Enhancing Text Classification: A Comparative Study of CNN, SVM, and RNN Models

Chen Wang, 1005281446
wangc.wang@mail.utoronto.ca

Yan Pan Chung, 1007598039
normanyp.chung@mail.utoronto.ca

Pas Panthasen, 1009776654
pas.panthasen@mail.utoronto.ca

Sebastian Tampu, 1004928572
seb.tampu@mail.utoronto.ca

Abstract—**The unprecedented growth of textual data across numerous platforms poses a significant challenge for efficient information management, search, and retrieval. Addressing this, our study conducts a comparative analysis of machine learning models, namely Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), in the context of enhancing topical text classification. Leveraging diverse datasets from different domains, we explore the effectiveness of these models in accurately categorizing textual content, focusing on aspects such as accuracy, computational cost, and adaptability to high-dimensional data. Our findings indicate that the performance of these models is significantly influenced by preprocessing methods and the nature of the datasets. SVMs exhibit superior performance on all datasets, as they are characterized by high-dimensional feature spaces, the textual data. While RNNs demonstrate comparable accuracy in handling larger datasets like AG News, CNNs consistently show competitive results for lengthy text, 10 Newsgroup, and those with a high number of labels as seen in Banking77, utilizing the power of local textual patterns. This study not only highlights the role of model selection but also underscores the impact of preprocessing in optimizing model performance. Through this analysis, we contribute valuable insights into the strategic application of machine learning models to meet the evolving demands of information retrieval in the vast digital landscape**

I. Introduction

In the digital age, the exponential growth of textual data across various platforms has made it a daunting task to efficiently manage, search, and retrieve information. From social media posts and news articles to academic journals and legal documents, the vast digital landscape is rich with information that is often challenging to navigate. This project aims to address this challenge by enhancing the precision and efficiency of topical text classification, a fundamental aspect of information retrieval.

The problem of efficiently classifying textual data into relevant categories is not new; however, it has become increasingly complex with the surge in digital content productions. Existing solutions leverage a variety of machine learning models, each with its strengths and limitations. For instance, Convolutional Neural Networks (CNNs) are adept at identifying local patterns within blocks of text, making them suitable for recognizing distinct topical patterns. Support Vector Machines (SVMs) offer robust performance in high-dimensional feature spaces, effectively discriminating between categories based on text-derived feature vectors. Recurrent Neural Networks (RNNs), on the other hand, excel in processing sequential data, enabling them to grasp the context and flow within texts. Despite these advancements, as highlighted in the comprehensive review by Zhang, Zhao, and LeCun on deep learning for text understanding, there remains a significant gap in developing a unified solution that combines the strengths of these models to achieve higher accuracy and efficiency in topical text classification [1]. This project proposes to bridge this gap by conducting a comparative study of CNN, SVM, and RNN models, aiming to develop a comprehensive machine-learning solution that enhances the effectiveness of topical text classification.

Building on the foundation of enhancing topical text classification, the proposed solution centers on the deployment of SVMs as the primary methodology. The choice of SVMs is motivated by their efficiency in handling high dimensional data, a characteristic feature of text classification tasks. Unlike other machine learning models that may struggle with dimensionality, SVMs thrive in such environments by constructing hyper-planes in a high-dimensional space to separate different classes. This capability is particularly advantageous when dealing with textual data, where the feature space can expand rapidly with the addition of words, n-grams, or TF-IDF (Term Frequency-Inverse Document Frequency) vectors.

II. System Design

A. SVM

The main idea behind utilizing SVMs in our project is to leverage their robustness in feature-based classification to accurately categorize textual content into predefined topics. SVMs attempt to find the decision boundary that best separates the classes while maximizing the margin between the nearest points of the classes (support vectors). This method not only contributes to the high accuracy of the classification but also enhances the model's generalizability to unseen data. Additionally, by experimenting with different kernel functions the SVM model can be tuned to capture the patterns and relationships within textual data. The proposed solution aims to systematically evaluate the effectiveness of SVMs in topical text classification, exploring various preprocessing techniques and kernel functions to optimize accuracy.

Several kernels and hyperparameters were evaluated to identify the optimal configuration that yields the highest accuracy, they are:

- **Kernels:** *linear*, *poly*, *rbf*, *sigmoid*
- **C values:** 0.1, 1, 10
- **Gamma:** 0.01, 0.1, 1

A grid search is employed to find the best possible configuration. The kernels considered in this evaluation are *linear*, *poly* (polynomial), *rbf* (radial basis function), and *sigmoid*. Each represents a different way of computing the similarity between data points in the feature space, altering decision boundaries constructed by SVM. *Linear* kernel facilitates linear decision boundaries, while *poly* allows for more complex, polynomial-shaped boundaries. The *rbf* kernel is particularly effective for non-linear data, creating boundaries that can conform to data with complex distributions, and *sigmoid* kernel, mimicking the sigmoid function, offers more approach to handling non-linear data, although less likely used compared to *rbf*.

Hyperparameters *C* and *gamma* play a crucial role in the model's performance. *C*, the regularization parameter, dictates the trade-off between achieving a low error on the training data and minimizing the norm of the weights, helping control overfitting. Lower values of *C* increase regularization, encouraging simpler models, while higher values push the model to fit it as closely to the training data as possible. *Gamma* influences the shape of the decision boundary in kernels like *rbf*, *poly*, and *sigmoid*. It defines how far the influence of a single training example reaches, with low values indicating *far*, and high values denoting *close*. The interaction between *gamma* and the chosen kernel significantly affects the model's capacity to handle the nuances in the data.

The model also employs cross-validation to assess the performance of the SVM classifier across a range of *C* and *gamma* values for each kernel types, except for the *linear* kernel where *gamma* is not applicable. This approach ensures that the evaluation of model performance is robust and not overly dependant on the partitioning of data into training and test sets. The best parameters and scores are recorded, showcasing a methodical search for the optimal SVM configuration.

B. CNN (TextCNN)

CNNs are typically associated with Computer Vision tasks, leveraging convolutional and pooling layers to extract essential features for object detection. Convolutional layers reduce the spatial relationship of pixels in image features to lower-dimension representations. However, Kim's experiment with pre-trained word vectors and a simple CNN achieved outstanding results for sentence-level classification tasks [2]. Despite transformer-based models outperforming others in various NLP tasks, TextCNN with minimal hyperparameter tuning can achieve excellent results in multi-class text classification.[3].

The structure of TextCNN is quite similar to the traditional CNN, except that TextCNN applies convolutions over the sequential structure of words within sentences or documents to capture local patterns and features. The key components of TextCNN include embedding layers, convolutional layers, pooling layers, and fully connected layers. The kernel size varied for different mini-batches allows the model to capture n-grams of varying lengths instead of square kernel size in traditional CNN as depicted in Figure1.

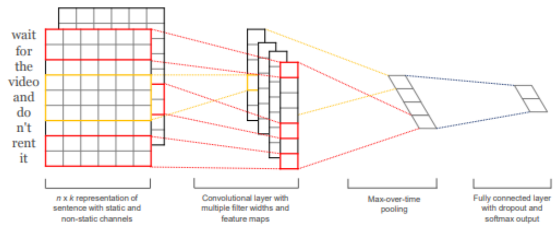
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 1: Model architecture with two channels for an example sentence [2]

In our TextCNN models, we utilize three different sizes of convolutional filters (2, 3, and 4 times the embedding dimension) to extract sequential features. Corresponding max-pooling layers are applied to downsample these features. The resulting hidden layers from each filter size are concatenated, allowing the model to capture various levels of detail in the input text. Finally, a fully connected layer classifies the concatenated features into the output class number.

In the context of hyperparameter tuning for TextCNN, we investigated the optimization of three crucial hyperparameters: kernel filter size, learning rate, and dropout rate. The different configurations are inspired by A. Bhavani and Kim's research to ensure reasonable choices.[3][2].

- **Learning Rate:** 0.001, 0.01
- **Kernel Size:** [2, 3, 4], [3, 4, 5]
- **Dropout Rate:** 0.2, 0.5, 0.6

C. RNN (BiLSTM)

Recurrent Neural Networks (RNNs) [4] are feedforward neural networks with a recurrent hidden state activated by the previous states at a certain time. This allows RNNs to dynamically model contextual information and handle variable-length sequences. Nevertheless, traditional RNNs are not sufficient to capture long-distance semantic connections, even if they can transfer semantic information between words [5]. During parameter training, the gradient may decrease gradually until it disappears, and hence the length of sequential data would be limited. To overcome the problem of gradient disappearance, Long Short-Term Memory (LSTM) [4] [5] is proposed by replacing the self-connected hidden units with memory blocks, applying purpose-built memory cells to store information for effectively finding and exploiting long-range context.

With traditional RNNs and LSTMs, information can only be propagated forward such that the state of time t only depends on the information right before time t. To make each moment contain the context information, Bidirectional LSTM (BiLSTM) [5] is proposed to access both the preceding and succeeding contexts or capture them by combining forward and backward hidden layers as in Figure 2.

A simple and complete BiLSTM model [6] is comprised of an embedding layer with text information as the input, a

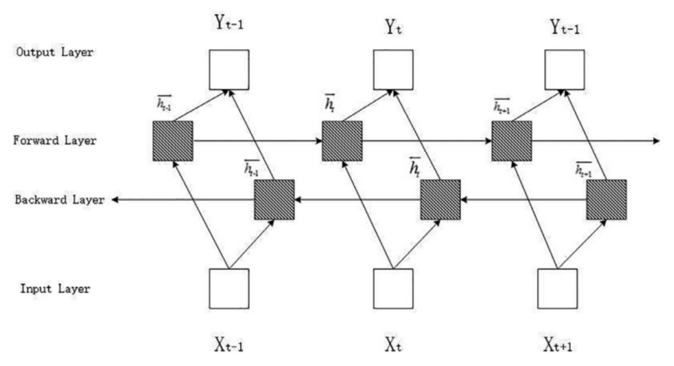

Fig. 2: Illustration of a BiLSTM model [4]

BiLSTM layer followed by a dropout layer, and a softmax layer for the output. Further details of each layer are provided in the following subsections.

1) Embedding Layer: At the beginning of the model, the input will carry data samples as a sequence of unique indices of the same length. The input will then be sent to the embedding layer where each index, corresponding to a unique word in the dataset, is transformed into a real-valued feature vector. These vectors are stacked together to form an embedding matrix such that each row of the embedding matrix depicts a unique index corresponding to a unique word in the vocabulary.

2) BiLSTM Layer: The BiLSTM layer consists of two LSTM networks and is capable of reading input reviews in forward and backward directions. The forward LSTM processes information from left to right and its hidden state can be shown as $\overrightarrow{h_t} = LSTM\left(x_t, \overrightarrow{h_{t-1}}\right)$ whereas the backward LSTM does the opposite, $\overleftarrow{h_t} = LSTM\left(x_t, \overleftarrow{h_{t+1}}\right)$. In the end, the output of BiLSTM layer can be concluded by combining the forward and backward states as $h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$. Additionally, dropout layers are included in the BiLSTM layer to reduce overfitting.

3) Output Layer: At the output, a softmax function is used to activate a dense layer to output a probability over a number of categories for each sample, then, a categorical cross-entropy loss is applied to support the multi-class classification.

D. Model Comparison

Comparing the efficacy of SVMs, TextCNNs, and BiLSTMs in the domain of topical text classification, each model presents a unique set of strengths tailored to specific aspects of the task at hand. TextCNNs excel at capturing local patterns within the text, utilizing convolutional layers to identify key phrases and word associations indicative of particular topics, making them highly effective for tasks where semantic patterns within fixed window sizes play a crucial role. On the other hand, BiLSTMs leverage their ability to process text sequences in both forward and backward directions, offering a profound understanding of context and the relationship between words over long stretches of text. This feature is particularly advantageous for understanding complex sentence structure and nuances in language that may be pivotal for accurate classification.

Despite the advanced capabilities of TextCNNs and BiLSTMs in handling textual data, SVMs hold a slight edge in scenarios where interpretability, simplicity, and computational efficiency are prioritized. SVMs, with their foundation in statistical learning theory, offer a robust framework for text classification, especially when dealing with high-dimensional sparse data, which is typical in text analysis. The efficiency of SVMs is underscored in their ability to handle extensive feature spaces with relatively fewer instances, a common occurrence in text data. Moreover, the flexibility in choosing kernels allows SVMs to adaptively model linear and non-linear relationships without the need for extensive data preprocessing or transformation. This adaptability, combined with the model's generalization capabilities, ensures that SVMs not only maintain competitive performance but also offer a more straightforward implementation and tuning process compared to the intricate architectures and computational demands of TextCNNs and BiLSTMs.

E. Datasets

The datasets utilized in this study consist of three corpora from two domains, as shown in TABLE I, covering various aspects of natural language tasks. The Banking77[7] dataset is composed of online banking queries labeled with 77 domains, while 10 Newsgroups[8] and AG's News[9] are collections of news articles, with the former excelling in text length and the latter having a higher number of data samples.

TABLE I: Dataset Characteristics

Dataset	**Size**	**#Classes**	**MaxLen**	**AvgLen**	**Domain**
Banking77	13.0k	77	433	58	Finance
10 Newsgroups	1.0k	10	55,227	2,574	General
AG News	127.6k	4	1,012	236	General

F. Preprocessing

As text data comes from diverse sources, preprocessing is required to transform it into a suitable format, often numerical, for model input. In natural language processing, this process includes cleaning, normalizing, and encoding the text data [10]. These steps consist of tokenization, expanding contractions, lowering case, removing words/characters, as well as lemmatization, stemming, and text encoding, either through TF-IDF vectorization or word indexing. The decision to use any of these techniques depends on the datasets and models. While some, such as word removal, can reduce noise, leading to decreased model complexity and training time, they may potentially discard useful information.

III. Results

The performance evaluation of this text classification relies primarily on accuracy measurement, supplemented by loss evaluation. Accuracy represents the proportion of correctly labeled data to the total number of data points, making it easily interpretable for humans and also fast and efficient to calculate. Since the test sets used for evaluation are all balanced, other metrics such as precision, recall, and F1 score may not be necessary. This is in contrast to imbalanced datasets where these metrics are crucial, as a model may achieve high accuracy by predicting only the majority class, potentially resulting in lower average precision and recall. Loss is another measurement used during the training of neural networks, penalizing misclassified data unequally, and is useful for indicating overfitting.

The following section will be divided to describe the tuning hyperparameters of each model, as well as an overall performance comparison at the end.

A. SVM

As mentioned, the grid search method was used to find the optimal configurations for each of the datasets. The optimal configurations along with their accuracy scores are shown in TABLE II. The full Banking77 and 10 Newsgroup datasets were used for this search as their size, seen in TABLE I, is suitable for SVM training. The AG News dataset is much larger than the other two, and so could have caused the SVM to require several hours to learn. As such, only 5,000 samples were used to find the optimal configuration for that dataset. The configurations presented in TABLE II are used to compare final scores against the RNN and CNN models.

TABLE II: Best SVM Parameters and Accuracy Scores for Different Datasets

Dataset	Kernel	C	Gamma	Accuracy Score
Banking77	*rbd*	10	0.1	88.8
10 Newsgroup	*sigmoid*	10	1	98.0
AG News	*rbf*	10	1	88.7

B. TextCNN

In our experimentation, we embarked on hyperparameter tuning for the TextCNN model, focusing on a subset of 5,000 AG News data. This involved exploring twelve unique combinations to find the most optimal configuration.

In the context of the result as shown in TABLEIIIwe observed that the model with a learning rate of 0.001 combined with a kernel filter size of [2, 3, 4] and a dropout rate of 0.5 achieve the highest performance, resulting in a validation accuracy of 80.24%. Thus, we will retain this configuration as the optimal one for comparison in the final stage.

TABLE III: Hyperparameter Tuning on 5,000 AG News Data with Twelve Different Combinations

LR	Kernel Size	Dropout Rate		
		0.2	0.5	0.6
0.001	[2, 3, 4]	79.3	80.2	78.6
	[3, 4, 5]	77.3	77.7	79.0
0.01	[2, 3, 4]	79.6	73.4	80.2
	[3, 4, 5]	76.3	76.6	77.6

C. RNN (BiLSTM)

TensorFlow library in Python is used to construct the proposed BiLSTM model and evaluate its performance with Banking77 and AG News.

The hyperparameters in the proposed BiLSTM model include the number of epochs, embedding dimension, learning rate, batch size, the number of hidden layers, and more. When considering the classification scores of the model based on Banking77, the two following hyperparameters have the greatest effect.

1) Embedding Dimension: In general, embedding dimension, also called the dimensionality of word embedding, refers to the number of dimensions in which the vector representation of a word is defined. Figure 3 shows that the growth of embedding dimensions causes the testing loss of the model to decrease, with some vibration. However, the loss increases again when the number of embedding dimensions is high.

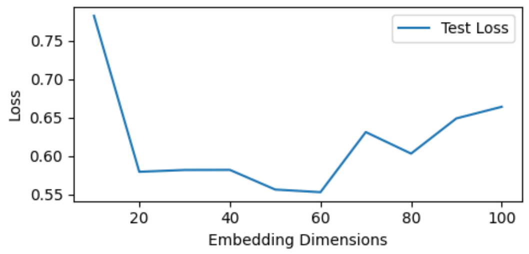

Fig. 3: Testing loss versus embedding dimensions

2) Batch Size: An appropriate choice of batch size is important for optimizing classification results. As shown in Figure 4 if the batch size is too large or too small, the testing loss will increase and the testing accuracy will decrease. Also, smaller batch size will extend training time.

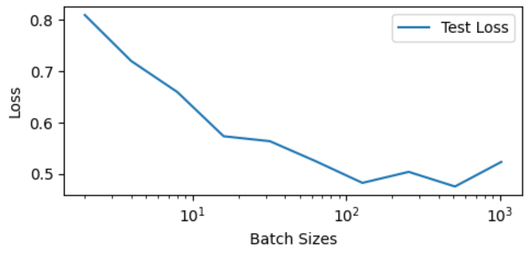

Fig. 4: Testing loss versus batch sizes

In summary, for the value of embedding dimensions, testing showed that a value of 60 may lead to a testing loss of 0.553 and a testing accuracy of 87.5%. For the value of batch size, a value of 512 results in the best performance: a loss of 0.476 and an accuracy of 87.6%. The aforementioned analysis of BiLSTM model is based on hyperparameters listed in Table IV.

When training the BiLSTM model using AG News [9] and hyperparameters in Table IV, the model takes around 20 minutes to train, receiving a testing loss of 0.246 and a testing accuracy of 92.3%. Moreover, training with more than 5 epochs or an unsuitable learning rate may lead to an overfitting problem. Lastly, increasing batch size can help reduce training time while retaining good performance.

TABLE IV: Hyperparameters applied to BiLSTM model with different datasets

Hyperparameters	**Banking77 [7]**	**AG News [9]**
Batch Size	16	200
Validation Ratio	0.15	0.1
Embedding Dimensions	70	100
Learning Rate	0.005	0.005
Hidden Layers	128	64
Epochs	20	5
Beta1	0.9	0.5
Beta2	0.99	0.5
Dropout	0.5	0.5

D. Overall

Testing all models with three datasets and four combinations of preprocessing steps, the accuracy of the test sets is as shown in Table V. The preprocessing methods are as follows: 0 - only splitting text by white space, 1 - text tokenization, case lowering, eliminating non-alphabet tokens, 2 - integrating contraction fixing, lemmatization, and stemming, and 3 - add stop word removal. All of these methods are then processed through TF-IDF vectorization for the SVM model and word indexing for words that appear more than three times for CNN and RNN models.

TABLE V: Accuracy Results of Machine Learning Models on Different Datasets with Various Preprocessing Methods

Dataset	**ML Models**	**Preprocessing Type**			
		0	**1**	**2**	**3**
Banking77	SVM	90.4	89.9	90.3	87.0
	CNN	82.6	86.2	88.1	82.8
	RNN	85.7	86.7	87.4	85.5
10 Newsgroup	SVM	97.5	97.0	98.0	96.0
	CNN	78.0	88.5	88.5	91.0
	RNN	38.0	35.5	56.5	66.5
AG News	SVM	92.2	91.5	92.2	92.0
	CNN	90.1	90.4	90.7	90.8
	RNN	91.9	91.7	91.9	91.7

The results indicate that SVM outperforms the other models across all datasets, while CNN secures its second position on Banking77 and 10 Newsgroups, and RNN achieves runner-up status for AG News. Several factors contribute to these outcomes, but mainly due to encoding technique and the unique advantages of each model. The utilization of TF-IDF vectorization in SVM enables it to effectively capture the significance of words relative to the entire corpus, aligning with SVM's strength in handling separable data in high-dimensional space. Conversely, the use of word indexing in CNN and RNN makes them suitable for capturing local patterns and sequencing data. In most cases, preprocessing helps these neural network models increase accuracy when it is type 0 to 2, but not when including word removal, except for the RNN case of 10 Newsgroups where the data are lengthy, thus significantly improving its performance.

IV. Concluding Remarks

In conclusion, our analysis reveals that SVMs, CNNs, and RNNs each have unique strengths in topical text classification, necessitating a nuanced approach to model selection and preprocessing to optimize performance across various datasets. SVMs excel in high-dimensional feature spaces, showing great efficiency in all datasets, while RNNs also stand out in processing larger datasets like AG News. Meanwhile, CNNs consistently deliver competitive results across all datasets, demonstrating their capability to capture local textual patterns effectively. Looking ahead, advancing these models' performance and adaptability involves exploring advanced preprocessing techniques, hybrid models, and the potential of transformer-based models for their context understanding and sequential data handling capabilities. Additionally, enhancing scalability and computational efficiency, especially for real-time applications, will be vital in making information retrieval systems more responsive, marking crucial next steps in our pursuit to navigate and manage the vast digital textual landscape efficiently.

References

[1] X. Zhang, J. Zhao, Y. LeCun, "Character-level Convolutional Networks for Text Classification", ArXiv:1509.01626v3 [cs.LG] (Submitted on 4 Sep 2015, last revised 4 Apr 2016), 2015, doi:10.48550/arXiv.1509.01626, 1509.01626.

[2] Y. Kim, "Convolutional Neural Networks for Sentence Classification", *in* A. Moschitti, B. Pang, W. Daelemans, eds., *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Association for Computational Linguistics, Doha, Qatar, Oct. 2014, doi:10.3115/v1/D14-1181, URL: https://aclanthology.org/D14-1181.

[3] A. Bhavani, B. Santhosh Kumar, "A Review of State Art of Text Classification Algorithms", *in 2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1484–1490, 2021, doi:10.1109/ICCMC51019.2021.9418262.

[4] G. Liu, J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification", *Neurocomputing*, vol. 337, pp. 325–338, 2019, doi:https://doi.org/10.1016/j.neucom.2019.01.078, URL: https://www.sciencedirect.com/science/article/pii/S0925231219301067.

[5] G. Xu, Y. Meng, X. Qiu, Z. Yu, X. Wu, "Sentiment Analysis of Comment Texts Based on BiLSTM", *IEEE Access*, vol. 7, pp. 51522–51532, 2019, doi:10.1109/ACCESS.2019.2909919.

[6] Z. Hameed, B. Garcia-Zapirain, "Sentiment Classification Using a Single-Layered BiLSTM Model", *IEEE Access*, vol. 8, pp. 73992–74001, 2020, doi:10.1109/ACCESS.2020.2988550.

[7] I. Casanueva, T. Temcinas, D. Gerz, M. Henderson, I. Vulic, "Efficient Intent Detection with Dual Sentence Encoders", *CoRR*, vol. abs/2003.04807, 2020, URL: https://arxiv.org/abs/2003.04807, data available at https://github.com/PolyAI-LDN/task-specific-datasets, 2003.04807.

[8] J. Baxter, "10 Newsgroups Dataset", https://www.kaggle.com/datasets/jensenbaxter/10dataset-text-document-classification, 2020.

[9] A. Gulli, "AG's corpus of news articles", http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, 2005.

[10] E. Haddi, X. Liu, Y. Shi, "The Role of Text Pre-processing in Sentiment Analysis", *Procedia Computer Science*, vol. 17, pp. 26–32, 2013, doi:10.1016/j.procs.2013.05.005, URL: https://www.sciencedirect.com/science/article/pii/S1877050913001385, first International Conference on Information Technology and Quantitative Management.