unsupervised learning.," *Expert systems with applications,* vol. 133, pp. 173-181, 2019.

[113] S. Bae, T. Kim, J. Kim and S. G. Lee, "Summary level training of sentence rewriting for abstractive summarization.," *arXiv preprint* , 2019.

[114] T. Ma, Q. Pan, H. Rong, Y. Qian, Y. Tian and N. Al-Nabhan, "T-bertsum: Topic-aware text summarization based on bert.," *IEEE Transactions on Computational Social Systems,* vol. 9, no. 3, pp. 879-890, 2021.

[115] Q. P. J. &. G. E. Grail, "Globalizing BERT-based transformer architectures for long document summarization.," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics,* 2021.

[116] W. Mann and S. Thompson, " Rhetorical structure theory: Toward a functional theory of text organization.," *Text 8,* pp. 243-281, 1988.

[117] S. Hou, S. Zhang and C. Fei, "Rhetorical structure theory: A comprehensive review of theory, parsing methods and applications.," *Expert Systems with Applications,* vol. 157, 2020.

[118] T. Ishigaki, H. Kamigaito, H. Takamura and M. Okumura, "Discourse-aware hierarchical attention network for extractive single-document summarization.," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing.,* 2019.

[119] J. Xu, Z. Gan, Y. Cheng and J. Liu, "Discourse-aware neural extractive text summarization.," *arXiv preprint,* 2019.

[120] Z. Liu and N. Chen, "Exploiting discourse-level segmentation for extractive summarization.," in *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, 2021.

[121] X. Feng, X. Feng, B. Qin, X. Geng and T. Liu, "Dialogue discourse-aware graph convolutional networks for abstractive meeting summarization.," *arXiv preprint,* 2021.

[122] J. Chen and D. Yang, "Structure-aware abstractive conversation summarization via discourse and action graphs.," *arXiv preprint,* 2021.

[123] J. Bogatinovski, L. Todorovski, S. Džeroski and D. Kocev, "Comprehensive comparative study of multi-label classification methods.," *Expert Systems with Applications,* vol. 203, pp. 1-18, 2022.

[124] J. Read, B. Pfahringer, G. Holmes and E. Frank, "Classifier chains for multi-label classification.," in *Joint European conference on machine learning and knowledge discovery in databases.,* Berlin, 2009.

[125] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification.," in *Pacific-Asia conference on knowledge discovery and data mining.,* Berlin, 2004.

[126] W. Weng, C. L. Chen, S. X. Wu, Y. W. Li and J. Wen, "An efficient stacking model of multi-label classification based on pareto optimum.," *IEEE Access,* vol. 7, pp. 127427-127437, 2019.

[127] E. Montañes, R. Senge, J. Barranquero, J. R. Quevedo, J. J. del Coz and E. Hüllermeier, "Dependent binary relevance models for multi-label classification.," *Pattern Recognition,* vol. 47, no. 3, pp. 1494-1508, 2014.

[128] Y. N. Chen, W. Weng, S. X. Wu, B. H. Chen, Y. L. Fan and J. H. Liu, "An efficient stacking model with label selection for multi-label classification.," *Applied Intelligence,* vol. 51, no. 1, pp. 308-325, 2021.

[129] B. B. Jia and M. L. Zhang, "Multi-dimensional classification via kNN feature augmentation.," vol. 106, p. Pattern Recognition, 2020.

[130] B. B. Jia and M. L. Zhang, "Multi-dimensional classification via selective feature augmentation.," *Machine Intelligence Research,* vol. 19, no. 1, pp. 38-51, 2022.

[131] H. Wang, C. Chen, W. Liu, K. Chen, T. Hu and G. Chen, "Incorporating label embedding and feature augmentation for multi-dimensional classification.," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence,* New York, 2020.

[132] P. De Handschutter, N. Gillis and X. Siebert, "A survey on deep matrix factorizations.," *Computer Science Review,* vol. 42, pp. 1-18, 2021.

[133] A. Luque, A. Carrasco, A. Martín and A. de Las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix.," *Pattern Recognition,* vol. 91, pp. 216-231, 2019.

[134] N. K. Mishra and P. K. Singh, "Feature construction and smote-based imbalance handling for multi-label learning.," *Information Sciences,* vol. 563, pp. 342-357, 2021.

[135] M. Han and H. Zhang, "Multiple kernel learning for label relation and class imbalance in multi-label learning.," *Information Sciences,* vol. 613, pp. 344-356, 2022.

[136] K. Duarte, Y. Rawat and M. Shah, "Plm: Partial label masking for imbalanced multi-label classification.," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2021.

[137] Y. Kim, Y. Lee and M. Jeon, "Imbalanced image classification with complement cross entropy.," *Pattern Recognition Letters,* vol. 151, pp. 33-40., 2021.

[138] J. Ortigosa-Hernández, I. Inza and J. A. Lozano, "Measuring the class-imbalance extent of multi-class problems.," *Pattern Recognition Letters,* vol. 98, pp. 32-38, 2017.

[139] Q. Yang, Y. Li, B. Li and Y. Gong, "A novel multi-class classification model for schizophrenia, bipolar disorder and healthy controls using comprehensive transcriptomic data.," *Computers in Biology and Medicine,* vol. 148, pp. 1-11, 2022.

[140] H. Xie, W. Lin, S. Lin, J. Wang and L. C. Yu, "A multi-dimensional relation model for dimensional sentiment analysis.," *Information Sciences,* vol. 579, pp. 832-844, 2021.

[141] N. Rai, D. Kumar, N. Kaushik, C. Raj and A. Ali, "Fake News Classification using transformer based enhanced LSTM and BERT.," *International Journal of Cognitive Computing in Engineering,* vol. 3, pp. 98-105, 2022.

[142] I. Ameer, N. Bölücü, M. H. F. Siddiqui, B. Can, G. Sidorov and A. Gelbukh, "Multi-label emotion classification in texts using transfer learning.," *Expert Systems with Applications,* vol. 213, 2023.

[143] T. Groza, S. Handschuh, K. Möller and S. Decker, "SALT-Semantically Annotated $\mbox {\LaTeX} $ for Scientific Publications.," in *European Semantic Web Conference,* Berlin, 2007.

[144] A. Brack, A. Hoppe, M. Stocker, S. Auer and R. Ewerth, "Analysing the requirements for an Open Research Knowledge Graph: use cases, quality requirements, and construction strategies.," *International Journal on Digital Libraries,* vol. 23, no. 1, pp. 33-55, 2022.

[145] A. Constantin, S. Peroni, S. Pettifer, D. Shotton and F. Vitali, "The document components ontology (DoCO).," *Semantic web,* vol. 7, no. 2, pp. 167-181, 2016.

[146] M. Firestone and B. L., "Paragraph Form: Definition, Types & Examples," 2022. [Online]. Available: http://study.com/academy/lesson/paragraph-form-definition-types-examples.html. [Accessed 26 10 2022].

[147] Dictionary.com, "Sentence," 2022. [Online]. Available: http://dictionary.reference.com/browse/sentence. [Accessed 26 10 2022].

[148] G. Sidorov, Syntactic n-grams in Computational Linguistics, Cham: Springer International Publishing, 2019.

[149] OxfordDictionary, "oxforddictionaries.com," 2015. [Online]. Available: http://www.oxforddictionaries.com/us/definition/american_english/word. [Accessed 8 11 2015].

[150] Merriam-Webster, "Word," 2022. [Online]. Available: https://www.merriam-webster.com/dictionary/word. [Accessed 26 10 2022].

[151] G. Pant, P. Srinivasan and F. Menczer, "Crawling the Web," in *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, 2004, pp. 153-178.

[152] D. Khyani, B. S. Siddhartha, N. M. Niveditha and B. M. Divya, "An Interpretation of Lemmatization and Stemming in Natural Language Processing.," *Journal of University of Shanghai for Science and Technology.,* vol. 22, no. 10, pp. 350-357, 2021.

[153] A. R. Martinez, "Part-of-speech tagging.," *Wiley Interdisciplinary Reviews: Computational Statistics,* vol. 4, no. 1, pp. 107-113, 2012.

[154] K. Janaki Raman and K. Meenakshi, Automatic text summarization of article (NEWS) using lexical chains and wordnet—A review., Singapore: Springer, 2021, pp. 271-282.

[155] N. &. C. S. Moratanch, "A survey on extractive text summarization.," in *International conference on computer, communication and signal processing (ICCCSP),* Chennai, 2017.

[156] NIST, "DUC 2002," 2002. [Online]. Available: http://www-nlpir.nist.gov/projects/duc/past_duc/duc2002/test.html. [Accessed 1 Nov 2015].

[157] J. Rojas-Simon, Y. Ledeneva and R. Garcia-Hernandez, "State-of-the-art Automatic Evaluation Methods," in *Evaluation of Text Summaries Based on Linear Optimization of Content Metrics*, Cham., Sprinher, 2022, pp. 107-136.

[158] C.-Y. Lin and E. H. Hovy, "Automatic evaluation of summaries using N-gram co-occurrence statistics," in *HLT NAACL*, 2003.

[159] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *ACL - Workshop on Text Summarization Branches Out.*, 2004.

[160] K. Papineni, S. Roukos, T. Ward and W. J. Zhu, "BLEU: a method for automatic evaluation of machine translation.," in *40th Annual meeting of the Association for Computational Linguistics.*, Philadelphia, 2002.

[161] A. Joshi, E. Fidalgo, E. Alegre and L. Fernández-Robles, "DeepSumm: Exploiting topic models and sequence to sequence networks for extractive text summarization.," *Expert Systems with Applications,* vol. 211, 2023.

[162] D. Parveen, H. M. Ramsl and M. Strube, "Topical coherence for graph-based extractive summarization.," in *Proceedings of the 2015 conference on empirical methods in natural language processing.*, 2015.

[163] C. Fang, D. Mu, Z. Deng and Z. Wu, "Word-sentence co-ranking for automatic extractive text summarization.," *Expert Systems with Applications,* vol. 72, p. 189–195, 2017.

[164] A. Jain and D. T. M. K. Bhatia, "Extractive text summarization using word vector embedding.," in *International Conference on machine learning and data science (MLDS)*, IEEE, 2017.

[165] A. Joshi, E. Fidalgo, E. Alegre and L. Fernández-Robles, "SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders.," *Expert Systems with Applications,* vol. 129, pp. 200-215, 2019.

[166] R. Nallapati, F. Zhai and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In," in *Thirty-first AAAI conference on artificial intelligence.*, 2017.

[167] K. Al-Sabahi, Z. Zuping and M. Nadher, "A hierarchical structured self-attentive model for extractive document summarization (HSSAS).," *IEEE Access,* pp. 24205-24212, 2018.

[168] X. Mao, H. Yang, S. Huang, Y. Liu and R. Li, "Extractive summarization using supervised and unsupervised learning.," *Expert systems with applications,* vol. 133, pp. 173-181, 2019.

[169] S. Ghodratnama, A. Beheshti, M. Zakershahrak and F. Sobhanmanesh, "Intelligent narrative summaries: From indicative to informative summarization.," *Big Data Research,* vol. 26, 2021.

[170] M. Aparício, P. Figueiredo, F. Raposo, D. M. de Matos, R. Ribeiro and L. Marujo, "Summarization of films and documentaries based on subtitles and scripts.," *Pattern Recognition Letters,* vol. 73, pp. 7-12, 2016.

[171] T. Kohonen, "The Self-Organizing Map," in *Proceedings of the IEEE 78,* 1990.

[172] L. Weigang, "A Study of Parallel Self-Organizing Map," in *Proceedings of the International Joint Conference on Neural Networks* , 1999.

[173] A. A. Saleh and L. Weigang, "A New Variables Selection And Dimensionality Reduction Technique Coupled With Simca Method For The Classification Of Text Documents," in *Proceeding of MakeLearn & TIM Joint International Conference 2015*, Bari, Italy, 2015.

[174] R. Brereton, Chemometrics: Data Analysis for the Laboratory and Chemical Plant, Chichester: Wiley, 2003.

[175] A. A. Saleh, M. Hegazy, S. Abbas and A. Elkosasy, "Development of distribution maps of spectrally similar degradation products by Raman chemical imaging microscope coupled with a new variable selection technique and SIMCA classifier.," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy,* vol. 268, pp. 1-11, 2022.

[176] J. Smith, J. Everhart, W. Dickson, W. Knowler and R. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus.," in *Proceedings of the 10th Symposium on Computer Applications and Medical Care*, Piscataway, 1988.

[177] R. P. Gorman, Sejnowski and T. J., "Analysis of hidden units in a layered network trained to classify sonar Targets," *Neural Networks,* vol. 1, no. 1, pp. 75-89, 1988.

[178] D. Dua and C. Graff, "UCI Machine Learning Repository," *https://archive.ics.uci.edu/ml/datasets/banknote+authentication,* 2019.

[179] V. G. Sigillito, S. P. Wing, L. V. Hutton and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks.," *Johns Hopkins APL Technical Digest,* vol. 10, pp. 262-266, 1989.

[180] R. A. Fisher, "The use of multiple measurements in taxonomic problems.," *Annals of eugenics,* vol. 7, no. 2, pp. 179-188, 1936.

[181] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik and S. Żak, "Complete gradient clustering algorithm for features analysis of x-ray images.," in *Information technologies in biomedicine,* Berlin, 2010.

[182] D. Dua and C. Graff, "UCI Machine Learning Repository," *https://archive.ics.uci.edu/ml/datasets/Abalone,* 2019.

[183] M. R. Boutell, J. Luo, X. Shen and C. M. Brown, "Learning multi-label scene classification.," *Pattern recognition,* vol. 37, no. 9, pp. 1757-1771, 2004.

[184] K. Trohidis, G. Tsoumakas, G. Kalliris and I. P. Vlahavas, "Multi-label classification of music into emotions.," *ISMIR* , vol. 8, pp. 325-330, 2008.

[185] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification.," *Advances in neural information processing systems,* vol. 14, 2001.

[186] M. L. Zhang and Z. H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning.," *Pattern recognition,* vol. 40, no. 7, pp. 2038-2048, 2007.

[187] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository.," *http://archive.ics.uci.edu/ml/datasets/solar+flare,* 2017.

[188] A. Karalič and I. Bratko, "First order regression.," *Machine learning,* vol. 26, no. 2, pp. 147-176, 1997.

[189] S. Džeroski, D. Demšar and J. Grbović, "Predicting chemical parameters of river water quality from bioindicator data.," *Applied Intelligence,* vol. 13, no. 1, pp. 7-17, 2000.

[190] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools.," *Energy and buildings,* vol. 49, pp. 560-567, 2012.

[191] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview.," *International Journal of Data Warehousing and Mining,* vol. 3, no. 3, pp. 1-13, 2007.

[192] J. Wang, F. Tian, H. Yu, C. H. Liu, K. Zhan and X. Wang, "Diverse non-negative matrix factorization for multiview data representation.," *IEEE transactions on cybernetics*, vol. 48, no. 9, pp. 2620-2632, 2017.