Selection of Intelligent Algorithms for Sentiment Classification Method Creation

Konstantinas Korovkinas
Institute of Applied Informatics, Kaunas Faculty
Vilnius University
Muitines Str. 8, Kaunas, Lithuania
konstantinas.korovkinas@knf.vu.lt

Gintautas Garšva
Institute of Applied Informatics, Kaunas Faculty
Vilnius University
Muitines Str. 8, Kaunas, Lithuania
gintautas4garsva@gmail.com

***Abstract*—The main goal of this paper is to select two single intelligent algorithms for sentiment classification method creation. We perform set of experiments to recognize positive or negative sentiment, using single intelligent methods and combination of them. It was observed that the better results were obtained by the single methods: Logistic regression, SVM and Naïve Bayes, also the combination of Logistic regression with SVM.**

***Index Terms*—Sentiment analysis, Logistic Regression, Naïve Bayes classification, Support Vector Machines, Random Forest.**

I. Introduction

Nowadays sentiment analysis is a very popular research area. A lot of works are done, but still there are no good enough method for sentiment classification. Many authors declare results of average slightly above 80%, but it is not enough if we need more accurate results.

Pang et al. in [17] evaluated the performance of Naïve Bayes, Maximum Entropy, and Support Vector Machines in the specific domain of movie reviews, obtaining that Naïve Bayes shown the worst and SVM the best results, although the differences aren't very large. Later Go et al. in [12] obtained similar results with unigrams by introducing a more novel approach to automatically classify the sentiment of Twitter messages as either positive or negative with respect to a query term. The same techniques were also used by Kharde and Sonawane in [14] to perform sentiment analysis on Twitter data, yet resulting in lower accuracy; again, SVM proved to perform best. Davidov et al. in [11] also stated that SVM and Naïve Bayes are the best techniques to classify the data and can be regarded as the baseline learning methods, by applying them for analysis based on the Twitter user defined hashtag in tweets. Tian et al. in [25] applied seven classification algorithms: J48, Random Forest, ADTree, AdaBoostM1, Bagging, Multilayer Perceptron and Naïve Bayes for imbalanced sentiment classification of Chinese product reviews. They found that their proposed method helps a Support Vector Machines (SVM) to outperform other classification methods. Singh et al. in [21] used a novel technique to predict the outcome of US presidential elections using sentiment analysis. To accomplish this task they used SVM. Jayalekshmi and Mathew in [13] proposed a system that automatically recognize the facial expression from the image and classify emotions for final decision. For classification they used SVM, Random Forest and KNN classifier. Ahmed et al. in [2] did investigation on a new approach of finding sentence level sentiment analysis using different machine learning algorithms. SVM (Support Vector Machines), Naïve Bayes and MLP (Multilayer Perceptron) were used for movie reviews sentiment analysis. Moreover they used two different classifiers of Naïve Bayes and two different types of SVM kernels to identify and analyze the difference in accuracy as well as to find the best outcome among all the experiments. Tayade et al. in [23] used sentiment analysis through machine learning to identify the targets of trolls, so as to prevent trolling before it happens. Naïve Bayes, Support Vector Machines (SVM) and Maximum Entropy (MaxEnt) classifiers have shown very promising results. Maheshwari et al. in [16] applied Support Vector Machines, Logistic Regression and Random Forests machine learners to identify the best linguistic and non-linguistic features for automatic classification of values and ethics. Pranckevičius and Marcinkevičius in [19] did experiments on short text for product-review data from Amazon in case to compare Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression classifiers implemented in Apache Spark by evaluating the classification accuracy, based on the size of training data sets, and the number of n-grams. Ahmad et al. in [1] did a review of different machine learning techniques and algorithms (Maximum Entrophy, Random Forest, SailAil Sentiment Analyzer, Multilayer Perceptron, Naïve Bayes and Support Vector Machines) which were applied by the researches on movie reviews and product reviews for the evaluation. Brito et al. in [7] presented how different hyperparameter combinations impact the resulting German word vectors and how these word representations can be part of more complex models. For prediction whether a user liked an app given a review with three different algorithms: Logistic Regression, Decision Trees and Random Forests. Ashok et al. in [4] proposed a social framework, which extracts user's reviews, comments of restaurants and points of interest such as events and locations, to personalize and rank suggestions based on user preferences. Naïve Bayes, Support Vector Machines with two different kernels (Gausian and Linear), Maximum Entropy and Random Forest have been used in this work.

Such results led to the conclusion that Logistic Regression, SVM, Naïve Bayes and Random Forest are still prominent for future research. Therefore, in this paper we perform experiments with each of them also with various combinations of two of them (depending on results of previous) to recognize positive or negative sentiment and to compare accuracy between them. The rest of the paper is organized as follows. In section II, a description of techniques used in research. In section III, presented method for combining results. In section IV, described preparation of dataset, experiments, experimental settings, effectiveness measure and results. In section V, we conclude and give tasks of our future works.

II. Description of techniques used in research

A. Logistic Regression

The logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in [0, 1]. The model has the form

$$\begin{aligned}\log\frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} &= \beta_{10}+\beta_1^T x\\ \log\frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} &= \beta_{20}+\beta_2^T x\\ &\vdots\\ \log\frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} &= \beta_{(K-1)0}+\beta_{K-1}^T x\end{aligned} \tag{1}$$

The model is specified in terms of $K-1$ log-odds or logit transformations (reflecting the constraint that the probabilities sum to one). Although the model uses the last class as the denominator in the odds-ratios, the choice of denominator is arbitrary in that the estimates are equivariant under this choice. A simple calculation shows that

$$\begin{aligned}Pr(G=k|X=x) &= \frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)},\\ &\quad k=1,\dots,K-1,\\ Pr(G=K|X=x) &= \frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)},\end{aligned} \tag{2}$$

and they clearly sum to one. To emphasize the dependence on the entire parameter set $\theta=\{\beta_{10},\beta_1^T,\dots,\beta_{(K-1)0},\beta_{K-1}^T\}$, we denote the probabilities $Pr(G=k|X=x)=p_k(x;\theta)$ (Hastie et al. in [26]).

B. Naïve Bayes Classification

A Naïve Bayes classifier is a simple probabilistic classifier based on Bayes' theorem and is particularly suited when the dimensionality of the inputs are high. In text classification, the given document is assigned a class

$$C^* = arg\max_c \; p(c|d)$$

Its underlying probability model can be described as an "independent feature model". The Naïve Bayes (NB) classifier uses the Bayes' rule Eq. (3),

$$p(c|d)=\frac{p(c)p(d|c)}{p(d)} \tag{3}$$

Where, *p(d)* plays no role in selecting C^*. To estimate the term $p(d|c)$, Naïve Bayes decomposes it by assuming the f_i*'s* are conditionally independent given *d's* class as in Eq.(4),

$$p_{NB}(c|d) := \frac{p(c)\left(\prod_{i=1}^{m} p(f_i|c)^{n_i(d)}\right)}{p(d)} \tag{4}$$

Where, *m* is the no of features and f_i is the feature vector. Consider a training method consisting of a relative-frequency estimation *p(c)* and *p* $(f_i|c)$ (Pang et al. in [17]).

In our experiments are used Multinomial Naïve Bayes, presented in [18]. It implements the Naïve Bayes algorithm for multinomially distributed data, and is one of the two classic Naïve Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y=(\theta_{y1},\dots,\theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi}=\frac{N_{yi}+\alpha}{N_y+\alpha n}$$

where $N_{yi}=\sum_{x\in T}x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y=\sum_{i=1}^{|T|}N_{yi}$ is the total count of all features for class y.

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha=1$ is called Laplace smoothing, while $\alpha<1$ is called Lidstone smoothing [18].

C. Support Vector Machines

Support vector machines were introduced by Boser et al. in [5] and basically attempt to find the best possible surface to separate positive and negative training samples. Support Vector Machines (SVMs) are supervised learning methods used for classification.

Given training vectors $x_i \in R^n$, $i=1,\dots,l$, in two classes, and an indicator vector $y \in R^l$ such that $y_i \in$ {1,-1}, $C-SVC$ (Boser et al. in [5]; Cortes and Vapnik in [9]) solves the following primal optimization problem [8].

$$\min_{w,b,\xi}\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \tag{5}$$

$$\text{subject to } \; y_i(w^T\phi(x_i)+b)\geq 1-\xi_i,$$

$$\xi_i \geq 0, \quad i = 1, \ldots, l$$

where $\phi(x_i)$ maps x_i into a higher-dimensional space and $C > 0$ is the regularization parameter. Due to the possible high dimensionality of the vector variable *w*, usually we solve the following dual problem.

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{6}$$

$$\text{subject to } \; y^T\alpha = 0,$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l$$

where $e = [1, ..., 1]^T$ is the vector of all ones, *Q* is an *l* by *l* positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j)$ is the kernel function.

After problem (6) is solved, using the primal-dual relationship, the optimal *w* satisfies.

$$w = \sum_{i=1}^{l} y_i\alpha_i\phi(x_i) \tag{7}$$

and the decision function is

$$sgn(w^T\phi(x) + b) = sgn\left(\sum_{i=1}^{l} y_i\alpha_i K(x_i, x) + b\right)$$

(Chang and Lin in [8])

D. Random Forests

Random Forests were introduced by Leo Breiman in [6] who was inspired by earlier work by Amit and Geman [3]. Random Forest is a tree-based ensemble with each tree depending on a collection of random variables. More formally, for a *p*-dimensional random vector $X = (X_1, \ldots, X_p)^T$ representing the real-valued input or predictor variables and a random variable *Y* representing the real-valued response, we assume an unknown joint distribution $P_{XY}(X, Y)$. The goal is to find a prediction function *f(X)* for predicting *Y*. The prediction function is determined by a loss function *L(Y, f(X))* and defined to minimize the expected value of the loss

$$E_{XY}(L(Y, f(X))) \tag{8}$$

where the subscripts denote expectation with respect to the joint distribution of *X* and *Y* [10].
Intuitively, *L(Y, f(X))* is a measure of how close *f(X)* is to *Y*; it penalizes values of *f(X)* that are a long way from *Y*. Typical choices of *L* are *squared error loss* $L(Y, f(X)) = (Y - f(X))^2$ for regression and *zero-one loss* for classification:

$$L(Y, f(X)) = I(Y \neq f(X)) = \begin{cases} 0 & \text{if } Y = f(X) \\ 1 & \text{otherwise.} \end{cases} \tag{9}$$

It turns out that minimizing $E_{XY}(L(Y, f(X)))$ for squared error loss gives the conditional expectation

$$f(x) = E(Y|X = x) \tag{10}$$

otherwise known as the *regression function*. In the classification situation, if the set of possible values of *Y* is denoted by $\mathcal{Y}$, minimizing $E_{XY}(L(Y, f(X)))$ for zero-one loss gives

$$f(x) = arg\max_{y\in\mathcal{Y}} P(Y = y|X = x) \tag{11}$$

otherwise known as the *Bayes rule* [10]. Ensembles construct *f* in terms of a collection of so-called "base learners" $h_1(x), \ldots, h_J(x)$ and these base learners are combined to give the "ensemble predictor" *f(x)*. In regression, the base learners are averaged

$$f(x) = \frac{1}{J}\sum_{j=1}^{J} h_j(x) \tag{12}$$

while in classification, *f(x)* is the most frequently predicted class ("voting")

$$f(x) = arg\max_{y\in\mathcal{Y}} \sum_{j=1}^{J} I(y = h_j(x)) \tag{13}$$

In Random Forests the *j*th base learner is a tree denoted $h_j(X, \Theta_j)$, where Θ_j is a collection of random variables and the Θ_j's are independent for $j = 1, \ldots, J$ (Cutler et al. in [10]).

III. The method for combining results

The method for combining results is presented in this section. Proposed method is based on our introduced method (Algorithm for sentences) in paper [15]. We modified this algorithm for using it with different machine learning algorithms. This algorithm is presented below.

Algorithm for combining results
Input: Let us denote *ML1* as the strongest classifier and *ML2* as the weakest classifier.

$R_{ML1} = \{ML1\_sent, p\}$ – set of the first algorithm results, obtained after performing machine learning algorithm *ML1* classification; *ML1_sent* – sentiment;

p – the probability of classification;

$R_{ML2} = \{ML2\_sent, v\}$ – set of the second machine learning *ML2* classification results obtained after performing *ML2*; *ML2_sent* – sentiment;

v – *ML2* results value, contains "positive" or "negative" sentiment;

$th_2 = 0.8$. The threshold value was selected by manually investigating the results;

$th_3 = \min(R_{ML1}\{p\}) + (\sigma_{R_{ML1}\{p\}} \setminus 2) - 0.01$ (used our proposed formula), where $\sigma_{R_{ML1}\{p\}}$ is the standard deviation of $R_{ML1}\{p\}$.

Algorithm for results combining:

1) Find results which are the same in both *ML1* and *ML2*. $Results = R_{ML1} \cap R_{ML2} = \{x : x \in R_{ML1}\{ML1\_sent\} \text{ and } x \in R_{ML2}\{ML2\_sent\}\}$
2) Find results which are different between *ML1* and *ML2*. $R_{ML1}\{ML1\_sent\}\Delta R_{ML2}\{ML2\_sent\}$ and $R_{ML1}\{p\} < th_2$

3) $Results = \begin{cases} Results \cup R_{ML1}, \text{if } |R_{ML1}\{p\}| < th_3 \\ Results \cup R_{ML2}, \text{if } |R_{ML1}\{p\}| \geq th_3 \end{cases}$

Output: set of classification results $Results = \{Sentence, Sentiment\}$ and Accuracy (Korovkinas et al. in [15]).

IV. Experiments and results

A. Dataset

In this paper are used two existing datasets: The Stanford Twitter sentiment corpus (sentiment140[1]) dataset and Amazon customer reviews dataset[2]. The Stanford Twitter sentiment corpus dataset is introduced by Go et al. in [12] and contains 1.6 million tweets automatically labeled as positive or negative based on emotions. The dataset is splitted into training dataset 70% (1.12M tweets) and testing dataset 30% (480K tweets). Amazon customer reviews dataset contains 4 million reviews and star ratings. The dataset is splitted into training dataset 70% (2.8M reviews) and testing dataset 30% (1.2M reviews).

Training and testing data has been preprocessed and has been cleaned before it was passed as the input of intelligent algorithm. It included removing redundant tokens such as hashtag symbols @, numbers, http for links, punctuation symbols, etc. After cleaning was performed all datasets were checked and empty strings were removed.

B. Experiments

In this paper are performed four experiments: two experiments with The Stanford Twitter sentiment corpus (sentiment140) dataset and two experiments with Amazon customer reviews dataset.

In the first and second experiments are used above described datasets, using split into 70% for training and 30% for testing, and apply them to four machine learning algorithms: Logistic Regression, Naïve Bayes classification, Support Vector Machines and Random Forest.

In the third and fourth experiments the best three machine learning algorithms are selected, depending on results of the previous experiments, for the creating various combinations of two different single methods and apply them on above described datasets.

C. Experimental settings

Data cleaning and preparing are performed with R [24]. The experiments are implemented with Python programming language and scikit-learn [18]: library for machine learning.

Machine learning algorithms are used with their default parameters. They are described below.

Logistic Regression default parameters [18]:

- *C* (Inverse of regularization strength): float, default: 1.0.
- *dual* (Dual or primal formulation): bool, default: False
- *fit_intercept* (Specifies if a constant should be added to the decision function): bool, default: True
- *intercept_scaling*: float, default 1

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

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

- *max_iter*(Maximum number of iterations taken for the solvers to converge): int, default: 100
- *multi_class*: str, default: 'ovr'. With 'ovr' a binary problem is fit for each label.
- *n_jobs* (Number of CPU cores used when parallelizing over classes if multi_class='ovr'): int, default: 1
- *penalty* (Used to specify the norm used in the penalization): str, 'l1' or 'l2', default: 'l2'
- *solver* (Algorithm to use in the optimization problem): 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga', default: 'liblinear'.
- *tol* (Tolerance for stopping criteria): float, default: 0.0001

Naïve Bayes default parameters [18]:

- *alpha* (Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing)): float, optional (default=1.0)
- *fit_prior* (Whether to learn class prior probabilities or not): boolean, optional (default=True)
- *class_prior* (Prior probabilities of the classes): array-like, size (n_classes), optional (default=None)

Support Vector Machines default parameters [18]:

- *C* (Penalty parameter C of the error term): float, optional (default=1.0)
- *kernel* (Specifies the kernel type to be used in the algorithm): string, optional (default='rbf'). We used 'linear' kernel instead.
- *loss* (Specifies the loss function): string, 'hinge' or 'squared_hinge' (default='squared_hinge'). 'squared_hinge' is the square of the hinge loss.
- *max_iter* (The maximum number of iterations to be run): int, (default=1000)
- *multi_class* (Determines the multi-class strategy if y contains more than two classes): string, 'ovr' or 'crammer_singer' (default='ovr'). 'ovr' trains n_classes one-vs-rest classifiers.
- *penalty* (Specifies the norm used in the penalization): string, 'l1' or 'l2' (default='l2')
- tol (Tolerance for stopping criteria): float, optional (default=0,0001)

Random Forest default parameters [18]:

- *n_estimators* (The number of trees in the forest): integer, optional (default=10)
- *max_features* (The number of features to consider when looking for the best split): int, float, string or None, optional (default="auto")
- *max_depth* (The maximum depth of the tree): integer or None, optional (default=None)
- *min_samples_split* (The minimum number of samples required to split an internal node): int, float, optional (default=2)
- *min_samples_leaf* (The minimum number of samples required to be at a leaf node): int, float, optional (default=1)
- *min_weight_fraction_leaf* (The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node): float, optional (default=0.0)

- *max_leaf_nodes* (Grow trees with *max_leaf_nodes* in best-first fashion. Best nodes are defined as relative reduction in impurity): int or None, optional (default=None)
- *min_impurity_decrease* (A node will be split if this split induces a decrease of the impurity greater than or equal to this value): float, optional (default=0.0)
- *bootstrap* (Whether bootstrap samples are used when building trees): boolean, optional (default=True)
- *oob_score* (Whether to use out-of-bag samples to estimate the generalization accuracy): bool (default=False)
- *n_jobs* (The number of jobs to run in parallel for both fit and predict): integer, optional (default=1)
- *verbose* (Controls the verbosity of the tree building process): int, optional (default=0)
- *warm_start* : bool, optional (default=False)
- *criterion* (The function to measure the quality of a split): string, optional (default="gini")

D. *Effectiveness*

Effectiveness is measured using statistical measures: accuracy (ACC), precision (PPV – positive predictive value and NPV – negative predictive value), recall (TPR – true positive rate and TNR – true negative rate) and F_1 (Harmonic mean of PPV and TPR). Formulas are presented below (Sammut and Webb in [20]):
Accuracy (ACC):

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

Positive predictive value (PPV):

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

Negative predictive value (NPV):

$$NPV = \frac{TN}{TN + FN}$$

True positive rate (TPR):

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

True negative rate (TNR):

$$TNR = \frac{TN}{TN + FP}$$

Harmonic mean of PPV and TPR (F_1):

$$F_1 = \frac{2}{\frac{1}{PPV} + \frac{1}{TPR}}$$

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.

E. *Results*

TABLE I contains the results of standard single machine learning algorithms with their default parameters. Results show that Logistic Regression (LR) obtained the best accuracy (ACC) in both experiments 79,67% and 90,21%. Other methods are arranged in the following order: SVM (ACC) – 79,16% and 90,00%, Naïve Bayes classification (ACC) – 76,72% and 84,18%, Random Forest (ACC) – 75,81% and 80,15%.

The better accuracy obtained when was used Amazon reviews dataset, while it significantly bigger than sentiment140 dataset. This happened because tweets are very short, contain noises, slangs, acronyms and etc.

Logistic Regression and SVM provided more uniform recognition of both classes; PPV, NPV, TPR, TNR, F_1, are almost even, compared to other methods.

Depending on results presented in TABLE I, for the further experiments were selected Logistic Regression, SVM and Naïve Bayes. Various combinations of two different single algorithms were performed in these experiments.

TABLE I
THE SINGLE METHODS EXPERIMENTS RESULTS

ML alg.	**Effectiveness (%)**					
	ACC	***PPV***	***NPV***	***TPR***	***TNR***	***F_1***
Experiment No 1						
LR	79,67	80,19	79,16	79,38	79,98	79,78
NB	76,72	73,18	80,26	78,76	74,95	75,87
SVM	79,16	79,49	78,82	78,97	79,35	79,23
RF	75,81	70,12	81,49	79,13	73,17	74,35
Experiment No 2						
LR	90,21	90,19	90,24	90,24	90,19	90,21
NB	84,18	81,46	86,89	86,14	82,42	83,74
SVM	90,00	90,03	89,98	89,98	90,03	90,01
RF	80,15	73,05	87,25	85,14	76,40	78,63

Table II shows that using proposed method (see Section III) for combination of two single methods let us to obtain the better accuracy to compare with a single method.

TABLE II
THE COMBINED METHODS EXPERIMENTS RESULTS

ML alg.	**Effectiveness (%)**					
	ACC	***PPV***	***NPV***	***TPR***	***TNR***	***F_1***
Experiment No 3						
LR-NB	79,81	79,49	80,12	80,00	79,62	79,75
SVM-NB	79,26	78,01	80,51	80,02	78,54	78,99
LR-SVM	81,83	79,98	83,69	83,06	80,69	81,49
Experiment No 4						
LR-NB	90,22	90,06	90,37	90,34	90,09	90,20
SVM-NB	89,98	89,81	90,15	90,12	89,84	89,96
LR-SVM	90,22	90,22	90,21	90,21	90,22	90,22

LR-SVM (Logistic Regression and SVM combination) shows the better accuracy (ACC) 81,83% and 90,22%, while (ACC) of other combinations are smaller: LR-NB (Logistic Regression and Naïve Bayes combination) – 79,81% and 90,22%, SVM-NB (SVM and Naïve Bayes combination) –

79,26% and 89,98%. Our introduced method also outperformed single LR algorithm in all experiments, except the fourth experiment where SVM-NB obtained accuracy (ACC) 89,98% to compare with Logistic Regression 90,21%.

Our method also provided more uniform recognition of both classes PPV, NPV, TPR, TNR, F_1.

V. Conclusions and future work

The main idea of this paper was to select two single intelligent algorithms to create a combined method for sentiment classification.

Results show that combination of two almost equal intelligent methods, which shown the best results (in our case Logistic Regression and SVM) can obtain the bigger accuracy (ACC) 81,83% and 90,22% to compare with the best results obtained single method like Logistic Regression 79,67% and 90,21%.

Combination between the strongest and the weakest method (in our case Naïve Bayes classification with accuracy (ACC) 79,81% and 90,22%) also outperform the best results obtained single method Logistic Regression.

The main advantage of methods combination is that combined method provided more uniform recognition of both classes PPV, NPV, TPR, TNR, F_1 to compare with Naïve Bayes and Random Forest.

Such results let to conclude that Logistic Regression and SVM, and combination of these methods fit the best for our further work. Our method presented in [15] can be applied with different algorithms and obtain the better classification accuracy. The goal of this approach was to test proposed method with existing datasets to be able in the future continue work with real-world data.

References

[1] M. Ahmad, S. Aftab, S. S. Muhammad, and S. Ahmad. Machine learning techniques for sentiment analysis: A review. *Int. J. Multidiscip. Sci. Eng*, 8(3):27–32, 2017.

[2] E. Ahmed, M. A. U. Sazzad, M. T. Islam, M. Azad, S. Islam, and M. H. Ali. Challenges, comparative analysis and a proposed methodology to predict sentiment from movie reviews using machine learning. In *Big Data Analytics and Computational Intelligence (ICBDAC), 2017 International Conference on*, pages 86–91. IEEE, 2017.

[3] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[4] M. Ashok, S. Rajanna, P. V. Joshi, and S. Kamath. A personalized recommender system using machine learning based sentiment analysis over social data. In *Electrical, Electronics and Computer Science (SCEECS), 2016 IEEE Students' Conference on*, pages 1–6. IEEE, 2016.

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

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] E. Brito, R. Sifa, K. Cvejoski, C. Ojeda, and C. Bauckhage. Towards german word embeddings: A use case with predictive sentiment analysis. In *Data Science–Analytics and Applications*, pages 59–62. Springer, 2017.

[8] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

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

[10] A. Cutler, D. R. Cutler, and J. R. Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.

[11] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 241–249. Association for Computational Linguistics, 2010.

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

[13] J. Jayalekshmi and T. Mathew. Facial expression recognition and emotion classification system for sentiment analysis. In *Networks & Advances in Computational Technologies (NetACT), 2017 International Conference on*, pages 1–8. IEEE, 2017.

[14] V. Kharde, P. Sonawane, et al. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971*, 2016.

[15] K. Korovkinas, P. Danėnas, and G. Garšva. Svm and naïve bayes classification ensemble method for sentiment analysis. *Baltic Journal of Modern Computing*, 5(4):398–409, 2017.

[16] T. Maheshwari, A. N. Reganti, S. Gupta, A. Jamatia, U. Kumar, B. Gambäck, and A. Das. A societal sentiment analysis: Predicting the values and ethics of individuals by analysing social media content. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 731–741, 2017.

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

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[19] T. Pranckevičius and V. Marcinkevičius. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221, 2017.

[20] C. Sammut and G. I. Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[21] P. Singh, R. S. Sawhney, and K. S. Kahlon. Forecasting the 2016 us presidential elections using sentiment analysis. In *Conference on e-Business, e-Services and e-Society*, pages 412–423. Springer, 2017.

[22] J. T. Starczewski, S. Pabiasz, N. Vladymyrska, A. Marvuglia, C. Napoli, and M. Woźniak. Self organizing maps for 3d face understanding. In *International Conference on Artificial Intelligence and Soft Computing*, pages 210–217. Springer, 2016.

[23] P. M. Tayade, S. S. Shaikh, and S. Deshmukh. To discover trolling patterns in social media: Troll filter. 2017.

[24] R. C. Team et al. R: A language and environment for statistical computing. 2015.

[25] F. Tian, F. Wu, K.-M. Chao, Q. Zheng, N. Shah, T. Lan, and J. Yue. A topic sentence-based instance transfer method for imbalanced sentiment classification of chinese product reviews. *Electronic Commerce Research and Applications*, 16:66–76, 2016.

[26] H. Trevor, T. Robert, and F. JH. The elements of statistical learning: data mining, inference, and prediction, 2009.

[27] A. Venckauskas, A. Karpavicius, R. Damaševičius, R. Marcinkevičius, J. Kapočiūte-Dzikienė, and C. Napoli. Open class authorship attribution of lithuanian internet comments using one-class classifier. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 373–382. IEEE, 2017.