Automatic Structured Abstract for Research Papers Supported by Tabular Format using NLP

Zainab Almugbel[1], Nahla El Haggar[2], Neda Bugshan[3]
Computer Science Department
Community College, Imam Abdulrahman Bin Faisal University, P. O. Box 1982, Dammam, Saudi Arabia

***Abstract*—The abstract is an extensive summary of a scientific paper that supports making a quick decision about reading it. The employment of a structured abstract is useful to represent the major components of the paper. This, in turn, enhances extracting information about the study. Regardless of the importance of the structured abstract, many computer science research papers do not apply it. This may lead to weak abstracts. This paper aims at implementing the natural language processing (NLP) techniques and machine learning on conventional abstracts to automatically generate structured abstracts that are formatted using the IMRaD (Introduction, Methods, Results, and Discussion) format which is considered as a predominant in medical, scientific writing. The effectiveness of such sentence classification, which is the capability of a method to produce an expected outcome of classifying unstructured abstracts in computer science research papers into IMRAD sections, depends on both feature selection and classification algorithm. This can be achieved via IMRaD Classifier by measuring the similarity of sentences between the structured and the unstructured abstracts of different research papers. After that, it can be classified the sentences into one of the IMRaD format tags based on the measured similarity value. Finally, the IMRaD Classifier is evaluated by applying Naïve Bayes (NB) and Support Vector Machine (SVM) classifiers on the same dataset. To conduct this work, we use dataset contains 250 conventional Computer Science abstracts for periods 2015 to 2018. This dataset is collected from two main websites: DBLP and IOS Press content library. In this paper, 200 xml based files are used for training, and 50 xml based files are used for testing. Thus, the dataset is 4x250 files where each file contains a set of sentences that belong to different abstracts but belong to the same IMRaD sections. The experimental results show that Naïve Bayes (NB) can predict better outcomes for each class (Introduction, method, results, Discussion and Conclusion) than Support Vector Machine (SVM). Furthermore, the performance of the classifier depends on an appropriate number of the representative feature selected from the text.**

***Keywords*—Natural language processing (NLP); Naïve Bayes (NB) classifier; SVM**

I. Introduction

The abstract is crucial to state the aim and the content of papers for authors. This is because it summarizes the scientific paper's key concepts and findings. The components of the abstract could be organized in a structured or an unstructured format. If the unstructured format is used, the abstract is called a conventional abstract. It is a set of sentences. The set briefly describes the scientific paper without following any format. This means the author may summarize the essential parts of the research paper from his/her point of view without considering any standards.

In contrast, the structured abstract follows a specific format to describe the paper [1]. This paper proposes employing the structured abstract based on IMRaD (Introduction, Methods, Results, and Discussion) format [2,3] in computer science research papers. The IMRaD format has many advantages for the authors, editors, and reviewers. This includes organizing ideas, remembering main elements, facilitating manuscripts evaluation process, improving computerized literature searching and enhancing the efficiency of finding specific information without skimming the entire paper [4,5,6]. For instance, researchers can make a quick decision about reading a paper based on its structured abstract [7]. Despite the advantages of the structured abstract, many computer science researchers prefer writing un-structured abstracts in their research papers. Therefore, this paper aims at applying the natural language processing (NLP) techniques and machine learning to automatically generate structured abstracts that are formatted using the IMRaD (Introduction, Methods, Results, and Discussion) format. This could indirectly contribute to enhancing the quality of the abstracts because it assists in identifying any missing IMRaD section. Moreover, this speeds up the process of finding specific information about the paper, such as methodologies or results, within the abstract. Thus, having a high-quality searchable abstract could increase the number of citations for the research paper.

The order of the paper as follows: Section 2 addresses a summary of previous related work in both automate structuring and similarity measurement. Section 3 presents what methodologies are used in this paper for structuring the conventional abstracts of the computer science research papers. This includes the term preprocessing method, the feature selection method, the training classifier, and cross-validation. Section 4 discusses the results of this work. Finally, conclusion and future work are stated in Section 5.

II. Related Work

Fatiregun et al. [1] examines the comparative advantage of structured abstracts over unstructured abstracts as documented by various articles on the subject and makes a recommendation for structuring abstracts in articles appearing in Nigerian Journals.

James Hartly et al. [8] illustrate the difference between structured and unstructured abstract. Structured abstracts are typically longer than traditional ones, but they are also judged

to be more informative and accessible. Authors and readers also judge them to be more useful than traditional abstracts. However, not all studies use "real-life" published examples from different authors in their work, and more work needs to be achieved in some cases.

Andrade [9] has provided recommendations on how to write an efficient abstract on conventions in abstract writing as well as on the advantages of structured abstracts.

G.H., Martín et al. [10] studies the similarity between research journals taking advantage of the semi-structured information that is usually available in the description of a research paper: abstract and additional features like their writers, keywords, and the journals in which they were published. After determining the elements included for similarity measurement, it uses the vector space model or by language modelling techniques to measure it.

S. Jeong et al. [11] is also used structured abstracts of the PubMed Central open access subset. It aims at developing an ontology-based abstract authoring support tool. This tool provides candidate lexical bundles organized according to IMRaD format and thereby helps to complete sentences in tabular format representation.

M. A. Morid et al. [12] uses two strategies feature-rich classifier and sentence location to classify the clinically useful sentences on PubMed abstracts. It shows that only results and conclusion headings contain the desired information.

The most recent study pointed out by S. Nam et al. [3] has explored the most useful linguistic features in MEDLINE papers where the constructed feature set consist of a bag of words, linguistic features, grammatical features, and structural features. The sentence's classification was improved when the feature set was evaluated on three datasets from the PubMed Central Open Access Subset. Indeed, this feature set influences the quality of classification.

III. Methodology

The methodology of the present investigation is introduced in this paper. In the first and second subsections, the source of data used to generate n-grams and the n-gram data preparation process are presented respectively. In the third subsection, classifier and a machine learning workbench utilized in the current study are suggested, including how the results are achieved and evaluated.

The proposed system, shown in Fig. 1 is divided into four parts: Getting raw data, pre-processing data, training classifier and cross-validation.

A. Dataset Preparation

In this paper, NLP is used to process the dataset in order to use it to the classifiers. The data from the XML file is used to create features and instances suitable for classification. To generate a classification file, we build a python program, called IMRaD Classifier, to extracts the features for each part of IMRaD describe these processes of feature extraction. The dataset contains 250 conventional abstracts that are first randomly selected from Computer Science research papers. Then, they are manually converted into structured abstracts. These papers met the following criteria [4, 10]:

Domain: Computer Science research papers

Source: the XML description of these papers is collected from two main websites: DBLP and IOS press content library [13].

Abstract length: the research papers are selected if their abstracts' word count is between 180 and 220.

Dataset: the following steps are used to assemble the dataset in this paper:

First, XML-based files are downloaded from DBLP. They contain the XML descriptors of the research papers, such as titles and authors, except their abstracts.

Second, the conventional abstract of each paper is transcribed manually from the IOS press content library into the related XML-based file.

Third, the conventional abstracts are structured manually using the (IMRaD) format (Introduction, Methods, Results, and Discussion). The sentences of the conventional abstracts are structured based on the descriptions of the IMRaD components (IMRaD tags) [4, 5, 6] that can be explained as follows:

- **Introduction tag** (<introduction>) contains the sentences that describe the research problem.
- **Method tag (**<method>) includes the sentences that describe what methodology is used to solve the research problem.
- **Results tag (**<results>) contains the sentences that describe the findings with respect to the method used.
- **Discussion and conclusion tag (**<discussion_conclusion >) contains the sentences that describe the results, the met objectives, major findings, and limitations.

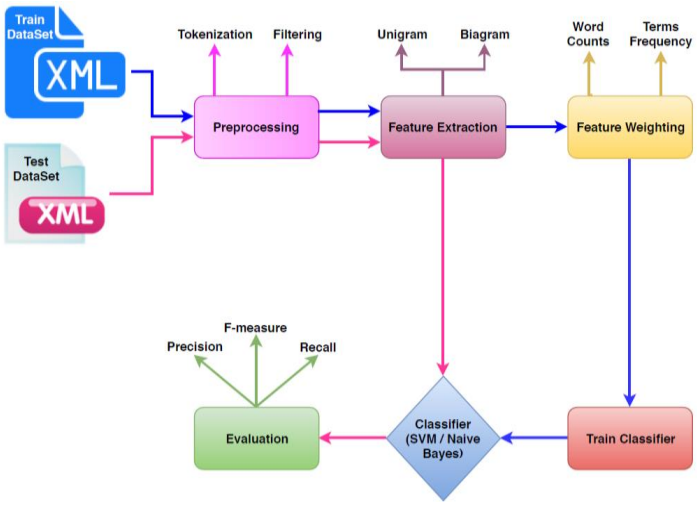

Fig. 1. Proposed System.

In our work, the dataset has two main properties:

1) It is divided into two sets of data (ratio = 75:25). The first dataset is used for the training set, and the other is used for the testing set. Thus, 4x200 XML-based files of the dataset are used for training, and 4x50 XML-based files are used for testing.

2) The IMRaD tags (<introduction>, <method>, <results>,<discussion_conclusion>) present the classes in the xml-based files

The IMRaD Classifier calls algorithms one and two in sequence. Both algorithm1 and algorithm2 are shown in Fig. 2 and Fig. 3. We will discuss them in details in the pre-processing and the feature extraction subsections.

B. Pre-Processing

The **preprocessing** stage is clarified in Algorithm 1. It starts with parsing the XML-based files to extract the structured abstracts of the training set. Then, each abstract's sentence is transcribed into a file based on its IMRaD XML-tag. Thus, four files are created at the end of this stage:

1) IntroFile includes the sentences that belong to <introduction>

2) MFile includes the sentences that belong to <method>

3) RFile includes the sentences that belong to <results>

4) DCFile includes the sentences that belong to <discussion_conclusion>

The preprocessing stage used in the framework includes the mostly used preprocessing tasks in NLP [14], which are:

- **Tokenization:** It is the process of dividing a sequence of string into pieces called tokens. The sentences converted into a list of terms by splitting into white spaces and removing punctuation.
- **POS Tagging:** The grammatical feature (Part of Speech) takes place to filter the available words in the sentences based on their part of speech using NLTK [15]. This helps to neglect the commonly used words such as propositions and pronouns.
- **N-gram Tagging:** In order to classify texts, a set of keywords that distinguish each class is required. In this paper, this is achieved by using the n-gram concept in which n-grams of different lengths are generated from a tag set. This set of n-grams (where n is set to 1 and 2) is primarily the result of moving a window of n characters along the text.

The word2vecort algorithm and nltk library are used to generate unigrams (where n=1) and bigrams (where n=2). They both applied to the four files mentioned in algorithm1 and to the merged file that contains the whole training set.

After extracting unigrams and bigrams, their frequency information is calculated for all related files. When the classification experiments are conducted, all frequency lists will be taken as inputs. By using n-grams, we do not need to perform word segmentation [16].

- **The Bag of Words:** Using machine learning methods to classify texts requires encoding the text as a feature vector. The most straightforward approach is to represent the document by a bag-of-words feature vector with the features being word occurrences.

C. Feature Extraction

In the feature extraction stage: the vector space model [17] generally utilizes to represent text documents from the training dataset as vectors of weighted features to classify it based on the maximization of the weight.

D. Abstract Representation using Various Weights

The Vector Space Model (VSM) is an algebraic model that represents text documents as vectors that makes use of the bag-of-words approach (BOW). Consequently, the M×N document-term matrix would be formed, where N is the number of documents, and M is the number of unique terms. Every unique term would be represented by a column, and each cell (i, j) keeps the number of term i which are in document j. Documents are described by word occurrences while completely ignoring the relative position [18] and [19].

Abstract A_j is then represented as a weighted vector $A_j = (w_1, w_2, \cdots, w_N)$. Each weight reflects the importance of that term in the abstract and/or in a given collection of IMRaD heading (Introduction, Methods, Results, and Discussion). The similarity between the two abstracts can then be assessed simply by comparing their vectors. "Abstract by the term" is constructed shown in Table 1, where T_i is n-gram, and each abstract is represented by a score of weight "w_{ij}".

w_{ij} = frequency of term T_i in abstract A_j, that is, TF_{ij} where: Generally, "w_{ij}" has been any of the following:

$$w_{ij} = TF_{ij} = \frac{n_{ij}}{\sum_k n_{ij}} \quad (1)$$

Algorithm1

```
1- Ask for the training set (set of XML-based files)
2- FOR each XMLFile in the training set
3-      FOR each XMLTag in the XMLFile
4-          IF XMLTag="introduction" THEN
5-                  Append IntroFile with XMLTag text
6-              ELSE IF XMLTag="method" THEN
7-                  Append MFile with XMLTag text
8-              ELSE IF XMLTag="resutls" THEN
9-                  Append RFile with XMLTag text
10-                     ELSE IF XMLTag=
                                " discussion_conclusion"
11-     THEN Append DCFile with XMLTag text
                            END FOR
                        END FOR
12- Apply tokenization then grammatical feature (POS)
to select terms from the four files (IntroFile, MFile,
RFile, DCFile)
13- Apply word2vector algorithm on the selected terms
to identify the keywords (unigrams) in each class
14- Apply the nltk library to determine the bigrams in
each class
```

Fig. 2. Algorithm1.

TABLE I. ABSTRACT BY TERM

Tag	Term				Class
	T_1	T_2		T_M	
Abs_{Intro}	W_{11}	W_{12}		W_{1M}	C_1
Abs_a	W_{21}	W_{22}		W_{2M}	C_2
Abs_R	W_{31}	W_{32}	…	W_{3M}	C_3
Abs_{DC}	W_{41}	W_{42}		W_{4M}	C_4
Abs_{IMRaD}	W_{N1}	W_{N2}		W_{NM}	C

$n_{i\,j}$ is the number of occurrences of the considered term in class c_i in abstract A_j where $\{A_j: T_i \in c_j , c_j \in A_i\}$ is the number of where the term T_i appears. Algorithm 2 version 1 in Fig. 3 conserves the sequence of IMRaD heading and the sentence position while classifying the sentences but Algorithm 2 version 2 does not.

E. Term Preprocessing within the Class

In the training set, each Term in dataset belongs to one class ci. Here, $c_i \in C$, $C = \{c_1, c_2..., c_n\}$, C is the class set defined before classification.

Algorithm2 version 1

// $T_{i\_\ F\ merged\ file}$: T_i Frequency in the merged file
// T_{i_FIntor}: T_i Frequency in Introduction File
// T_{i_FM}: T_i Frequency in Method File
// $T_{i\_FR:}$ T_i Frequency in Result File
// T_{i_FDC}: T_i Frequency in Discussion and Conclusion File
// **CA**: conventional abstract
// **SA**:structured abstract
1-Create a merged file of the four mentioned files
2-Apply word2vector algorithm to find the similar terms in each separated file and the merged file (a term and its similarities are considered as one term if similarity value is high)
3-Calculate the terms frequency in the merged file
4-Calculate the terms frequency in the four files separately
5-FOR EACH term T_i
6-Calculate the weight of T_i in each IMRaD heading :
a. $Ti\_Intor = T_{i\_FIntor} / T_{i\_\ F\ merged\ file}$
b. $Ti\_M = T_{i\_FM} / T_{i\_\ F\ merged\ file}$
c. $Ti\_R = T_{i\_FR} / T_{i\_\ F\ merged\ file}$
d. $Ti\_DC = T_{i\_FDC} / T_{i\_\ F\ merged\ file}$
7- Store T_i, its frequency, IMRAD heading (KB)
8- END FOR
9- Ask for the conventional abstract
10- FOR EACH sentence in **CA**
11- Retrieve the weights of its terms from KB
12- Sum its terms' weights for each specific IMRaD heading
13-Classify the sentence based on its maximum total weight
14- END FOR
15- Return **SA**

Fig. 3. Algorithm 2 Version 1.

"Abstract by the term" is constructed as shown in Table 1, where T_i includes all the n-grams (where n=1,2) extracted in the class c_i and T is n-gram set in all classes selected by Algorithm 3. However, the n-grams frequency in each class is higher than 9,000 on an average. Most of them occur only one or two times. Three kinds of weight "w_{ij}" are compared in this paper:

$$w_{ij} = TF_{ij} = \frac{n_{ij}}{\sum_{ij} TF_{ij} of\ \ key\ words\ in\ related\ class} \quad (2)$$

$$w_{ij} = TF_{ij} = \frac{n_{ij}}{\sum_{ij} TF_{ij}\ of\ key\ words\ in\ all\ classes} \quad (3)$$

$$w_{ij} = TF_{ij} = \frac{n_{ij}}{\sum_{ij} TF_{ij}\ \ in\ all\ classes} \quad (4)$$

We choose $\alpha = 0.5$ as the threshold in order to keep features as many as possible in each class.

F. Classification

Finally, once the feature is selected, it's the time to train the classifier. Classification is one of the critical steps in all machine learning's tasks.

Classification is a method of identifying to which set or category a new observation belongs, on the basis of a training dataset including observations whose class is known. Since we already have labelled all the instances, we only need to choose supervised learning classifiers.

Whenever the data to be used for training a supervised classifier is relatively little, the machine learning theory recommends to use a classifier with high bias/low variance (Naïve Bayes, SVM logistic regression, and decision trees) [20]. Based on that we decided to use Naïve Bayes and SVM in this research.

1) The naive bayes (NB): The Naive Bayes (NB) classifier [21, 22], in machine learning, is a supervised learning algorithm that uses a simple probability to determine the maximum likelihood of the occurrence of a possible solution. This algorithm is based on applying the Bayes' Theorem with the naive assumption of independence between every pair of features [21]. This classifier is very popular because classification using Naive Bayes algorithm is easy, quick and efficient.

Assume a variable C indicates the class of an observation O. The class of the observation O can be predicted using the Naive Bayes rule; we need to calculate the highest posterior probability of [23]:

$$P(C|O) = \frac{P(C)P(O|C)}{P(O)} \quad (5)$$

In the NB classifier, using the assumption of features O_1, O_2 O_n are conditionally independent on each other given the class, we get [23]:

$$P(C|O) = \frac{P(O)\prod_{i=1}^{n} P(O_i|C)}{P(O)} \quad (6)$$

2) Support vector machine (SVM): Another common method that is used to perform supervised learning using different classifiers in order to predict possible future solutions is Support Vector Machine (SVM).

Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, the algorithm outputs an optimal hyperplane for given training data which categorizes new examples. In spite of being a complicated process, SVM is widely regarded as one of the best text classification algorithms because of its effectiveness, accuracy, efficiency, and versatility. For implementing SVM, the training steps from 1 to 13 of the algorithm (1) are re-applied to the dataset. Then, the dataset is represented in a format that suits the inputs of LIBSVM [24]. The LIBSVM is used to evaluate the results of the different classifiers.

IV. Results and Discussion

We use dataset contains 250 conventional Computer Science abstracts for periods 2015 to 2018. This dataset is collected from two main websites: DBLP and IOS Press content library. First, the XML based descriptors of research papers are selected from DBLP to include papers with abstracts of 180-220 words length. Second, the papers' conventional abstracts are transcribed manually from IOS press content library into the XML descriptors. Third, the conventional abstract are converted into structured abstracts based on the (IMRaD) format (Introduction, Methods, Results, and Discussion). In this paper, 200 XML based files are used for training, and 50 XML based files are used for testing. Thus, the dataset is 4x250 files where each file contains a set of sentences that belong to different abstracts but belong to the same IMRaD section.

A. Natural Language Toolkit (NLTK)

NLTK [15] module is a huge toolkit, aimed at helping us with the entire Natural Language Processing (NLP) methodology. NLTK helped us with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words and then even with assisting the machine in understanding what the text is all about. Python's package NLTK is one of the most important packages for this paper. NLTK is a very suitable tool to work with while working with natural language and machine.

B. Analysis

For the analysis of our research, we use the F1 measure (F-Score) which is a measure of a test's accuracy. It considers the test's measurements: precision and recall to compute the score.

The F1 score takes a value between 0 and 1, where 0 is the worst possible score, and 1 is the top possible score. It is calculated using the precision (p) and recall (r) measures, defined as:

Precision is called the positive predictive value. It is the percentage of correctly predicted positive data (TP) overall predicted positive data.

$$Precision(P) = \frac{TP}{TP+FP} \quad (7)$$

Where TP is true positives number where the predicted outcome matches the actual value as positive, and FP is the false positives number or false alarms that occur when the prediction indicates that the result is positive, but the real value is negative. The computation of the classifier's performance is based on Precision [25].

The recall is the percentage of correctly predicted overall positive data. The recall is the ratio given by:

$$Recall\ (R) = \frac{TP}{TP+FN} \quad (8)$$

Where TP is the true positives number and FN is the number of false negatives that occur when the predicted solution is negative, but the actual value is positive.

The F- score can be interpreted as a weighted harmonic mean of the precision and recall, where it reaches its best value at one and worst score at zero.

$$F - Score = 2 * \frac{P*R}{P+R} \quad (9)$$

For multi-classes, the F- scores are summarized over the different categories using the Micro-averages and Macro-averages of F-Scores:

- Micro F-Score = average in documents and classes.
- Macro F-Score = average of within-category F values.

C. Comparison of Text Representation Weights

All experiments were validated using 10-fold cross-validation in which, the whole dataset is broken into ten equal sized sets and classifier is trained on nine datasets and tested on remaining dataset. This process is repeated ten times, and we take a mean accuracy of all fold. 1-, 2-gram combination has better performance than n-gram. Consequently, we set our experiments by comparing three kinds of feature selection methods by using 1-, 2-gram combination. That is, both 1-grams and 2-grams in the dataset are extracted as terms. We design three kinds of vector weights referred to in equation (2), (3) and (4). During the test process, the algorithm (2) was maintained to check if better results are possible. This includes the following:

1) Changing the weight calculation formula for each term . The formula in equation (4) gives better testing results than equation (2) and (3). Therefore, it is chosen.

2) Checking if conserving the sequence of (IMRaD) headings and the sentence position has to influence on the results. Based on the results in Tables 2 and 3, this has no significant influence on the performance of the algorithm as shown by the result by Table 4.

D. Analysis of NB and SVM

In this paper, we perform experiments using Naive Bayes (NB) and Support Vector Machine (SVM) classifiers. We use the F-Score which combines recall and precision as in equation (9) as shown in Fig. 5.

TABLE II. ALGORITHM 2 (V1) & ALGORITHM 3

	Algo.2 (V1)		
	Precision	**Recall**	**F-Score**
Overall	0.420142	0.46389	0.41433
Intro	1	0.79739	0.88727
Method	1	0.28205	0.44
Results	1	0.02703	0.05263
Dis&Con	1	0.20755	0.34375

TABLE III. ALGORITHM 2 (V2) & ALGORITHM 3

	Algo. 2 (V2)) & Algo.3		
	Precision	**Recall**	**F-Score**
Overall	0.42531	0.458333	0.432515
Intro	1	0.66667	0.8
Method	1	0.4359	0.60714
Results	1	0.02703	0.05263
Dis&Con	1	0.20755	0.34375

TABLE IV. ACCURACY COMPARISON BETWEEN ALGO.2 (V1, V2) & ALGO.3

Overall Accuracy	***Algorithm2 Ver 1 with Conserving IMRaD and sentence position***	***Algorithm2 Ver 2 without Conserving IMRaD and sentence position***
	0.46	0.458

Machine learning classifiers Naïve Bayes (NB) and SVM were trained and tested using the features created previously. A confusion matrix (as shown in Tables 7 and 8) is giving a more detailed description of the accuracy, and it is describing the types of errors that are being made by a model. This confusion matrix is often called a contingency table; accurate decisions are formed along the diagonal, in which each column represents prediction labels, and each row represent

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

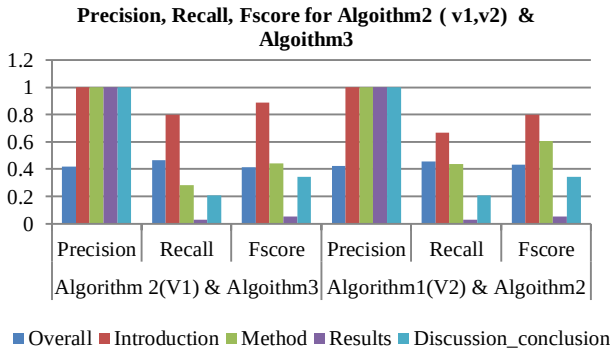

Fig. 4. Precision, Recall, F-Score Comparison between Algo.2 (V1, V2) & Algo.3.

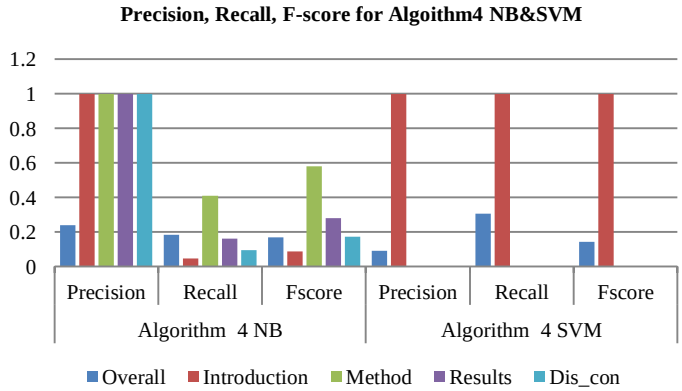

Fig. 5. Precision, Recall, F-Score Comparison between NB &SVM.

Machine learning classifiers Naïve Bayes (NB) and SVM were trained and tested using the features created previously. A confusion matrix (as shown in Tables 7 and 8) is giving a more detailed description of the accuracy, and it is describing the types of errors that are being made by a model. This confusion matrix is often called a contingency table; accurate decisions are formed along the diagonal, in which each column represents prediction labels, and each row represents actual labels.

In Table 7, the confusion matrix shows the predictions made by our model. It is a result of classification on the test set using 9,000 1- and 2-grams. The rows correspond to the known classes of the data, i.e. the labels in the data. The columns correspond to the predictions produced by the model. The diagonal elements show correct classifications number for each class.

TABLE V. PRECISION, RECALL, F-SCORE FOR NB

	Algorithm4 NB		
	Precision	**Recall**	**F-Score**
Overall	0.2402225	0.183333	0.171379
Intro	1	0.045752	0.0875
Method	1	0.410256	0.58182
Results	1	0.16216	0.27907
Dis&Con	1	0.09434	0.17241

TABLE VI. PRECISION, RECALL, F-SCORE FOR SVM

	Algorithm4 NB		
	Precision	**Recall**	**F-Score**
Overall	0.0931439	0.305195	0.142728
Intro	1	1	1
Method	0	0	0
Results	0	0	0
Dis&Con	0	0	0

TABLE VII. NAÏVE BAYES PERFORMANCE (CONFUSION MATRIX)

	Intro	Method	Results	Dis&Con
Intro	7	74	18	21
Method	12	48	13	17
Results	2	21	6	2
Dis&Con	4	33	1	5

Accuracy for NB = 0.18

Error rate = 1 – Accuracy = 0.81

TABLE VIII. SVM PERFORMANCE (CONFUSION MATRIX)

	Intro	Method	Results	Dis&Con
Intro	47	0	0	0
Method	51	0	0	0
Results	23	0	0	0
Dis&Con	33	0	0	0

Accuracy for SVM = 0.31

Error rate = 1 – Accuracy = 0.69

TABLE IX. MACRO-F AND MICRO-F FOR NB AND SVM

	Precision	Recall	F-Score
Macro-Average NB	47	0	0
Macro-Average SVM	51	0	0
Micro-Average NB	23	0	0
Micro-Average SVM	33	0	0

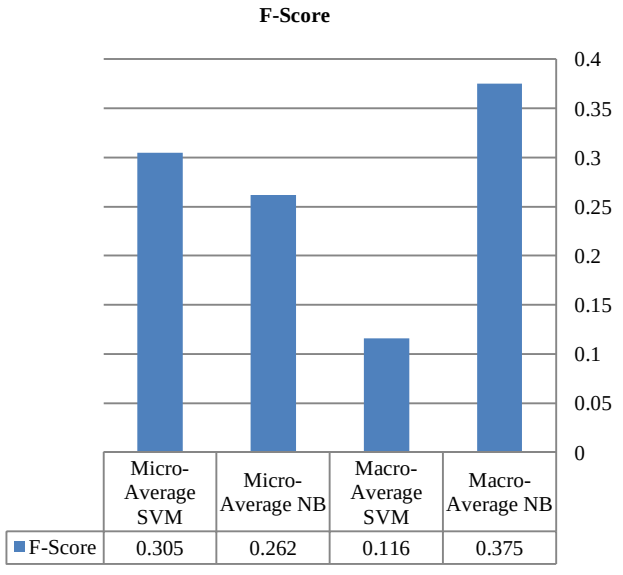

Fig. 6. Comparison of Macro & Micro F-Score Results.

The accuracy of classification techniques is evaluated based on the selected classifier algorithm like Naïve Bayes (NB) and Support Vector Machine (SVM). The predictive accuracy (Precision, Recall, F-Score) of Naïve Bayes (NB) and SVM on the testing sets which include 50 datasets are showed in Tables 5 and 6. From Table 7, the Overall accuracy of Precision, Recall and F-score for Naïve Bayes classifier is 24%, 18%, and 17% respectively. On the same way from Table 8, we calculated the overall accuracy of Precision, Recall and F-score for SVM which is 9%, 30%, and 14%. As we can see, the accuracy of SVM is slightly higher than Naïve Bayes.

Moreover, the values to measure the performance of each the classifiers (i.e. Precision, Recall, F-score) are derived from the confusion matrix presented in Tables 7 and 8. The confusion matrix used to evaluate the performance of the four-class classification problem. A macro-average results are shown in Table 9 is computed the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average results are aggregated the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable if there might be a class imbalance (i.e. there are many more examples of one class than of other classes). Fig. 6 depicts all previous described results.

From the experiments above, we could find that Macro F-score and Micro F-score give inconsistent results. As a result, we could compare them for each classifier NB and SVM. As shown in Fig. 4, SVM has better performance than NB, which indicates that feature selection based on 1- and 2-gram frequency in all classes is better than that depend on text frequency (Keyword in absolute or relative classes).

V. CONCLUSION AND FUTURE WORK

In this paper, a new technique was suggested by using Natural Language Processing (NLP) techniques and machine learning to generate automatic structuring of unstructured abstract according to IMRaD (Introduction, Methods, Results, and Discussion) format. This approach has been applied to short text for classification the unstructured abstracts then measure the similarity between sentences unstructured and structured abstracts that are found in the other research papers. Finally, evaluate the extracting feature technique by applying Naïve Bayes (NB) classifier sentences.

The results showed that text representation using TF weight formula in all classes gives better testing results than TF weight formula in keywords in related class and TF weight formula in keywords in all class. Therefore, it is chosen.

The accuracy of classification techniques is evaluated based on the selected classifier algorithm Naïve Bayes (NB) and Support Vector Machine (SVM) where the accuracy of SVM = 0.31 is slightly higher than Naïve Bayes =0.18. The reason for increasing the error rate may be caused by the existing similarity between some classes. It would be better to construct a multi-label classifier.

The performance of SVM calculated by Micro F-Score =0.305 has better performance than the performance of NB where Micro F-Score= 0.262. The reason for the decrease in performance is the unbalanced class distributions. Our future work will try to solve these problems. A promising direction for future work is using Tf*idf weight to represent the text and investigate the performance of feature selection methods on different machine learning classifiers.

REFERENCES

[1] Fatiregun AA, Asuzu MC, "Structured and unstructured abstracts in journal articles: a review".. 2003 Sep;10(3):197-200.

[2] James Hartley and Guillaume Cabanac. Thirteen ways to write an abstract. Publications, 5(2):11, 2017.

[3] Sejin Nam, Sang-Kyun Kim, Hong-Gee Kim, Victoria Ngo, Nansu Zong, et al. Structuralizing biomedical abstracts with discriminative linguistic features. Computers in biology and medicine, 79:276285, 2016.

[4] Jianguo Wu. Improving the writing of research papers: Imrad and beyond, 2011.

[5] Christian W Dawson. Projects in computing and information systems: a student's guide. Pearson Education, 2005.

[6] [PN08] WC Peh and KH Ng. The basic structure and types of scientific papers. Singapore medical journal, 49(7):522525, 2008.

[7] Grace Y Chung, "Sentence retrieval for abstracts of randomized "controlled trials." Published online 2009 Feb 10. BMC Med Inform Decis Mak

[8] James Hartley. Current findings from research on structured abstracts. Journal of the Medical Library Association, 92(3):368, 2004.

[9] Andrade, C. (2011). How to write a good abstract for a scientific paper or conference presentation. Indian Journal of Psychiatry, 53(2), 172–5. doi:10.4103/0019-5545.82558

[10] Germán Hurtado Martín, Steven Schockaert, Chris Cornelis, and Helga Naessens. Using semi-structured data for assessing research paper similarity. Information Sciences, 221:245261, 2013.

[11] Senator Jeong, Sejin Nam, and Hyun-Young Park. An ontology-based biomedical research paper authoring support tool. Science Editing, 1(1):37 42, 2014.

[12] Mohammad Amin Morid, Siddhartha Jonnalagadda, Marcelo Fiszman, Kalpana Raja, and Guilherme Del Fiol. Classification of clinically useful sentences in Medline. In AMIA Annual Symposium Proceedings, volume 2015. American Medical Informatics Association, 2015.

[13] https://content.iospress.com/& https://dblp.uni-trier.de/xml/

[14] Jurafsky D., & Martin J. H., (2000), An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice Hall Englewood Cliffs.

[15] Bird, Steven. "NLTK: the natural language toolkit." Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics, 2006.

[16] Xiaoyan Tang and Jing Cao "Automatic Genre Classification via N-grams of Part-of-Speech Tags " Procedia - Social and Behavioral Sciences 198 (2015) 474 – 478

[17] Salton G., Wong A., & Yang C. S., (1975), Vector Space Model for Automatic Indexing, Communications of the ACM, vol. 18, pp. 613–620.

[18] Daniel Jurafsky and James H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Prentice Hall, second edition, 2008.

[19] http://scikitlearn.org/stable/modules/feature_extraction.html

[20] Banko M. & Eric B., (2001), Scaling to Very Very Large Corpora for Natural Language Disambiguation, ACL '01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics Stroudsburg, PA, USA, pp. 26-33.

[21] H. Zhang (2004). “The Optimality of Naive Bayes.” Retrieved from: < http://scikit-learn.org/stable/modules/naive_bayes.html>

[22] Rish I., (2001), an Empirical Study of the Naive Bayes Classifier, IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence.

[23] Sonat T., & Musa M., (2013), Learning the Naïve Bayes Classifier with Optimization Models, International Journal of Applied Mathematics and Computer Science, vol. 23, no. 4, pp. 787–795.

[24] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification

[25] Sharma A, Dey S (2012) A document-level sentiment analysis approach using artificial neural network and sentiment lexicons. ACM SIGAPP Appl Comput Rev 12(4):67–75.