Domain Ontology Graph Approach Using Markov Clustering Algorithm for Text Classification

Rajinder Kaur and Mukesh Kumar

Abstract Text categorization means dividing a set of input documents into the two or more classes to which these documents belong. Because of increase in availability of data in digital form in large amount, it becomes necessary to organize it. Feature extraction is the crucial step in text classification. Most of the existing text classifiers are lacking in finding out the relations among the terms. We proposed a probabilistic approach for text classification in which the nonlinear relations among the terms are also considered. This model uses the domain ontology graph (DOG) with Markov clustering (MCL) algorithm. Here, ontology graph is constructed using DOG model and then clustering of ontology graph is done by MCL algorithm. This approach is scalable to huge dataset also and its classification power is not affected if relations among terms are large. Experimental results have shown that our system is 91% accurate for 8 categories and decreases, as we increase the classes from 8 to 10 and then to 12, from 91 to 88% and then to 85%, respectively. We have compared our classifier with existing Naive Bayes and *k*-Nearest Neighbor classifiers. Experimental results show that our proposed model is more accurate than these two classifiers. The better results demonstrated that our presented system is developed effectively.

Keywords Domain ontology graph · Markov clustering algorithm
Text mining · Text classification

R. Kaur (✉) · M. Kumar
Computer Science & Engineering, University Institute of Engineering & Technology, Panjab University, Chandigarh, India
e-mail: mrajinder14@gmail.com

M. Kumar
e-mail: mukesh_rai9@yahoo.com

S.S. Dash et al. (eds.), *International Conference on Intelligent Computing and Applications*, Advances in Intelligent Systems and Computing 632,
https://doi.org/10.1007/978-981-10-5520-1_47

1 Introduction

Text mining also known as text data mining or knowledge discovery in text (KDT) is the base of extracting high-quality information from raw data or text. High quality means the information should be according to user's need. Text classification is an active research field of text mining. As computers take text as sequence of strings, they can't extract useful information. So, specific algorithms and techniques should be used for preprocessing of raw data in order to get useful information or patterns [1].

Text (document) classification is the active research area of text mining in which assigning of text documents into classes or categories is done [2, 3]. These text documents include letters, newspapers, articles, blogs, proceedings, journal papers, etc. Text categorization means dividing a set of input documents into the two or more classes to which these documents belong. Because of increase in availability of data in digital form in large amount, it becomes necessary to organize it.

Text classification techniques can be divided into two categories: supervised document classification and unsupervised document classification (or document clustering). Supervised classification is one in which for defining the classes and classifying the documents, an external mechanism (e.g., human feedback) provides the information. Supervised machine learning techniques like Support Vector Machine, *k*-Nearest Neighbors, Naive Bayes, Decision Tree are applied frequently in text classification [1].

In unsupervised classification, the system doesn't have any pre-defined classes and it works without any external reference. Classification mode can also be semi-supervised in which some documents are pre-classified by external means for better learning of classifier. *k*-means, hierarchical clustering, etc., are commonly used as unsupervised learning techniques in text classification.

Text classification is divided into two phases: training phase and testing phase as shown in Fig. 1. Set of pre-classified or labeled documents $D = \{d_1, d_2, d_3, \ldots, d_n\}$ as training set is belonging to set classes $C = \{c_1, c_2, c_3, \ldots, c_p\}$. The training set is used for machine learning, i.e., to train the classifier. Depending upon the features selected, classifier is trained and classification algorithm is defined. The set of unlabeled documents referred as test set is used to test the classifier's accuracy by comparing the result driven by classifier for known label of document in the test set.

In this paper, we propose a probabilistic approach for text classification. It generates the domain ontology for each pre-defined class using training dataset. This model needs no human intervention in the process of ontology learning. Here, DOG is generated using MCL algorithm to train the classifier. The rest of the paper is composed of background, detailed methodology used for generating the DOGs and text classification algorithm, observations, conclusions and future scope, and limitations.

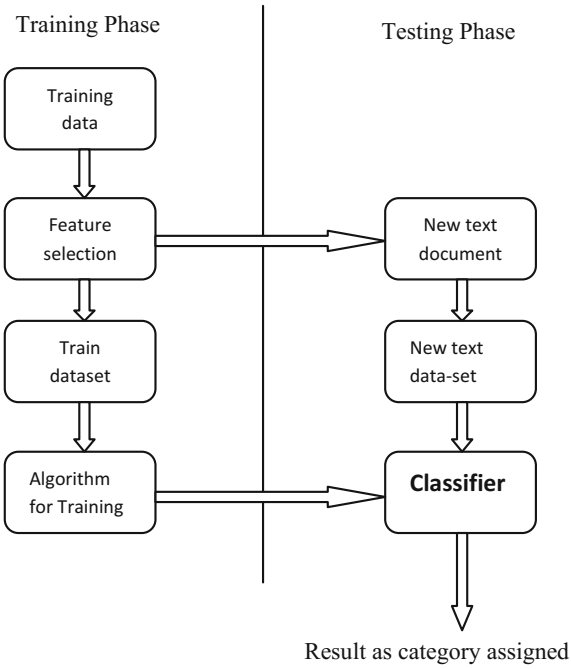

Fig. 1 Text classification steps

2 Background

2.1 Ontology

Ontology is basically a representation of real world's knowledge. Ontology defines a set of representative parameters for designing the model of domain of knowledge. These representational primitives are in machine readable format and are understandable by the human beings also [4, 5]. These formats are composed of attributes (properties), classes, and relationship among them. Ontology helps to develop knowledge-based systems, like Web search engines, text classification systems, content management systems, very effectively and efficiently. Ontology helps in real-time applications also. So we can conclude that ontology can be widely used as standard for semantic-based Web systems.

2.2 Ontology Learning

Ontology learning from textual data is very useful method as text data is the real source of human knowledge. Analyzing textual data requires some natural language processing approach [6, 7]. In recent years, for ontology learning most researchers

have used artificial intelligence approaches like machine learning or statistical analysis approaches. The knowledge in textual data is implicit and vague, but these techniques compute knowledge explicitly. So there are difficulties in both quality and quantity.

2.3 Text Classification

Text Classification assigns class to the unlabeled documents. It is a task of assigning a value to every $(d_i \times c_j) \in D \times C$; here, D is the set of all unlabeled documents, defined as $D = \{d_1, d_2, d_3, \ldots, d_n\}$ and C is domain of all pre-defined categories defined as $C = \{c_1, c_2, c_3, \ldots, c_p\}$. The main target of TC is to approximate the value of function $\text{ø} : D \times C \rightarrow \{T, F\}$. The value $\{d_i \times c_j\} \rightarrow T$ indicates that d_i belongs to class c_j, and the value calculated as $\{d_i \times c_j\} \rightarrow F$ indicates that d_i doesn't belongs to class c_j [8].

Most of the existing techniques [9–13] for text classification are lacking in finding the relation among the different terms of the document belonging to particular class. Sometimes, the results are biased and give error while classification. So relation among different terms of a class is very important point for classification and thus making the ontology. As existing system for text classification is not considering the term relation and treating every term as a unique identity for classification, error rate is high in them. If some systems have used the ontology for relation of terms, they are very complex and not much efficient. Ontology-based text classification improves the traditional system performance in terms of accuracy and also reduces the problem of over fitting. In this paper, we propose a probabilistic approach for text classification. It generates the domain ontology for each pre-defined class using training dataset. This model needs no human intervention in the process of ontology learning. Here, DOG is generated using MCL algorithm to train the classifier.

3 Methodology

Text classification process is divided into two phases: training phase and testing phase. In training phase, DOG is generated using feature extraction and MCL algorithm. During testing phase, text classification is done for unlabeled documents.

To model the ontology of knowledge in domain, ontology graph approach is used by the knowledge seeker system. Ontology graph is made up of four levels of conceptual units (CUs), linked together by different types of associations. The four CUs can be defined as:

Term T: the smallest conceptual unit extracted from the text which is relevant to the user's need.
Concept C: grouping up of related terms together with conceptual relation (CR) build the concepts, these are the basic units for concept graph (CG).
Concept Cluster CC: group of related clusters form a concept cluster CC. It tightly binds up the clusters to form hierarchy of knowledge.
Ontology Graph: grouping up of all CC forms a big and largest cluster of knowledge, termed as ontology graph.

3.1 Ontology Learning

At this stage, the domain of knowledge in the form of DOG is created. Graph creation is a knowledge-extraction process. A bottom–up approach is defined for extracting the features and designing the DOG in the form of CU (cluster unit) and CR (cluster relations). Bottom–up means the extraction is started from the smallest unit, i.e., term T and it ends up with the highest level, i.e., DOG. The five learning sub-processes are defined for ontology learning [14, 15]. These are the following:

I. **Term Extraction**: It is the process in which all the relevant terms are extracted from the dataset. A candidate term list T: $\{t_1, t_2, t_3, \ldots, t_n\}$ is extracted by eliminating the irrelevant terms from the text corpus. Stop word removal, stemming and lemmatization are done at this step.
II. **Term-to-Class Relationship Mapping**: The next step is term-to-class relationship mapping. The term-to-class mapping is done by using the nonlinear relation among the term and classes mutual information and information entropy is used for mapping. The information entropy for each term t and class c is calculated using Eq. 1.

$$H(X) = -\sum_{x \in X} P(x) \log p(x) \tag{1}$$

Then, mutual information among the term and class is calculated using Eq. 2.

$$I(t|c) = -\sum_{c \in C} \sum_{t \in T} P(t, c) \log \frac{p(t|c)}{p(t)p(c)} \tag{2}$$

Then, $R(t, c)$ relationship factor is calculated as:

$$R(t, c) = \frac{2I(t|c)}{H(t) + H(c)}. \tag{3}$$

If,

$R(t,c) < 1$: term a is negative dependence and not considered in the class c.

$R(t,c) > 1$: then term t is positive dependence and considered in the class c.

All the terms having negative dependence on class are ignored. And the terms having positive dependence are considered for class. So term lists are prepared for all the pre-defined classes/categories in this step.

III. **Term-to-Term Relationship Mapping**: Further interrelationship among the different terms in class is measured by term-to-term relationship mapping. Similarly, the term-to-term mapping is done by using the *R*-factor value as given by Eq. 4.

$$R(t_a, t_b) = \frac{2I(t_a|t_b)}{H(t_a) + H(t_b)} \tag{4}$$

Here, we will get the relationship factor among the terms extracted at previous step for each class. The *R*-factor visualization for two classes as an example for medical and space class is as shown in Fig. 2. In this figure, only first 11 terms are considered for each class.

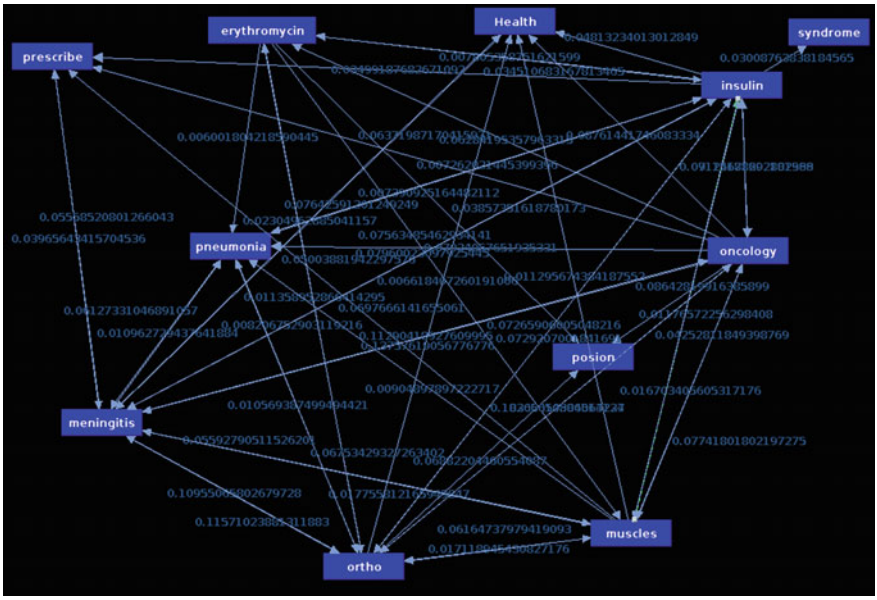

Fig. 2 Term-to-term relationship mapping for medical class

IV. **Concept Clustering using MCL algorithm**: at this step, the graph generated at previous level is clustered into tight semantic-related group. In this paper, Markov clustering algorithm [16] defined as Algorithm 1 for graph clustering is used.

Algorithm 1: The MCL algorithm for graph clustering

I. Input is an un-directed graph, power parameter e, and inflation parameter r.(by default e = r =2)
II. Creation of associated matrix;
III. Normalization of matrix;
IV. Expanding the matrix by taking up to eth power;
V. Inflation of resulting matrix with parameter r;
VI. Repetition of steps 4 and 5 until a steady state is reached;
VII. Interpretation of resulting matrix to find clusters.

V. **DOG Generation**: using the different concepts and relations defined in previous levels, DOG is generated. The node having maximum relation value among all terms in a cluster is selected as a label for that cluster.

Algorithm 2: DocOG generation algorithm

I. Input an unlabelled text document;
II. Obtain the term list of document as: T= $\{t1, t2, t3, \ldots, ti\}$;
II. Compute the term and relation set for document by comparing with all DOGs as T_d and R_d;
IV. Calculate the weight of every term in T_d by using $wi = tfi/n$; n=number of terms in document
V. For every pair of related terms calculate the weight of edges as $Wti, tj = wi * wj$;
VI. DocOG is generated using the created relation set and term set.

3.2 *Text Classification*

Text classification is achieved by finding the similarity of unlabeled document with the DOG. For this, text classification process is comprised of three steps: (1). Generation of DocOGs for the unlabeled document corresponding to each pre-defined class DOG. (2) Deriving the score vector of document for each class. (3) Select the class having highest score as category for the document.

3.2.1 Generation of DocOG

DocOG will be created by using the DOGs generated at previous level. An unlabeled document is input to the system, and then DocOGs of this document corresponding to all pre-defined classes are generated. Algorithm 2 is used to generate DocOG

3.2.2 Score Vector Calculation

Here the score vector for the document as given by Eq. 5.

$$\text{Score}\left(\text{doc}, \text{DOG}_j\right) = \text{score}\left(\text{DocOG}, \text{DOG}_j\right) \tag{5}$$

$S = \{S_1, S_2, S_3, \ldots, S_n\}$ is calculated for all the n-pre-defined categories. These scores are calculated by finding the number of nodes matching of all DocOGs with corresponding DOG.

3.2.3 Category Selection

After obtaining the score vector of all DocOGs here, comparison among the different scores for classes is done and the document is assigned to the class having highest score, i.e., the highest scored DocOG is selected as classified domain.

4 Software and Dataset

This proposed approach is implemented in Linux operating system using java, python, and C. Twenty Newsgroups dataset is downloaded from Jason Rennie's page. This dataset is a collection of 20,000 newspaper documents approximately, partitioned in 20 categories. This dataset is freely available. We have filtered the documents of only 12 classes, i.e., Advertisement, Automobile, Computers, Cryptography, Electronics, Games, Medical, Politics, Religion, Science, Graphics, and Windows for our research from these 20 categories dataset. Each file contains on an average of 70 words. We have used different number of classes for comparison and for checking the efficiency of classifier. First we had taken 8 categories then, we took 10 categories and then we had taken 12 categories for text classification. This variation of classes is done in order to check the effect of number of categories on classification power of classifier (Fig. 3).

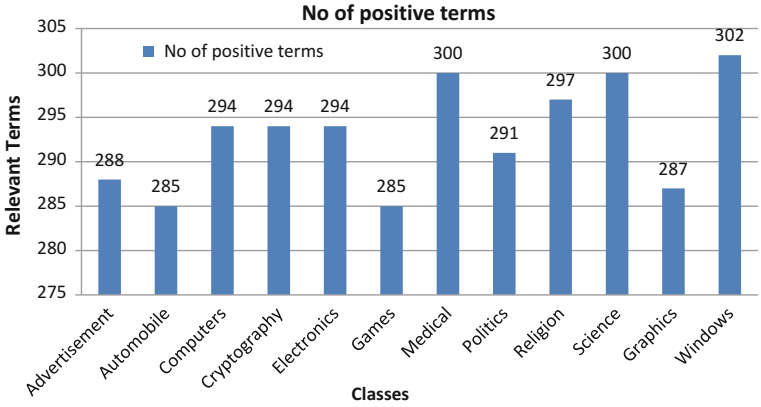

Fig. 3 Number of positive terms for each class

Table 1 Distribution of dataset

S. No.	Class	No. of documents	Used for learning	Used for testing	No of positive terms	No of negative terms
1	Advertisement	250	220	30	288	1423
2	Automobile	300	245	55	285	1444
3	Computers	249	215	34	294	1396
4	Cryptography	251	219	32	294	1414
5	Electronics	250	215	35	294	1398
6	Games	250	210	40	285	1405
7	Medical	250	210	40	300	1389
8	Politics	250	215	35	291	1379
9	Religion	252	215	37	297	1412
10	Science	250	220	30	300	1300
11	Graphics	250	220	30	287	1362
12	Windows	250	220	30	302	1317

4.1 Performance Measures

Error rate is the performance measure used to evaluate the classifier efficiency and accuracy. In this precision, recall and *f*-measure [8] are the basic performance measuring parameters. These can be defined as following (Table 1).

4.1.1 Precision

Precision is also known as positive predictive value. It calculates the accuracy by finding the percentage of documents correctly retrieved to the total retrieved documents.

Table 2 Contingency table

Category set $\mathbf{C} = \{c_1, c_2, c_3, \ldots, c_j\}$		Expert judgments	
		Yes	No
Classifier judgments	Yes	TP	FP
	No	FN	TN

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

Table 2 is the contingency table, in which TP is True Positive, FP is False Positive, FN is False Negative, and TN is True Negative.

- TP: number of documents correctly labeled as belonging to positive class.
- FP: number of documents incorrectly labeled as belonging to the class.
- FN: number of documents which are not labeled as belonging to the positive class but should have been.
- TN: number of documents which are correctly labeled as not belonging to the class.

4.1.2 Recall

Recall is also known as sensitivity. It calculates the ability of the classifier by measuring the percentage of correctly classified documents to the total classified documents.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

4.1.3 *F*-measure

It is the measure of harmonic mean of precision and recall. It gives the closeness between precision and recall. It is defined by as mentioned in Eq. 3.

$$F\text{-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{8}$$

4.2 Experimental Results

The proposed algorithm for text classification is implemented and compared with Naive Bayes and *k*-Nearest Neighbor classifier. Naive Bayes and *k*-Nearest Neighbor classifiers are implemented in Python using the inbuilt library "sklearn." In *k*-Nearest Neighbor algorithm, ten nearest neighbors are considered for measuring the distances in classification.

The three classifiers are implemented using 8, 10, and 12 classes or categories to measure the performance of classifiers effectively. It is done to evaluate and compare the effect of number of categories on the classification power of the classifier. To evaluate the power of classifiers, the comparison is done using precision, recall, and *f*-measure.

4.2.1 Experiment 1

In Experiment 1, we have considered the number of categories $N = 8$ for text classification. These are Automobile, Electronics, Religious, Sports, Medical, Cryptography, Science, and Politics. Then, the performance is evaluated using precision, recall, and *f*-measure. Table 3 shows the values of precision, recall, and *f*-measure for different models using number of classes $N = 8$. Figure 4 gives the representation for comparison of *f*-measure for different classifiers for different classes.

The accuracy power for DOG is 91%, while those of Naive Bayes are 84% and that of *k*-NN is 77%. This *f*-measure value shows that the DOG proposed model performs better than other two classifiers. *k*-NN has lowest accuracy. And *k*-NN has lower classification power as compared to others. Proposed model shows maximum of 97% accurate results for class Electronics and minimum of 81% for class Science. This result shows that proposed model gives better result as compared to

Table 3 Precision, recall, and *f*-measure for $N = 8$

Class	Precision			Recall			*f*-measure		
	NB	*k*-NN	DOG	NB	*k*-NN	DOG	NB	*k*-NN	DOG
Automobile	0.89	0.75	0.93	0.96	0.85	0.95	0.92	0.79	0.94
Electronics	0.95	0.84	0.94	0.97	0.88	0.99	0.96	0.86	0.97
Religious	0.58	0.71	0.92	0.98	0.9	0.96	0.73	0.8	0.94
Sports	0.93	0.79	0.89	0.7	0.62	0.87	0.8	0.69	0.88
Medical	0.92	0.87	0.93	0.88	0.62	0.87	0.9	0.72	0.9
Cryptography	0.88	0.87	0.92	0.94	0.82	0.96	0.91	0.84	0.94
Science	0.93	0.64	0.9	0.61	0.73	0.74	0.73	0.68	0.81
Politics	0.98	0.76	0.79	0.51	0.78	0.88	0.67	0.77	0.83
Average	0.88	0.78	0.91	0.84	0.77	0.91	0.84	0.77	0.91

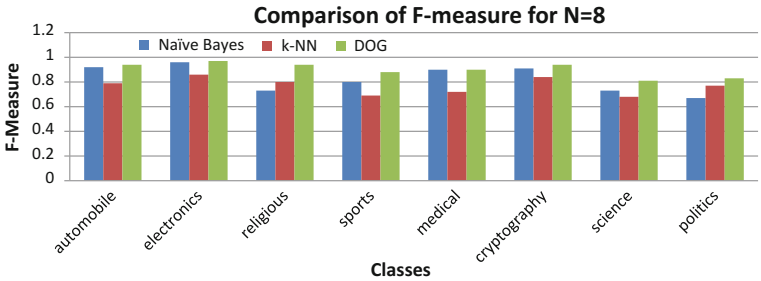

Fig. 4 Representation for comparison of *f*-measure for $N = 8$

Table 4 Precision, recall, and *f*-measure for $N = 10$

Class	Precision			Recall			*f*-measure		
	NB	*k*-NN	DOG	NB	*k*-NN	DOG	NB	*k*-NN	DOG
Automobile	0.83	0.61	0.84	0.86	0.73	0.9	0.84	0.67	0.87
Electronics	0.94	0.61	0.84	0.74	0.63	0.91	0.83	0.62	0.87
Religious	0.83	0.76	0.91	0.95	0.77	0.93	0.89	0.77	0.92
Sports	0.94	0.83	0.92	0.97	0.84	0.98	0.95	0.84	0.95
Medical	0.56	0.72	0.91	0.98	0.89	0.96	0.71	0.79	0.94
Cryptography	0.85	0.72	0.9	0.65	0.54	0.7	0.74	0.62	0.79
Science	0.9	0.86	0.93	0.88	0.58	0.87	0.89	0.69	0.9
Politics	0.87	0.85	0.9	0.94	0.81	0.96	0.9	0.83	0.93
Advertisement	0.92	0.64	0.91	0.6	0.71	0.72	0.72	0.67	0.8
Computer	0.98	0.75	0.78	0.5	0.76	0.87	0.66	0.76	0.82
Average	0.86	0.74	0.89	0.82	0.73	0.89	0.82	0.73	0.88

the other two techniques. This comparison can also be expressed using graphical representation. Figure 4 shows the graphical representation for comparison of the three techniques such as proposed model, Naive Bayes, and *k*-NN algorithm for text classification.

4.2.2 Experiment 2

In Experiment 2, we have considered the number of categories $N = 10$ for text classification. These are Automobile, Electronics, Religious, Sports, Medical, Cryptography, Science, Politics, Advertisement, and Computer. Then, the performance is evaluated using precision, recall, and *f*-measure. Table 4 shows the values of precision, recall, and *f*-measure for different models using number of classes $N = 10$. Figure 5 gives the representation for comparison of *f*-measure for different classifiers for different classes. The accuracy power for DOG is 88%, while those of Naive Bayes are 82% and that of *k*-NN is 73%.

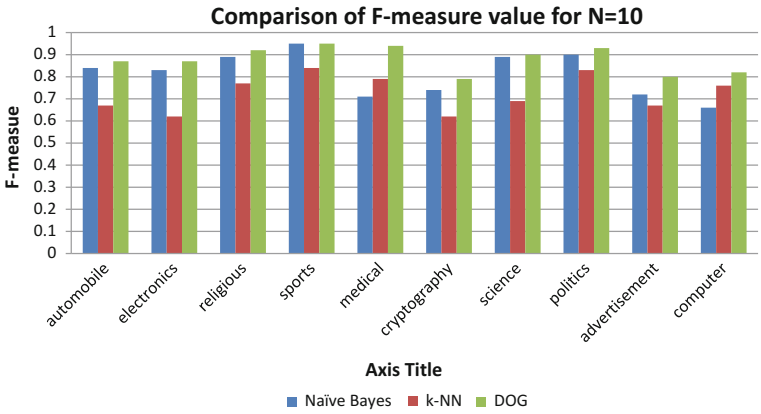

Fig. 5 Representation showing comparison of *f*-measure for $N = 10$

This *f*-measure value shows that the DOG performs better as compared to other two classifiers. *k*-NN has lowest accuracy. And *k*-NN has lower classification power as compared to others. Proposed model shows maximum of 95% accurate results for class Sports and minimum of 79% for class Cryptography. This result shows that proposed model gives better result as compared to the other two techniques. This comparison can also be expressed using graphical representation. Figure 5 shows the graphical representation for comparison of the three techniques such as proposed model, Naive Bayes, and *k*-NN algorithm for text classification.

4.2.3 Experiment 3

In Experiment 3, we have considered the number of categories $N = 12$ for text classification. These are Automobile, Electronics, Religious, Sports, Medical, Cryptography, Science, Politics, Advertisement, Computer, Graphics, and Windows. Then, the performance is evaluated using precision, recall, and *f*-measure. Table 5 shows the values of precision, recall, and *f*-measure for different models using number of classes $N = 10$. Figure 6 gives the representation for comparison of *f*-measure for different classifiers for different classes.

The accuracy power for DOG is 85%, while those of Naive Bayes are 79% and that of *k*-NN is 69%. This *f*-measure value shows that the DOG proposed model performs better than other two classifiers. *k*-NN has lowest accuracy. And *k*-NN has lower classification power as compared to others. Proposed model shows maximum of 92% accurate results for class Medical and Cryptography, and minimum of 76% for class Politics. This result shows that proposed model gives better result as compared to the other two techniques. This comparison can also be expressed using graphical representation. Figure 6 shows the graphical representation for comparison of the three techniques such as proposed model, Naive Bayes, and *k*-NN algorithm for text classification.

Table 5 Precision, recall, and *f*-measure for $N = 12$

Class	Precision			Recall			*f*-measure		
	NB	*k*-NN	DOG	NB	*k*-NN	DOG	NB	*k*-NN	DOG
Automobile	0.85	0.52	0.88	0.7	0.71	0.81	0.77	0.6	0.84
Electronics	0.87	0.57	0.83	0.67	0.6	0.83	0.76	0.58	0.83
Religious	0.7	0.59	0.78	0.8	0.64	0.76	0.75	0.62	0.77
Sports	0.95	0.61	0.83	0.73	0.54	0.91	0.82	0.57	0.87
Medical	0.82	0.77	0.91	0.95	0.73	0.93	0.88	0.75	0.92
Cryptography	0.91	0.82	0.87	0.97	0.82	0.98	0.94	0.82	0.92
Science	0.47	0.7	0.86	0.97	0.88	0.96	0.63	0.78	0.91
Politics	0.84	0.73	0.89	0.63	0.52	0.66	0.72	0.61	0.76
Advertisement	0.9	0.86	0.92	0.88	0.58	0.87	0.89	0.69	0.89
Computer	0.84	0.87	0.86	0.93	0.79	0.95	0.88	0.83	0.91
Graphics	0.93	0.62	0.89	0.56	0.7	0.7	0.7	0.66	0.78
Windows	0.98	0.73	0.74	0.45	0.77	0.86	0.62	0.75	0.8
Average	0.83	0.7	0.86	0.79	0.69	0.86	0.79	0.69	0.85

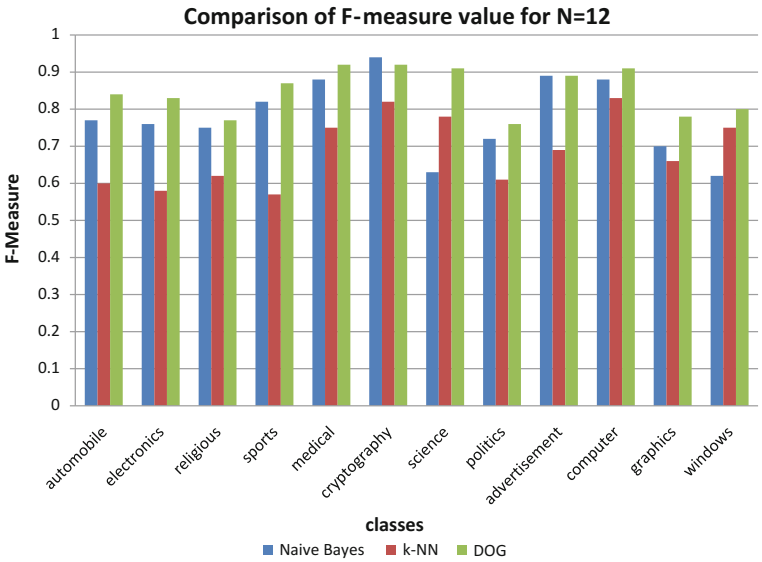

Fig. 6 Representation showing comparison of *f*-measure for $N = 12$

Table 6 Average value of *f*-measure for all classifiers

f-measure			
	Naive Bayes	*k*-NN	DOG
Classes = 8	0.84	0.77	0.91
Classes = 10	0.82	0.73	0.88
Classes = 12	0.79	0.69	0.85

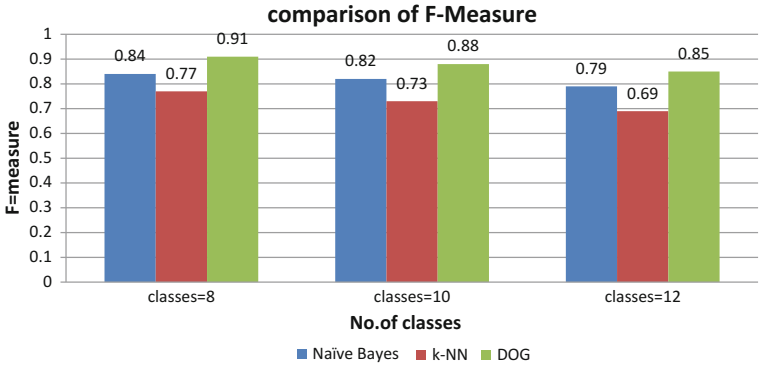

Fig. 7 Comparison of average *f*-measure value

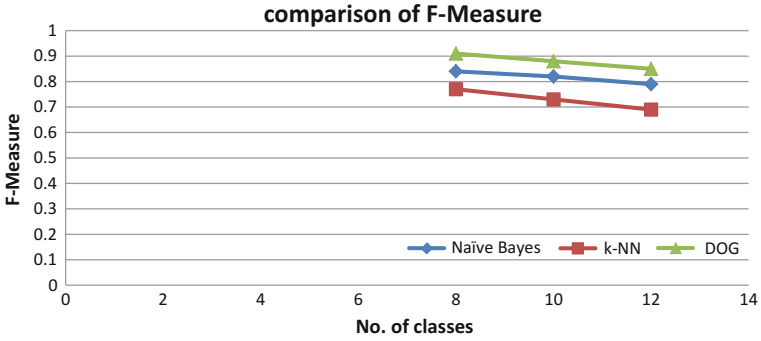

Fig. 8 Line-graph representations for *f*-measure comparison

4.2.4 Experiment 4

Experiment 4 shows the average value of precision, recall, and *f*-measure for all the three classifiers (Table 6).

Figures 7 and 8 show the graphical representation of average *f*-measure for all the three classifiers. These show that with increase in number of categories, the accuracy of the classifier decreases. As we can see that the *f*-measure value decreases for every classifier with increase in value of *N*. By comparing the average

values, it is also proved that the proposed classifier is more accurate than traditional Naive Bayes and *k*-NN approaches for text classification.

Figure 8 shows using line-graph representation how the accuracy power of different classifiers decreases with increase in number of categories. From this figure, it is also concluded that proposed classifier is the more accurate among the three classifiers.

5 Conclusion and Future Scope

This proposed scheme, to classify English texts by using probabilistic approach, is a fully automatic system. We just give the dataset and pre-defined classes as input to our system. DOG with hierarchical clustering was used for Chinese text. It is for the first time that DOG model with MCL clustering is used for English text. DOG model increases the classification power. Effective feature extraction is done by considering the nonlinear relations among the terms. Here, the domain ontology graph model is designed to generate the knowledge representation and MCL clustering algorithm is used to cluster the terms of the graph. The use of MCL algorithm makes the system efficient as it is mathematical approach, so is more accurate. This approach is scalable to huge dataset also and its classification power is not affected if relations among terms are large. But there are limitations also. We have not used the synonyms and antonyms while designing the ontology. Also, in MCL clustering, we perform the matrix multiplication as we are using the mathematical probabilistic approach. This matrix multiplication is of $O(n^3)$. So it is highly complex system.

This work has devised a text classification system. It is probabilistic approach for classifying the texts. We have used the DOG model with MCL clustering algorithm for English text classification. Experimental results have proved that it is an accurate and effective text classifier. This DOG model is used for the first time for text classification. But there are many things to do. In near future, this work can be extended. Some of the things which can be done are:

- Synonyms and antonyms can also be added while generating the domain ontology graph.
- Semantic-based learning approach can also be used in future for improving the system.
- It can also be applied to other languages.

References

1. Hotho, Andreas, Andreas Nürnberger, and Gerhard Paaß. "A brief survey of text mining." *Ldv Forum*. Vol. 20. No. 1. 2005.
2. W. Fan, L. Wallace, S. Rich, and Z. Zhang, "Tapping into the Power of Text Mining", Journal of ACM, Blacksburg, 2005.
3. C. Aggarwal, and C. Zhai, "A survey of text classification algorithms". In Mining text data. Springer. 2012. pp 163–222, 2012.
4. James N. K. Liu, Yu Lin He, Edward H.Y. Lim, "A New Method For Knowledge and Information Management Domain Ontology Graph Model" IEEE Transactions On Systems, Man, and Cybernetics: Systems, VOL. 43, No.1, January 2013.
5. P. Buitelaar and P. Ciomiano, Ontology Learning and Population: Bridging the Gap between Text and Knowledge. Amsterdam, The Netherlands: IOS Press, 2008.
6. M. Y. Dahab, H. A. Hassan, and A. Rafea, "TextOntoEx: Automatic ontology construction from natural English text," Expert Syst. Appl., vol. 34, no. 2, pp. 1474–1480, Feb. 2008.
7. R. Gacitua, P. Sawyer, and P. Rayson, "A flexible framework to experiment with ontology learning techniques," Knowl.-Based Syst., vol. 21, no. 3, pp. 192–199, Apr. 2008.
8. F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys. Vol. 34, No. 1, pp. 1–47, March 2002.
9. F. Sebastiani, "Text categorization", In Alessandro Zanasi (ed.), Text Mining and its Applications, WIT Press, Southampton, UK, pp. 109–129, 2005.
10. J. Han, M. Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann publishers, USA, 70–181, 2001.
11. S. Rasane, D. V. Patil, "Handling Various Issues In Text Classification: A Review", International Journal of Emerging Trends in Technology, Vol. 3, 2016.
12. J. Han, M. Kamber, "Data Mining: Concepts and Techniques," Elsevier, Second Edition, 2006.
13. Y. Yang "An Evolution of statistical Approaches to Text Categorization" Information Retrieval Vol. 1, pp. 69–90, 1999.
14. M. Y. Dahab, H. A. Hassan, and A. Rafea, "TextOntoEx: Automatic ontology construction from natural English text," Expert Syst. Appl., vol. 34, no. 2, pp. 1474–1480, Feb. 2008.
15. P. Soucy, and G. Mineau, "Feature Selection Strategies for Text Categorization", AI 2003, LNAI 2671, pp. 505–509, 2003.
16. Van Dongen, S. "Graph Clustering by Flow Simulation". PhD Thesis, University of Utrecht, The Netherlands, 2000.