TEXT MINING AND CLASSIFICATION OF PRODUCT REVIEWS USING STRUCTURED SUPPORT VECTOR MACHINE

Jincy B. Chrystal[1] and Stephy Joseph[2]

[1]M.Phil. Computer Science, IIITM-K, Trivandrum, Kerala
jincyb.mphilcs2@iiitmk.ac.in
[2]M.Phil. Computer Science, IIITM-K, Trivandrum, Kerala
stephyjoseph.mphilcs2@iiitmk.ac.in

ABSTRACT

Text mining and Text classification are the two prominent and challenging tasks in the field of Machine learning. Text mining refers to the process of deriving high quality and relevant information from text, while Text classification deals with the categorization of text documents into different classes. The real challenge in these areas is to address the problems like handling large text corpora, similarity of words in text documents, and association of text documents with a subset of class categories. The feature extraction and classification of such text documents require an efficient machine learning algorithm which performs automatic text classification. This paper describes the classification of product review documents as a multi-label classification scenario and addresses the problem using Structured Support Vector Machine. The work also explains the flexibility and performance of the proposed approach for efficient text classification.

KEYWORDS

Text classification, Multi-label classification, Structured Support Vector Machine

1. INTRODUCTION

With the rapid growth of technology and its applications, text data has become one of the important information sources in real world scenarios. In such a scenario, text classification plays an important role in organizing the text documents into different categories. Considering the convenience and relevance of text classification, the dataset used in this work encompasses a large collection of product reviews of electronic gadgets. This paper presents the construction of a classification model in multi-label domain for the classification of product review documents. As the first phase of this work, text mining is carried out inorder to model and structure the information content of textual sources. The result of text mining process generates text documents with relevant and high quality information which indeed contributes to efficient text classification. The work deals with the general problem of text classification, but using a new approach of Multiclass-Multilabel classification using Structured Support Vector Machine.

The Structured SVM is a supervised learning algorithm designed for complex outputs and structured output spaces and it performs the learning by using discriminant function over input-

Dhinaharan Nagamalai et al. (Eds) : CSITA, ARIN, ISPR - 2015
pp. 21–31, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.50803

output pairs. The learning phase of the specified method involves the feature extraction and grouping of features into multiple classes and hence multiple labels. Here the text classification is a multi-labeled problem, where each document can belong to more than one class. We propose a multi-label text classification model that maps a set of categories to each input document and so the output of the classifier will be a vector rather than a single class label. The resultant model thus performs multi-label text classification of product review documents and it also focuses on the precision, accuracy and performance of the system by the creation of a confusion matrix which measures the degree of prediction and classification of text documents.

2. METHODOLOGY

The proposed work describes Structured Support Vector Machine as a Multi-label text classifier for the classification of product review documents. The entire system is organized into four major modules namely, Preprocessing, Learning, Classification and Evaluation. The preprocessing stage involves the techniques and processes which completes task of text mining. The structured SVM is formulated by the training and testing modules which indeed represents the learning and classification tasks. Finally the evaluation phase measures the efficiency and performance of the system. The workflow of the proposed system is represented as follows.

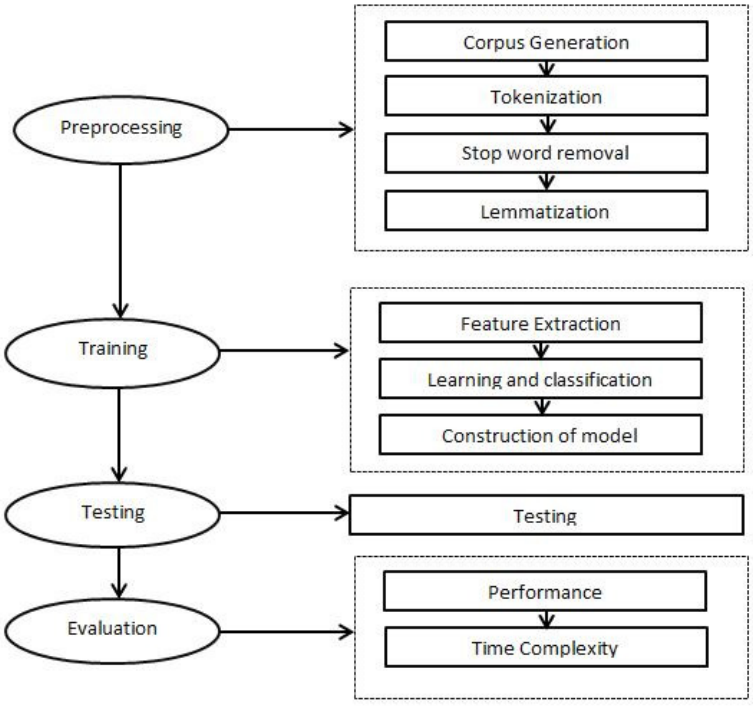

Figure 1. Proposed System

2.1. Corpus

The experiment of this work is carried out on a text corpus which is a collection of product reviews of various electronic gadgets. The electronic gadgets include Mobile Phones, Tablets, Laptops, Pendrives, Televisions, Datacards, Memory cards, Printers, Speakers, Washing Machines, Air conditioners, Vacuum Cleaners, Fans, Microwave Ovens etc. The corpus is

organized as a multi-label dataset with 105 features, 25 classes, 1500 review documents for training, and 500 review documents for testing.

2.2. Pre-processing

2.2.1. Tokenization

Tokenization is the first pre-processing stride of any natural language processing and corpus generation. It is the process of replacing the meaningful sentence in to individual words with space as the delimiter and it retain all the valuable information's. Each individual words are known as tokens. These tokens are the key elements of the NLP.

In our experiment, tokenization is one of the pre-processing steps of our corpus generation. One lakhs corpus was considered in our work. The tokenization methodology is used to split the corpus sentence into words. The python script are using for corpus tokenization. In our corpus, the tokenization help us to knowing the total frequency of words in corpus.

2.2.2. Stop word removing

Stop word removing is one of the pre-processing stage of natural language processing. It is the method of removing the common stop words in English like 'is', 'was', 'where', 'the', 'a', 'for', 'of', 'in' exe. The stop words in corpus make little difficult for coups processing and feature extraction. To avoid this issues we are choose stop word remover.

In our experiment, the python stop word scrip and NLTK toolkit are used for stop word removing processing. The stop word remover help to remove the extra common words from the corpus and help to reduce the size of the corpus and it will help us easy to identify the key words in the corpus and frequency distribution of concept words in overall concepts.

2.2.3. Lemmatization

Lemmatization is the process of obtaining the "lemma" of a root word with involves reducing the word forms to its root form after understanding the parts of speech and the context of the word in the sentence [1]. In our methodology the lemmatization is one of the NLP procedure for getting the root words from the huge corpus. It will help us to label the most frequent word in that corpus.

Here the lemmatization works the following way. First to establish a link between the corpuses an English dictionary. The English dictionary contains all form of words and also had the all inflected form of verbs. Based on the dictionary the lemmatization process are takes place. This process is to replace inflected and past form of the word with corresponding bass or root words. After the lemmatization process we got the root words and these root words are used to processing the labelling process in the support vector machine algorithm.

2.3. Training

2.3.1. Feature Extraction

The text feature extraction of this work is performed by using Term Frequency – Inverse Document Frequency approach and similarity matching of words. The general problem of text feature extraction can be done by tf-idf method, but there are situations in which the term frequency criteria fail to do so. For example, we may have a review document which doesn't find the frequency of a particular term and thus couldn't map to a feature explicitly. In such cases, the

similarity of words and their synonyms are to be considered and grouping of such words is done to extract the features. The following section describes these methods in detail.

Term Frequency – Inverse Document Frequency (tf-idf) is a popular feature extraction method which reflects the relevance of a word in a particular document among the corpus. It is a numeric statistical approach which is often considered as a weighing factor in Information Retrieval and Text Mining and its value is directly proportional to the number of times a word appears in a particular document. Denote a term by 't', a document by 'd' and a corpus by 'D', the Term Frequency TF (t, d) is defined as the number of times the term 't' appears in document 'd' while Document Frequency DF(t, D) is defined as the number of documents that contains the term 't'. However, some frequent terms may not provide any relevance for the task of feature extraction and the weight of such terms should be diminished. For this, the 'Inverse Document Frequency' approach is used to distinguish relevant and non-relevant keywords which results in minimization of weight of frequently occurring non-relevant terms and maximisation of weight for terms that occur rarely. The idf gives the measure of specificity of a term which can be expressed as the inverse function of the number of documents in which the term occurs.

The tf-idf based feature extraction is performed by modelling the documents in vector space. The first step in modelling is the creation of an index vocabulary (dictionary) of terms present in the training documents. Now the term frequency gives the measure of how many times the words in the dictionary are present in the testing documents. Mathematically, tf and idf are calculated as follows:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d) : w \in d\}}$$

Where f(t, d) denotes the raw frequency of the term, 't' and f(w, d) represents the raw frequency of any term in the document.

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|}$$

where N denotes the total number of documents in the corpus and the denominator denotes the occurrence of term t in document d.

Similarity matching and grouping of words is performed inorder to extract features from review documents which does not explicitly specify the features. For this, we formulate a perceptual grouping of similar words that falls under a single class. Inorder to map a group of such words to a particular feature, it is essential to find out the hyponyms and hypernyms for each relevant word in the document. This work uses the WordNet Interface of NLTK Python for generating the hyponyms and hypernyms of words in each document. An illustration of the grouping of features is given below:

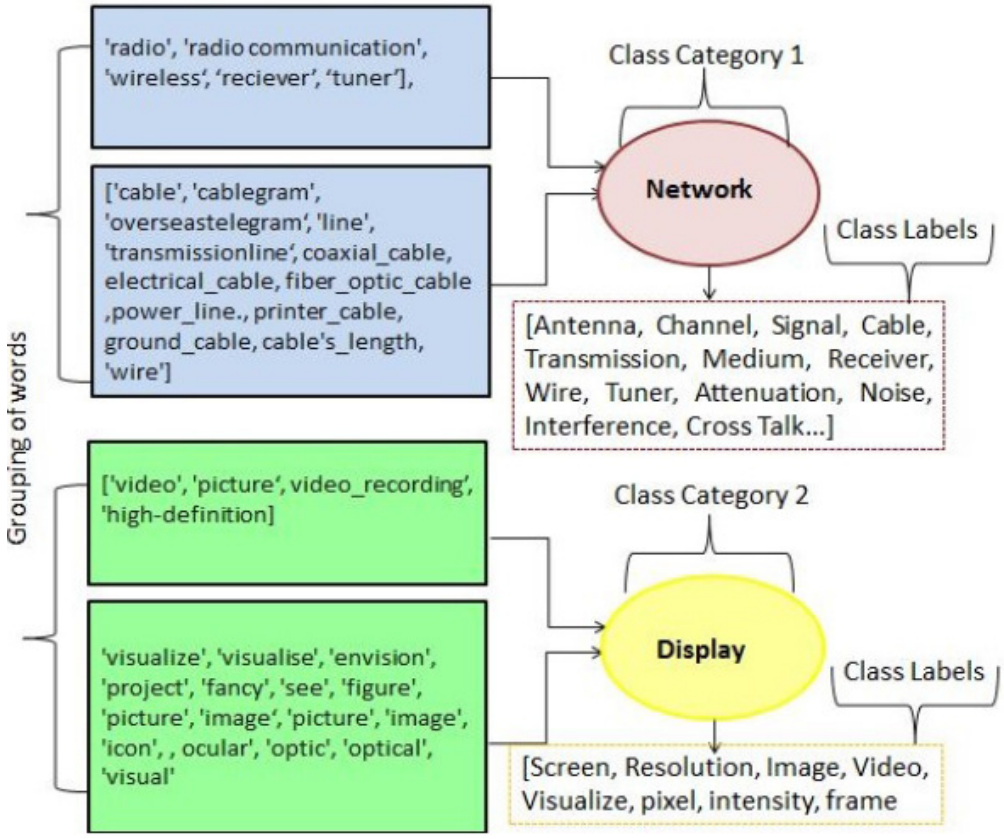

Figure 2. Grouping of Features

2.3.2. Learning and Classification

The paper presents the classification of words in product reviews to different class labels based on the various features of the product. Since the corpus is associated with text reviews, a grouping approach of feature extraction is done in this work, which results in the formulation of multiple classes and multiple class labels. Hence the classification problem is represented as a multi-class multi-label problem and this work proposes a new approach called 'Structured Support Vector Machines' for learning and classification. The problem addresses the issues of complex outputs including multiple dependent output variables and structured output spaces. The proposed method is to perform Multi label classification using Structured SVM. The method approaches the problem by generalizing large margin methods to the broader problem of learning structured responses. This approach specifies discriminant functions that exploit the structure and dependencies within structured output spaces. The maximum margin algorithm proposed in Structured SVM has the advantages in terms of accuracy and tenability to specific loss functions.

2.3.3. Structured Support Vector Machines

Structured SVM is a Support Vector Machine (SVM) learning algorithm for predicting multivariate or structured outputs. It performs supervised learning by approximating a mapping h: X→Y using labelled training samples (x1,y1),….(xn,yn). Unlike the regular SVMs which consider only univariate predictions like in classification and regression, SSVM can predict complex objects like trees, sequences or sets. Examples of problems with complex outputs are

natural language parsing, sequence alignment in protein homology detection and Markov models for Parts Of Speech (POS) tagging. The algorithm can also be used for linear-time training of binary and multi-class SVMs under linear kernel. The algorithm uses quadratic programming and is several orders of magnitude faster than prior methods. SVM^{struct} is an instantiation of the Structured Support Vector Machine algorithm and it can be thought as an API for implementing different kinds of complex prediction algorithms. In this work, Python interface to SVM^{struct} is used for implementing the multi-label classification.

In the SSVM model, the initial learning model parameters are set and the pattern-label pairs are read with specific functions. The user defined special constraints are then initialised and then the learning model is initialised. After that, a cache of combined feature vectors is created and then the learning process begins. The learning process repeatedly iterates over all the examples. For each example, the label associated with most violated constraint for the pattern is found. Then, the feature vector describing the relationship between the pattern and the label is computed and the loss is also computed with loss function. The program determines from feature vector and loss whether the constraint is violated enough to add it to the model. The program moves on to the next example. At various times (which depend on options set) the program retrains whereupon the iteration results are displayed. In the event that no constraints were added in iteration, the algorithm either lowers its tolerance or, if minimum tolerance has been reached, ends the learning process. Once learning has finished, statistics related to learning may be printed out and the model is written to a file and the program exits.

After the learning process, a model is created and written to a file for classification. For the testing phase, the learned model is read with and the testing pattern-label example pairs are loaded with. Then, it iterates over all the testing examples, classifies each example, writes the label to a file, finding the loss of this example, and then may evaluate the prediction and accumulate statistics. Once each example is processed, the classification summary statistics are printed out with and the program exits. The learning and classification module of struct SVM model is represented as follows:

Structured output SVMs extends SVMs to handle arbitrary output spaces, particularly o**nes with non-trivial structure** (E.g. textual translations, sentences in a grammar, etc.). Learning a structured SVM requires solving an optimisation problem by choosing the highest scoring output for each input

$$\hat{\mathbf{y}}(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y}) \rangle$$

The evaluation of a structured SVM requires solving the following problem and the efficiency of using structured SVM (after learning) depends on how quickly the inference problem is solved.

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y}) \rangle$$

Then we define a loss function Δ(y, y^) measuring how well the prediction y^ matches the truth label y. Finally we define a constraint generation function which captures the structure of the problem. Generating a constraint for an input-output pair (X, y) means identifying what is the most incorrect output that the current model still deems to be compatible with the input. The margin rescaling formulation and slack rescaling formulation are represented as follows:

$$\max_{\hat{y}\in\{-1,+1\}} \Delta(y,\hat{y}) + \langle \Psi(\mathbf{x},\hat{y}),\mathbf{w}\rangle - \langle \Psi(\mathbf{x},y),\mathbf{w}\rangle$$

And

$$\max_{\hat{y}\in\{-1,+1\}} \Delta(y,\hat{y})[1 + \langle \Psi(\mathbf{x},\hat{y}),\mathbf{w}\rangle - \langle \Psi(\mathbf{x},y),\mathbf{w}\rangle].$$

Where w is the parameter vector to be learned, $\Psi(x, y) \in R^2$ is the joint feature map and F(x,y; w) is an auxiliary function.

The SVM^{struct} implements the 1-slack cutting plane algorithm which is an equivalent formulation of the Structural SVM quadratic program and is several orders of magnitude faster than prior methods.

2.3.4. Pseudo code

2.3.4.1. SVM_Python_Learn ()

1. Check out all the command line arguments.
2. Load the Python Module
3. Parse_Parameters
   Sets the attributes of sparm based on command line arguments.
4. Read_Examples
   Reads and returns x, y example pairs from a file.
5. Initialize_model
   Initializes the learning model
6. Construct cache of Ψ(x, y) vectors used in training.
7. Train model and iterate over all training examples until no constraints are added.
8. Return a feature vector describing the input pattern x and correct label y.
   - If Margin scaling, find the most violated constraint and then classify example. Return y' associated with x's most violated constraint.
   - If Slack scaling, find the most violated constraint slack and then classify example. Return y' associated with x's most violated constraint.
   - Return a feature vector describing pattern x and most violated label y'.
   - Return the loss of y' relative to the true labelling y.
   - If the new constraint significantly violates the existing Quadratic Programming, add it to the SVM QP.
   - Print_Iteration_Stats
   - Print the statistics once learning has finished.
9. Train model, and iterate over all training samples until no constraints are added.
10. Print_Learning_Stats
    Print statistics once learning has finished.
11. Write_Model
    Dump the struct model to a file.
12. Exit

2.3.4.2. SVM_Python_Classify ()

1. Check out all the command line arguments.
2. Load the Python Module

3. Parse_Parameters_Classify
   Process the custom command line arguments
4. Read_Model
   Load the struct model from a file
5. Read_Examples
   Reads and returns x, y example pairs from a file.
   - Classify_example
     - Given a pattern x, return the predicted label
   - Write_label
     - Write a predicted label to an open file.
     - Return the loss of y' relative to the true labelling y
   - Eval_Prediction
     - Accumulate statistics about a single training example.
6. Iterate over all examples
7. Print_testing Stats
8. Print statistics once classification has finished.
9. Exit

3. CONCLUSION AND PERFORMANCE EVALUATION

In the experiment phase of this work, 500 testing samples are selected for testing and performance evaluation. The confusion matrix provides an idea about the actual and predicted classifications done by the classification system. It is also created for identifying the miss classifications and missed classifications. The confusion matrix generated after the testing process is as follows:

n=500	**Learn NO**	**Learn YES**	
Actual NO	TN = 27	FP = 50	77
Actual YES	FN = 48	TP = 375	423
	75	425	

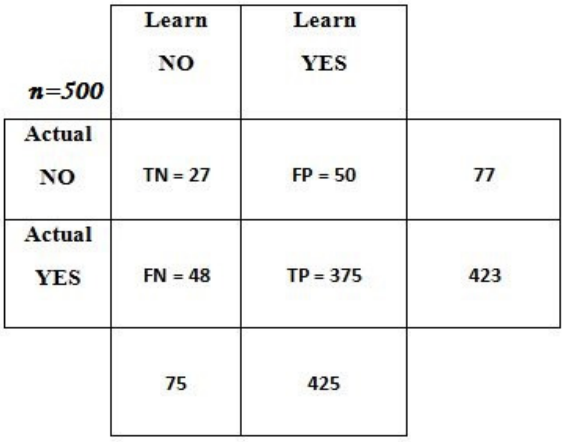

Figure 3. Confusion Matrix

The following is the list of measures that are often computed from the above confusion matrix:

Table 1. Confusion Matrix measurements

Measures	**Values**
Accuracy	80.4 %
Misclassification Rate (Error Rate)	19.6 %
True Positive Rate (Recall)	88 %
False Positive Rate	64 %
Specificity	35 %
Precision	88 %
Prevalence	84 %

n= number of population
True positive (TP)
True negative (TN)
False positive (FP)
False negative (FN)

The below table values are representing the accuracy of the proposed structured supporting vector machine learning algorithm for text mining and classification of the product reviews.

Table 2. Accuracy table

Accuracy = (TP+TN)/total	**False Positive Rate**: FP/actual no
Misclassification Rate= (FP+FN)/total [Also known as "Error Rate"]	**Specificity**: TN/actual no
True Positive Rate: TP/actual yes [Also known as **"Recall"**]	**Precision**: TP/predicted yes **Prevalence**: actual yes/total

4. CONCLUSION

We formulated a Structured Support Vector Machine learning paradigm for the classification of texts from various product reviews. The problem is represented as a multi-class multi-label problem and addressed by Struct SVM Python Implementation. The system results in an overall accuracy of 80.4% with enough flexibility and ability to handle specific loss functions. The remarkable characteristic feature of this algorithm is its capability for training complex models. The final outcome of this work is the classified words in the review text into multiple class labels according to the extracted features. The accuracy and performance of the system is measured and found to be an optimized method in the case of a Multi-label text classification scenario.

REFERENCE

[1] Bullinaria, John A., and Joseph P. Levy. (2012) "Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD." Behavior research methods 44.3: 890-907.

[2] Chrupała, Grzegorz. (2006) "Simple data-driven contextsensitive lemmatization."Procesamiento del Lenguaje Natural 37 :121-127.

[3] Fagan, Joel L., et al.(1991) "Method for language-independent text tokenization using a character categorization." U.S. Patent No. 4,991,094. 5 Feb. 1991.

[4] Joachims, Thorsten. "Transductive inference for text classification using support vector machines." ICML. Vol. 99. 1999.

[5] Korenius, Tuomo, et al.(2004) "Stemming and lemmatization in the clustering of finnish text documents." Proceedings of the thirteenth ACM international conference on Information and knowledge management. ACM.

[6] Plisson, Joël, Nada Lavrac, and Dr Mladenić. (2004) "A rule based approach to word lemmatization.".

[7] Torres-Carrasquillo, Pedro A., Douglas A. Reynolds, and J. R. Deller Jr(2002). "Language identification using Gaussian mixture model tokenization."Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on. Vol. 1. IEEE, 2002.

[8] De Heer T (1982) “The application of the concept of homeosemy to natural language information retrieval. Information Processing & Management,” 18:229–236.

[9] Harman D (1992)” Relevance feedback revisited.” In: Proceedings of the 15th International ACMSIGIR Conference on Research and Development in Information Retrieval (SIGIR-92), pp. 1–10.

[10] Harman D et al. (1995) Performance of text retrieval systems. Science, 268:1417–1418.

AUTHORS

1. Jincy B. Chrystal

Jincy B. Chrystal
Student, MPhil-Computer Science
Indian Institute of Information Technology and Management- Kerala (IIITM-K),
Trivandrum, Kerala,
jincyb.mphilcs2@iiitmk.ac.in

Short Biography

My interests lie with Machine learning. Throughout my professional career I've worked on technologies how to train and learn the machines. Now I'm focusing on the supervised machine learning algorithms. And doing the research on how these algorithms are used to train the machine perfectly.

2. Stephy Joseph

Student, MPhil-Computer Science
Indian Institute of Information Technology and Management- Kerala (IIITM-K),
Trivandrum, Kerala,
stephyjoseph.mphilcs2@iiitmk.ac.in

Short Biography

My interests lie with multimedia technologies. Throughout my professional career I've worked on technologies that identify, extract features from multimedia resources. Now I've focused on the voice features extracted from video files.