Text Classification and Topic Modeling for Online Discussion Forums

Chapter · January 2020

DOI: 10.4018/978-1-5225-9373-7.ch006

3 authors, including:

Xin Zhao
University of Alabama
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Jeff Gray
University of Alabama
370 PUBLICATIONS **3,971** CITATIONS
SEE PROFILE

Chapter 6
Text Classification and Topic Modeling for Online Discussion Forums:
An Empirical Study From the Systems Modeling Community

Xin Zhao
University of Alabama, USA

Zhe Jiang
University of Alabama, USA

Jeff Gray
University of Alabama, USA

ABSTRACT

Online discussion forums play an important role in building and sharing domain knowledge. An extensive amount of information can be found in online forums, covering every aspect of life and professional discourse. This chapter introduces the application of supervised and unsupervised machine learning techniques to analyze forum questions. This chapter starts with supervised machine learning techniques to classify forum posts into pre-defined topic categories. As a supporting technique, web scraping is also discussed to gather data from an online forum. After this, this chapter introduces unsupervised learning techniques to identify latent topics in documents. The combination of supervised and unsupervised machine learning approaches offers us deeper insights of the data obtained from online forums. This chapter demonstrates these techniques through a case study on a very large online discussion forum called LabVIEW from the systems modeling community. In the end, the authors list future trends in applying machine learning to understand the expertise captured in online expert communities.

DOI: 10.4018/978-1-5225-9373-7.ch006

1. INTRODUCTION AND BACKGROUND

Systems modeling is the process of developing abstract models that represent multiple perspectives (e.g., structural, behavioral) of a system. Such models also provide a popular way to explore, update, and communicate system aspects to stakeholders, while significantly reducing or eliminating dependence on traditional text documents. There are several popular systems modeling tools, such as Simulink (MathWorks, 2019) and LabVIEW (National Instruments, 2019).

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments. LabVIEW offers a graphical programming approach that helps users visualize every aspect of the system, including hardware configuration, measurement data, and debugging. The visualization makes it simple to integrate measurement hardware from any vendor, represent complex logic on the diagram, develop data analysis algorithms, and design custom engineering user interfaces. LabVIEW is widely used in both academia (Ertugrul, 2000, 2002) and industry, such as Subaru Motor (Morita, 2018) and Bell Helicopter (Blake, 2015). There are more than 35,000 LabVIEW customers worldwide (Falcon, 2017).

Text summarization refers to the technique of extracting information from a large corpus of data and represents a common application area of machine learning and natural language processing. With the increasing production and consumption of date in all aspects of our lives, text summarization helps to reduce the time to digest and analyze information by extracting the most valuable and pertinent information from a very large dataset.

There are two main types of text summarization: extractive text summarization and abstractive text summarization. Extractive text summarization is a technique that pulls keywords or key phrases from a source document to infer the key points from original documents. Abstractive text summarization refers to the creation of a new document for summarizing the original document. The result of abstractive text summarization may include new words or phrases not in the original documents.

To understand the current best practices and tool-feature needs of the LabVIEW community, we collected user posts from the LabVIEW online discussion forum. An online discussion forum is a website where various individuals from different backgrounds can discuss common topics of interest in the form of posted messages. Online discussion forums are useful resources for sharing domain knowledge. The discussion forums can be used for many purposes, such as sharing challenges and ideas, promoting the development of community, and giving/receiving support from peers and experts. Several researchers have identified benefits of online discussion forums from different aspects, such as education (Jorczak, 2014), individual and society development (Pendry & Salvatore, 2015) and socialization (Akcaoglu & Lee,

2016). The LabVIEW discussion forum has very rich resources for text summarization because most of the user-generated content in the forums is text-based. We applied text classification based on supervised machine learning techniques and topic modeling based on unsupervised machine learning techniques to the large collection of LabVIEW forum posts. After downloading all the post questions through web scraping, we first used supervised machine learning to classify all the questions into four categories (i.e., "program", "hardware", "tools and support" and "others"). We compared three popular methods, including Multinomial Naive Bayes, Support Vector Machine and Random forest. After this, we applied unsupervised machine learning techniques to delve into the largest category ("program") to find subtopics. In this chapter, we examine three unsupervised machine learning approaches: K-means clustering, hierarchical clustering and Latent Dirichlet Allocation (LDA). We use the LabVIEW discussion forum as our case study with empirical results.

The contributions of this chapter are two-fold. First, we demonstrate how text summarization techniques can be used to extract online discussion forum key information. Second, we describe future trends and research directions based on the analyses of text summarization results, which give direction toward future areas of investigation for the text summarization research community.

This chapter is structured as follows: In Section 2, we first introduce the process and technical details of supervised machine learning techniques in the context of text classification. After this, we use the LabVIEW discussion forum as a case study to introduce our empirical experiment that demonstrates how to apply supervised machine learning techniques to classify posts with predefined categories for the LabVIEW posts. In Section 3, unsupervised machine learning algorithms are first presented, followed by the empirical application of unsupervised machine learning technique in gaining subtopics of LabVIEW posts in a specific category. Future trends in this area represented by this paper are summarized in Section 4 and concluding remarks are offered in Section 5.

2. TEXT CLASSIFICATION

Text classification is the process of assigning a set of predefined categories to text. Text classification is the heart of many applications, such as spam filtering (Kumar, et al., 2016), sentiment analysis (Cambria, 2016) and readability assessment (Miltsakaki & Troutt, 2008).

Text classification has been a popular research topic over the past decade. Nenkova and Mckeown (2012) surveyed several text summarization techniques from the perspective of different phases involved in the summarization. A newer and complete summarization work is described by Altınel and Ganiz (2018). In

their work, they divided text summarization into two major categories: traditional text summarization and semantic text summarization.

- **Traditional text classification:** Traditional text summarization is based on the concept of Bag-of-Words (BOG, Salton & Yang, 1973). This classification technique separates document into individual words and it only considers their corresponding frequencies in a document. The information of word locations are discarded during the processing of classification.
- **Semantic text classification:** Semantic text classification tries to overcome the shortcoming in traditional text classification techniques by including semantic relations between words. Altınel and Ganiz summarized five categories for semantic text summarization:
  - domain knowledge based approaches: a technique that classifies documents based on common knowledge bases (such as Wikipedia, Suganya and Gomathi, 2013);
  - corpus-based approaches: similar to domain knowledge based approaches, but based on a training corpus instead of knowledge (Deerwester et al., 1990);
  - deep learning based approaches: text classification based on deep learning methodologies (we will give a more detailed explanation on semantic text classification in Section 4);
  - word/character sequence enhanced approaches: In this method, string-matching techniques are applied to find string sequences in a document (Razon & Barnden, 2015); and
  - linguistic enriched approaches: lexical and syntactic rules are applied to extract key information compared with other approaches (Abbasi et al., 2011).

In this chapter, we focus on text classification analysis based on an existing corpus. We applied machine learning techniques to analyze a corpus to predict new documents. In general, text classification based on machine learning techniques includes four steps:

Step 1: Data Pre-processing
Step 2: Feature Extraction
Step 3: Model Training
Step 4: Model Evaluation

With the help of the model we obtain from Step 3, predictions on new data can be achieved. In this section, we first describe these four steps in detail. At the end

of this section, the case study from the modeling community is introduced as an example application of how these techniques are applied in a real scenario.

2.1 Data Pre-Processing

The first step is to refine the text data in the dataset to remove noise and errors. In the text that is mined from a discussion forum, there are several characters (such as '[' and '/') and words (such as 'I' and 'we') that are meaningless and not important for extracting the key information. To remove these characters, a common approach is to convert these characters into their corresponding Unicode characters. These Unicode characters will be removed when they are recognized. After removing meaningless characters, tokenization, a technique for breaking sentences into pieces (these pieces are called tokens) is applied to the text data. Tokens can be individual words or phrases. Usually, tokens are individual words. In this step, two operations are needed: stemming (e.g., converting words into common bases, such as 'windows' to 'window') and filtering (e.g., dropping stop words like 'a', 'an' and 'the').

2.2 Feature Extraction

After data cleaning, each document consists of a sequence of tokens (or symbols). In order to apply machine learning algorithms to these documents, we need to represent the sequence of tokens in each document as a numeric feature vector, which is recognizable for machines. One simple way to extract features is to use a word-document frequency matrix (also called bag-of-words) (Zhang, Jin, & Zhou, 2010). In this approach, the frequency of occurrence of each word is used as a feature for training a classifier. A more prevalent approach is TF-IDF (Sparks Jones, 1972), which is a measure that adopts two statistical methods - Term Frequency (TF) and Inverse Document Frequency (IDF). TF is the total number of times a given term appears in the text, and IDF measures the weight of a given word in the entire text. It is a measure of how much information the word provides. TF-IDF is the product of term frequency and inverse document frequency. There are several schemes for calculating TF and IDF (Manning, Raghavan, & Schütze, 2008). Recently, Word2Vec (Lilleberg, Zhu, & Zhang, 2015) based on deep learning has become increasingly popular. In this section, we discuss the bag-of-words model and TF-IDF model.

2.2.1 Bag-of-Words Model

A bag-of-words model (BoW) is a simple way to extract features from text for use. It is a representation of text that describes the occurrence of words in a document. The name "bag" comes from the fact that in the document, the order or the structure

of words are irrelevant. BoW model is primarily concerned with whether known words occur in the document. If one word exists, the frequency of occurrence of this word is used as a feature for training a classifier. If one word does not exist, the feature of this word is simply marked as 0. Consider a simple example showing how to build a BoW model from existing documents: imagine we have text documents containing three posts (abstracted here for example purposes, CompactRIO is a controller made by National Instruments for industrial control systems):

- **Post 1:** "LabVIEW is easy to use. It is popular, too."
- **Post 2:** "CompactRIO is a hardware supported by LabVIEW."
- **Post 3:** "How can you debug a program?"

The first step is to build a vocabulary vector for the document. The length of the vocabulary is equal to the number of unique words in the documents. In this example:

```
Vocabulary = ['LabVIEW', 'is', 'easy', 'to', 'use', 'it',
'popular', 'too', 'CompactRIO', 'a', 'hardware',
           'supported', 'by', 'how', 'can', 'you', 'debug',
'program']
```

Each word in the document is compared with words in the vocabulary. If a word appears in the vocabulary, the frequency counter for that word is incremented. For example, in Post 2, the third word is 'a,' we look up the dictionary and find out word 'a' exists and its position is 10. So we put 1 in the Post 2 vector at position 10 (marked in red below). Applying the vocabulary to the three posts, we transfer the posts to feature vectors as follows:

- **Post 1:** [1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- **Post 2:** [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
- **Post 3:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1]

2.2.2 TF-IDF Model

TF-IDF model is a very popular Natural Language Processing (NLP) technique to transform text into vectors. It is composed of two components: TF and IDF. TF is defined as "The weight of a term that occurs in a document is simply proportional to the term frequency" (Luhn, 1957). There are several schemes to calculate TF weight, such as term frequency and log normalization. The most commonly used approach is term frequency, which is defined as:

$$TF\left(t\right) = \frac{Number\ of\ times\ term\ t\ appears\ in\ the\ document}{Total\ number\ of\ terms\ in\ the\ document}.$$

However, some common words, such as 'is' and 'the,' may appear in a document many times but have little importance. IDF measures how important a term is by weighing down the frequent terms while scaling up the rare ones. It is defined as:

$$IDF\left(t\right) = \log \frac{Total\ number\ of\ documents}{Number\ of\ times\ term\ t\ appears\ in\ all\ documents}.$$

TF-IDF is then calculated as:

$$TF - IDF\left(t\right) = TF\left(t\right) \times IDF\left(t\right).$$

For example, suppose we have two documents:

- **Document 1:** This post is related to category *program*.
- **Document 2:** This question is related to category *hardware*.

Based on the definition above, we have the TF-IDF results for each word shown in Table 1.

2.3 Model Training

The third step of the technique is to apply machine learning algorithms to train the machine to build the classification model. There are various techniques that can be used for text classification. The training process starts with a manually labeled training set, whereby each document is manually labeled into predefined categories. The machine then learns a classification model that can automatically predict the class label of a new text document. A test set of documents are often used to verify the accuracy of the classifier. In this section, we introduce three widely used algorithms for text classification: Multinomial Naive Bayes, Support Vector Machines and Random Forest. The reason why we chose these three algorithms is because they are easy to implement and have relatively good classification results.

Table 1. TF-IDF calculation example

Word Vocabulary	*TF*		*IDF*	*TF* ×*TDF*	
	Document 1	**Document 2**		**Document 1**	**Document 2**
the	$\frac{1}{7}$	$\frac{1}{7}$	$\log\frac{2}{2}=0$	0	0
post	$\frac{1}{7}$	0	$\log\frac{2}{1}=0.3$	0.043	0
question	0	$\frac{1}{7}$	$\log\frac{2}{1}=0.3$	0	0.043
is	$\frac{1}{7}$	$\frac{1}{7}$	$\log\frac{2}{2}=0$	0	0
related	$\frac{1}{7}$	$\frac{1}{7}$	$\log\frac{2}{2}=0$	0	0
to	$\frac{1}{7}$	$\frac{1}{7}$	$\log\frac{2}{2}=0$	0	0
category	$\frac{1}{7}$	$\frac{1}{7}$	$\log\frac{2}{2}=0$	0	0
program	$\frac{1}{7}$	0	$\log\frac{2}{1}=0.3$	0.043	0
hardware	0	$\frac{1}{7}$	$\log\frac{2}{1}=0.3$	0	0.043

2.3.1 Multinomial Naive Bayes

Multinomial Naive Bayes classifier is a probabilistic algorithm based on Bayes' Theorem (Bayes, 1763). It is a widely adopted algorithm in NLP, such as text classification, spam email detection, and sentiment analysis. Because of the nature of Bayes' Theorem, one precondition is assumed when applying Naive Bayes classifiers: All features are conditionally independent given the underlying class category. Bayes' Theorem states that:

$$P(C \mid X) = \frac{P(X \mid C) \times P\left(C\right)}{P\left(X\right)}$$

where C represents the category of the text and X is the feature of the text. P(C|X) is called the *posterior probability*, which is the statistical probability of the category for a given document based on its numeric feature. P(X|C) is called *likelihood*. It means that for a given category, the probability of a feature X occurs. P(C) and P(X) are called *prior probability*. P(C) is the probability of the occurrence of category C in the documents and P(X) is the probability of the occurrence of feature X in the document. Based on this theorem, given an instance with feature X_1, X_2, …, X_n, the possibility of this instance belonging to category C_k is defined as:

$$P\left(C_k \mid X_1, X_2, \ldots X_n\right) = P\left(C_k\right) \times P\left(X_1 \mid C_k\right) \times P\left(X_2 \mid C_k\right) \times \ldots \times P(X_n \mid C_k) .$$

This is known as the Bayes Classifier. We will illustrate Bayes Classifier with an example. Suppose we have a document set (treated as training set) containing three posts belonging to two categories as seen in Table 2.

A key question is to decide which category a new post (e.g., "*Test a program in LabVIEW*") belongs to. There are 15 distinct words in the example training set: ["LabVIEW", "program", "is", "hard", "to", "debug", "share", "your", "example", "with", "others", "in", "the", "NI", "community"]. Adding the words from the new post, we have 22 distinct words: ["LabVIEW", "program", "is", "hard", "to", "debug", "CompactRIO", "cannot", "connect", "to", "share", "your", "example", "with", "others", "in", "the", "NI", "community", "test", "a", "in"]. Based on Bayes' Theorem, we have:

$$P(program \mid Test\ a\ program\ in\ LabVIEW) = \frac{P(Test\ a\ program\ in\ LabVIEW \mid program) * P\left(program\right)}{P\left(Test\ a\ program\ in\ LabVIEW\right)} \quad (1)$$

Table 2. Example posts with their categories

Post	Category
LabVIEW program is hard to debug.	*program*
CompactRIO cannot connect to LabVIEW.	*hardware*
Share your Example Program with others in the NI community.	*program*

$$P(hardware \mid Test\ a\ program\ in\ LabVIEW) = \frac{P(Test\ a\ program\ in\ LabVIEW \mid hardware) * P\left(hardware\right)}{P\left(Test\ a\ program\ in\ LabVIEW\right)} \quad (2)$$

Because we assume all the events are independent, for Equation (1), we have

$$P\left(Test\ a\ program\ in\ LabVIEW \mid program\right) = P(test \mid program) * P(a \mid program) * P(program \mid program) * P(in \mid program) * P(LabVIEW \mid program)$$

Equation 1 can be reduced to calculate the occurrence of words "test," "a," "program," "in," "LabVIEW" in the existing training set *program*. For a word that does not appear in training set *program* (such as 'test'), we apply *Laplace Smoothing* (a technique for smoothing categorical data): we first add 1 to every word count and then add the number of possible words to the divisor (so the result will never be greater than 1). Thus,

$$P(test \mid program) = \frac{0+1}{15+22}$$

$$P(a \mid program) = \frac{0+1}{15+22}$$

$$P(program \mid program) = \frac{2+1}{15+22}$$

$$P(in \mid program) = \frac{0+1}{15+22}$$

$$P(LabVIEW \mid program) = \frac{1+1}{15+22}.$$

Based on the training set, we have $P\left(program\right) = \frac{2}{3}$. (2 posts are categorized as program in total of 3 posts). Applying the same approach to Equation 2, we have

$$P\Big(program \mid Test\ a\ program\ in\ LabVIEW\Big) = \frac{1}{37} \times \frac{1}{37} \times \frac{3}{37} \times \frac{2}{37} \times \frac{2}{3} \approx 5.768 * 10^{-8}$$

$$P\Big(hardware \mid Test\ a\ program\ in\ LabVIEW\Big) = \frac{1}{27} \times \frac{1}{27} \times \frac{1}{27} \times \frac{1}{27} \times \frac{1}{3} \approx 4.646 * 10^{-8}$$

Therefore, the new post "Test a program in LabVIEW" is classified as a *program* related post. To simplify the calculation process, data preprocessing (such as removing stop words) is always performed in advance.

In Multinomial Naïve Bayes, the algorithm configuration is related to the setting of the *smoothing parameter* (which is usually called α). When α is set as 1 (also known as "*Laplase Smoothing*"), it means we add 1 to each word counting. This is shown in the example introduced above. When α is set as 0 (also known as "*Lidstone Smoothing*"), we add 0 to each word counting. In the following discussion throughout this chapter, we choose $\alpha = 1$.

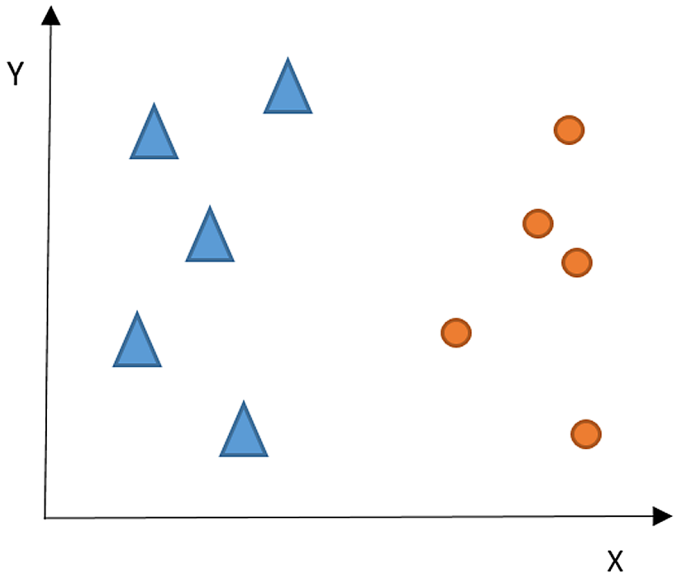
2.3.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm that is widely used in classification problems. Like other machine learning algorithms, some data pre-processing is also executed in SVM, such as data cleaning and feature extraction. However, unlike calculating the frequency of each feature in the document in Multinomial Bayes, SVM plots each feature vector as an n-dimensional point (where n is the number of total features in a post) with the value of each feature corresponding to the value of a particular coordinate. Then, we perform classification by finding the ***hyperplane*** that separates points from the different classes ("support vectors" are the points near the boundary hyperplane). Thus, the core problem of learning a SVM classifier is to find the best separation hyperplane.

We will illustrate SVM with a simple example. Suppose we have two categories (blue triangle and orange circle) and the data has two features (x and y). Figure 1 shows this example.

There are several possible hyperplanes that can be used to separate points from these two categories (such as the dotted black line and bold black line in Figure 2). Research has shown that the hyperplane with largest margin (shown in the bold black line) is the best (Joachims, 2006). For a dataset that is inseparable in 2 dimensional space, SVM transforms the data to a higher dimension to do separation. Figure 3 shows such a situation.

Figure 1. Two categories containing different data points



For a dataset with n features (in other words, a dataset in an n-dimension), SVM finds an n-1 dimension hyperplane to separate it. When data is not linearly separable, SVM transforms the dataset to a higher dimensional dataset where it becomes separable. However, it is almost impossible to try out all the transformations from lower dimension to higher dimension. To solve this, SVM computes the pair-wise dot products. For a given pair of vectors and a transformation into a higher-dimensional space, there exists a function (which is known as a "*kernel function*") that can compute the dot product in the higher-dimensional space without explicit transformation. This is called "*kernel tricks*," which saves computation and makes SVM feasible in high dimension datasets.

The algorithm configuration for SVM is much more complicated than Multinomial Naïve Bayes. There are several parameters used in SVM, such as penalty parameter C, kernel type, whether to use shrinking heuristic, and the size of kernel. In this section, we discuss the two most important parameters for SVM: C and Kernel.

Parameter C is called the penalty parameter (or complexity parameter). It shows the tolerance of error. The larger the value of C, the less tolerance for the algorithm. Overfitting is more likely to happen with higher values of C. On the contrary, a smaller C value offers more tolerance of error and under-fitting is more likely to happen. The kernel parameter refers to which kernel is to be used by the SVM engine. Common kernel functions include linear kernel, poly kernel and RFB kernel. In the experiment we conducted, we chose C = 1 and linear kernel.

Figure 2. Possible separations of two categories

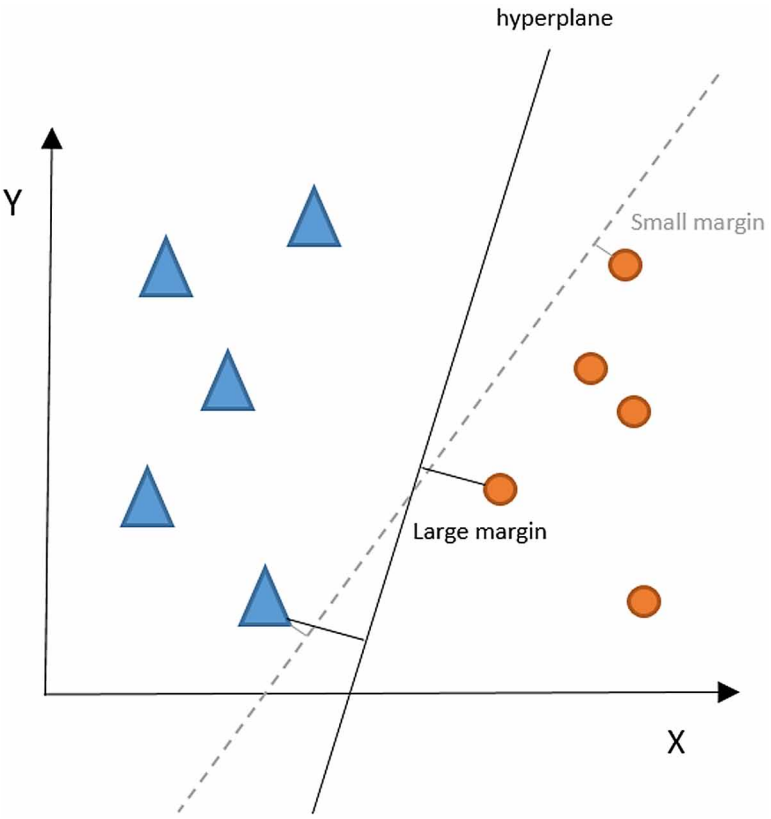
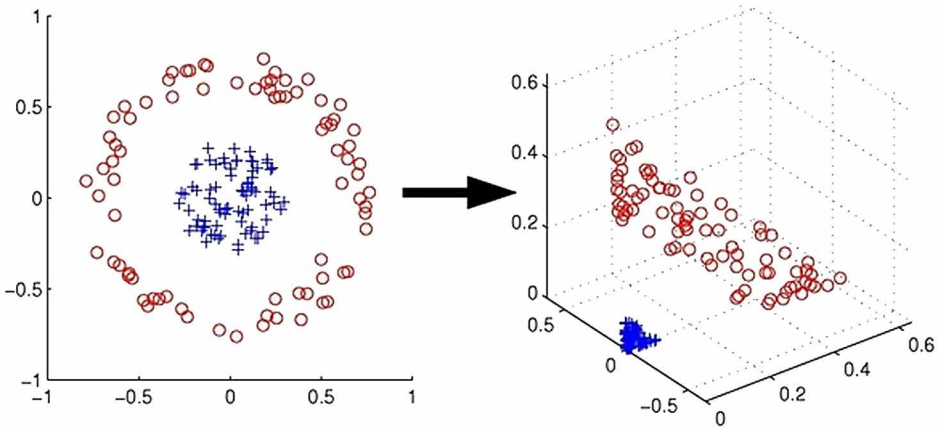


Figure 3. Data inseparable in 2D space is separable in 3D space. Example is taken from
(Deshpande, 2018)



2.3.3 Random Forest

Random Forest is a supervised machine learning technique based on decision trees. A decision tree is a tree-like structure where internal nodes represent a test condition on a feature attribute. The answer for the test is binary (only "yes" or "no") or multi-way. Each leaf is a final predicted class. When using a decision tree to classify text, we start from the root, and select a sub-branch based on the test result on the root node. This decision process is repeated until a leaf node is reached, where a class is predicted. Figure 4 is a decision tree showing a simple post classification strategy (the number below each category indicates the prediction probability).

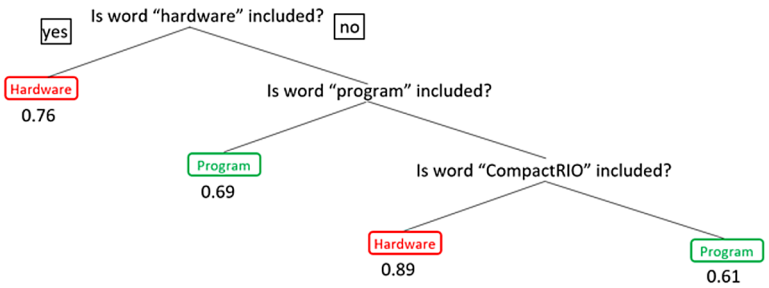
A random forest classifier is an algorithm based on the ensemble of decision trees. It involves learning a collection of decision trees to make a final prediction based on voting. The random forest algorithm first creates several decision trees randomly to evaluate the performance of each decision tree constructed and then averages the results to improve the predictive accuracy.

The algorithm configuration for random forest is closely related to the setting of random trees. In general, three parameters are crucial to the performance of random forest. First, the depth of each tree. A deep tree may increase the performance of the algorithm, but it increases the calculation time. Second, number of max features, which decides how many attributes a tree uses every time a subtree is created. The higher value this parameter holds, the more attributes a tree has. Usually, increasing this value could improve the performance of a random forest. Similarly, it also increases the running time. In our experiment, we chose unlimited tree depth and

$$\log \frac{Number\ of\ feature}{2}$$

as the max number assigned to each tree in the forest.

Figure 4. An example for decision tree based classification strategy



2.4 Evaluation

2.4.1 Evaluation Approaches

The most dominant way to evaluate the model built from a ML algorithm is to apply *K-fold cross validation* (Kohavi, 1995), which divides the training data into K parts equally. Suppose K = 10 and the 10 parts are labelled as *A, B, C, D, E, F, G, H, I, J*. In the first iteration, we use part *A* through part *I* as training datasets (9 parts in total) and use part *J* as a testing dataset. In the second iteration, we choose part *B* through part *J* as training datasets and use part *A* as a testing dataset. Proceeding similarly for the other permutations, we will have 10 iterations in total. We calculate the average value for these 10 iterations as the final evaluation result. The reason why we apply K-fold cross validation is because cross-validation is a good evaluation technique when there are no sufficiently large training and testing sets (Duda, Hart, & Stork, 2012).

2.4.2 Evaluation Metrics

There are several evaluation metrics in the context of text classification problems. The most basic and simple one is *accuracy*, which can be defined as:

$$accuracy = \frac{Number\ of\ correct\ predictions}{Total\ predictions\ made}$$

Another metric is *Receiver Operating Characteristic curve (ROC curve)*, which is a plot that is used in a binary classification. The ROC curve represents a model's ability to discriminate between positive and negative classes, where an area of 1.0 means a perfect prediction and 0.5 represents a model that is as good as a random guess.

Confusion Matrix (also known as *error matrix*) is an evaluation metric that presents the accuracy of a model with two or more categories, taking the following form in Table 3:

True positive (TP) means that an instance is predicted as positive and the prediction is true (i.e., in the example from Section 2.2, a post related to *program* is predicted as *program)*. Similarly, true negative (TN) means that an instance is predicted negative and the prediction is true (a post not related to *program* is not predicted as *program*). False positive (FP, also known as Type 1 Error) indicates that an instance is predicted positive, but it is negative (a post is related to *hardware*, but it is predicted as *program*). False negative (FN, also known as Type 2 Error)

Table 3. Different combinations of predicted and actual values

		Actual Values	
		Positive	**Negative**
Predicted Values	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

indicates that an instance is predicted negative, but it is positive (a post related to *program*, but it is predicted as *hardware*).

Precision, *recall* and *F-measure* are widely used metrics in performance evaluation. Precision (also called "positive predictive value") is the "fraction of relevant instances among the retrieved instances" (Alvarez, 2002). It usually refers to the accuracy of the result and is defined as:

$$Precision = \frac{TP}{TP + FP}$$

Recall (also known as "sensitivity") is the "fraction of relevant instances that have been retrieved over the total amount of relevant instances" (Alvarez, 2002). It usually refers to the completeness of the prediction result and it is defined as:

$$Recall = \frac{TP}{TP + FN}$$

Combining the measurements of Precision and Recall, we can obtain a metric that evaluates the correctness and completeness at the same time: F-measure. It is defined as:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

2.5 Empirical Classification

In this section, we present our post classification from empirical mining on the LabVIEW discussion forum. We first introduce web scraping – a technique used in our experiment to obtain the text data from the forum. Following this, we discuss how supervised machine learning techniques are applied to find post distribution

among predefined categories. A result is given based on our approach in the end of this section.

2.5.1 Data Source and Pre-Processing

Background: Web Scraping

Web scraping is the process of automatically mining data to collect information from a rendered web page. There are several techniques for web scraping, ranging from manual examination to fully automated approaches. The most naive approach for web scraping is human copy-and-paste. However, this approach requires much human effort when the dataset is large. The most popular approach for web scraping is based on parsing text represented as HTML (Hypertext Markup Language, a standard language for creating web pages and web applications). In this section, we focus our discussion on HTML parsing.

In general, web scraping based on HTML parsing for an online discussion forum consists of three main parts:

- ***Data Acquisition.*** This step builds a connection between a client and a website. It uses an HTML request library or a headless browser (a web browser without a graphical user interface). A popular method for data acquisition is to use the Node.js request-promise module (Simpson, 2015). There are several tools supporting this method, such as Puppeteer (De Lara, Wallach, & Zwaenepoel, 2001). By sending a request to the URL, the method receives a promise that specifies the format of the returned object. A client will receive the raw HTML format from the web page if the connection is built successfully.
- ***Source Inspection.*** This step finds the exact pattern/tag that a client is looking for in the code. Most current web browsers support the function of inspecting the source code of a web page. An HTML ***selector*** is often used to locate the target element and extract HTML and text within the selected element.
- ***Data Parsing.*** The last step is to parse the HTML by calling handler methods when start tags, end tags, text, comments, and other markup elements are encountered. Clients can customize the output to export and save the results to different file formats, such as a simple text file, Excel file, or CSV file.

Dataset

Our dataset was collected from the official LabVIEW discussion forum. With our web scraping approach mentioned above, we implemented a JavaScript web crawler to automatically retrieve all the user posts from the LabVIEW discussion forum.

We collected 184,203 posts from May 18th, 1999 to February 15th, 2019. Among these posts, we removed: 1) Posts that are not written in English: 4,116 posts in total; 2). Meaningless posts (such as empty posts, posts with characters only, etc.): 2,592 posts in total. After filtering, 177,495 posts were remained as our dataset.

We first preprocessed text data in the posts through stemming and lemmatization (find the root of a word, such as converting 'playing' to 'play', 'books' to 'book'), and removed stop words (common words such as 'the,' 'an,' 'a'). Then, we represented each post with a numeric vector based on TF-IDF.

2.5.2 Machine Training

We manually labelled 1,800 posts randomly chosen from the dataset. Based on our exploration of the dataset, we labelled each post belonging to one of the four categories: program, hardware, tools and support, and others. The "program" category includes any posts related to program problems, such as program design/debugging, graph processing, coding/language, function/control, program performance etc. For example:

" … Does LabVIEW 7 have an easy way allow the user at runtime to change the default values of controls? …"

" … Hello. I am new in the LabView, and I have to do a task. When I will get lowercase it has to be display as an uppercase and uppercase to lowercase … Any ideas how to fix it?"

Any question related to hardware issues is classified into the "hardware" category, such as hardware driver, computer hardware, controller, chassis, sensor and any external hardware supported/developed by National Instruments. A complete hardware list can be found at the National Instruments official website. Examples of hardware related posts are:

"… Does anybody have experience booting LabVIEW RT disklessly over the internet using PXE on a PXI Embedded slot 1 controller…?"

"... Hello has somebody developed a LabView driver for the Diamond-MM-16 board. If yes, I'm interested ..."

We label a post as "tools and support" if the question is asked about LabVIEW itself (such as version, license and installation) or any existing external tools/ architectures/libraries supported by LabVIEW. For example:

" … How can I download all NI software and packages that our academic license covers? I know that the license covers many tools, and I wish to download and install them all …"

" ... I want to know if there is any easy way that I can update LabVIEW 11.0 to 16.0. Do I have to uninstall the old one and download the new one? …"

Any posts that do not belong to the above categories will be classified as "others." For example:

"Is it the word "vi" pronounced like "vee" or "v-i" (each letter separately). All opinions welcomed."

"Happy New Year to all the LabVIEW users."

2.5.3 Evaluation Results

In our work, 10-fold cross validation is used to evaluate the classification performance. Our evaluation metrics are *Precision* and *Recall* and *F-measure*. The performance of applying Multinomial Naive Bayes, SMO (a fast implementation of SVM) (Bach, Lanckriet, & Jordan, 2004) and Random Forest on our training set is shown in Table 4.

2.5.4 Prediction

From Table 4, we can see Multinomial Naive Bayes (minTermFreq = 1) has the best performance (marked in red in Table 2). So we chose this model as our prediction model. Applying Multinomial Naive Bayes (minTermFreq = 1) to predict all the

Table 4. Evaluation results. MinTermFreq-N means only keep words that appeared at least N times in all documents. Pre = Precision, Rec = Recall

	MinTermFreq-1	**MinTermFreq-2**	**MinTermFreq-3**	**MinTermFreq-5**
Multinomial Navie Bayes	Pre:0.766 Rec:0.753 F-1:0.758	Pre:0.753 Rec:0.728 F-1:0.740	Pre:0.761 Rec:0.733 F-1:0.745	Pre:0.761 Rec:0.733 F-1:0.745
SMO	Pre:0.701 Rec:0.726 F-1:0.711	Pre:0.709 Rec:0.727 F-1:0.715	Pre:0.704 Rec:0.719 F-1:0.710	Pre:0.702 Rec:0.714 F-1:0.707
Random Forest	Pre:0.704 Rec:0.721 F-1:0.648	Pre:0.699 Rec:0.714 F-1:0.641	Pre:0.701 Rec:0.724 F-1:0.656	Pre:0.701 Rec:0.724 F-1:0.656

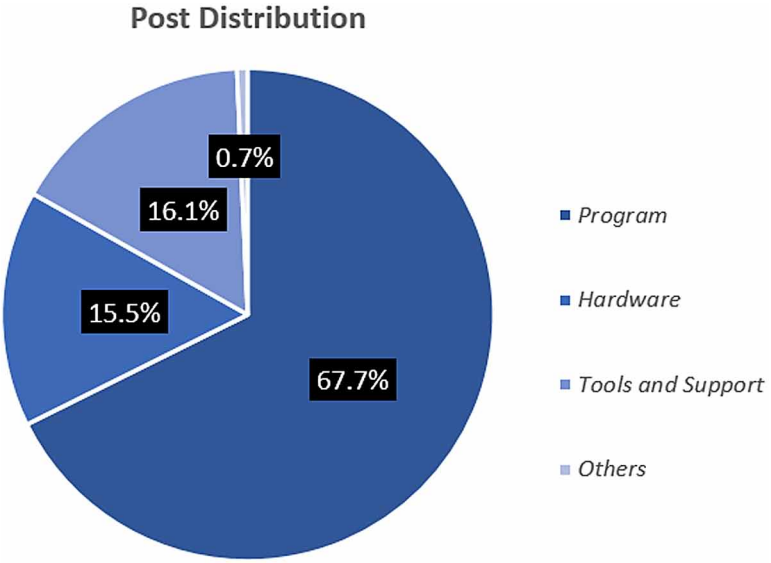
posts in our dataset, we obtained a post distribution shown in Figure 5. We can see that posts related to "program" have the largest portion (120,085 posts are related to program out of 177,495 posts). There are 28,636 posts that are predicted as "tools and support," which is a little more than posts predicted as "hardware" (27,541 posts). 1,233 posts are predicted as "others."

The prediction result shows that most LabVIEW discussion forum users are interested in the topic of *program*. Therefore, it would be beneficial in our analysis to understand what subtopics are discussed most among *program* related posts. However, with a wide range of topics, it is difficult to suggest predefined subcategories. To this end, we applied unsupervised machine learning techniques to delve into subtopic clustering among different "program" categories.

3. TOPIC MODELING

Applying unsupervised ML techniques to find topics in the text is referred to as "topic modeling" or "topic clustering." The task of topic modeling is to find groups/clusters of similar topics in the dataset. The similarity is computed through a similarity function. The term "unsupervised" indicates this process is done automatically without any human seeding or initiation of the process. The cluster can be in different levels of granularities, such as sentences, phrases and words. Conventional unsupervised

Figure 5. Post distribution prediction for LabVIEW discussion forum



ML techniques for topic clustering include k-means clustering (Hartigan & Wong, 1979), hierarchical clustering (Johnson, 1976), and Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003). A comparison of these approaches is summarized by Steinbach (Karypis, Kumar, & Steinbach, 2000). Unsupervised machine learning for topic modeling mainly consists of four steps:

Step 1: Data Pre-processing
Step 2: Feature Extraction
Step 3: Unsupervised Clustering
Step 4: Results Interpretation

The approaches used in Step 1 and Step 2 are similar to supervised machine learning techniques for text classification. Therefore, in this section, we mainly discuss the techniques used in Step 3. The empirical example from LabVIEW online discussion forum is introduced in the end of this section to show how unsupervised clustering is adopted in practice.

3.1 Unsupervised Machine Learning Algorithms

3.1.1 K-Means Clustering

K-means clustering (Hartigan & Wong, 1979) is a popular technique for cluster analysis in data mining based on vector quantization. It helps to identify similar groups of data (Xiong, Hua, Lv, & Li, 2016). K-means clustering partitions n observations into K clusters in which each observation belongs to the cluster with the nearest mean. The basic steps of K-Means clustering include:

1. Initialize K centroids randomly (also known as the K seeds);
2. Convert data to vector and compute the distance between observation data and each centroid, and assign this observation to the closest centroid;
3. Recompute the centroid based on the data assigned to the respective cluster;
4. Repeat step 2 and step 3 until stopping criteria are reached. Stopping criteria include
    a. Centroids do not change location;
    b. No data change its cluster;
    c. Reach the iteration time defined manually.

Clustering is sensitive to the distance measure. Distance measure is used to provide a relationship between each element in the dataset. The most simple distance measure is called *Manhattan Distance*. This measure simply calculates the sum of

the horizontal and vertical distances between two points *p* and *q*. For example, in a 2D space, Manhattan Distance is defined as:

$$d\left(p,q\right)=\left|q_x-p_x\right|+\left|q_y-p_y\right|$$

In K-means clustering, pairwise *Euclidean distance* is applied implicitly. In mathematics, Euclidean distance represents the length of the line segment connecting two points. In a 2-dimension space, it is defined as:

$$d\left(p,q\right)=d\left(q,p\right)=\sqrt{\left(p_1-q_1\right)^2+\left(p_2-q_2\right)^2}$$

One of the advantages of Euclidean distance is that the distance between any two objects is not affected by the addition of new objects to the analysis.

Another way to measure the distance is to apply *cosine distance*. In K-means clustering, using cosine distance rather than Euclidean distance is referred to as *Spherical K-means clustering*. Cosine distance is defined as:

$$d\left(\vec{A},\vec{B}\right)=1-\frac{\vec{A}\cdot\vec{B}}{\vec{A}\vec{B}}$$

In this equation, $\vec{A}\cdot\vec{B}$ is the dot product of vector A and vector B and $\vec{A}\vec{B}$ is the multiplication of the norm of vector A and vector B. $\frac{\vec{A}\cdot\vec{B}}{\vec{A}\vec{B}}$ is also known as *cosine similarity*.

Each distance calculation approach has its own advantages and disadvantages. In the context of text mining, Li and Han (Li & Han, 2003) conducted an empirical evaluation on different distance measuring metrics on three text datasets (20 Newsgroups, containing 20 categories with 11,293 training examples and 7,528 testing examples; Reuters 52c, containing 52 categories with 6,532 documents for training, and 2,568 documents for test; and Sector, containing 104 categories with 6,412 training samples and 3,207 test samples). Their results suggest that cosine distance has the best performance on average.

The algorithm configuration of K-Means cluster mainly includes: 1) a distance function to compare the similarity between different instances (we introduced this parameter in the above discussion; 2) max iterations to define the iteration time for the algorithm running; and 3) number of clusters (i.e., the value of K).

3.1.2 Hierarchical Clustering

Hierarchical clustering (Johnson, 1976) (also known as hierarchical cluster analysis, HCA) is an alternative approach to K-means clustering for identifying groups in the dataset. It is a method of cluster analysis in data mining to build a hierarchy of clusters. Hierarchical clustering is the hierarchical decomposition of the data based on group similarities. The result of clustering resembles a tree structure, called a *dendrogram*. Unlike k-means clustering, hierarchical clustering does not need to specify the number of clusters.

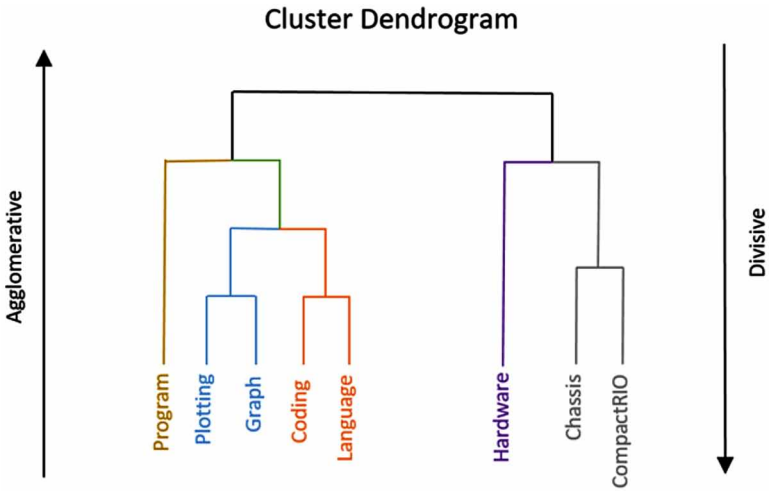
Based on the manner of how a dendrogram is constructed, HCA can be divided into two types: agglomerative HCA and divisive HCA.

1. **Agglomerative HCA**: In agglomerative HCA, the dendrogram is built in a bottom-up manner. At first, each text data is considered as a single cluster. At each step, the similarity between each cluster is calculated (the similarity calculation approach is the same as the techniques for K-means cluster we discussed earlier in this section). The two clusters that are most similar are combined into a new larger cluster. This process iterates until all the clusters are combined into one final cluster.
2. **Divisive HCA**: In divisive HCA, the dendrogram is built from top to bottom. Initially, all the text data is treated as one single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. This process iterates until all objects are in their own cluster.

Figure 6 shows the process of building a dendrogram in an agglomerative and divisive manner. In this example, we can see that from the agglomerative perspective over the first three iterations, plotting and graph (marked as blue in Figure 6), coding and language (marked as orange in Figure 6), chassis and compactRIO (marked as gray in Figure 6) are grouped as one cluster. Then, the blue cluster and the orange cluster are merged into a larger cluster (marked as green in Figure 6) at the fourth iteration. At the fifth and sixth iteration, program cluster and green cluster, hardware and gray cluster merged to a bigger cluster respectively. In the last iteration, all of the clusters are combined into just one cluster (marked as black in Figure 6). The clustering process stops because all of the clusters are merged into one cluster. The divisive HCA is executed in the same manner, but from top to bottom.

The parameters used in hierarchical clustering are similar to configurations in K-means clustering. In particular, hierarchical clustering also needs a provided distance function. However, in order to calculate the similarity between two clusters, another parameter is needed. This parameter is called the link type. Usually, there are three kinds of link types: nearest neighbor, complete linkage and group average.

Figure 6. An example showing the process of hierarchy clustering



Nearest neighbor connects two clusters based on the nearest points in two different clusters and complete linkage does the opposite: choose the furthest points in two clusters to combine into one cluster. The group average calculates the distance of all the points among different clusters and computes the average value. The cluster link is based on this average value.

3.1.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a statistical topic modeling approach that is widely used in NLP. It automatically discovers topics in the documents. The topic inference for LDA is built upon Dirichlet-multinomial distribution (Ronning, 1989), a family of discrete multivariate probability distributions from finite non-negative integers. The number of clusters needs to be specified manually in LDA at the beginning of the process. There are two principles in LDA:

- Each document contains multiple topics.
- Each topic contains multiple words.

For example, given three documents (e.g., posts from the LabVIEW discussion forum):

Post 1: “LabVIEW program is hard to debug.”
Post 2: “CompactRIO cannot connect to LabVIEW.”

Post 3: "Share your example program with others in the NI community."

Suppose we want to find two topics among these posts. Applying LDA to thee three posts, it may produce a result like:

Building topic for documents:

Post 1: 100% belongs to Topic A;
Post 2: 100% belongs to Topic B;
Post 3: 80% belongs to Topic A, 20% belongs to Topic B.

Building words for topics:

For Topic A: 95% program, 70% debug, 30% share …
For Topic B: 90% CompactRIO, 60% connect …

At this point, LDA cannot specify what is Topic A and what is Topic B. The result of LDA needs human (usually domain expert) interpretation. Based on our domain knowledge, Topic A could be interpreted as *program* and Topic B could be interpreted as *hardware*.

Compared with k-means clustering and hierarchical clustering, LDA has many advantages. First, it is a probabilistic model that can be interpreted easily by humans. Second, since LDA does not involve similarity comparison between different data objects, which sometimes can be extremely expensive to compute, it saves calculation space and time. LDA has revolutionized the field as a popular technique for topic modeling since it was first introduced in 2002.

The parameters LDA need are more complicated than K-means and hierarchical clustering. The most important parameters include:

- The number of topics $\boldsymbol{K}$: the number of requested latent topics to be extracted from the training corpus. It is similar to the number of clusters in K-Means clustering.
- Prior estimator $\boldsymbol{\alpha}$: the average frequency that each topic within a given document occurs.
- Topic-word estimator $\boldsymbol{\beta}$: a collection of k topics where each topic is given a probability distribution over the vocabulary used in a document corpus.

In general, higher alpha values mean documents contain more similar topic contents. The same is true for beta – a higher beta value means a topic contains more similar topic words. In our experiment, we chose 3 as the value of $\boldsymbol{K}$. We implemented our algorithm based on an LDA Python library – Gensim (Khosrovian, Pfahl &

Garousi, 2008). In Gensim, the value of and $\boldsymbol{\alpha}$ are $\boldsymbol{\beta}$ are tailored to the structure of a dataset. This is known as "asymmetric prior."

3.2 Empirical Example of Topic Modeling

In this section, we discuss topic clustering using our case study from the LabVIEW discussion forum. Section 2.4 in this chapter introduced our post classifications based on the LabVIEW discussion forum and the results suggested that forum participants are more interested in *program* related questions. Based on this observation, we want to look deeper into program related posts. Because the posts in the forum spanned a very wide range of topics, it is difficult to suggest predefined subcategories. To this end, we applied unsupervised machine learning techniques to identify several important sub-topics. Because the LDA algorithm is a powerful tool for discovering and exploiting hidden topics (Liu, Tang, Dong, Yao, & Zhou, 2016), we applied LDA for topic clustering on our discussion forum.

The first step for LDA is to choose a clustering number. In our experiment, we examined three different clusters: 3, 5 and 10. Experimental results suggest that when using 5 and 10 clusters, there are many overlapping concepts across each cluster. Therefore, we chose 3 topic clusters in the following empirical discussion.

3.2.1 Dataset

As mentioned in Section 2.4, there are 120,085 posts (out of 177,495) that are related to the topic of *program*. Among the 120,085 posts that were predicted as *program* related, we chose posts with prediction probability equals to 1, which means the prediction model is very confident about the selected posts are related to *program* category. After filtering according to this confidence level, 92,655 posts remained.

3.2.2 Clustering Results

We developed a Python program for LDA based on pyLDAvis (Sievert & Shirley, 2014), a library for helping users interpret the results in a topic model that has been fit to a corpus of text data. We applied LDA topic clustering on the dataset with 3 clusters. The result for one iteration is as follows:

```
(1, '0.014×"control" + 0.014×"array" + 0.009×"button"
+ 0.009×"event" + 0.008×"panel" + 0.008×"problem" +
0.007×"change" + 0.007×"try" + 0.007×"front" + 0.007×"value"')
(2, '0.008×"try" + 0.008×"array" + 0.008×"write" +
0.008×"example" + 0.007×"problem" + 0.007×"error"
```

```
+ 0.007×"signal" + 0.007×"graph" + 0.006×"input" +
0.006×"sample"')
(3, '0.014×"graph" + 0.009×"problem" + 0.008×"control"
+ 0.007×"try" + 0.007×"display" + 0.007×"output" +
0.007×"waveform" + 0.006×"array" + 0.006×"input" +
0.005×"could"')
```

Because LDA is a probabilistic model, when running multiple times, we may get different results. We applied LDA to the same dataset five times. In order to find uniqueness in each cluster, we eliminated repeated words across different clusters. Table 5 shows the frequency of unique terms after 5 times LDA running. From Table 3, we can interpret the three clusters as: program change, file operation and graph processing (this interpretation also requires domain knowledge of the dataset).

The combination of supervised and unsupervised machine learning techniques revealed several interesting insights. For example, the results tell us that most forum posters are concerned about program related questions compared with other categories. Additionally, in the program related posts, many posts focused on program change, file operation and graph processing.

Our experiment reveals several interesting research implications. First, from the results we obtained in Section 2.3, the "change" always appears in the largest cluster. Motivated by this, we are particularly interested in understanding posts containing the keyword 'change,' in which many of the posts implied an intention of "refactoring," as used in software engineering. By examining posts belonging to the first cluster, we found many posts that discussed the effects of bad smells for LabVIEW system models. Some posts contain bad smells that are common in source code that also appear in LabVIEW. For example,

Table 5. Unique Term frequency in three clusters after 3 iterations. Cluster 1 represents the largest topic distribution, cluster 3 represents the smallest distribution.

Unique Term Frequency	**5**	**4**	**3**	**2**	**1**
Unique terms in Cluster 1		change	event, button	graph, panel, event	example, control, display, front
Unique terms in Cluster 2			output, value, function	write, number, file	attach, example
Unique terms in Cluster 3		graph	waveform, sample	point, display	create, value, however, input, signal, control, output, could

"… A lot of duplicated code. Most of it is the same except selected cluster element changes..."

-We are currently investigating the summarization of bad smells in LabVIEW system models from an end user's perspective. Second, LabVIEW users expressed concern about the best way to design software that eliminates redundancies and can be maintained for a long period. For example, the user post

""...we have a big LV-project with some conditional disabled or with a switch unused VI's. But LabVIEW still tries to load them when building an application (even when they are not used). Something like ignore unused/missing VI's during built?..."

suggests that errors were made during the design that eventually led to some unreached model elements during the execution. We are also working on design analysis of LabVIEW system models, but all of these topics are outside of the scope of the context of this paper and are focused more on the example domain

4. FUTURE TRENDS

A recent trend has emerged towards applying deep learning methods to address the challenges of text classification and topic modeling. In this section, we first introduce deep learning in text classification and then we move to the future trends in applying deep learning to topic modeling.

4.1 Deep Learning in Text Classification

Convolutional Neural Networks (CNNs) (LeCun, Bottou, Bengio, & Haffner, 1998) and Recurrent Neural Networks (RNNs) (Rumelhart, Hinton, & Williams, 1986) are the two most popular approaches for text classification based on deep learning. Compared with traditional machine learning algorithms, deep learning techniques reduce the need for feature extraction and have shown better performance (Joulin, et al., 2016).

Research into Convolutional Neural Networks (CNNs) is inspired from how animal visual cortex works for seeing objects. CNNs have shown impressive results in several areas, such as image recognition (Girshick, 2015) and text classification (Conneau, et al. 2017). Given a corpus of documents, CNN first vectorizes all the documents. The input for CNN-based text classification is a matrix. The width of the matrix (d) is the word embedding (a technique that is capable of capturing context of a word in a document and relation with other words. Word2Vec is one of the most

popular methods to learn word embedding dimension). The height of a matrix is the number of words in the document. After receiving the input matrix, CNN performs a set of continuous operations named convolution and pooling.

A Recurrent Neural Network (RNN) is a sequence of artificial neural network blocks linked to each other, thus generating a directed graph. This connection allows RNNs to exhibit temporal dynamic behavior. RNN is based on Rumelhart's work (Rumelhart et al., 1986). It is widely used in connected handwriting recognition and speech recognition through the power of processing sequences of input (Graves, Mohamed, & Hinton, 2013). Some researchers have applied RNN to NLP applications (Laiet al., 2015) (Kowsari et al., 2017). There are several classification architectures built upon RNN, such as Zhou's work (Zhou et al., 2015) and Cho's work (Cho et al., 2014). Zhou's work is based on Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and Cho's work is based on Gated Recurrent Units (GRUs), a variation to LSTM. Compared with LSTM, GRU modifies the structure of the network unit.

Another technique based on deep learning for text classification is Hierarchical Attention Network (HAN) (Yang et al., 2016). This approach contains a hierarchical structure that mirrors the hierarchical structure in documents. It also incorporates mechanisms that enable this approach to recognize the importance of content at two levels: word level and sentence level.

There is not a fixed answer to the question of which approach is better to solve text classification problems. Yin (Yin et al., 2017) conducted an empirical study to compare the performance of three deep learning based approaches: CNN, GRU and LSTM on 7 datasets. The result shows that given different dataset, different approaches exhibited varied performance. Although applying deep learning techniques to solve text classification problems is a future trend, these techniques usually require a large amount of data and are computationally expensive (Chen and Lin, 2014). However, many investigations have shown that deep learning algorithms outperform traditional machine learning algorithms (Zhang, Zhao, & LeCun, 2015) (Majumder et al., 2017) (Hassan & Mahmood, 2018). Utilizing deep learning techniques for text classification remains an open research topic.

4.2 Deep Learning in Topic Modeling

Xu (Xu et al., 2015) proposed a topic modeling approach called STCC (Short Text Clustering via Convolutional neural networks), a clustering technique that considers a single constraint on learned features through a self-taught learning framework without using any external tags/labels. By applying their proposed work to two datasets, the authors compared their results with the best baselines and found that their approach achieved several improvements. Based on this, Xu (Xu et al., 2017)

extended their original work and proposed a flexible Self-Taught Convolutional neural network framework for Short Text Clustering (dubbed STC2). It could incorporate more useful semantic features and learn non-biased deep text representation in an unsupervised manner.

Wang (Wang, Mi & Ittycheriah, 2016) proposed a semi-supervised methodology to combine the representation learning process and K-Means clustering to achieve short text clustering via deep representation learning. In their approach, texts are presented as distributed vectors with neural networks. Some labeled data are also adopted to specify the intention of clustering.

Compared with applying deep learning techniques to text classification, topic modeling approaches based on deep learning has not been widely investigated. We believe applying deep learning techniques in topic modeling will be a future trend in the area of text summarization.

5. FUTURE WORK AND CONCLUSION

This chapter introduced two key techniques related to text summarization in the context of understanding online discussion forums - supervised machine learning techniques for text classification and unsupervised machine learning techniques for topic modeling. As a supporting technique, web scraping - an approach to automatically download text from forum posts, is also addressed in this chapter. We also discussed two text classification techniques based on deep learning: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Currently, we are working on a full examination of the LabVIEW discussion forum through the application of supervised and unsupervised machine learning techniques. To improve the model performance, we are examining more posts to enlarge training dataset. To validate our results, we are planning to conduct an empirical analysis among LabVIEW experts. The empirical analysis will be done through a survey asking the participants 1): to what extent do the participants agree with our classification strategy used in supervised machine learning technique; 2) to what extent do the participants agree with the post distribution obtained from end-users.

We believe the results will offer insights into future research directions, thus leading to advanced capabilities that improve the effectiveness of LabVIEW end-users who develop systems models for their domain-specific work. Moreover, our proposed technique is generalizable to other modeling forums. We are also working on the examination of another popular systems modeling discussion forum – Simulink. By cross comparisons between LabVIEW and Simulink, our work will come up with new needs and visions for the future development of systems modeling tools. At an even more generalized level, we believe that our work can be applied to many

discussion forums that capture domain expertise that is spread across a broad range of user posts.

REFERENCES

Abbasi, A., France, S., Zhang, Z., & Chen, H. (2010). Selecting attributes for sentiment classification using feature relation networks. *IEEE Transactions on Knowledge and Data Engineering*, *23*(3), 447–462. doi:10.1109/TKDE.2010.110

Akcaoglu, M., & Lee, E. (2016). Increasing social presence in online learning through small group discussions. *The International Review of Research in Open and Distributed Learning*, *17*(3), 1–17. doi:10.19173/irrodl.v17i3.2293

Altınel, B., & Ganiz, M. C. (2018). Semantic text classification: A survey of past and recent advances. *Information Processing & Management*, *54*(6), 1129–1153. doi:10.1016/j.ipm.2018.08.001

Alvarez, S. A. (2002). *An exact analytical relation among recall, precision, and classification accuracy in information retrieval*. Boston College. Technical Report BCCS-02-01.

Bach, F. R., Lanckriet, G. R., & Jordan, M. I. (2004, July). Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the twenty-first international conference on Machine learning* (p. 6). Banff, Canada: ACM. 10.1145/1015330.1015424

Bayes, T. (1763). LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical Transactions of the Royal Society of London*, (53): 370–418.

Blake, R. (2015). *A Test Cell for Mission-Critical Aerospace Gearbox Testing*. Retrieved from https://www.gsystems.com/case-study-bell-helicopter

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Cambria, E. (2016). Affective computing and sentiment analysis. *IEEE Intelligent Systems*, *31*(2), 102–107. doi:10.1109/MIS.2016.31

Chen, X. W., & Lin, X. T. (2014). Big data deep learning: Challenges and perspectives. *IEEE Access: Practical Innovations, Open Solutions*, *2*, 514–525. doi:10.1109/ACCESS.2014.2325029

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724-1734). Doha, Qatar: Association for Computational Linguistics. 10.3115/v1/D14-1179

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 1107-1116). Valencia, Spain. Association for Computational Linguistics. 10.18653/v1/E17-1104

De Lara, E., Wallach, D. S., & Zwaenepoel, W. (2001). Puppeteer: Component-based adaptation for mobile Computing. In *USENIX Symposium on Internet Technologies and Systems* (pp. 159 - 170). San Francisco, CA: USENIX Association.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Deshpande, M. (2018). *Classification with Support Vector Machines*. Retrieved from https://pythonmachinelearning.pro/classification-with-support-vector-machines/

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Ertugrul, N. (2000). Towards virtual laboratories: A survey of LabVIEW-based teaching/learning tools and future trends. *International Journal of Engineering Education*, *16*(3), 171–180.

Ertugrul, N. (2002). *LabVIEW for electric circuits, machines, drives, and laboratories.* Prentice Hall PTR.

Falcon, J. (2017). Facilitating Modeling and Simulation of Complex Systems through Interoperable Software. In *International Conference on Model Driven Engineering Languages and Systems*. Austin, TX: ACM.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448). San Diego, CA: IEEE.

Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). Vancouver, Canada: IEEE.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C, Applied Statistics*, *28*(1), 100–108.

Hassan, A., & Mahmood, A. (2018). Convolutional recurrent deep learning model for sentence classification. *IEEE Access: Practical Innovations, Open Solutions*, *6*, 13949–13957. doi:10.1109/ACCESS.2018.2814818

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735 PMID:9377276

Joachims, T. (2006, August). Training linear SVMs in linear time. In *Proceedings of the 12th ACM international conference on Knowledge discovery and data mining* (pp. 217-226). Philadelphia, PA: ACM.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254. doi:10.1007/BF02289588 PMID:5234703

Jorczak, R. (2014). Differences in classroom versus online exam performance due to asynchronous discussion. *Online Learning Journal*, *18*(2), 1–9.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 3-7), Valencia, Spain: ACM. 10.18653/v1/E17-2068

Karypis, M. S. G., Kumar, V., & Steinbach, M. (2000, August). A comparison of document clustering techniques. In *TextMining Workshop at KDD 2000*. Boston, MA: ACM.

Khosrovian, K., Pfahl, D., & Garousi, V. (2008). GENSIM 2.0: a customizable process simulation model for software process evaluation. In *International Conference on Software Process* (pp. 294-306). Springer, Berlin, Heidelberg.

Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence (IJCAI)*, *14*(2), 1137-1145.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., & Barnes, L. E. (2017, December). Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 364-371). Cancun, Mexico: IEEE.

Kumar, S., Gao, X., Welch, I., & Mansoori, M. (2016, March). A machine learning based web spam filtering approach. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)* (pp. 973-980). Crans-Montana, Switzerland: IEEE. 10.1109/AINA.2016.177

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence* (pp. 2267-2273). Austin, TX. ACM.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi:10.1109/5.726791

Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 611-618). Hefei, China: Springer. 10.1007/978-3-642-41278-3_74

Lilleberg, J., Zhu, Y., & Zhang, Y. (2015, July). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)* (pp. 136-140). Beijing, China: IEEE. 10.1109/ICCI-CC.2015.7259377

Liu, L., Tang, L., Dong, W., Yao, S., & Zhou, W. (2016). An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, *5*(1), 1608. doi:10.118640064-016-3252-8 PMID:27652181

Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, *1*(4), 309–317. doi:10.1147/rd.14.0309

Majumder, N., Poria, S., Gelbukh, A., & Cambria, E. (2017). Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, *32*(2), 74–79. doi:10.1109/MIS.2017.23

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Scoring, term weighting and the vector space model. *Introduction to Information Retrieval, 100*, 2-4.

MathWorks. (2019). *Simulink Introduction*. Retrieved from https://www.mathworks.com/products/simulink.html

Miltsakaki, E., & Troutt, A. (2008, June). Real-time web text classification and analysis of reading difficulty. In *Proceedings of the third workshop on innovative use of NLP for building educational applications* (pp. 89-97). Columbus, OH: Association for Computational Linguistics. 10.3115/1631836.1631847

Morita, T. (2018). *Advancing Subaru Hybrid Vehicle Testing Through Hardware-in-the-Loop Simulation.* Retrieved from http://sine.ni.com/cs/app/doc/p/id/cs-15982#

National Instruments. (2019). *LabVIEW Introduction*. Retrieved from http://www.ni.com/en-us/shop/labview/labview-details.html

Nenkova, A., & McKeown, K. (2012). A survey of text summarization techniques. In *Mining text data* (pp. 43–76). Boston, MA: Springer. doi:10.1007/978-1-4614-3223-4_3

Pendry, L. F., & Salvatore, J. (2015). Individual and social benefits of online discussion forums. *Computers in Human Behavior*, *50*, 211–220. doi:10.1016/j.chb.2015.03.067

Razon, A., & Barnden, J. (2015). A New Approach to Automated Text Readability Classification based on Concept Indexing with Integrated Part-of-Speech n-gram Features. In *Proceedings of the International Conference Recent Advances in Natural Language Processing* (pp. 521-528). Academic Press.

Ronning, G. (1989). Maximum likelihood estimation of Dirichlet distributions. *Journal of Statistical Computation and Simulation*, *32*(4), 215–221. doi:10.1080/00949658908811178

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling, 5*(3), 1.

Salton, G., & Yang, C. S. (1973). On the specification of term values in automatic indexing. *The Journal of Documentation*, *29*(4), 351–372. doi:10.1108/eb026562

Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces* (pp. 63-70). Baltimore, MD: Association for Computational Linguistics. 10.3115/v1/W14-3110

Simpson, K. (2015). *You Don't Know JS: Async & Performance*. O'Reilly Media, Inc.

Sparks Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *The Journal of Documentation*, *28*(1), 11–21. doi:10.1108/eb026526

Suganya, S., & Gomathi, C. (2013). Syntax and semantics based efficient text classification framework. *International Journal of Computers and Applications*, *65*(15).

Xiong, C., Hua, Z., Lv, K., & Li, X. (2016, November). An Improved K-means text clustering algorithm By Optimizing initial cluster centers. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* (pp. 265-268). Macau, China: IEEE.

Xu, J., Peng, W., Guanhua, T., Bo, X., Jun, Z., Fangyuan, W., & Hongwei, H. (2015). Short text clustering via convolutional neural networks. In *13th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 62 - 69). Denver, CO: Association for Computational Linguistics. 10.3115/v1/W15-1509

Xu, J., Xu, B., Wang, P., Zheng, S., Tian, G., Zhao, J., & Xu, B. (2017). Self-taught convolutional neural networks for short text clustering. *Neural Networks*, *88*, 22–31. doi:10.1016/j.neunet.2016.12.008 PMID:28157556

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1480-1489). San Diego, CA: Association for Computational Linguistics.

Yin, W., Kann, K., Yu, M., & Schütze, H. (in press). *Comparative study of CNN and RNN for natural language processing.* Clinical Orthopaedics and Related Research. Retrieved from https://arxiv.org/abs/1702.01923

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649–657). Montreal, Canada: Neural Information Processing Systems.

Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1-4), 43–52. doi:10.100713042-010-0001-0

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016, December). Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 3485-3495). Osaka, Japan: The COLING 2016 Organizing Committee.