Baltic J. Modern Computing, Vol. 7 (2019), No. 1, pp. 47–60
https://doi.org/10.22364/bjmc.2019.7.1.04

SVM and k-Means Hybrid Method for Textual Data Sentiment Analysis

Konstantinas KOROVKINAS[1], Paulius DANĖNAS[2], Gintautas GARŠVA[1]

[1] Institute of Social Sciences and Applied Informatics, Vilnius University, Muitines Str. 8, Kaunas, Lithuania
[2] Center of Information Systems Design Technologies, Department of Information Systems, Kaunas University of Technology, Studentu Str. 50-313a, Kaunas, Lithuania

konstantinas.korovkinas@knf.vu.lt, paulius.danenas@ktu.lt, gintautas4garsva@gmail.com

Abstract. The goal of this paper is to propose a hybrid technique to improve Support Vector Machines classification accuracy using training data sampling and hyperparameter tuning. The proposed technique applies clustering to select training data and parameter tuning to optimize classifier effectiveness. The paper reports that better results were obtained using our proposed method in all experiments, compared to results of method presented in our previous work.

Keywords: SVM, k-Means, sentiment analysis

1 Introduction

Emergence of social networks and spread of Internet-connected smart devices was followed by explosion in data available for collection and processing, which offered serious technological and computational challenges together with new attractive possibilities in research, adoption and application of new and existing data science and machine learning techniques. It soon became obvious that novel techniques are required to effectively adopt and apply existing approaches. Support Vector Machines (abbr. as SVM) is one of the most widely used techniques which has proved its efficiency in different tasks and domains. It is very flexible to parameter tuning, as well as internal modifications, which allows to improve its performance and accuracy. Pang et al. (2002), Amolik et al. (2016), Tripathy et al. (2016) evaluated the different machine learning algorithms like Support Vector Machines, Naïve Bayes and Maximum Entropy for movie reviews sentiment classification tasks and obtained best accuracy with SVM. Go et al. (2009), Kolchyna et al. (2015), Kharde and Sonawane (2016), Hamoud et al. (2018) also proved

that the single SVM or SVM as a part of ensemble method perform the best by automatically classify the sentiment of Twitter messages. Korovkinas et al. (2017) used SVM and combination of it with Naïve Bayes for sentiment analysis in different domains of: movie reviews, Twitter and Amazon reviews. SVM achieved the better results as the standalone method to compare with NB. Rathor et al. (2018), Haque et al. (2018) showed that SVM can produce better results than other methods in sentiment analysis on Amazon product reviews. Liu and Lee (2018) reported SVM algorithm being the best option for Email sentiment classification. Al-Smadi et al. (2018) also reported that the SVM approach outperformed deep RNN approach in aspect-based sentiment analysis of Arabic hotels' reviews. Medhat et al. (2014), Ahmad et al. (2017), Manikandan and Sivakumar (2018) concluded in their reviews that SVM is one of the most frequently used machine learning algorithm for solving sentiment classification problem.

However, despite all advantages, it is often reported to be slow in terms of training time and general performance with big data arrays. The higher number of features is, the longer computation time it requires. There have been a number of efforts to speed up SVM, and most of them focus on reduction of the training set (Lee and Mangasarian, 2001; Lei and Govindaraju, 2005; Graf et al. 2005; Nandan et al., 2014; Wang et al., 2014; Mao et al., 2016; Mourad et al., 2017). These authors conclude, that properly selected training data can improve executing time with no losing or similar accuracy. In paper Korovkinas et al. (2018) we also applied training set reduction to speed-up SVM training, with only slight decrease in accuracy.

Manual hyperparameter selection is still one of the biggest issues, related to practical SVM research and application. Recent literature still does not provide any heuristic rules or rules of thumb for this task (Steinwart and Christmann, 2008), and it is usually required to test multiple classifiers with different sets of hyperparameters to achieve satisfactory performance. Grid search is often applied to solve this problem (Chen et al., 2011; Ahmad et al., 2018); it is also often integrated into SVM related packages, such as LibSVM (Chang et al., 2011) or *scikit-learn* (Pedregosa et al., 2011), to simplify research pipelines. Multiple attempts to tackle hyperparameter optimization problem can be identified in the literature, particularly using simulated annealing (Boardman and Trappenberg, 2006), its adoption to grid search (Jimenez et al., 2009), evolutionary techniques (Wu et al., 2007; Friedrichs and Igel, 2005), particle swarm optimization (Li-xia et al., 2011; Yongqi, 2012; Garšva and Danėnas, 2014), firefly algorithm (Chao and Horng, 2015). Other works focus on combined selection of both features and hyperparameters (Maali and Al-Jumaily, 2012; Yao et al., 2009; Sunkad, 2016). Osman et al. (2017) showed empirically that optimized hyperparameters significantly improved performance for k-nearest neighbours algorithm while prediction accuracy of support vector machines either improved or was at least retained.

k-Means (MacQueen, 1967) is one of the most popular and widely known techniques, used as standalone technique or in combination with others. Gu and Han (2013) proposed Clustered Support Vector Machine (CSVM) method, using k-Means for data dividing in clusters, in each to train linear SVM. Yao et al. (2013) used k-Means clustering algorithm to select the most informative samples into small subset from original training set for SVM training. Kurasova et al. (2014) presented an overview of techniques used for big data clustering and also identified k-means as one of the most popu-

lar and efficient techniques. Gan et al. (2017) used k-Means to construct a pre-selection scheme, which obtains a subset of important instances as training set for SVM. They reported that proposed KA-SVM has the outstanding performance on both of classification accuracy and computation efficiency. Wang et al. (2018) improved the spam filtering speed and filtering accuracy using a fast content-based spam filtering algorithm with fuzzy-SVM and k-Means. k-Means was used to compress data with retain most of the effective information. The conceptual simplicity and efficiency helped us to choose it for evaluation in instance selection step of our approach.

The main goal of this paper is to present technique to increase accuracy of method presented in previous paper Korovkinas et al. (2018) and evaluate it for textual data sentiment classification. The rest of the paper is organized as follows. Section 2 describes SVM and k-Means clustering algorithms which were used in the experiment. In section 3, our method is introduced, whereas Section 4 gives a description of datasets and experimental settings used to evaluate proposed approach, together with results obtained during experimenting. Finally, Section 5 outlines the conclusions and sets guidelines for future work.

2 Relevant algorithms

This section describes techniques which are relevant to research, presented in this paper, such as Support Vector Machines (Cortes and Vapnik, 1995) and its highly optimized implementation in LibLinear library (Fan et al., 2008), as well as k-Means (MacQueen, 1967) technique.

2.1 Support Vector Machines

Support Vector Machines (abbr. as SVM) were initially introduced in (Boser et al., 1992; Cortes and Vapnik, 1995). Linear support vector machines (SVM) is originally formulated for binary classification. Given training data and its corresponding labels (x_n, y_n), $n = 1, \ldots, N$, $x_n \in \mathbb{R}^D$, $t_n \in \{-1, +1\}$, SVMs learning consists of the following constrained optimization:

$$\begin{gathered} \min_{w,\xi_n} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n \\ \textit{s.t. } w^T x_n t_n \geq 1 - \xi_n \ \ \forall n \\ \xi_n \geq 0 \ \ \forall n \end{gathered} \tag{1}$$

where w is vector variable, C determines the trade-off between the maximum margin and the minimum classification error, ξ_n are slack variables which penalizes data points which violate the margin requirements. Note that we can include the bias by augment all data vectors x_n with a scalar value of 1. The corresponding unconstrained optimization problem is the following:

$$\min_{w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - w^T x_n t_n, 0) \tag{2}$$

The objective of Eq. 2 is known as the primal form problem of L1-SVM, with the standard hinge loss. Since L1-SVM is not differentiable, a popular variation is known as the L2-SVM which minimizes the squared hinge loss:

$$\min_w \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - w^T x_n t_n, 0)^2 \tag{3}$$

L2-SVM is differentiable and imposes a bigger (quadratic vs. linear) loss for points which violate the margin. Class label for test instance x is predicted using:

$$\arg\max_t (w^T x) t \tag{4}$$

(Tang, 2013)

2.2 k-Means

k-Means (MacQueen, 1967) is one of the oldest and widely research clustering algorithm. It is often preferred due to its simplicity and generally very fast performance. The main idea is to partition the input dataset into *k* clusters, represented by adaptively-changing centroids (also called cluster centers); they are initialized using so-called seed-points. k-Means computes the squared distances between the input data points and centroids, and assigns inputs to the nearest centroid. Formally, to solve problem of clustering *N* input data points $x_1, x_2, \ldots, x_N$ into *k* disjoint subsets C_i, *i* = 1,...,*k*, each containing n_i data points, $0 < n_i < N$, the following mean-square-error (MSE) cost-function is minimized:

$$J_{MSE} = \sum_{i=1}^{k} \sum_{x_t \in C_i} \|x_t - c_i\|^2 \tag{5}$$

x_t is a vector representing the *t*-th data point in the cluster C_i and c_i is the geometric centroid of the cluster C_i. Finally, this algorithm seeks to minimize J_{MSE}, where $\|x_t - c_i\|^2$ is a chosen distance measurement between data point x_t and the cluster centre c_i. An input data point x_t is assigned to cluster i if it satisfies the following condition:

$$I(x_t, i) = \left\{ \begin{array}{l} 1 \text{ if } i = \arg\min(\|x_t - c_j\|^2) \;\; j = 1, \ldots, k \\ 0 \text{ otherwise} \end{array} \right\} \tag{6}$$

Cluster centers $c_1, c_2, c_j, \ldots, c_k$ can be obtained with the following steps (Žalik, 2008):

Step 1: Initialize *k* cluster centres $c_1, c_2, \ldots, c_k$ by some initial values called seed-points, using random sampling.
For each input data point x_t and all *k* clusters, repeat steps 2 and 3 until all centres converge.
Step 2: Calculate cluster membership function $I(x_t, i)$ by Eq. (6) and decide the membership of each input data point in one of the *k* clusters whose cluster centre is closest to that point.
Step 3: For all *k* cluster centres, set c_i to be the centre of mass of all points in cluster C_i.

3 The proposed technique

The proposed hybrid method (abbr for further use – *kmLSVM*) contains three techniques: number of cluster selection, training data selection (denoted as *kMeans-Part*) and SVM hyperparameter selection. The main goal is to select representative training dataset and penalty parameter of the error term (*C*) for SVM training to increase accuracy of method presented in Korovkinas et al. (2018). For training dataset selection is used k-Means clustering algorithm. In experimental settings, it is assumed that the testing subset is 30%, therefore, the training data should be 70%. "Results" is the final results set with the following classified sentiment: "positive" or "negative". Data was converted to a matrix of TF-IDF (term frequency - inverse document frequency) features. The diagram and algorithm of the proposed method are presented in Fig. 1. Dashed line in figure means extra calculations steps, which are needed to be executed before the concrete step of main algorithm is joined with.

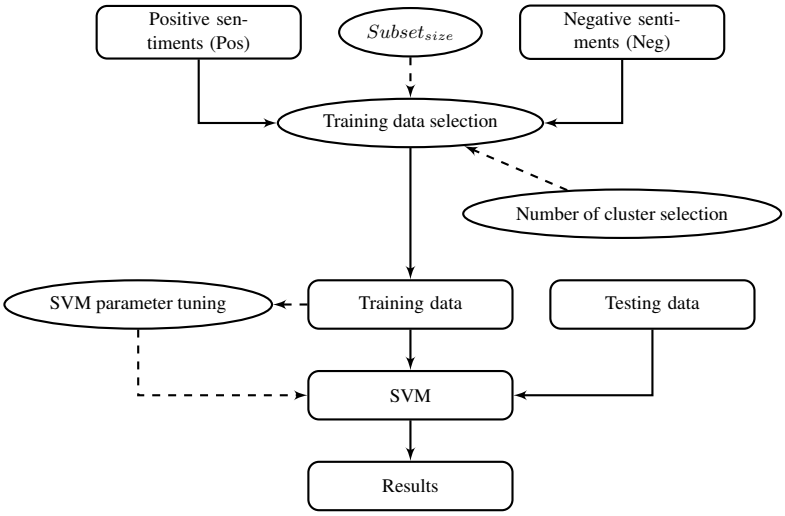


Fig. 1: Proposed method

Further, detailed formalization of the algorithm is presented. Here, function *Performance(function(.))* denotes performance obtained after running particular function; hence, the main goal is to find the optimal cluster configuration which would be optimal in terms of time and separation. Visual inspection was applied for selection in this experiment, although, other measures could be applied as well.

D – set of data;
D_{size} – dataset size;
$D_{results}$ – set of results from *SelectTrainData* function;

k_{opt} – optimal number of clusters;
cluster_range – max number of clusters;
$Subset_{size}$ – size of testing data subset is divided into;
$k\text{-}Means_{res}$ – set of results after k-Means algorithm;
Pos_{train} – set of positive sentiments;
Neg_{train} – set of negative sentiments;
Pos_{data} – set of selected sentiments from Pos_{train} set;
Neg_{data} – set of selected sentiments from Neg_{train} set;
Train – set of training data;
Test – set of testing data;
$Sample_{count}$ – number of samples to randomly select;
C_opt – optimal value for SVM penalty parameter of the error term;
Effect – effectiveness of SVM classification (see subsec 4.4).

1. Select optimal number of clusters:

   return $k_{opt} \leftarrow \arg\max(Performance(kmeans(k))), \;\; k \in cluster\_range$

2. Training data selection:

   SelectTrainData(D, $\mathbf{D_{size}}, \mathbf{Subset_{size}}$):
   $D_{results} \leftarrow \{\}$
   $n \leftarrow 0$
   $m \leftarrow Subset_{size}$
   while$(len(D_{results}) <= D_{size})$:
   $EvaluateKMeans(k_{opt}, random.sample(D, Subset_{size}))$

$$k\text{-}Means_{res} = \begin{cases} val1, \text{ value with } MAX \text{ distance to cluster center} \\ val2, \text{ value with } MIN \text{ distance to cluster center} \\ val3, \text{ value closest to } MEAN \text{ distance to cluster center} \end{cases}$$

   $D_{results} \leftarrow D_{results} \cup k\text{-}Means_{res}$
   $n \leftarrow m + 1$
   $m \leftarrow (n + Subset_{size}) - 1$
   if $m >= len(D)$:
   $n \leftarrow 0$
   $m \leftarrow Subset_{size}$
   return $D_{results}$

3. SVM parameter tuning:

   TrainClassifier(D, $\mathbf{C_{min}}, \mathbf{C_{max}}$):
   $Neg_{data} \leftarrow random.sample(Neg_{train}, Sample_{count})$
   $Pos_{data} \leftarrow random.sample(Pos_{train}, Sample_{count})$
   $train_{neg}, test_{neg} \leftarrow train\_test\_split(Neg_{data}, test\_size)$
   $train_{pos}, test_{pos} \leftarrow train\_test\_split(Pos_{data}, test\_size)$
   $Train \leftarrow train_{pos} \cup train_{neg}$

$Test \leftarrow test_{pos} \cup test_{neg}$
$C_{opt} \leftarrow \arg\max(SVM_{ACC}(C)),\ \ C_{min} <= C <= C_{max}$
$SVM_{opt} \leftarrow trainSVM(C_{opt})$
$Effect, sentiment \leftarrow predict(SVM_{opt})$
return *Effect*, *sentiment*

Output: set of SVM classification results *Results* = {*Effect, sentiment*}

4 Experiments and results

4.1 Dataset

In this paper are used two existing datasets: The Stanford Twitter sentiment corpus (sentiment140[3]) dataset and Amazon customer reviews dataset[4]. The Stanford Twitter sentiment corpus dataset is introduced by Go et al. (2009) and contains 1.6 million tweets automatically labeled as positive or negative based on emotions. The dataset is split to 70% (1.12M tweets) for training and 30% (480K tweets) for testing. Amazon customer reviews dataset contains 4 million reviews and star ratings; it was also split to select 70% (2.8M reviews) entries for training and 30% (1.2M reviews) for testing. Training and testing data has been cleaned and preprocessed before passing it as the input for training classifier. Preprocessing pipeline included removing redundant tokens such as hashtag symbols @, numbers, *http* for links, punctuation symbols, empty strings, etc.

4.2 Experiments

The main goal of this research is to compare our proposed technique with results of presented method in Korovkinas et al. (2018). Two experiments are performed in this paper: one experiment with the Stanford Twitter sentiment corpus dataset (sentiment140) and second experiment with Amazon customer reviews dataset (Amazon reviews). We reuse technique presented in Korovkinas et al. (2018) (abbr. for further use *Subset30K*) with testing dataset divided into subsets, which contain 30K rows of a dataset and applied our proposed technique (see Section 3) on it. Testing data for first experiment contains (480K tweets) and for second experiment (1.2M reviews). For *kmLSVM* training data is selected using proposed method (see Section 3) and contain 70K rows of sentiment140 training dataset for the first experiment and 70K of Amazon reviews training dataset for the second. For *Subset30K* training data (70K rows of training datasets) is selected randomly (Korovkinas et al. 2018). Table 1 shows detailed experimental settings for methods (see subsec 4.3). For more detailed results to see the impact of each part of our proposed method we perform set of experiments with the only *kMeans-Part* by using SVM parameters as they are in LinearSVC module (the part of *scikit-learn* package (Pedregosa et al., 2011)) and after the set of experiments with the whole method *kmLSVM* including "SVM parameter tuning" part. The results are compared with *Subset30K*.

[3] http://help.sentiment140.com/

[4] https://www.kaggle.com/bittlingmayer/amazonreviews/

4.3 Experimental settings

Table 1: Experimental settings (Korovkinas et al., 2018)

Exp. No.	Dataset	Testing data size (TDs)	Subset size (Ss)	Subsets quantity (SQ) trunc(TDs/Ss)	Remainder TDs-(Ss*SQ)	Calculated training data dependently on Ss
1	sentiment140	480K	30K	16	0	70K
2	Amazon reviews	1.2M	30K	40	0	70K

Python programming language was used to implement and evaluate the proposed technique. For SVM classification was used LinearSVC module, implemented in terms of LibLinear (A Library for Large Linear Classification [5]), considering its flexibility in parameter tuning and better fitting to large numbers of samples (Pedregosa et al., 2011). Simple iterative search was applied to select C parameter in range [1, 10]. k-Means clustering are implemented by using Kmeans module from *scikit-learn* package. k-Means were run three times with random initialization and different seeds, with best output selected as final result; the number of iterations was set to 100. Data was converted to a matrix of TF-IDF features, before passed to SVM and k-Means algorithms. To get more various dataset for training data, stopwords were removed before was passed to k-Means input. Initialize *k* cluster centres are defined using random sampling (default parameter of KMeans module). Workstation with processor Intel(R) Core(TM) i7-4712MQ CPU @ 2.30 GHz and 16.00 GB installed memory (RAM) was used to run experiments.

4.4 Effectiveness

Effectiveness is measured using statistical measures which are used often for similar tasks, particularly, accuracy, precision, recall and $F_1 score$. Formulas are presented below (Sammut and Webb, 2011):

Accuracy: $ACC = \frac{TP + TN}{TP + TN + FP + FN}$

Precision. Positive predictive value: $PPV = \frac{TP}{TP + FP}$

Precision. Negative predictive value: $NPV = \frac{TN}{TN + FN}$

Recall. True positive rate: $TPR = \frac{TP}{TP + FN}$

Recall. True negative rate: $TNR = \frac{TN}{TN + FP}$

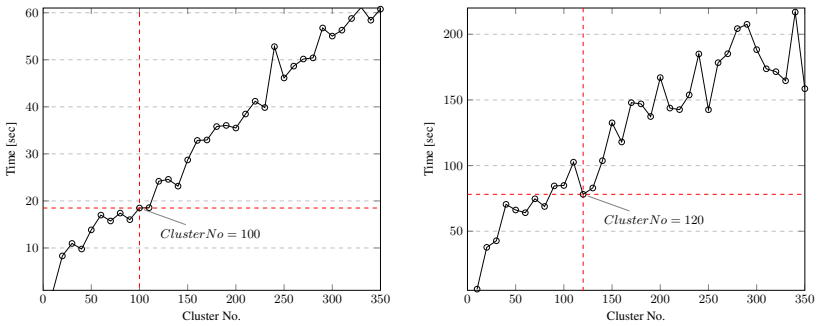
[5] `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`

Harmonic mean of PPV and TPR: $F_1 score = \frac{2}{\frac{1}{PPV} + \frac{1}{TPR}}$

where *TP* – count of correctly classified "positive" sentiments, *TN* – count of correctly classified "negative" sentiments. *FP* – count of incorrectly classified "positive" sentiments. *FN* – count of incorrectly classified "negative" sentiments.

4.5 Results

Two experiments were performed to evaluate the effectiveness of proposed technique in terms of accuracy, precision, recall and $F_1 score$ (see subsec 4.4). Fig.2 presents clustering step results. It was assumed that optimal number for clusters will be selected dependently on k-means execution time (for sentiment140 max time 20 sec. and for Amazon reviews max time 100 sec.) and number of clusters should be closest to 100. Visual output identified that optimal number of cluster for sentiment140 dataset (Fig. 2a) is 100 and for Amazon reviews – 120 (Fig. 2b).



(a) sentiment140 (b) Amazon reviews

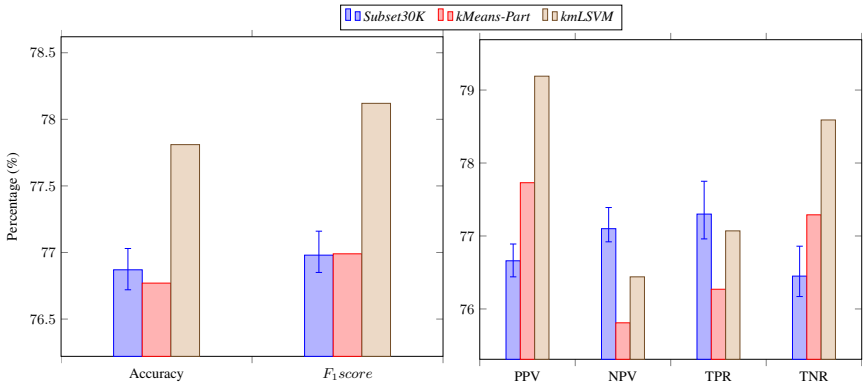
Fig. 2: Clustering step results

Table 2 shows averaged results for sentiment140 dataset, obtained using *kMeans-Part* and *kmLSVM* techniques; they are compared to results obtained with 30K rows subset (*Subset30K*), generated according to Korovkinas et al. (2018).

Table 2: Results of the proposed method applied on the sentiment140 dataset

Method	Function	ACC	PPV	NPV	TPR	TNR	$F_1 score$
Subset30K	MIN	76,72%	76,44%	76,92%	76,96%	76,17%	76,85%
(Korovkinas et al.,	MAX	77,03%	76,89%	**77,39%**	**77,75%**	76,86%	77,16%
2018)	AVG	76,87%	76,66%	77,10%	77,30%	76,45%	76,98%
kMeans-Part	–	76,77%	77,73%	75,81%	76,27%	77,29%	76,99%
kmLSVM	–	**77,81%**	**79,19%**	76,44%	77,07%	**78,59%**	**78,12%**

Average accuracy (ACC) of *Subset30K* is 76,87%, which is slightly higher than *kMeans-Part* (76,77%), but lower than *kmLSVM* (77,81%). Moreover, the results of *kMeans-Part* and *kmLSVM* are higher in terms of *PPV, TNR,* F_1score are higher, compared to *Subset30K* average, which indicate better identification of *positive* and *negative* classes. Results are visually depicted in Fig. 3.



(a) Accuracy and F_1score (b) PPV, NPV, TPR, TNR

Fig. 3: sentiment140 results

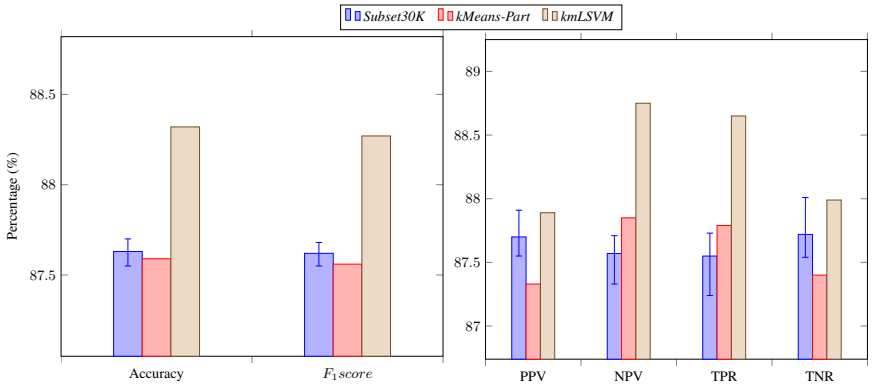
Table 3 shows averaged results of *Subset30K*, *kMeans-Part* and *kmLSVM* for Amazon reviews dataset.

Table 3: Results of the proposed method applied on Amazon customer reviews dataset

Method	Function	ACC	PPV	NPV	TPR	TNR	F_1score
Subset30K	MIN	87,55%	87,55%	87,33%	87,24%	87,54%	87,55%
(Korovkinas et al.,	MAX	87,70%	**87,91%**	87,71%	87,73%	**88,01%**	87,68%
2018)	AVG	87,63%	87,70%	87,57%	87,55%	87,72%	87,62%
kMeans-Part	–	87,59%	87,33%	87,85%	87,79%	87,40%	87,56%
kmLSVM	–	**88,32%**	87,89%	**88,75%**	**88,65%**	87,99%	**88,27%**

The average accuracy of *Subset30K* is 87,63%, which is slightly higher than *kMeans-Part*, but however, it was outperformed by our proposed *kmLSVM* (88,32%). Moreover, the results of *kmLSVM* in terms of *PPV, NPV, TPR, TNR,* F_1score are higher than *Subset30K* as well; *kMeans-Part* resulted in higher *NPV*(0.28%) and *TPR*(0.24%). Again, these results are visualized in Fig. 4.

However, the main goal of this research was to increase effectiveness of our method proposed in previous paper (Korovkinas et al. 2018), comparison was done only between them, to show effectiveness of *kmLSVM*. In this case its rather difficult to directly



(a) Accuracy and $F_1 score$ (b) PPV, NPV, TPR, TNR

Fig. 4: Amazon reviews results

compared obtained results to results obtained by other authors, considering heterogeneity of testing platforms, subtleties in implementations or their configurations, etc. Experimental alignment of our implementation to be tested with similar approaches is among our future works.

5 Conclusions and future work

This paper proposes the method to improve SVM classification accuracy by subsetting training data using clustering. The experimental results show that our method is characterized by higher accuracy than the method presented in our previous paper Korovkinas et al. (2018). The main advantage of introduced method, compared with aforementioned method, is the training data selection. Training data for *Subset30K* is selected randomly, which might negatively affect accuracy in different runs; therefore, multiple runs are required for more objective results. In this paper we advocate the use of clustering-based instance selection method (*kMeans-Part*) using data points with MAX, MIN and AVG distances to each cluster center. Results, obtained with *kMeans-Part*, are comparable with effectiveness of *Subset30K*(Korovkinas et al., 2018), with slightly lower accuracy, but higher *PPV, TNR,* $F_1 score$ values. Therefore, effectiveness of the whole technique (*kmLSVM*) are higher than *Subset30K* in both terms of average accuracy and other evaluated metrics for both sentiment140 and Amazon reviews datasets, which is considered as a positive and significant step towards more efficient sentiment analysis and overall SVM-based classification techniques.

There are several directions to work on increasing the classification accuracy of proposed method. Advanced feature engineering techniques might have significant impact on classifier effectiveness. Moreover, it would benefit of more extensive application of natural language techniques, including part-of-speech (POS) tagging, named entity recognition, lemmatization, abbreviation resolution, relation extraction, etc. Aspect-based sentiment analysis is also one of the fields, which could make use of proposed

techniques. Hence, thorough investigation of novel approaches in the context of our proposed techniques are among our future works.

References

Ahmad, M., Aftab, S., Muhammad, S.S., Ahmad, S. (2017). Machine Learning Techniques for Sentiment Analysis: A Review. *Int. J. Multidiscip. Sci. Eng*, 8(3), 27–32.

Ahmad, M., Aftab, S., Bashir, M.S., Hameed, N., Ali, I., Nawaz, Z. (2018). SVM Optimization for Sentiment Analysis. *Int. J. Adv. Comput. Sci. Appl*, 9(4).

Al Hamoud, A., Alwehaibi, A., Roy, K., Bikdash, M. (2018). Classifying Political Tweets Using Naïve Bayes and Support Vector Machines. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 736–744). Springer, Cham.

Al-Smadi, M., Qawasmeh, O., Al-Ayyoub, M., Jararweh, Y., Gupta, B. (2018). Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews. *Journal of computational science*, 27, 386–393.

Amolik, A., Jivane, N., Bhandari, M., Venkatesan, M. (2016). Twitter sentiment analysis of movie reviews using machine learning techniques. *International Journal of Engineering and Technology*, 7(6), 1–7.

Boardman, M., Trappenberg, T. A Heuristic for Free Parameter Optimization with Support Vector Machines. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, BC, 2006, pp. 610–617.

Boser, B.E., Guyon, I.M., Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152.

Chang, C.C., Lin, C.J. (2011). LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.* 2(3), Article 27.

Chao, Ch.F., Horng, M.H. (2015). The construction of support vector machine classifier using the firefly algorithm. *Computational Intelligence and Neuroscience*, Article 2.

Chen, M.Y. (2011). Bankruptcy prediction in firms with statistical and intelligent techniques and a comparison of evolutionary computation approaches. *Computers & Mathematics with Applications* 62(12) 4514-4524.

Cortes, C., Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273–297.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J. (2008). LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 1871–1874.

Friedrichs, F., Igel, Ch. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing* 64, 107—17.

Jiménez, A.B., Lázaro J.L., Dorronsoro, J.R. (2009). Finding optimal model parameters by deterministic and annealed focused grid search. *Neurocomputing* 72(13), 2824–2832.

Gan, J., Li, A., Lei, Q.L., Ren, H., Yang, Y. (2017). K-means based on active learning for support vector machine. In *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on* (pp. 727–731). IEEE.

Garšva, G., Danėnas, P. (2014). Particle swarm optimization for linear support vector machines based classifier selection. *Nonlinear Analysis: Modelling and Control*, 19(1), 26–42.

Go, A., Bhayani, R., Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford, 1*(2009), 12.

Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V. (2005). Parallel support vector machines: The cascade svm. In *Advances in neural information processing systems* (pp. 521–528).

Gu, Q., Han, J. (2013). Clustered support vector machines. In *Artificial Intelligence and Statistics* (pp. 307–315).

Haque, T.U., Saber, N.N., Shah, F.M. (2018). Sentiment analysis on large scale Amazon product reviews. In Innovative Research and Development (ICIRD), 2018 IEEE International Conference on (pp. 1–6). IEEE.

Kharde, V., Sonawane, P. (2016). Sentiment analysis of twitter data: a survey of techniques. arXiv preprint *arXiv:1601.06971*.

Kolchyna, O., Souza, T.T., Treleaven, P., Aste, T. (2015). Twitter sentiment analysis: Lexicon method, machine learning method and their combination. *arXiv preprint arXiv:1507.00955*.

Korovkinas, K., Danėnas, P., Garšva, G. (2017). SVM and Naïve Bayes Classification Ensemble Method for Sentiment Analysis. *Baltic Journal of Modern Computing*, 5(4), 398–409.

Korovkinas, K., Danėnas, P., Garšva, G. (2018). SVM Accuracy and Training Speed Trade-Off in Sentiment Analysis Tasks. In International Conference on Information and Software Technologies (pp. 227–239). Springer, Cham.

Kurasova, O., Marcinkevicius, V., Medvedev, V., Rapecka, A., Stefanovic, P. (2014). Strategies for big data clustering. In *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on* (pp. 740–747). IEEE.

Lee, Y.J., Mangasarian, O.L. (2001). RSVM: Reduced support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining* (pp. 1–17). Society for Industrial and Applied Mathematics.

Lei, H., Govindaraju, V. (2005). Speeding up multi-class SVM evaluation by PCA and feature selection. *Feature Selection for Data Mining*, 72.

Liu, S., Lee, I. (2018). Email Sentiment Analysis Through k-Means Labeling and Support Vector Machine Classification. *Cybernetics and Systems*, 49(3), 181–199.

Li-xia, L., Yi-qi, Z., Liu, X. (2011). Tax forecasting theory and model based on SVM optimized by PSO, *Expert Systems with Applications* 38(1) 116-120.

Maali, Y., Al-Jumaily, A. (2012). A novel partially connected cooperative parallel PSO-SVM algorithm: Study based on sleep apnea detection. *2012 IEEE Congress on Evolutionary Computation*, pp. 1-8.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281–297).

Manikandan, R., Sivakumar, R. (2018). Machine learning algorithms for text-documents classification: A review. *Machine learning*, 3(2).

Mao, X., Fu, Z., Wu, O., Hu, W. (2016). Fast kernel SVM training via support vector identification. In *Pattern Recognition (ICPR), 2016 23rd International Conference on* (pp. 1554–1559). IEEE.

Medhat, W., Hassan, A., Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4), 1093–1113.

Mourad, S., Tewfik, A., Vikalo, H. (2017). Data subset selection for efficient SVM training. In *Signal Processing Conference (EUSIPCO)*, 2017 25th European (pp. 833–837). IEEE.

Nandan, M., Khargonekar, P.P., Talathi, S.S. (2014). Fast SVM training using approximate extreme points. *The Journal of Machine Learning Research*, 15(1), 59–98.

Osman, H., Ghafari, M., Nierstrasz, O. (2017). Hyperparameter optimization to improve bug prediction accuracy. In *Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE Workshop on* (pp. 33–38). IEEE.

Pang, B., Lee, L., Vaithyanathan, S. (2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 79–86). Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), 2825–2830.

Rathor, A.S., Agarwal, A., Dimri, P. (2018). Comparative Study of Machine Learning Approaches for Amazon Reviews. *Procedia Computer Science*, 132, 1552–1561.

Sammut, C., Webb, G.I. (Eds.). (2011). Encyclopedia of machine learning. Springer Science & Business Media.

Steinwart, I., Christmann, A. (2008). Support vector machines. Springer Science+Business Media, LLC.

Sunkad, Z.A. (2016). Feature Selection and Hyperparameter Optimization of SVM for Human Activity Recognition. In *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, (pp. 104–109). IEEE.

Tang, Y. (2013). Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239.*

Tripathy, A., Agrawal, A., Rath, S.K. (2016). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications, 57*, 117–126.

Wang, S., Li, Z., Liu, C., Zhang, X., Zhang, H. (2014). Training data reduction to speed up SVM training. *Applied intelligence*, 41(2), 405–420.

Wang, S., Zhang, X., Cheng, Y., Jiang, F., Yu, W., Peng, J. (2018). A Fast Content-Based Spam Filtering Algorithm with Fuzzy-SVM and K-means. In *Big Data and Smart Computing (BigComp), 2018 IEEE International Conference on* (pp. 301–307). IEEE.

Wu, C., Tzeng, G., Goo, Y., Fang, W. (2007). A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy, *Expert Systems with Applications* 32(2): 397–408.

Yao, Q.Z., Cai, J., Zhang, J.L. (2009). Simultaneous Feature Selection and LS-SVM Parameters Optimization Algorithm Based on PSO. *2009 WRI World Congress on Computer Science and Information Engineering* 1(2) 723-727.

Yao, Y., Liu, Y., Yu, Y., Xu, H., Lv, W., Li, Z., Chen, X. (2013). K-SVM: An Effective SVM Algorithm Based on K-means Clustering. *JCP*, 8(10), 2632–2639.

Yongqi. C. (2012). LS_SVM Parameters Selection Based on Hybrid Complex Particle Swarm Optimization, *Energy Procedia*, 17, 706-710.

Žalik, K.R. (2008). An efficient k'-means clustering algorithm. *Pattern Recognition Letters*, 29(9), 1385–1391.

Received September 25, 2018 , revised November 11, 2018, accepted January 28, 2019