VILNIUS UNIVERSITY

TOMAS PRANCKEVIČIUS

INVESTIGATION OF MULTI-CLASS CLASSIFICATION METHODS FOR TEXTUAL DATA

Doctoral Dissertation

Technological Sciences, Informatics Engineering (07 T)

Vilnius, 2017

The dissertation was written from 2011 to 2016 at Vilnius University Institute of Mathematics and Informatics.

Scientific Supervisor:

Dr. Virginijus Marcinkevičius (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

VILNIAUS UNIVERSITETAS

TOMAS PRANCKEVIČIUS

DAUGIAKLASIŲ TEKSTINIŲ DUOMENŲ KLASIFIKAVIMO METODŲ TYRIMAS

Daktaro disertacija

Technologijos mokslai, informatikos inžinerija (07T)

Vilnius, 2017

Disertacija rengta 2011–2016 metais Vilniaus universitete, Matematikos ir informatikos institute.

Mokslinis vadovas:

dr. Virginijus Marcinkevičius (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Acknowledgements

Firstly, I would like to express my sincere gratitude to my thesis supervisor Dr. Virginijus Marcinkevičius for a continuous support of my Ph.D. study and related research, for his patience, motivation, research experience, and knowledge. His guidance helped me all the time of research and writing of the thesis.

Apart from my advisor, I would like to thank Prof. Dr. Tomas Krilavičius, Prof. Dr. Rimantas Vaicekauskas, and other colleagues not only for their insightful reviews, comments, and encouragement but also for the challenging questions which incented me to widen my research from various perspectives.

I would like to express my special thanks to colleagues and friends Dr. Frank Rothhaas, Harald Fritz, Reinhard Müller, Krishna Prasad, Zigmas Bigelis for their well-rounded support, insights, and advice.

Last but not least, I would like to thank my family: parents, my brother, and grandmother, and my girlfriend for supporting me spiritually throughout writing this thesis and my life in general.

And finally, I would like to thank all the people who have been indirectly involved in the preparation of this dissertation, and somehow supported me.

Abstract

Today, a largely scalable computing environment provides a possibility of carrying out various data-intensive natural language processing and machine-learning tasks. One of them is a classification of textual data with some issues recently investigated by many data scientists. In this dissertation, big data-classification tasks will be completed by using the machine learning toolkit MLlib on the Apache Spark[1], the in-memory intensive data analytics framework. Such intensive in-memory computations open the door to classification methods that are effective in solving big-data multi-class text-classification tasks. In this thesis, a multi-class classification of *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, Logistic Regression* and *Multilayer perceptron* classifiers are experimentally examined and compared with a focus on evaluating the classification accuracy, based on the size of training datasets, and the number of *n*-grams. The proposed data feature selection such as a combination of *n*-grams, term frequency, inverse document frequency, part of speech, noise reduction, and used classifiers, determines multi-class classification problem with a higher classification accuracy. Findings indicate the optimal data feature selection that can be used in a variety of short texts, such as product-review classification within sentiment analysis. Applied data analytics frameworks are horizontally scalable in the multi-node cloud computing environment and allow us to run the mostly known classification algorithms to understand and predict the textual data that support knowledge gathering and decision-making processes. In the experiments, short texts for product-review data from Amazon[2] were analyzed.

[1] Apache Spark is a registered trademark: More: https://apache.org

[2] Amazon is a registered trademark. More: https://amazon.com

Santrauka

Nutolusių kompiuterinių skaičiavimo technologijų prieinamumas suteikia galimybes spręsti didelės apimties tekstinių duomenų klasifikavimo problemas, pavyzdžiui apdorojant natūralią kalbą mašininio mokymo kontekste. Tekstinių žinučių klasifikavimas naudojant klasifikavimo metodus yra vienas iš aktualių uždavinių, kurios sprendžia daugelis tyrėjų. Autorius šiame darbe tyrinėja nutolusių kompiuterinių technologijų poveikį klasifikavimo algoritmams ir natūralios kalbos apdorojimo metodams. Darbe atlikti eksperimentai naudojantis Apache Spark[3] technologija, kuri vykdo skaičiavimus kompiuterio operatyvioje atmintyje. Eksperimentinėje dalyje atliktas tekstinių duomenų klasifikavimas naudojant šiuos metodus: *paprastasis Bajesas, sprendimų medžiai, atraminiai vektoriai, logistinė regresija ir daugiasluoksniai perceptronai*. Šie klasifikavimo metodai eksperimentiškai ištirti ir palyginti, įvertinant tekstinių duomenų klasifikavimo tikslumą, naudojant skirtingus duomenų rinkinių dydžius, duomenų atrankos funkcijas: žodžių derinių *n*-gramas, terminų dažnius, triukšmo mažinimą ir kitas natūralios kalbos apdorojimo funkcijas. Šių duomenų atrankos funkcijų taikymas leidžia pasiekti aukštesnį duomenų klasifikavimo tikslumą taikant daugiaklasius klasifikavimo metodus. Remiantis eksperimentiniais rezultatais, darbe pasiūlytos kompleksinės duomenų atrankos funkcijos, kurios naudingos trumpų tekstų klasifikavimui. Eksperimentas atliktas naudojant duomenų analitikos paradigmą, kuri yra tinkama darbui su ypač dideliais duomenimis bei leidžia panaudoti klasikinius klasifikavimo algoritmus, siekiant pagerinti tekstiniais duomenimis paremtų sprendimų priėmimą ir žinių išgavimo procesą. Eksperimentuose buvo panaudoti Amazon[4] elektroninės prekyvietės duomenys.

[3] Apache Spark yra registruotas prekės ženklas. Plačiau: https://apache.org

[4] Amazon yra registruotas prekės ženklas. Plačiau: https://amazon.com

Contents

List of figures

List of tables

Abbreviation and acronyms

A	the output variable as the classification accuracy;
c_i	the classification attribute or class assigned to the review with a numerical rating value, where $C = \{c_1, c_2 \dots, c_i, \dots c_l\}$;
D	a set of documents $D = \{d_1, d_2 \dots, d_i, \dots d_N\}$ in the corpus, where document index $i \in [1; N]$;
D^L	the training set as the input $D^L = \{d_1^l, d_2^l \dots, d_i^l, \dots d_N^l\}$ for machine learning model;
d_i	document of data corpus D and i is the index;
M	the total number of layers in multilayer perceptron;
f	the classification function with the mapping $f : D \rightarrow C$;
fp_i	false positive classification examples;
fn_i	false negative classification examples;
N	the total number of documents in data corpus D;
n	the total number of words in the given document;
l	the total number of classes;
L	the total number of output nodes L in the artificial neural network;
K	the total number of trees $k \in \{1, \dots, K\}$;
$\mathcal{K}(x)$	the kernel function;
k^t	the size of the training context;
$P(C\|X)$	the posterior probability shows the probability that event C will occur when an event X has occurred;
$P(X\|C)$	the likelihood probability shows the probability that event X will occur when an event C occurred;
$P(C)$	the class prior probability of class C;
$P(X)$	the prior probability of predictor as feature vector X;
t_i	the term and i is the term sequence number;
t^l	the training word that is predicting the context in the given text;
tp_i	true positive classification examples;
tn_i	true negative classification examples;
T_{net}	the size of the training context for a neural network;
X	the feature vector $X = (\, x_1, x_2, \dots x_n)$ as a value and array as the input for the machine learning model corresponding to the documents;
W	the weight vector $W = (\, w_1, w_2, \dots, w_i, \dots w_n)$;
θ	the hyperparameter that belongs to the set of hyperparameters $\Theta \in \{1, \dots, \theta\}$.

Glossary

ANN (or *net*)	Artificial neural network.
API	Application programming interface.
DT	Decision Tree classification method.
Dataset A	Customers’ product-review dataset for android apps.
Dataset B	Customers’ product-review dataset for movies and TV.
GraphX	GraphX is a distributed graph processing framework on top of Apache Spark.
Hadoop	An open source in-disk computing framework that enables distributed processing of large datasets across clusters of computers using simple programming models.
HDFS	Hadoop Distributed File System is a distributed file system designed for Hadoop.
LR	Logistic Regression classification method.
ML	Machine Learning.
MLlib	A distributed machine learning framework on top of Apache Spark core.
MP	Multilayer Perceptron classification method.
NB	Naïve Bayes classification method.
n-gram	*an n*-gram is a contiguous sequence of tokens.
NLP	Natural Language Processing.
POS	Part-of-Speech Tagger.
RDD	Resilient Distributed Dataset is a fundamental data structure of Apache Spark.
RF	Random Forest classification method.
Spark	An open source very fast and general engine for big data processing, with built-in modules for the stream, machine learning and graph processing.
SVM	Support Vector Machine classification method.
TF	This is a feature that calculates the Term-Frequency (counting the frequency of words).
YARN	A cluster management technology (Yet Another Resource Negotiator).

Chapter 1 Introduction

1.1. Research context

A largely scalable and distributable computing environment provides a possibility of carrying out various data-intensive, natural language processing, and machine-learning tasks. One of them is a multi-class text classification into predefined classes, with issues involving text classification, recently investigated by many data scientists. Text classification into predefined classes is basically recognized as a sentiment analysis that analyzes the emotional tone for the given content and by the classification task assigns the meaning of sentiment, e.g., either positive or negative. Text classification interrelates with a variety of elements and subjects which renders technical possibilities of big textual data classification, involving mathematical, statistical, data engineering, pattern recognition, machine learning, modeling, high-performance computing, and natural language processing methods and techniques. Otherwise, this is nothing else but only new forms of data analysis including knowledge gathering by using in-memory computing and computer network possibilities. It involves an integral part of intelligence and new emerging fields that contain collection and analysis of natural language data by delivering solutions for decision makers. The focus of the investigation is on comparing multi-class classifiers by evaluating the text classification accuracy, based on the size of training data, the number of *n*-grams, tokens, and other modern methods, such as a part of speech, term frequency, etc. In experiments, product-review data from *Amazon* are analyzed. Particularly, such a product-review collects useful information that might help each potential customer to decide whether to order this product or service or not. On the other hand, some product-reviews are useless, i.e., they do not provide significant information and have a negative rating because the delivery happened one day later than expected. Also, there are other types of product-reviews that provide neither useful nor useless information about the product or service.

This is a challenging issue to deal with all the data and informatics engineering must prepare the tools and modern methods, based on how to find the effective and accurate ways to work with such challenges. The thesis investigates the impact of cloud computing technology on the classification and modern natural language processing methods. The research and experiments are implemented in *Apache Spark*, i.e., the in-memory intensive computing platform managed in the cloud computing environment. This dissertation aims to propose a combination of data feature selection, and classifiers that determines a multi-class classification problem with a higher classification accuracy for large-scale short text product-review data.

The research questions are as follows:

1. What is the impact of cloud computing technology on the classification algorithms?
2. What are the features of cloud computing that will enable the execution of multi-class text classification methods?
3. What are multi-class classification algorithms useful for textual data?
4. What is an exactly theoretical background of these methods and algorithms?
5. What are the constraints of multi-class text classification methods?
6. What are the data feature selection and natural language processing techniques that help to increase the classification accuracy?

1.2. Statement of the problem

The text is a valuable source of information when it comes to knowledge gathering about online and purchase behavior or emotional influence effect on what to buy. Usually product-reviews influence the role of emotion and decision of a consumer whether to buy this product or service or not. Precisely, the attention of internet-based retailers is always focused on the tools and methods how to promote their products and increase the sales generated revenue and, at the same time, to collect the customer's feedback. Product-reviews (or online-reviews) usually consist of complex, large-scale textual data. This type of data is used for buying or selling online, e.g., in specialized online data stores, and other specific internet directories. However, some product-reviews are more helpful and influential to the customers and sellers than others, but they are not always very evident because of the big data impact. Product-reviews might also be professional, unprofessional, short or long, with psychological and behavioral or perception and judgment aspects, which indicates distinctive personality types. Several papers analyze the classification problem of product-reviews, but the authors have not proposed a multi-class classification method that determines a higher classification accuracy of large-scale textual data, using data feature selection with a combination of *n*-grams. Comparison studies usually include Naïve Bayes and Support Vector Machine, but not Logistic Regression as a comparable classification method. Also, some authors have shown that changing parameters of classification methods have a lower impact on the classification accuracy than reasonably preparing the text corpora, by applying natural language processing and data feature selection techniques.

Therefore, the goal of this dissertation is to propose a combination of data feature selection and classifiers that determines the product-review classification problem with a higher classification accuracy for large-scale product-review data. In other words, the idea is to identify and accurately assign prediction of a class to unknown product-reviews, when a training set of review data with class labels is given. New in-memory computing technology evolution capabilities open the doors for innovative ways of processing and classifying large-scale natural language data. This scientific investigation is about the impact changes in large-scale natural language data, development of in-memory data analytics frameworks, machine learning, and multi-class classification methods.

1.3. Research object

The research object is the natural language processing methods for multi-class classification, based on modern cloud computing technology solutions and data analytics frameworks.

1.4. Research aim and objectives

The aim is to propose a combination of data feature selection and classifiers that determines the multi-class classification problem with a higher classification accuracy for large-scale short text product-review data.

To accomplish the aim of the research, the following tasks were performed:

1. To investigate data-intensive technologies, multi-class classification, and natural language processing methods and techniques.
2. To compare data-intensive technologies, multi-class classification methods, and data feature selection for large-scale textual data, by performing a comparative analysis of the realized and investigated methods, using real datasets, based on multi-class classification performance criteria: *classification accuracy, precision, recall, error rate, F1-measurement.*
3. To propose data features selection that improves the multi-class classification accuracy with the given data, e.g., for short texts such as product-review messages.
4. To propose a modified workflow model, including the corresponding methods and techniques for multi-class classification, suitable to classify short-text messages more accurately.

1.5. Research methods

The whole research methodology, applied in this thesis, is mainly based on:

1. Bibliographic research of the stated research questions and objectives was used and helped to identify, select, and evaluate the research evidence relevant to these questions and objectives.

2. The analysis of the scientific, experimental and practical achievements in the fields of machine learning with textual data classification in the cloud computing technology, the use of information retrieval, organization, analysis, benchmarking and aggregation methods.
3. Quantitative and qualitative information gathering was used in the problem-solving procedures, e.g., a collection of experimental data for multi-class classification methods.
4. Constructive research procedure for producing new constructions found to offer the solution to the real-world challenges and to make some contribution to the theory of the discipline in which it can be applied.
5. Case-based and controlled experiments were used in the experimental part in this thesis. The experimental research methodology is described in Chapter 3.
6. Software development methods were used in the experimentation phase for constructing multi-class classification methods, and data feature selection techniques for large-scale textual data, based on multi-class classification performance criteria: *classification accuracy, precision, recall, error rate, F1-measurement.*

1.6. Scientific contribution of the research

The following scientific contribution is presented in the dissertation:

1. The multi-class classification methods for large-scale short text product-review data are experimentally investigated.
2. The proposed combination of *n*-grams (uni, bi, tri-gram) is effective in selecting term frequency data features, applied in *Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest, Decision Tree, Multilayer Perceptron* classifiers, and determines the multi-class classification problem with a higher classification accuracy.
3. The presented methodology for multi-class text classification tasks using the resampling of data feature selection (noise reduction, bags of words and term frequency) are suitable for datasets that contain a lot of

short texts such as product-review messages. This type of data usually is used for buying or selling online, e.g., in specialized online data stores, internet-based retailers, telecommunication, and specific internet directories.

4. Comparative analysis of cloud-based big data analytics frameworks has shown that *Apache Spark* analytics framework, used in the cloud computing technology, is suitable to scale the amount of textual data and apply a variety of classifications using machine learning algorithms in a horizontally distributable computer network.

1.7. Practical value of the research

The results of presented methodology for multi-class text classification and the proposed feature selection method can be used in a variety of large-scale textual data processing systems and tools:

1. To deal with large-scale data, select and classify negative and positive or neutral product-reviews, and promote the most accurate, positive ones that will allow us to increase the incomes when selling various products and services, to provide additional sale services, or to support a customer retention program, e.g., to detect unsatisfied customers.
2. To find the optimal structures and their values, to implement the algorithms, understand and predict the textual data, to support decision making and the knowledge gathering process in science or business.
3. To investigate large network datasets for market research, to detect antisocial online behavior, to classify text documents that are related to cybersecurity area, e.g., malicious domains, source code vulnerability, phishing identification, etc.

1.8. Statements to be defended

The following statements to be defended in the dissertation are presented:

1. Big data analytics frameworks can be successfully used in the in-memory intensive operations for machine learning, e.g., classification algorithms.
2. The proposed method of data feature selection, based on a combination of *n*-grams (uni, bi, tri-gram), term frequency allows us to achieve a higher classification accuracy with short texts such as product-review messages, when the method is used with *Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest, Decision Tree, Multilayer Perceptron*.
3. The *Logistic Regression method that has outperformed Support Vector Machine, Naïve Bayes, Random Forest, Decision Tree, and Multilayer Perceptron* classification methods with the given large-scale multi-class textual datasets that contains a proposed combination of *n*-grams (uni, bi, tri-gram) as compared to unigrams, and applied to term frequency, and noise reduction techniques such as tokenization, stop-word removal, lowercasing, and stemming.
4. The proposed multi-class classification method with a combination of *n*-grams (uni, bi, tri-gram) achieves a higher classification accuracy using short texts such as product-review data.

1.9. Approbation of the results

The main results of the dissertation were published in the following scientific publications.

Papers in the reviewed scientific journals:

[A1] Pranckevičius T. Parallel data processing services based on Cloud computing technology. Information Sciences, 73: 64–73, 2015. ISSN 1392-0561.

[A2] Pranckevičius T., Marcinkevičius V. Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification.

Baltic J. Modern Computing, Vol. 5, No. 2, 221–232, 2017. ISSN 2255-8950.

Papers in the reviewed conference proceedings:

[A3] Pranckevičius T., Marcinkevičius V. Logistic Regression and Tokenization Methods Applied for Multi-Class Text Classification. *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE),* Vilnius, 2016. ISBN: 978-1-5090-4473-3.

Summaries in other conference proceedings:

[A4] Pranckevičius T., Marcinkevičius V. Classification and visualization algorithms on cloud computing: issues and advantages. *Data analysis methods for software systems: 7th international workshop* [abstract book], Druskininkai, December 3–5, Vilnius: Vilniaus universiteto Matematikos ir informatikos institutas, 2015. ISBN 978-9986-680-58-1.

Presentations in international scientific conferences:

1. Pranckevičius T. Comparison of Naive Bayes and Random Forest classifiers on product-review data. 28th European Conference on Operational Research. Poznan, Poland. July 3–6, 2016.
2. Pranckevičius T. Cloud computing and applications based on software services. 2nd Workshop on Software Services. Timisoara, Romania. November 11–14, 2011.

Presentations in international scientific conferences hosted in Lithuania:

3. Pranckevičius T. Application of Logistic Regression with Part-of-speech Tagging for Multi-Class Text Classification. The 4th Workshop on AIEEE'16, Vilnius, Lithuania. November 10–12, 2016.
4. Pranckevičius T. Parallel data processing services based on cloud computing technology. 56th International conference: Computer Days

– 2015. Panevėžys, Lithuania. September 17–19, 2015.

5. Pranckevičius T. Classification and visualization algorithms on cloud computing: issues and advantages. 7th International Workshop: Data Analysis Methods for Software Systems. Druskininkai, Lithuania. December 3–5, 2015.
6. Pranckevičius T. Investigation of the impact of cloud computing technology on the visualization and classification algorithms. International doctoral consortium. Informatics and Informatics Engineering Education Research: Methodologies, Strategies, and Implementation. Druskininkai, Lithuania. November 30–December 4, 2011.

1.10. Outline of the dissertation

The dissertation is organized as follows: Chapter 1 is an introduction, Chapter 2 presents an overview of machine learning for multi-class classification; Chapter 3 describes the research methodology; Chapter 4 presents the results of experiments; Chapter 5 includes general conclusions; references are presented at the end of the thesis. The dissertation consists of 125 pages, 54 figures, and 29 tables.

Chapter 2 Classification using Machine Learning

Text classification is an area investigated by many data scientists, with the demand for a data classification set to continue growing in the future due to a number of reasons: firstly, analyzing customers' generated data that are related to the quality level of products and services; secondly, classification allows the investigation of global social and information networks to acquire special knowledge, derived from hundred millions of users around the globe, for detecting antisocial online behavior, antisocial users in a community, or that which act strangely or even appear dangerous [12]; thirdly, for analyzing large data networks generated in communities, including images, videos, sound and text. The main goal of classification is to identify and assign the predefined class to a selected object when the training set of objects with class labels is given. The increasing size of data capacities has overstepped the capabilities of the available computing resource. Nowadays, largely scalable computing technologies can provide capabilities for classification using machine learning [1]. Advances and issues in computing are linked to how the data processing is carried out, fast-growing datasets, and an increased interest in the research of machine learning methods used for data analysis. Today, cloud computing technologies [2] are more and more available for many different purposes, even for data processing and transformation, classification, and visualization. The in-memory intensive computing, utilized in cloud computing technologies, extends data analysis capabilities to deal with the increased amount of data by applying the classification using machine learning. Also, remote computing resources lead to an increasing demand for a new type of data processing services and data science subjects, such as knowledge extraction, data analysis, information design, interactive data visualization, descriptive statistics, etc. One of them is data classification by natural language processing applying *Naive Bayes, Random Forest, Decision Tree, Support Vector Machines,* *Logistic Regression*, and *Multilayer Perceptron*. A full overview and definitions of largely scalable computing technologies and the text classification using machine learning are provided in the sections of Chapter 2. Therefore, the purpose of this chapter is to overview text classification using machine learning, including data feature selection, popular classification methods, computing technology concepts and data analytics frameworks that can deal with a large-scale natural language processing today.

2.1. Big data

Nowadays, to reasonably classify a small amount of data is not a challenge anymore. The challenge is to find and adjust modern technologies that can run classification methods in a horizontally scalable computing environment that can simply deal with a large amount of data. Current issues are following an increasing amount of the data that humans and technology are facing. The amount of data is continuously increasing and becoming available to uncover large hidden values of big datasets that are diverse, complex, and of a massive scale [3]. The term "*big data*" is not very old and in the past decade quite often was used by information and communication technology (ICT), enterprise and science specialists. There are many definitions of big data available today, but usually, it refers to the following properties: volume (size), variety (structure), veracity, velocity (intensity), value (meaning), viability, visualization, variability, and validity. All these aspects of big data are challenges and can only be solved using certain methods and technologies. These representations of data could be expressed by a combination of *V properties* that are used as a definition to describe big data, but these properties are not absolutes [4] [5]. Fig. 1 illustrates *3^2 Vs Venn* diagrams in a hierarchical model that has three interrelated aspects established [6]. Today this diagram is mainly used to define the big data problems (see Fig. 2). Each *Venn* diagram has single *V* and it contains some attributes that can be form-independent triangle diagrams. Therefore, the meaning of big data is presented within the complexity and relation of *3Vs* attributes. The semantic meaning establishes the relationship of data, business intelligence, and statistics. In the center of this diagram, there is machine learning because learning to understand big data without computer interaction would be an impossible mission.

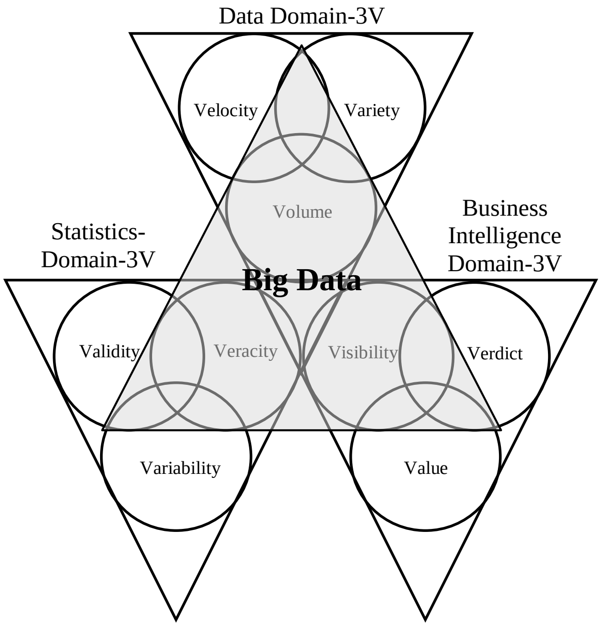


Fig. 1. 3^2 Vs Venn diagrams in the hierarchical model [6]

Below in the text, more insights and forms to define a difference in the data is presented. However, the "big data" are not defined completely because there is still a lot of discussions about what is "big". In some cases, it might be *Terabytes* ($10^{12}\,bytes$) and *Petabytes* ($10^{15}\,bytes$), even more – *Exabyte* ($10^{18}\,bytes$), *Zettabyte* ($10^{21}\,bytes$) or *Yottabyte* ($10^{24}\,bytes$). The size of data and properties are not only few meanings on how big data can be described. Understanding "big data" is not very simple, because "big data" can also be separated into various categories that are illustrated in Fig. 2. These categories help us to gather the knowledge and understanding about the structure of big data. Mainly, big data consist of four approaches: types of source, structure, data store, and data staging [3]. The presented structure is important in the aspect of dealing with big data processing when selecting the correct tools and methods. The first aspect is named as data sources. It contains social media, machine data, sensing, transactions, and internet-of-things.

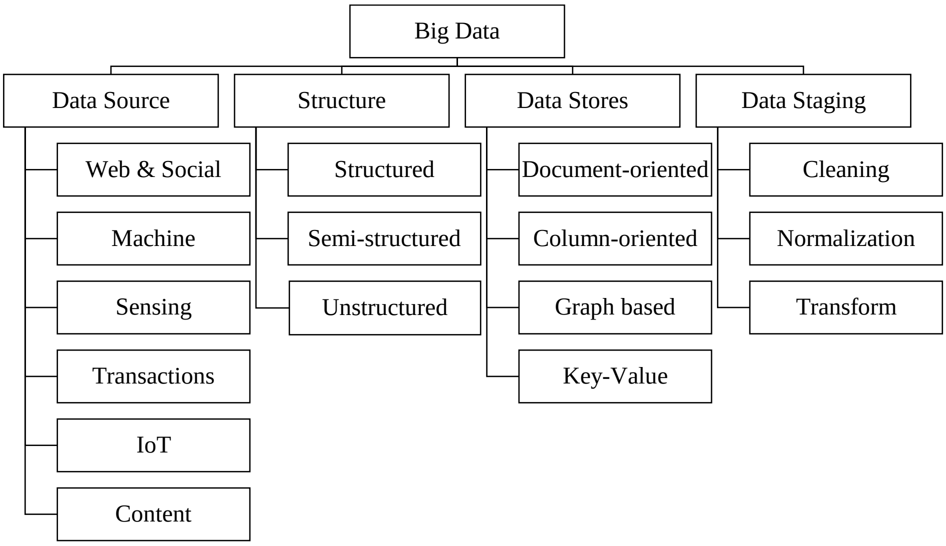


Fig. 2. Big data classification

This type of data is usually generated from sources such as virtual communities and social networks, hardware and software, financial transactions, smartphones and other mobile devices. The data structure defines the second one. The data structure consists of unstructured, semi-structured, and structured data. Structured data are that is easier to organize, e.g., input, query, store or analyze, and they are of the same format. Unstructured data are typical of textual data, video, and social media data that do not have a common or special structure format. And semi-structured data are typically more difficult to organize, because they are of a different format, and require complex rules necessary to decide further processes after capturing such a type of data. To deal with this type of data has become a very challenging task today. The third aspect is called as data stores. Data stores mainly serve as means how to store the data, e.g., document-oriented, column-oriented, graph-oriented or key-valued. Each has its attributes: a document-oriented data store supports complex data forms, retrieve documents based on the content and uses various standard file formats, e.g., *JSON*, *XML, PDF,* and *MS Word*; while the column-oriented data store keeps the content in the column way; the graph data store stores the data according to the relations one to another that is based on nodes, edges, and properties; the key-value data store is designed to store, retrieve, and manage associative arrays as the values that use specially created keys.

Transformation of data to the key-value database helps us to deal with a very large size of data. Today, the key-value database is also known as a dictionary or hash function. The fourth aspect is called data staging. It contains cleaning, transformation, and normalization. Data cleaning (or noise reduction) is the process of identifying and removing meaningless data. Transformation is the process of changing the data in a form that is meaningful to further analysis, e.g., classification or clustering. Data normalization is the process of minimizing the redundancy of data, e.g., removing duplicate items.

2.2. Classification using machine learning

Machine learning is an algorithm that is constructed to learn from the data [7]. Mitchell (1997) has created a used quote widely about machine learning (Definition 1) that says:

> "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.*" [8].

This quote includes a word "*tasks*", that is not only the task of learning itself. It simply describes that the learning process can help by its influence to accomplish the given task. For instance, if we want a car to be able to perform self-driving from *San Francisco* to *New York*, then driving is the task [7]. For that reason, machine learning utilizes computer resources to solve very difficult tasks that usually are not easy to handle for humans or other default software. If applied machine-learning algorithms can solve human problems, that means it can be used as part of artificial-intelligence and data-mining initiatives, e.g., to assign labels to a given input class or to solve many different human problems for which other tools are not capable of giving valuable results [9]. So, that is why fundamentally is known that machine learning is a very interesting scientific problem that allows humans to improve and acquire new knowledge about intelligence in general [7].

Definition 1. *Machine learning is a technique for effectively using computing resources to solve problems by understanding data through training and getting them to learn and to use unique data-processing features (representation of data) and models (representation of patterns and rules)* [10].

The machine learning system applies its parameters by some algorithms to generate the desired output patterns from a given input pattern. Pattern recognition is a part of machine learning that studies machine abilities to observe the environment including patterns and categories. A pattern can be realized as an object, process or event with its given name. Patterns can include a class that belongs to commonly shared attributes known as categories. Pattern recognition assigns a label to a given data value. The best case of pattern recognition is the problem of classification, but it might be realized to solve the several types of machine learning problems as well. Machine learning is constructed to solve many different tasks, and one of them is classification (Definition 2). This one is called a classification using machine learning (Definition 3). While solving this kind of task, the machine learning algorithm specifies which of k categories the input belongs to [7].

Definition 2. *Classification is a process of systematically grouping objects or ideas that have similarities and can be recognized by one or few attributes, e.g., by kingdom, class, size, shape, density or other classification aspects.*

Definition 3. *Classification using machine learning is a process of determining the probability, if a certain feature, appearing in a certain class, thereby allows us to predict a likely classification, based on an input containing one or more of those features* [A2].

Let the machine learning algorithm be a function f that is formalized as follows:

$$f\colon R^n \to \{1, \ldots, l\}, \tag{1}$$

where R is a set of all real numbers, n is a integer number (a dimension of vector space), and l is the total number of classes. The function assigns a valued array of input of the feature vector $X = (\, x_1, x_2, \ldots, x_i, \ldots x_n)$ to the classification attribute or a class denoted C and is formally defined:

$$C = f(X) \tag{2}$$

Also, there exist other options in classification tasks, for instance, where the function f yields the output of probability distribution over the multiple classes.

The generalized machine learning process design presents the stages how by applying machine learning, to solve the given machine learning problem, and the generalized workflow model that consists of three stages (Fig. 3.): 1) data extraction (evaluation and preparation) as the input; 2) model construction; 3) prediction and result evaluation as the output.

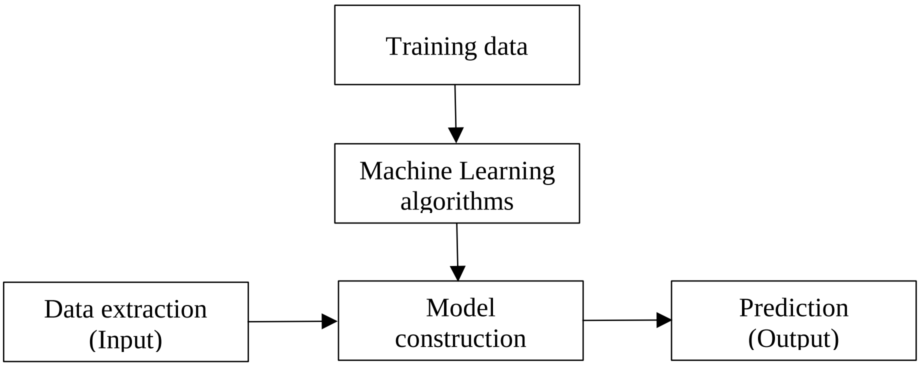


Fig. 3. Model of generalized machine learning workflow

The first and initial step is that data must be prepared before running any machine learning algorithms. It is important to analyze, evaluate, clean, and select only the necessary data, remove unnecessary data (not valuable information), and to reduce the dimensionality of data. In the model construction stage, machine learning algorithm is necessary to be selected, depending on the task formulation, e.g., data classification. Training the data by the selected classification method allows us to develop a model, that could be used to generate the output, i.e., to make predictions with the new and never used data. Big data movement through the described workflow model is the issue because transferring or extracting substantial amounts of data should be performed following the best practices, applying various methods and functions, and finally, to extract and predict only relevant and meaningful data.

2.2.1. Supervised and unsupervised learning

Machine learning can be basically categorized into two core groups: supervised, and unsupervised learning [11]. Also, there are two more so-called hybrid categories inherited from the core groups, – semi-supervised, and reinforcement learning. These machine learning categories and methods are illustrated in Fig. 4. Supervised learning (human-defined classes and labels that are used in training documents [12]) means that there is a rating or class already assigned to a data unit and usually dealing with the assigned class label, e.g., analysis of customer opinions or sentiments [13].

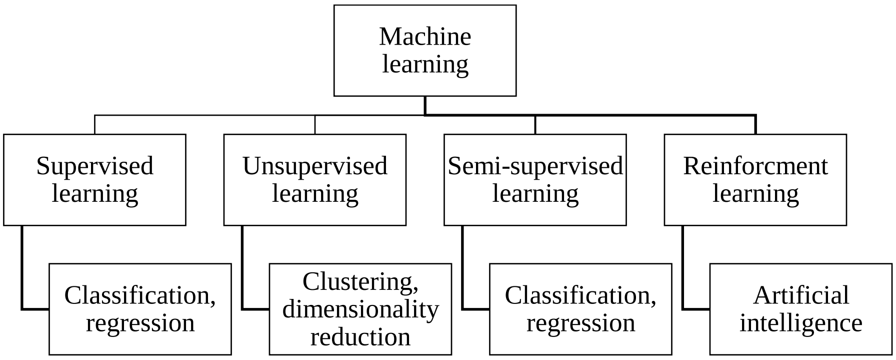


Fig. 4. Machine learning categories and problems

Let a supervised learning be S with the given training input D^L as a model is a function that produces a classifier f with the output:

$$S(D^L|\theta) \to f, \tag{3}$$

where θ is a set of hyperparameters (Definition 4), also known as learning parameters.

Definition 4. *Hyperparameters (or learning parameters) are settings used to control the learning algorithm. Classification algorithms with predicted probability distributions or confidences are also known as discriminant values or support values* [14].

These parameters are known as learning parameters, which usually deal with the settings used to control the learning algorithm. On the contrary, unsupervised learning has no rating or class assigned, and the training instances are unknown. It means there is no trainer, so the learner must build concepts by experimenting with the data and create models only by realizing the experiments.

Meanwhile, the semi-supervised learning method is a combination of supervised and unsupervised methods, and both are used in the training process. And finally, reinforcement learning is constructed to maximize the output by a feedback loop between the learning system and its experiences, which helps to train the model and learn from experience by interacting with the environment or given problem [7]. It is typically used for building artificial intelligence.

2.2.2. Text classification

Classification methods are unique data-processing features of machine learning [1] and allow us to run a multi-class text-classification. Text classification into predefined classes can be acknowledged as a sentiment or polarity analysis that indicates the emotional tone for a given content and assigns the meaning of sentiment, e.g., either positive or negative. Application of sentiment analysis can be used almost in every aspect of the modern world from products and services such as healthcare, online retail, social networks, to financial services or political elections, and other possible domains where humans leave their feedback. Organizations are usually seeking to collect consumer or public opinions about their products and services. To this end, many surveys or opinion gathering techniques and methods are conducted with a focus on targeted groups or by using any other information available. Therefore, developed concepts and techniques of informatics engineering can suggest modern solutions including sentiment analysis that explores topics such as classification using machine learning and works with collections of humans' opinions or customer feedback data expressed in short text messages, e.g., product-reviews. Text classification is an active area of investigation in machine learning that is applied in very different domains such as analysis of customers' feedback about the product quality or spam detection [15], to classify unclassified product-review data that will help the customer to decide whether to order products and services or not. The main idea of using text classification is to apply selected classification methods that can assign single (or multiple) classes to the given text object that has an unknown class. Textual data always have a relation to certain objects that represent attributes, values, and classes. One can argue that the calculated class definition is more accurate than an assigned class by a human since the assigned class is very subjective. On the other hand, the assigned class and the actual text object might not be related. A formal definition of text classification is described in Definition 5.

Definition 5. *Text classification is a process that assigns a predefined set of classes or attributes to the given text object that is the textual content of elements or attributes such as text documents, news articles, emails, tweets, customer feedbacks, product-reviews, etc.*

Today, machine learning enables an effective use of high-performance computing resources that help to solve big textual data classification problems through the training and learning process, including the use of data-processing (data-pre-processing), and application of classification methods that help to construct machine learning models. The ability to classify the textual data links to the selection of data preparation and analysis toolkits, unique features, classification methods, computing resources, and other techniques that deal with these challenges. A data classification algorithm (Definition 6) is a model that classifies by the given data (training set) and numbers of hyperparameters.

Definition 6. *The classification algorithm is a function that produces a classifier, given a training dataset and a set of hyperparameters* [16].

The theoretical description with the focus on text classification that defines the text classification issue formally is presented in the following section.

Let $d \in D$ be an instance. The instances are usually a document (textual data) object that belongs to a set of documents $D = \{d_1, d_2 \dots, d_i, \dots d_N\}$, where the document index $i \in [1; N]$, N is the total number of documents. Let c_i be a classification attribute or class assigned to the document (usually human-defined or sometimes automatically) object in the given set of documents D with a rating numerical value $C = \{c_1, c_2 \dots, c_i, \dots c_l\}$, where l is the total number of classes and i is the index. For binary classification, we have $2 > l > 0$, and for multi-class classification – $2 < l \ll \infty$. The classification function f presents how instance objects d are assigned with c and that can be formally defined as follows [12]:

$$f : D \to C \tag{4}$$

where the classifier is learning the documents from the given D^L training set, and then the classifier, as the function f, maps all instances (documents) to the attributes of the classes l. Ideally, the instance object d_i cannot be assigned to multiple attributes of the l classes, except only one single class c. But there might be scenarios that one instance (document) can belong to several classes simultaneously. Such a classification is called *multi-class*, *multi-label*, or *multi-value* classification [12]. Each classification algorithm entails a search through an implicit or explicit hypothesis space to find the preferred model structure and/or parameters [16]. Fig. 5 presents an example of classified data with the space that is broken up into regions, using horizontal and vertical lines. Examples in these regions are classified and, they share the same attributes, values or classes. The main idea to have such regions is because it allows to build up a model that predicts the target variable of a new and unseen instance by determining which segment it falls into [17]. Therefore, text classification can be structured by two approaches: topic-based classification and sentiment classification. The idea of topic-based classification is to detect the key words that are related to the topic of the classes.

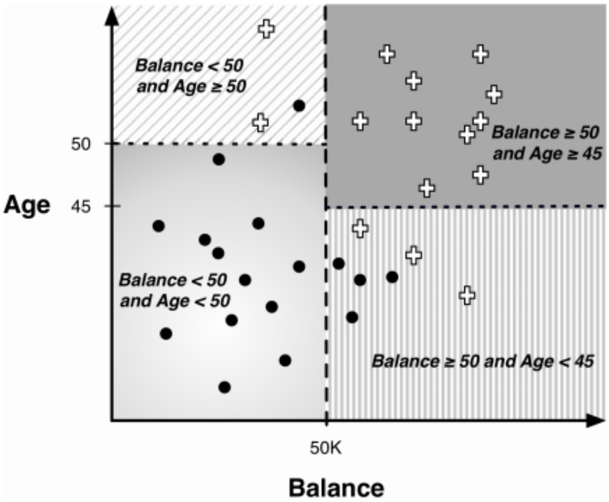


Fig. 5. An example of data classification by class

Meanwhile, the sentiment classification consists of positive and negative opinions about the words such as good, bad, excellent, normal or great, but there are more sentiment classification options available, as well.

Though, the sentiment analysis is different from the classical topic-based classification which is based on the context or given data topic, the sentiment analysis is used for understanding and extracting human feelings from data, e.g., classifying product reviews by positive and negative emotions, or classifying if e-mails are spam or not, analyzing tweet to find positive or negative emotions [18]. The sentiment classification is a special process of text classification to classify texts according to the sentimental polarities of opinions available, e.g., favorable or unfavorable, positive or negative [19]. In other words, sentiment classification helps to identify and categorize the opinions that are expressed by the human being. These expressions are usually related to the determination of the opinion about the object or product, i.e., positive or negative, or neutral.

The popularity of sentiment analysis is still rapidly increasing to analyzing large textual datasets. The rapid development of sentiment analysis enables us to make revolutionary links between words and real-world human behaviors [20].

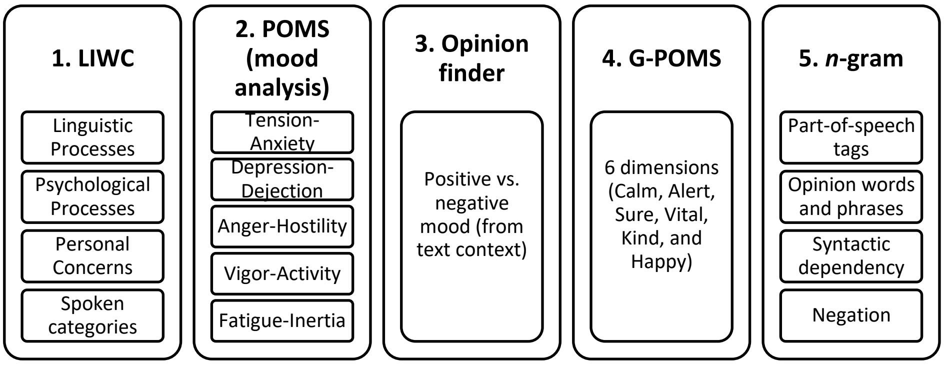


Fig. 6. Sentiment analysis categories

Thus, the sentiment analysis is categorized into five areas, presented by *J. Park* (Fig. 6): linguistic inquiry and word count (LIWC) [20], profile of mood states (POMS) [21], opinion finder, *Google* profile of mood states (G-POMS) [22], and *n*-grams. The linguistic inquiry and word count is a process and software toolkit (including a psychometrically validated internal dictionary) of the text context analysis that includes linguistic, psychological processes, personal concerns, spoken categories, and identifies emotional, cognitive, and structural components of text samples. Opinion finder is a process of the text context analysis in the given date that creates a positive vs. negative output of daily time series public mood. The profile of mood states is the process of text context analysis that provides a detailed view of changes in the public mood of different dimensions: calm, alert, sure, vital, kind, and happy. The supervised learning method (Fig. 7) can be adapted to the sentiment classification by applying classical classification methods using machine learning, such as *Naïve Bayes* [23], *Random Forest* [24], *Decision Tree* [25], *Support Vector*

Machines [26], *Logistic Regression* [27], *Multilayer Perceptron* and other classifiers.

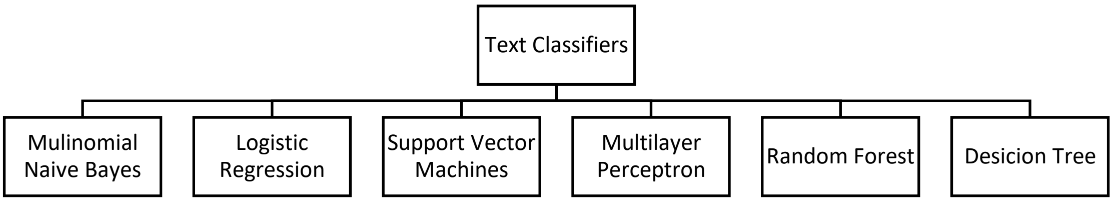


Fig. 7. Classical supervised classification methods

Pang et al. (2002) used this approach to classify movie reviews. This author has concluded that positive and negative classes are classified by using *n*-grams (unigrams) and the classification task is performed well by *Naïve Bayes* and *Support Vector Machine*, but neutral reviews were not used and it might be concluded that the classification problem was not very much challenged. Later on, the research was improved by prediction of the class review, using the so-called five-star rating system [28]. This research was considered as part of a regression problem since the review classes are ordinal. One of the mostly known sentiment classification problems is the classification closely related to the training data used in the domain. For example, the classifier performs not accurately when it was trained in one domain and applied in another. The most likely reason is that phrases and words or languages have different meanings, such as, in some cases, positive can be mixed with negative. However, *Joachim* [29], in his comparative work on the text classification with supervised machine learning, has concluded that *Support Vector Machine* is one of the best classifiers, compared to that of *Decision Tree* or *Naïve Bayes*. Other authors also demonstrated the superiority of *Support Vector Machine* over *Decision Tree*, and *Naïve Bayes* [30]. One of the earliest comparative works on text classification using the supervised machine learning methods has revealed that Support Vector Machines is the top-notch classifier, compared to *Decision Tree* or *Naïve Bayes* [31]. *Dumais et al.* [32] also demonstrated the superiority of *Support Vector Machines* over *Decision Tree* and *Naïve Bayes*. Despite the domination of *Support Vector Machines*, *Bayesian* methods maintained their popularity and are often selected as the baselines. It is necessary to mention that *Naïve Bayes* with a multinomial model is more often selected instead of a simple *Naïve Bayes* with the *Bernoulli* model because the *Naïve Bayes* method with a multinomial performs obviously better using larger feature sets [33]. Moreover, some researchers report that the multinomial *Naïve Bayes* method a can even outperform popular *Support Vector Machines* [34], [35]. But later, for future investigations, the *Support Vector Machine* method is taken by many researchers and became a most popular method for text classification tasks not only in English but also in many other languages.

2.2.3. Naïve Bayes

The *Naive Bayes* classifier is a machine learning algorithm, a probabilistic classification method very often used for classifying textual data using machine learning, e.g., sentiment analysis, recommendation analysis, spam filtering, and is intensively studied since 1950. This classification method is also known for real-time predictions, high accuracy, speed, support of large-scale and multi-dimensional data, and multi-class classification. Probabilistic classification is the learning process of the probability of the given object that has the same or certain similar attributes and belongs to a group or class. For example, let us identify orange fruit from the given set of fruits only by the color, shape, and taste. So, if the fruit is sweet citrus, and has the orange color and a spherical shape, then highly persuasively it is the orange fruit. These attributes are related to each other, and together or even individually represent these properties and contribute to the probability that this is an orange fruit. Therefore, it is called “naïve”. As to the *Bayes* part, it refers to a statistician and philosopher *Thomas Bayes*. The construction and idea of the *Naïve Bayes* method consist of knowledge of probabilities, statistics, and it can be expressed as follows:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}, \tag{5}$$

where $P(C|X)$ is a posterior probability of the class C, $P(X|C)$ is likelihood related to the given class, $P(C)$ is a class prior probability of class, $P(X)$ is a prior probability of predictor. To solve the classification problem using machine learning, there might be not only binary, but also multiple classes $C = \{c_1, c_2 \dots, c_i, \dots c_l\}$, where l is the total number of classes and i is the index. The goal is to calculate the posterior probability of an object (or class) with the feature vector $X = (x_1, x_2, \dots, x_n)$ that belongs to the class c_i,

$$P(c_i|x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n|c_i)P(c_i)}{P(x_1, x_2, \dots, x_n)}, \tag{6}$$

for $1 \leq i \leq l$. The numerator of the fraction on the right-hand side of the equation above is:

$$P(x_1, x_2, \dots, x_n|c_i)\, P(c_i) = P(x_1, x_2, \dots, x_n, c_i), \tag{7}$$

The conditional probability term $P(x_1, x_2, \dots, x_n|\, c_i)$ becomes $\prod_{j=1}^{j=n} P\left(x_j \middle| c_i\right)$, because of the assumption that features are independent. The *Bayes* theorem can be expressed, based on the following assumption:

$$P(c_i|x_1, x_2, \dots, x_n) = \left(\prod_{j=1}^{j=n} P\left(x_j \middle| c_i\right)\right) \frac{P(c_i)}{P(x_1, x_2, \dots, x_n)}, \tag{8}$$

for $1 \leq i \leq l$. The expression $P(x_1, x_2, \dots, x_n)$ is constant for all the classes, so the expression can be simplified:

$$P(c_i|x_1, x_2, \dots, x_n) = \left(\prod_{j=1}^{j=n} P\left(x_j \middle| c_i\right)\right) P(c_i), \tag{9}$$

for $1 \leq i \leq l$.

2.2.4. Support Vector Machine

Support Vector Machine as a machine learning algorithm is used to model the relationship between a categorical dependent variable and one or more explanatory variables. *Support Vector Machine* was introduced by *V. Vapnik* [36]. The main task of *Support Vector Machine* method is to find a hyperplane that can maximize the path with two vectors between two classes. In other words, to find and create a straight line (two-dimensional), a plane (three-dimensional) or a hyperplane (*n*-dimensional case *l* is greater than three), that will find the best way to distinguish objects, belonging to different classes. In this case, a hyperplane (a plane, or a line) is considered the best, if the distance from the hyperplane to the nearest objects, belonging to different classes, is equal and maximal. An example of two-dimensional objects separated by the hyperplane is presented in Fig. 8.

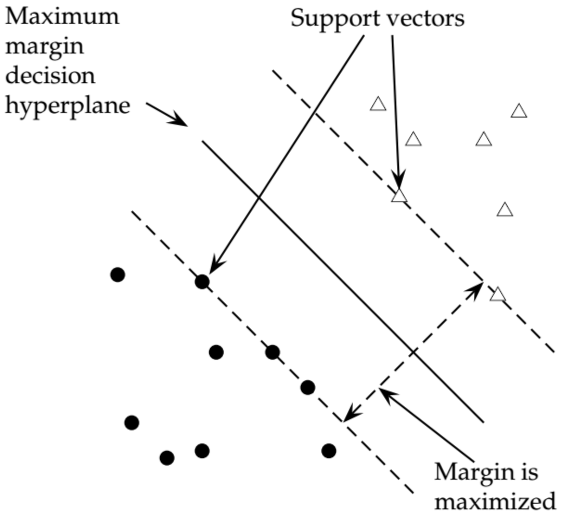


Fig. 8. Support vectors and decision hyperplane

The vectors in between the hyperplane are called support vectors [12]. If the distance from the hyperplane to the nearest objects, belonging to different classes, are larger, than the hyperplanes that separate the objects, it is considered more effective. The formal mathematical abbreviation of classes that separate a hyperplane equation can be simply defined:

$$WD^L + b = 0, \tag{10}$$

where $W = (w_1, w_2, \dots, w_i, \dots w_n)$ is the weight vector; D^L is the training set, $D^L = \{d_1^l, d_2^l \dots, d_i^l, \dots d_N^l\}$ that contains the number of d_N^l objects, and b is a constant. The most common case is to deal with two classes denoting one class as positive and the other as negative, e.g. $c_i \in \{-1;\ 1\}$.

If different classes of data can be linearly separated, then there is a need to find two hyperplanes (positive and negative):

$$WD^L + b = 1, \tag{11}$$

$$WD^L + b = -1, \tag{12}$$

that divide the data into classes so that the curves of the restricted space do not include any object from the training set. Furthermore, the distance between the hyperplane and can be represented by the formula $2/||W||$ and it must be maximized. Hence it follows the need to minimize $||W||$, e.g., the *Euclidean* norm of vector W. To facilitate this process, the decision rule $||W||$ can be substituted by a member of $||W||^2/2$. Thus, the optimization task can be expressed as follows:

$$\min_{W,b} \frac{1}{2} ||W||^2. \tag{13}$$

The hyperplane can be found by using the sequential optimization algorithm and classifies a test document by finding:

$$\sum_{x=1}^{m} \alpha_i y^{(i)} \mathcal{K}(x^{(i)}, x) + b, \tag{14}$$

where α_i and b are parameters of the optimal hyperplane, $\mathcal{K}$ is a function kernel. In the text classification using machine learning, the textual data must be transformed into the feature vectors representations by using specific mapping. Meanwhile, kernel functions can operate in a highly dimensional feature space without memorizing the coordinates of data points. The kernel function allows us to compute the required data points among all the data, located in this feature space. With such a kernel trick, it enables us to reduce the unnecessary computing performance. There are several commonly used types of kernel functions, such as linear, polynomial, and radial-based functions.

2.2.5. Random Forest

The *Decision Tree* [25] classification method is known for its ability to solve many different classification problems, and it uses constructions and structures based on the posterior probability of class. Also, the *Decision Tree* classifier is known by an extended version of *Random Forest* that is illustrated in Fig. 9.

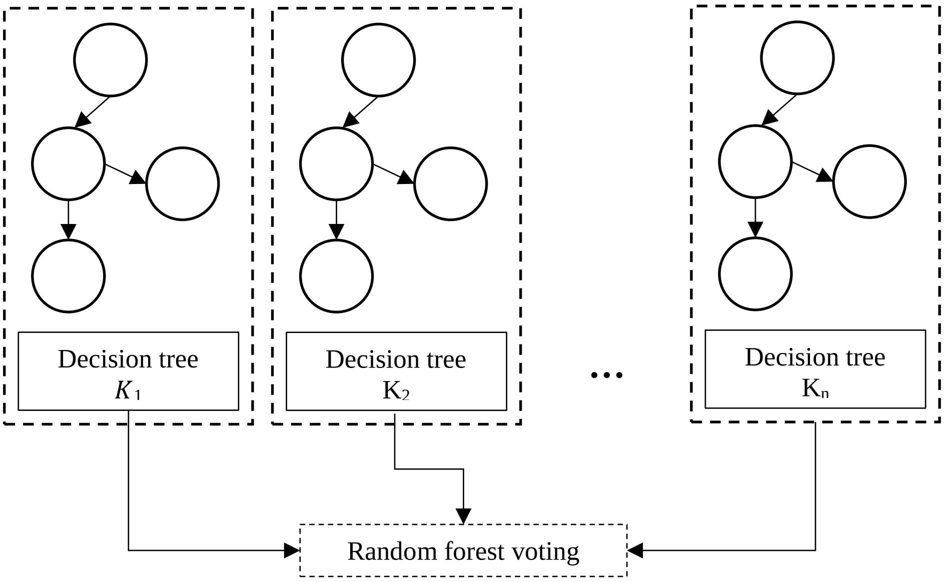


Fig. 9. Random Forest

The *Random Forest* classifier is just a collection (or a bunch) of trees that make up a forest. The difference between *Random Forest* and *Decision Tree* lies in the overfitting, e.g., when the decision tree creates a single large and deep tree. On the contrary, *Random Forest* is constructed to avoid the overfitting issue, because it creates random feature subsets and simply builds smaller trees inside, but according to the number of trees and the size of data, the training time increases. The main idea of how the classification is implemented by calculating the variable importance in the tree. In *Random*

Forest, the highest importance variable becomes the root node in that tree and this variable becomes the most important for classification. The prediction is treated as a vote for one class, and the final class is declared which has the majority of votes. The trees are trained independently and, in the testing phase, each initial node x is pushed until it gets to the corresponding point. In the end, the output of every node x in the tree must be evaluated by the vote for a class and such voting should be equal to the total number of classes available. Only then the maximum posterior probability is considered. The formal expression of posterior probability can be formulated as follows:

$$P(\mathrm{C}|\mathrm{X}) = \frac{1}{N}\sum_{n=1}^{K} P_n(C|X), \tag{15}$$

where K is the total number of trees $k \in \{1, \dots, K\}$, N is the total number of documents in data corpus D, $P(C|X)$ the posterior probability shows the probability that event C will occur when an event X has occurred, n is the total number of words in the given document.

2.2.6. Logistic Regression

Linear regression [37] is a field of statistics that studies the models for understanding the relationship between the input and output and has been recently used in the classification using machine learning. It is the model that assumes a linear relationship between the input as a feature vector X and the single output variable y.

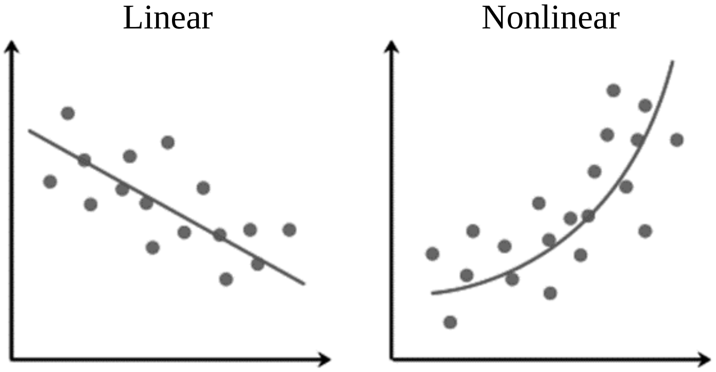


Fig. 10. Linear and nonlinear regression

An example of linear and nonlinear regression is illustrated in Fig. 10. More specifically, that single output variable y can be calculated from a linear combination of the input variables X. One of the methods commonly used for classification is *Logistic Regression* classifier. The *Logistic Regression* classifier measures the relationship between the categorical dependent variable and one or more independent variables in estimating probabilities using a logistic function, which is a cumulative logistic distribution. This method is useful for the analysis of data where few independent variables calculate the output. *Logistic Regression* works better, if the dataset is linearly separable, so small datasets are almost linearly separable and the classification accuracy yields higher results.

The logarithmic transformation log is necessary to normalize the distribution of logistic regression. The log transformation of P allows us to create a link with the normal regression equation. The logistic transformation of P is also called as $logit$ of P or $logit(P)$. $Logit(P)$ is the log (to base e) of the *odds ratio or likelihood ratio* the dependent variable of which is 1. Formally it is defined by the following mathematical equation [37]:

$$logit(P) = \log\left(\frac{P}{1-P}\right) = ln\left(\frac{p}{1-p}\right), \tag{16}$$

where p can only range from 0 to 1, the $logit(p)$ scale ranges from the negative infinity to the positive infinity and is symmetrical approximately $logit$ of 0.5 (which is zero). The formula of the logistic regression equation is [37]:

$$logit(P(C|X)) = \log\left(\frac{P(C|X)}{1-P(C|X)}\right) = a + b_1x_1 + b_2x_2 + \cdots, \tag{17}$$

where $P(C|X)$ is a posterior probability that shows the probability that event C will occur when an event X has occurred, a is the constant of the equation, b_i are the coefficients of predictor variables. Classification is implemented by creating data points onto a set of hyper-planes, the distance to which is used to determine a class membership probability. The probability P can be expressed mathematically [37]:

$$P = \frac{e^{a+b_1x_1+b_2x_2+\cdots}}{1+e^{a+b_1x_1+b_2x_2+\cdots}}, \tag{18}$$

where a is the constant of the equation, b is a coefficient of predictor variables, e is a base of natural logarithms (approximately 2,72), $P(C|X)$ is a posterior probability shows the probability that event c will occur when an event x has occurred.

Logistic Regression is a binary classification method, but the multi-class classification can also be implemented by using optional multi-class properties that implement the binary problem fitting to each label, e.g., one-vs-all.

2.2.7. Artificial Neural Network

Artificial neural networks as a learning algorithm consist of the information processing structures that inaccurately simulate some living organisms, which occurs in the brain information processing [38]. Artificial neural networks were invented in 1940. An artificial neural network is based on interactions of the smallest elements – neurons. Neurons are interconnected in various strength connectors through which signals are transmitted. Connection strength coefficients are called weights since they belong to the transferred signal modification.

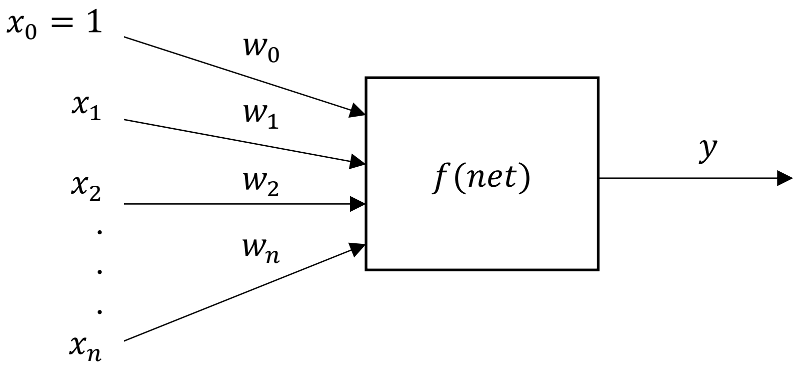


Fig. 11. An artificial neuron

The biggest benefits of neural networks are the ability to learn, adapt, and adjust. The mathematical function of a single artificial neuron (perceptron) is illustrated in Fig. 11, Neuron excitation signal is obtained by calculating the weighted sum of input signals minus the value of the threshold and f is a transfer function, y is the output value of an artificial neuron:

$$net = \sum_{i=1}^{N} w_n x_i - w_0. \qquad (19)$$

where x_i is input signal and w_n is the weight parameter. The transfer function transforms the excitation signal to a neuron in the output signal:

$$y = f(net). \qquad (20)$$

The activation function is most common. It is the sigmoid logistic function that represents nodes in the intermediate layers of the artificial neural network:

$$f(net) = \frac{1}{1 + e^{-net}}, \qquad (21)$$

Moreover, there might be another activation function used. Step, linear, Tanh, ReLu, softplus, and softplus functions are commonly used [7]. Unfortunately, there is no single answer which activation function is the best, but many data scientists widely apply it. For classification tasks, a sigmoid logistic function is typically used, as it is faster in the training process and convergence in comparison to other activation functions.

External data are used to determine parameters of the system, so neural networks are trained on given input and output values. The goal is to find the system parameters that minimize the difference between the output value and the desired response. Originally selected parameters are inadequate, and the system makes a lot of mistakes, but changing the values of the coefficients helps to find the best set of parameters.

For the text classification, a modified artificial neural network with n layers including hidden layers of neurons, is constructed. In the case of building a multilayer artificial neural network, the text classification approach can be structured by three core elements: pattern matching, algorithms, and artificial neural nets.

One of wideset artificial neural network methods is a multilayer perceptron classifier. A multilayer perceptron consists of multiple layers of simple neurons as presented in Fig. 12. These multilayers represent the input, hidden and output layer nodes. As usual, each layer is completely connected with the neighboring layer in this neural network. The classification accuracy in terms of configuration of an artificial neural network depends on the number of hidden layers and neurons. The first layer is called input, and the length of the first layer is always equal to the size of the feature vector. The input layer nodes accept the input values and represent the data from the input perspective. Thus, the hidden layers in-between the input and output are mapping the input values by creating a relationship, i.e., the way to the output layer by performing a linear combination of inputs by adding the node weights w and bias b, and applying the activation function f.

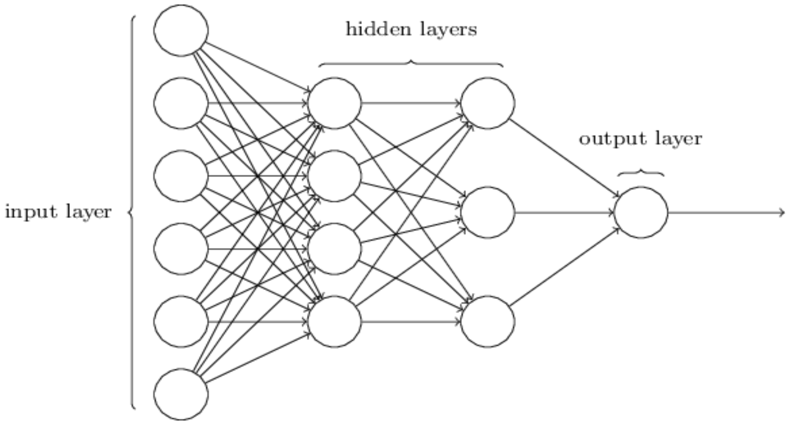


Fig. 12. An example of a multilayer perceptron

This relationship of multilayer perceptron with $m+1$ layers can be expressed as follows:

$$y(X) = f_{m+1}(\dots f_2(W_2^T f_1(W_1^T x + b_1) + b_2) \dots + b_m). \qquad (22)$$

The nodes in between of hidden layers for a multilayer perceptron are formally defined by a sigmoid logistic function. The formal definition of a sigmoid function (21) is presented earlier in this section.

Respectively, the last layer is called an output layer in the multilayer perceptron network and returns the output of classification with a multilayer perceptron by using a sigmoid logistic function. According to the properties of the given multilayer perceptron network, the output represents the results and learning performance of the previous layers. The output nodes for a multilayer perceptron can be formally defined by using a softmax function that is as follows:

$$f(net_i) = \frac{e^{net_i}}{\sum_{m=1}^{L} e^{net_m}}, \tag{23}$$

where the number of output nodes L is equal to the total number of classes l.

However, nowadays a sigmoid for the last layer, softmax, and softplus for internal layers as activation functions of a multilayer perceptron network might be considered.

2.3. Natural language processing

A Homo sapiens, 100,000 years ago learned how to speak, and about 7,000 years ago learned how to write. Today there are millions of pages and sources on the web written by humans, and most all of them can be defined as a natural language. Language is challenged by many official and unofficial rules that make it one of the largest problems in the science world. Computing revolution has touched lots of our life aspects, including the natural language. Nowadays, the language is becoming very much integrated into written form, and digital media challenges are producing lots of contents created by a single user. The computer challenge today is to find ways how to understand this mass of multi-language contents. Unfortunately, humans are not able to analyze and understand even some part of this information, but powerful computers might be used together with the natural language processing (Definition 7) methods and techniques.

Definition 7. *Natural language processing is a theory-motivated range of computational techniques for the automatic analysis and representation of human language* [39].

Natural language technologies deal with methods, tools, lexical features and everything else that helps to prepare a natural language for machine learning algorithms. One of the important phases of data analysis is the natural language processing and pre-processing. However, large-scale textual data have domain-specific issues that can be solved using only special strategies capable of dealing with scalable data-intensive applications. The natural language processing is based on developing an efficient model that typically applies various special methods and techniques for processing sequential data. In most of the cases, a sequence of words, contrary to a sequence of characters is used in natural language processing models. A number of words in the large textual data is so huge, that word-based (rule-based) language models must be considered to overcome high the cause of an extremely high dimensionality problem. There are methods used to clean and prepare data for further analysis. The language analysis always starts from the pre-processing data stage which helps to create a text corpus (Definition 8).

Definition 8. *A text corpus is a collection of authentic machine-readable texts* [40].

Text corpus preparation includes the natural-language processing features, such as tokenization, sentence segmentation, removing stop-words, stemming or lemmatization, bags of words, part-of-speech tagging, term-frequency, word-embedding, etc. The natural language processing depends on the use case or issues that must be solved. Practically, it might be document classification, information extraction, social media analysis, text message classification, analysis of customer communication, media news monitoring – all such examples require special natural language processing models and features.

2.3.1. Noise reduction

The noise reduction consists (Fig. 13) of various natural language pre-processing techniques such as tokenization, stop-word removal, dropping punctuation characters, reducing all capital letters to a lowercase form, term-normalization, spelling correction and other techniques [41]. They are used for removing meaningless noise and frequently used words that usually keep no data. In exceptional cases, someone can consider keeping all the data inside the corpus and ignore the noise, but statistically, the classification accuracy will stay within the limits of noise. Therefore, much more should be loaded into the selected classification method and that is usually not very reasonable.

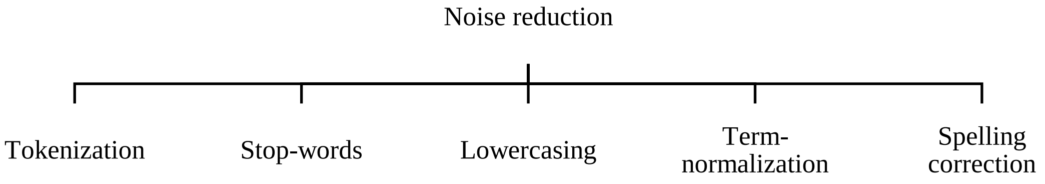


Fig. 13 Noise reduction

So, the main idea of noise reduction is to filter the useful terms from the meaningless ones. All these pre-processing techniques are one of the first stages that are used in the field of natural language processing. The pre-processing stage (punctuated square) in a simplified data processing and the classification workflow model is illustrated in Fig. 14.

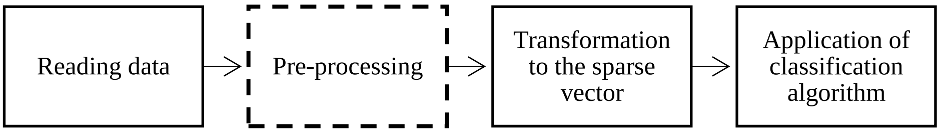

Reading data
Pre-processing
Transformation to the sparse vector
Application of classification algorithm

Fig. 14. Model of data processing and the classification workflow

This natural language pre-processing stage features consist of concepts that are defined in Definition 9, and Definition 10.

Definition 9. *Word is a unit or element of the content with its meaning and place in the sentence* [11].

Definition 10. *Tokenization (the segmentation of a text into sentences and words) is a natural language processing feature that breaks a stream of text into individually isolated word units or even letters. The token is a word element in the meaning of a natural language processing subject* [40].

Word tokenization in *English* and some other languages using punctuation or whitespace is the most usable way that brakes the sentences into the word tokens. An example of tokenization is presented below:

```
Input: Friends, romans, country, lend me your ears
Output: Friends romans country lend me your ears
```

However, this problem is not trivial due to the usage of the full stop character, which may or may not also terminates a sentence. On the other hand, word tokenization is different in ideographic languages as compared to the alphabetic ones [40].

The next key step is removing all stop-words (Definition 11). Stop-words are the words that have no meaning or have lots of conflicting meanings which is not suitable for a further text analysis.

Definition 11. *Removing stop-words is a process to exclude all the stop-words from the given document* [42].

There exist special toolkits and dictionaries that help to find stop-words in the text and remove them. One of them, widely used in data science, is a natural language toolkit [43]. The natural language toolkit (*NLTK*) module comes with a set of stop-words for many languages pre-packaged. The stop-word corpus contains determiners such as *the, a, an, another* that are followed by nouns. The stop-word corpus available in the *NLTK* website [43] contains thousands of stop-words that are prepared for many popular languages. An example of the stop-words corpus is presented in the following section:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', 'she', 'her', 'hers', 'herself', 'it'].
```

The stop-word corpus contains conjunctions and prepositions. Conjunctions, such as *for, an, nor, but, or, yet, so,* serve as connections between separate words, phrases, and clauses. Prepositions, such as *in, under, towards, before,* create expressions between temporal or spatial relations. In summary, all these words as separate instances are very often met in the texts and keep very meaningless information about the semantic tone in the text. Apart from that, some researchers are using stop-words in the construction of corpora because they believe that stop-words can assist to detect meaningful phrases such as "*state-of-the-art*". Otherwise, in most research cases, stop-words simply help to filter high-frequency words from the document that usually have a lower impact on the lexical content of the document before a further construction of the text corpus is continued. The main reason is to reduce the classification capacity, e.g., make less overload by constructing *n*-grams and the feature vector, respectively. After removing the stop-words, there is a distinction that influences the same words that consist of the upper-case (minuscule) and lower-case (capital) letters.

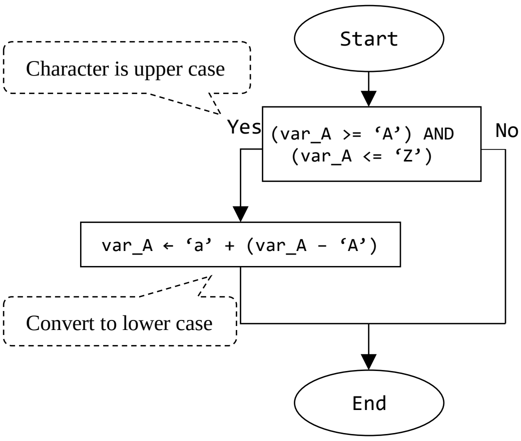

Start
Character is upper case
Yes
(var_A >= 'A') AND (var_A <= 'Z')
No
var_A ← 'a' + (var_A – 'A')
Convert to lower case
End

Fig. 15. Conversion to a lower-case algorithm

The conversion from the capital to lower-case letters must be. It helps to normalize the given text. All the alphabet letters have an association with the numbers that are sequentially distributed. For example, whatever is the number associated with capital "A", capital "B" is sequentially the next one and up to "Z". Identically, lower-case characters have the same relation with the sequential numerical values. To convert, it is necessary to use the relationship between numerical values that are associated particularly between the uppercase and lowercase characters. The relationship is checked by the algorithm (Fig. 15): if the given character is presented in a variable (var_A) is in the upper-case, then it converts to a lower-case character.

Textual data consists of many forms of words with their special endings, depending on the present, past, or future tenses, plural or singular forms, or other formal linguistic rules.

Definition 12. *Term normalization (word stemming or lemming) is the process that removes language rule dependencies on the words and implements reduction of the words to their root (stemma form) form that is required for increasing the consistency of indexing and retrieval.*

Multiple words can be normalized and reduced to their root or base forms. Their main use is as part of a term normalization process that is used in natural language processing tasks.

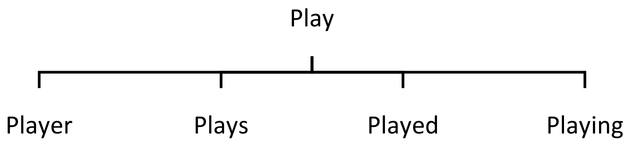


Fig. 16. An example of the normalized word "play"

The normalized word "*play*" is illustrated in Fig. 16. It is a dictionary and rule-based morphological analysis process that removes or replaces the endings of words [44]. Word normalization can also be applied almost to all languages, but it requires a special customization, based on their linguistic rules.

One of the wide used term normalization algorithms is based on Porter stemming (or Porter stemmer) [45] [46]. It is a dictionary-based morphological analyzer that allows us to eliminate all endings including some suffixes (without prefixes). The stemmer can deal with names and other words if they are defined in this dictionary. Thus, it is known as a rule-based method, but sometimes the stemmer can make a loss of meaning and ambiguity of the word [47].

There are more methods and techniques available, and their summary descriptions are presented in Table 1.

Table 1. Additional NLP methods

Name	Description
Named entity recognition	*Name entity recognition contains all the named entities, which are phrases that contain the names of persons, organizations, locations, times, and quantities* [48].
Apostrophes	*Used for possession and contractions (aren't, don't, can't).*
Specific tokens	*Applied to identify contact related information such as phone numbers, e-mails, addresses, invoice payment days, bank account or VAT number.*
Hyphens	*Considering that white space is not always required to every case, such as Mercedes-Benz, San Francisco-Los Angeles.*
Normalization	*Typically, normalization is used to deal with accents and diacritics that is related to the regional language. It also reduces all letters to a lower case.*
Equivalence classes	*Create relations between two or more normalized tokens, such as chair and furniture.*
Phrase	*Identify phrases as a small group of words that creates a meaningful unit. There are several types of phrases: noun (vase of roses, book about), verb (had been living, will be going), adjective (very interesting), adverbial (very slowly), prepositional (near the sea).*
Syntactic parsing	*Recognize the sentence and its grammatical correctness by assigning a grammar function to each word in the corpus.*
Chunking	*Implements word and sentence segmentation and creates labels on multi-token sequences. It presents word-level tokenization and Part-of-speech tagging. Large boxes present a higher-level of chunking. Chunking usually makes selections from a subset of tokens* [49].
Emoticons	*The process that extracts the emotion symbols from the text. Emoticons usually express different states of mind (thoughts and emotions) and can be used as an advanced feature for classification and sentiment analysis.*

2.3.2. *N*-grams

At present, the popularity of statistical linguistic is increasing in the text classification, whereas the algorithms that make a semantic relations analysis are conditionally slow and very complex. In the case of statistical linguistic words, terms are used as separate units or in other terms. Such a type of simple approach leads to the computational calculation of the importance of a term or group of terms in the given text corpus. In the artificial neural networks, groups of words are usually made according to the semantical meaning, whether one word is on the left side or the right side of another word. These words are considered as similar words by their semantical meaning. Meanwhile, using statistical linguistics in developing ordinary relations between the different terms or a group of terms helps us create similar semantical relations as by the artificial neural networks. To establish such relations, *n*-grams are the main object in the statistical linguistics. *N*-grams are, basically a contiguous sequence of n items (tokens) from a given sequence of the text. But sequentially applying *n*-grams (Definition 13), it is not always enough to classify text accurately, because usually much more relations exist between different terms or their groups. Therefore, some ordinary combinations of *n*-grams should be investigated and proposed for the given problem rather to complex semantical feature selection algorithms.

Definition 13. *n-gram is a continuous sequence of tokens.*

Any sentence from the given text is written of sequences of composed symbols or tokens – words, letters, digits, punctuation, white spaces and other characters. Regarding how the *n*-gram model is defined, the token can be a word, letter, digit, punctuation, white space, and another character.

Today, in the area of natural language processing, *n*-gram feature allows us to create *n*-grams from the continuous sequence of words. Instead of building *n*-grams from the sentences, typically a continuous text flow is in use. This is because the task of a classifier does not attempt to understand the meaning of a sentence. It creates the input to a classifier with all features (tokenized terms, and term groups), the classifier builds the model that assigns the class as accurately as possible. Models, based on *n*-grams, are the core building block of statistical language modeling for many decades.

Definition 14. *The class-based language model is a process that improves the statistical efficiency by introducing a notion of word categories and then shares statistical strength among the words that are in the same category* [7].

These class-based language models of *n*-grams with the help of classification algorithms partition the set of words into classes by their attributes, that are based on their appearance frequencies, with relation to other words (the idea of term frequency is presented in the section 2.3.4). The strength of this model is to use the groups of words with the assigned class attribute, rather than individual words that represent the context.

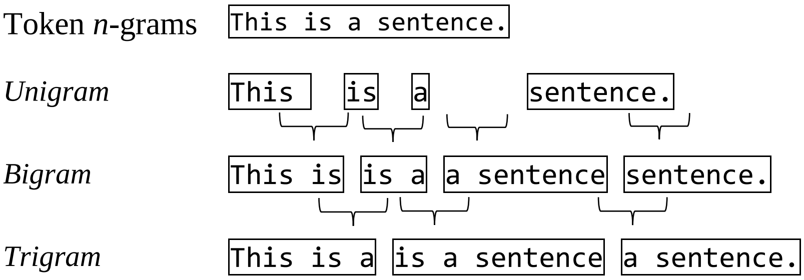
The number of n defines how many items are grouped in each segment. It allows us to create word groups or small phrases with regards to the definition of *n*-gram size. The structure of *n*-gram is presented in Table 2.

Table 2. An example of *n*-gram composition

n-gram	Example
unigrams ($n = 1$)	*['better', 'chance', 'enjoy', 'good', 'weather']*
bigrams ($n = 2$)	*['better chance', 'chance enjoy', 'enjoy good', 'good weather']*
trigrams ($n = 3$)	*['better chance enjoy', 'chance enjoy good', 'enjoy good weather']*

An *n*-gram size represents the number of words, e.g., if $n = 1$, then it is called one-word unigram; if $n = 2$, two-word bigram; and if $n = 3$, then – three-word trigram. The construction of *n*-gram feature is presented as follows:

- *Unigram* is a sequence of words in the given sentence that splits each single word, based on the whitespace between the words;
- *Bigram* takes the output of a unigram and splits the sequence of words by starting from the first one;
- *Trigram* takes the output of a unigram and splits the sequence of words starting with from the first one.



An *n*-gram includes sequences of the text or speech and uses computational linguistics that deals with the statistical (or rule-based) properties of the *n*-gram and natural language processing.

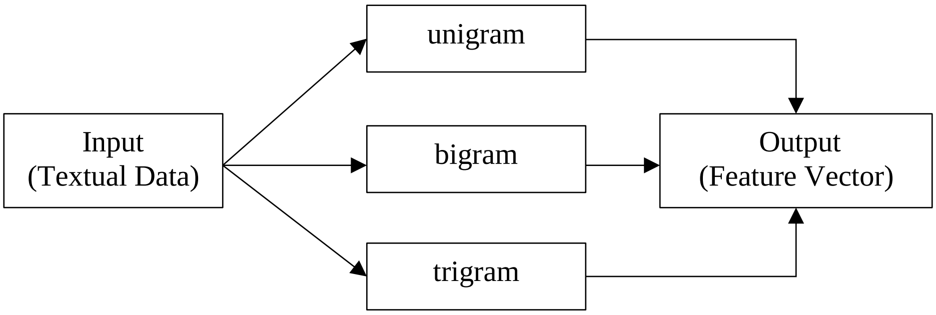


Fig. 17. A workflow model example of the *n*-gram feature

A workflow model example of the *n*-gram feature is presented in Fig. 17 and contains input as textual data, continues text flow conversion to the *n*-gram and output as the construction of a feature vector.

Another known application of *n*-grams is a probability distribution over the sequence of words, the model considers multiple words at the same time. A model of the probability distribution of *n*-words sequences is called as an *n*-gram model as text prediction using *n*-grams. An *n*-gram model is defined as a Markov chain of order $n - 1$. In the Markov chain, the probability of word w_i

depends only on the previous words, not on any other words [18]: n-gram as a sequence of written words of length n is applied “unigram” for unigram $P(w_i)$ “bigram” for bigram $P(w_i|w_{1:i-1})$, and “trigram” $P(w_i|w_{i-2:i-1})$, where w_i is the word in the sentence and p is a probability. In the Markov chain, the probability of word w_i depends only on the immediately preceding words, not on any other words. The trigram model (Markov chain of order 2) can be formally defined as follows:

$$P(w_i|w_{1:i-1}) = P(w_i|w_{i-2:i-1}) \tag{24}$$

where the probability of a word w_i depends only on the immediately preceding words, not on any other words. The probability of a sequence of words $P(w_{1:N})$ under the trigram model is first factoring with the chain rule and then using the Markov assumption:

$$P(w_{1:N}) = \prod_{i=1}^{N} P\,(w_i|w_{1:i-1}) = \prod_{i=1}^{N} P\,(w_i|w_{i-2:i-1}) \tag{25}$$

For a trigram word model in a text dataset of 10000 reviews or separate text units, $P(w_i|w_{i-2:i-1})$ has millions of words and can be estimated by counting word sequences in a body of text units of millions of words and more. For instance, n-grams can be used for language identification or probability of item distribution in a sequence, e.g., next-letters or next-words. For instance, computers can identify languages very accuracy, but still there, are many confusions with closely related languages.

2.3.3. Part of speech

It is already known that extraction of nouns, verbs or adjectives is a strong indication to make a more accurate sentiment analysis because these terms are context keepers. Some studies compared the impact of the word forms, and conclusions lead to the effectiveness of adjectives, verbs, and adverbs, where subcategorization usually makes a reasonable impact on the sentiment analysis

[50] [51] [28]. The idea is to identify and tag terms as nouns, verbs, adjectives, adverbs, etc. It simply marks the words in a text with special labels, corresponding to the part of speech (Definition 15) of the word in this context.

Definition 15. *A part-of-speech tagging is a process that annotates every term with a part-of-speech tag, a label assigns to each term, e.g., single noun (N), a plural noun (NNS), verb (VB), verb, past tense (VBD), etc.*

Part-of-speech tagging is thought of to be a crude kind of word sense disambiguation [52]. Such text subcategorization of nouns and verbs can become a strong indication for more accurate sentiment analysis when applying classification algorithms in a large-scale textual data analysis. Two approaches use the tagging rules: statistical or rule-based and trained, using corpora manually labeled (i.e., single noun (N), a plural noun (NNS), verb (VB), verb, past tense (VBD)). An example of the universal set of the part-of-speech tag is presented in Table 3.

Table 3. An example of the universal set of the part-of-speech tag set

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	ad position	*on, of, at, with, by, into, under*
ADV	adverb	*Already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, university*

2.3.4. Term frequency hashing

At present, the feature term frequency hashing is very commonly used in the field of text analysis. The feature term-frequency hashing allows us to perform tasks much faster at the classification time because it is using specially created hash-values versus that of a string. Feature term-frequency hashing converts all textual documents to predefined fixed-length feature vectors that are suitable to be passed to the classification stage. The main idea of feature term-frequency hashing (or hashing function) is to reduce the dimensionality and perform a compression of textual data, e.g., to encode variable length textual documents (every single term) into equal-length numerical format feature vectors. In a very primitive example, the hash function will encode and sum up the output of all the letters from alphabet "a" to 1, "b" to 2, "c" to 3 and so on, up to "z" being 26. For the *Abraham Lincoln* quote "*whatever you are, be a good one*" the output of the hash function is:

Table 4. An example output of the hash function

Input	Output
$(whatever)$	$23 + 8 + 1 + 20 + 5 + 22 + 5 + 18 +$
(you)	$25 + 15 + \ 21 +$
(are)	$1 + 18 + 5 +$
(be)	$2 + 5 +$
(a)	$1 +$
$(good)$	$7 + 15 + 15 + 4$
(one)	$15 + 14 + 5$
Total	$= 270$

As per the given an example, the output represents a quantitative meaning of the input quote. The idea of feature hashing can be successfully applied to words, for instance (book, 0), (internet, 1), (application, 3), etc.

Meanwhile, the feature term frequency (Definition 16) is the process that groups all terms and calculates their term frequency with the output: the hashing key of the term and its frequency value.

Definition 16: *Term frequency is a statistical process that generates fixed-size feature vectors from text documents and assigns the weight parameter by its frequency in the given text corpus.*

These terms are imported to a specially created hashing vector assembler. With the help of the vector assembler, a transformation into one column can be processed as the input for the classification algorithm. A range of arrays input/output is presented in Fig. 18.

[interest, tale, lawyer, take, million, dollar, firm, fake, take, love, money, usual, hook...]

$$\underbrace{[1000,}_{\text{Length of a feature vector}}\ \underbrace{[15,192,230,236,266,354,371,210\ldots],}_{\text{Position of words / hashing key}}\ \underbrace{[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,\ \ldots]}_{\text{Count of a word at position / frequency}}\ ]$$

Fig. 18 An example of word array before and after transformation

Let us say a term t is a word that exists in corpus D with a set of documents $D = \{d_1, d_2 \dots, d_i, \dots d_N\}$. Given some term t, and a document d, the term count n_{td} is the number of times that particularly the term t occurs in the document d. Given a collection of k terms and corpus D with a set of documents, the term-frequency $tf(t, d)$ is:

$$tf(t, d) = \frac{n_{td}}{\sum_{k=1}^{T} n_{kd}}. \tag{26}$$

The term frequency $tf(t, d)$ is a frequency parameter that defines how often the term t is found in the document d. One of the most known and often used models is *Zipf's law*. It says that, if the term t_1 is the most common term in the text corpus, then the term t_2 is the next most common. Meanwhile, the collection frequency $tf_i(t, D)$ of the ith most common term is proportional to $\frac{1}{i}$ [23]:

$$tf_i(t, D) \propto \frac{1}{i}. \tag{27}$$

It means that, if the most common term t_a in the text corpus D occurs tf times, then the second most frequent term t_b has less occurrences, the third one three times less, and so on. The term-frequency tf defines the number of documents that inhere to the term t. However, some terms are highly distributed in the corpus D, so it will not hold any useful information about a special document. Meanwhile, the idea of the inverse document frequency idf can be formally expressed by a numerical measurement parameter that calculates how much information or weight the term t is holding:

$$idf(t,D) = log\frac{|N|+1}{df(t,D)+1}, \tag{28}$$

where $|N|$ is the total number of documents in data corpus D. If such a term becomes noticeable in all the documents of data corpus D, the value must be equal to 0. Otherwise, the inverse document frequency will make a greater weight to a term that occurred less times in the given documents.

2.3.5. Word embedding

Currently, word embedding gained a lot of interest in the natural language processing area. Word embedding creates a good representation of words, including similarities that exist between these words in the given context. The main idea behind the word embedding is to create a multilayer artificial neural network, using mapped words to input vectors that learns word embedding vectors by maximizing the corpus likelihood by neural network training [53].

Definition 17: *Word embedding is a set of language modeling and feature learning techniques applied using natural language processing to map words to vectors that represent the corpus of text* [54].

Therefore, the output of word vector represents converted numerical values that artificial neural network can learn these words. Fundamentally, this method is mostly used to function with deep artificial neural networks performing natural language processing tasks. Comparing to other feature vectors, word vectors create much more semantic meaning and the relationship between the words. The most known word embedding can be performed by the *word2vec* technique [53] that can be constructed by using *skip-gram* that learns representations of the word vector, and they are valuable at predicting its context in the given text of the training words $t_1^l, t_2^l, \ldots, t_N^l$. The core idea of a skip-gram is that each single word (term) during the training is associated with two vectors v_t' and v_t that are vector representations of t as a words (term) and context, accordingly. This skip-gram model is illustrated in Fig. 19 presented by *Mikolov* [54].

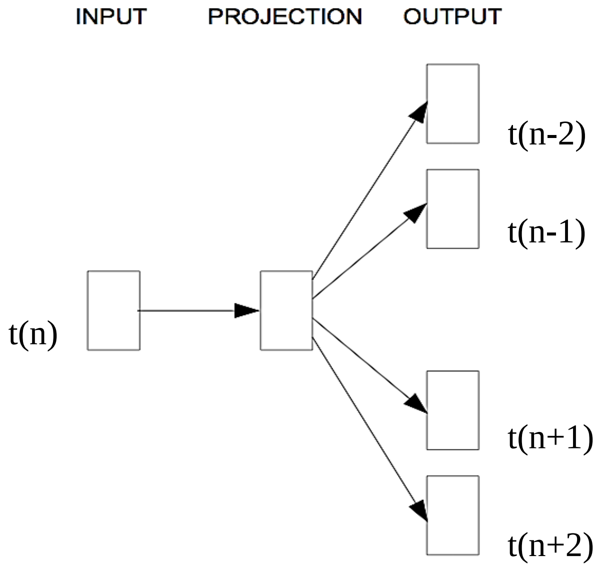


Fig. 19 A skip-gram model

An example of skip-gram architecture demonstrates that it finds the central word and predicts the surrounding word (term) that exists in the given text instance. The skip-gram model architecture contains input, projection, and output. The *skip-gram* tries to maximize the average log-likelihood, and it can be formally defined:

$$\frac{1}{n}=\sum_{n=1}^{n}\sum_{-k^t\leq j\leq k^t,j\neq 0} log\, p\left(t_{n+j}\middle|t_n\ \right), \tag{29}$$

where k^t is the size of the training context (also can be as a function of the center word (term) t), t_n is a word (term), n is the total number of words (terms) in the given text corpus, p is the probability of a correctly predicted word. Larger k^t results in more training examples lead to a higher classification accuracy [54]. Thus, the probability of correctly predicts the term (word) t_i, given the word t_j, can be formalized by using a softmax model:

$$p(t_i|t_j) = \frac{\exp({v'_{t_i}}^{\top} v_{t_j})}{\sum_{t=1}^{n} \exp({v'_t}^{\top} v_{t_j})} \tag{30}$$

where v_t is the input, and v'_t is the output vector representation of words t (terms) and n is the total number of words (terms) in the given text corpus.

2.4. Performance measurement of multi-class classification

This section presents the measurements for multi-class classification tasks with the average accuracy error rate, precision, and recall. The average per-class effectiveness of a classifier is one of the most known and used for machine learning multi-class classification tasks. Assume that, for every individual class c_i, the assessment is defined by tp_i, fp_i, fn_i, tn_i, where tp_i are true positive classification examples, fp_i are false positive ones, fn_i are false negative ones, and tn_i are true negative ones, and l is the number of classes.

The classification accuracy is calculated by actual labels that are equal to a predicted label, divided by the total corpus size into test data. The average accuracy formula for multi-class classification can be presented as follows [55]:

$$Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l}. \tag{31}$$

The average per-class classification error rate formula for multi-class classification can be expressed as follows [55]:

$$Error\ Rate = \frac{\sum_{i=1}^{l} \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{l}. \tag{32}$$

The precision is calculated by an average per-class agreement of data class labels with that which is assigned. The precision rate formula for multi-class classification can be written as follows [55]:

$$Precision = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \tag{33}$$

The recall rate formula for multi-class classification can be presented as follows [55]:

$$Recall = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{34}$$

The recall is calculated by the average per-class effectiveness of a classifier to identify class labels. Thus, F_1 (*F1-measurement or balanced F-score*) is calculated according to relations between positive data labels and that given by a classifier, based on a per-class average [54]:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{35}$$

The presented measures for multi-class classification are the main ones that measure the classification performance. These measurements are usually categorized into two levels: *micro* and *macro*-averaging. A macro measure is the average of the measures calculated $c_1, c_2 \ldots, c_l$ or the sum of counts to get cumulative tp_i, fp_i, fn_i, tn_i. The *micro*-averaging favors lager classes, and *macro*-averaging treats all classes equally [54].

2.5. Computing resources and data analytics frameworks

The computer industry in terms of ICT witnessed several major fractures. Firstly, the computing technology began to slowly migrate from the universities and public institutions to the first personal computers. Mostly all mainframe computers have been decentralized to the client-server systems, which accelerated the appearance of the first personal computer at homes, and with a rapid spread of technological progress. Then, there was the so-called second technological transformation, when computers were connected to the global network, which is called the internet today. The third and crucial turning point happened when users have started to share the available ICT infrastructure and services in the form of purchasing or renting them from the cloud computing technology (Definition 18) service providers. The cloud computing technology became a model for on-demand network access to a shared pool of widely configurable computing resources, i.e., networks, servers, storage, applications, and services.

Definition 18. *Cloud computing is a technology where all intensive computations act at present, i.e., technologically allow us to extend remote computing capabilities without human interaction, ensures availability over the high-speed network, automatically monitor, control, and optimize modern computing resources according to different consumers' needs with a sense of location independence.*

Now, the cloud computing technology can be rapidly provisioned and released with the minimal management effort or service provider interaction [56]. In technological terms, it is a transition from a client-server to centralized systems with distributed parallelism, like a cyclical return to the past, but this is done to optimize time or cost and focus on the core organizational assets.

Currently, centralization of the computing resources becomes more popular and reasonable to use in today's ICT world and there are many advantages such as cost reduction impact: virtualization, lower quantity of hardware, less energy consumption, no up-front investment and pay-as-you-go service, lower operating costs, rapid allocation and deallocation, scalability when service demands are changing, accessibility through different devices and reducing business risk while outsourcing from infrastructure providers. On the other hand, disadvantages are related to trust and data protection, centralization of infrastructure, high-costs of network broadband between various locations, and investment costs for infrastructure. Independent of the mentioned and not mentioned disadvantages, the popularity of cloud computing technologies is increasing, and large numbers of users are starting to use it.

2.5.1. Cloud computing technology

The first time the idea and term of cloud computing technology were expressed by *McCarthy* in 1966. *McCarthy* (1966) has envisioned:

> *"Extended and remote computing facilities can be delivered to the end users as a service"* [57].

Later the term was discussed again by the *Google* founder and developer *Schmidt* in 2006. His vision was clearly based on the innovative sales model as most of the services can be delivered to the customers over the internet and worldwide [58]. Today, the cloud computing technology [2] is available for many different purposes, and one of them is machine learning, big data (including textual) processing and transformation that leads to a new type of digital services. The analysis of large data deals with understanding and using innovative ways and tools how to process, operate, and reuse large multidimensional datasets. Therefore, cloud computing technology capabilities enable new tools and transition of data to run software as a service, platform as a service, infrastructure as a service, and hardware as a service.

The amount of data is increasing rapidly in comparison with CPU performance, and it creates limits on computing resources [59]. But today, the cloud computing technology has overcome such limitations with a possibility to adjust the computing resource to deal with the increasing capacity of data in a very distributive network where applications, data, and computing resources are spread out across more than one hardware unit and distributed over the computer network. The cloud computing technology consists of many other necessary components that are usually invisible to the end users: network, database instances, identity management, monitoring, run time, security, storage, capacity scheduling, redundancies and high availability, process automation, etc. However, the cloud computing technology enables virtualization technologies in multiple levels (hardware and application platform) to realize resource sharing and dynamic resource provisioning in comparison to grid computing where it also employs distributed resources to achieve application-level objectives [58].

The national institute of standards and technology (NIST) in the United States of America [60] has defined the core characteristics of the cloud computing technology that is as follows: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [56]. These characteristics are based on possibilities to extend computation capabilities on demand and self-service based without special interaction of planning or maintenance of infrastructure, to ensure the availability over the network, to provide computing resources for different consumers in the sense of location independence or automatically monitor, control, and optimize resources. A layered model can express the cloud computing architecture and represented in the way of which it explains four elements or four service abstractions: hardware, infrastructure, platform and application [56]. The hardware service layer is responsible for managing the physical resources of the cloud computing technology. For this purpose, hardware service layer can manage and assign the required physical resources. The infrastructure service layer creates a pool of resources using virtualization technologies and ensures access to the virtual processing or memory resources [61]. It contains a distributed storage and processing of computer clusters. It is usually composed of commodity hardware and located inside huge data centers. For instance, parallel computing, in-memory or disk-based data processing services expand computing capabilities just on demand or by a pay-as-you-go model for all being involved in a large data analysis with machine learning. The platform service layer consists of the operating system and application frameworks, such as *Apache Hadoop* or *Apache Spark* (described in the following section). The application layer runs computer programs under earlier described layers.

2.5.2. Data analytics frameworks

The open source organization *Apache Software Foundation*[5] is developing many software projects, and only very few of them are dedicated to the data processing and machine learning technologies such as *Apache Hadoop*, and *Apache Spark*. The main concept of *Apache Spark* is an immutable resilient distributed dataset. In contrast to *Apache Hadoop*, *Spark* can run distributed processing in-memory rather than only on disc [62]. *Apache Spark* is known as a new generation open source cluster computing framework in comparison to *Hadoop*, but its origin is derived from *Apache Hadoop MapReduce*. Both framework enables processing automation, fault tolerance, high availability, and scalability between computer nodes or even physical racks. Such data-intensive computations open the door to text classification that is effective in solving big-data classification tasks. *Apache Hadoop* is a framework for the distributed processing of big data across clusters of computers using *MapReduce* programming data model[6]. *MapReduce* is a programming model that provides support for parallel computing, locality-aware scheduling, fault-tolerance, and scalability on commodity clusters [63].

Nowadays *Apache Spark* and *Apache Hadoop* are classified as third-generation data processing, originated for horizontal scalability when adding more computers to the pool or cluster, and contain machine learning technologies as compared to *SAS*, *R* and *Weka*. The latter ones belong to the first-generation that are built mostly for vertical scalability when adding more power to the existing computer. Some add-on tools support the first-generation machine learning technologies to make them more advanced, but there are still cost and technological issues when classifying large datasets. Table 5 summarizes three generation view of data processing and machine learning technologies [5]. The third-generation technologies that belong *Apache Spark* and *Apache Hadoop*

[5] Apache Software Foundation. More: http://spark.apache.org

[6] MapReduce Tutorial. More: http://hadoop.apache.org

frameworks, support fault-tolerance, and are horizontally scalable, e.g., applicable to an increased number of large datasets and can process tasks on multiple nodes within the cloud or grid computing environments. The first-generation consists of traditional tools, whereas the second and third-generation, include progressively innovated big data technologies capable of dealing with a cloud computing environment. However, this thesis focuses on the third-generation approaches that work over *Apache Spark* and *Apache Hadoop*. More details about selected data processing frameworks are presented in this section.

Table 5. Three generation view of data processing and machine learning technologies

Generation	*1st Generation*	*2nd Generation*	*3rd Generation*
Examples	SAS, R, Weka, SPSS, KNIME, KEEL	Mahout, Pentaho, Cascading	Spark, Hadoop, GraphLab, Pregel, Giraph, ML over Storm
Scalability	Vertical	Horizontal (over Hadoop)	Horizontal (beyond Hadoop)
Classification algorithms available	Huge collections of algorithms	Small subset: sequential logistic regression, linear SVMs, Stochastic Gradient Descendent, k-means clustering, Random forest, etc.	Logistic regression, Naive Bayes, Random forest, Decision Tree, Support Vector Machine, Multilayer perceptron classifier.
Classification algorithms not available	Practically nothing	Vast no.: kernel SVMs, Multivariate Logistic Regression, Conjugate Gradient Descendent, ALS, etc.	Work is in progress to expand the set of available algorithms
Fault-tolerance	Single point of failure	Most tools are FT, as they are made on top of Hadoop	FT: Hadoop, Spark Not FT: Pregel, GraphLab, Giraph

Thus, *Apache Spark* is an extension of *Apache Hadoop* [64] that supports interactive queries and stream processing. *Apache Spark* was invented by M. Zaharia at the *University of California* and was started as a research project in 2009 and as an open source in 2010. It has become as the Apache top-level project in 2014. *Google founders created Hadoop M. Cafarella* and *D. Cutting* in 2005 with the main purpose to solve a searching and indexing problem on the internet. A very high-end usage of *Hadoop* is known for even a more advanced computing system, named *IBM Watson*. *Watson* uses *IBM's DeepQA* software with *Hadoop* to provide distributed computing [65]. It is a humans' question answering system that uses the natural language and returns a precise answer to the question. The questions can be very generic or specific. However, *Watson* has made a semantic analysis and applied numerous of detection rules that help to make a deep analysis of the question and find the way how best to approach answering it [66]. It applies advanced natural language processing, information retrieval, knowledge representation, automated reasoning, and machine learning technologies to the field of open domain question answering[7]. *Apache Hadoop* and *Apache Spark* are largely scalable cluster-computing and data analytics frameworks that can provide computing abilities for various types of natural language processing [39] and machine-learning tasks [1] by using data-intensive applications [67]. The implementation of *Apache Hadoop* is customizable and is based on the application requirements. Fig. 20 illustrates the core components of *Apache Hadoop* [68]. *Hadoop Common* is responsible for libraries and utilities needed by other *Hadoop* modules.

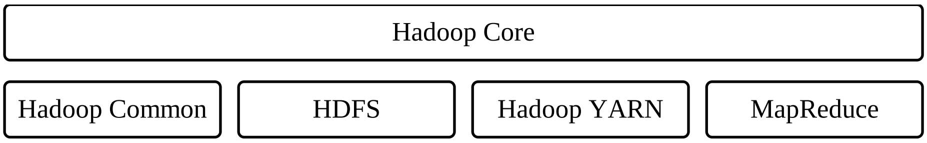


Fig. 20 Architecture of Apache Hadoop

Hadoop YARN provides a management platform for computing resources and is responsible for scheduling applications to run. As already mentioned, *Hadoop* consists of two abstractions, *Map* and *Reduce*, and responsible programming model for data-intensive computing. The *Hadoop MapReduce* model consists of five phases: 1) it gathers unstructured data from the input

[7] IBM. (2011). DeepQA Project: FAQ

source, 2) imports it to the *HDFS* [68] (*Hadoop Distributed File System*) file system, 3) runs the map phase (groups and specifies the per-record computation), 4) runs reduce phase (collects, sorts data, specifies aggregated results in associative and commutative manner) and 5) writes the output to the file system [5]. The programmer must specify the *Map* and *Reduce* functions within a job. Then, the job usually divides the input dataset into independent subsets that are processed in parallel by the *Map* tasks. *Apache Hadoop* is a framework of a distributed processing computation model. To enter data into *Hadoop*, firstly all data must be converted to *HDFS*. *Apache Hadoop* processes every task and returns the result to the disk after mapping and reduction actions are completed, in other words, it is a two-stage and disk-based architecture. Only the *HDFS* file system is designed to implement parallel computing by *Apache Hadoop*. There are significant differences, as compared with other distributed file systems, and advantages are considered as highly fault-tolerant and are deployed on low-cost hardware. *HDFS* files must be divided into 64 or 128 MB fragments and only afterward can be delivered to the nodes (create three copies) to implement data-intensive processing [68]. Besides, *Hadoop* can be scaled up to 1000s of nodes. *HDFS* can run a very distributed environment and manage multiples nodes. *HDFS* can store large files across multiple machines, multiple geo-servers ensures reliability by replicating the data across multiple servers and locations. Apache Foundation manages the Hadoop HDFS project as an open source project. All data transfer occurs directly between clients and data nodes and communications with the name node only involve the transfer of metadata. The essential properties of *HDFS* name nodes are as follows [5]:

- to keep all the details of the directory structure, exactly knowing the locations of all blocks and jobs;
- to coordinate all client communications with data nodes;
- to ensure the failure tolerance and the health of the entire system, as it uses replicas of the data blocks.

Today, *Apache Spark* is knowns as a key framework for data science that helps to explore, understand and transform big datasets, to create models for classification using machine learning. *Apache Spark* is one of the best frameworks that runs applications with large multidimensional data and machine learning algorithms. It is an intensive in-memory computing platform designed to be one of the fastest available and very general-purpose regarding running various kinds of computing tasks [69] and it contains *RDD* (*Resilient Distributed Datasets*) file systems. In 2014, *Spark* was tested for a large-scale sorting and achieved a world record. *Spark* could approximately process 100 times faster than *MapReduce*[8], but it processes all data in the memory, so it needs a lot of computing resources. *Apache Spark* uses memory as the standard database, – loads and processes in the memory until a further action has started. In this case, there are always possibilities for computing performance degradations, and it depends on the size of data. *Spark* could be used to run *MapReduce* and keeps all the most important aspects of data distribution, fault tolerance, and parallelization (Table 6). The structure of *Apache Spark* consists of four elements: *Shark SQL*, *Spark Streaming*, *MLlib*, and *GraphX* graph. All the components are targeted at large scale commodity clusters or cloud computing technologies. To run any application in the *Apache Spark* execution environment, the essential matter is to use a *Spark* context. The *Spark* context is illustrated in Fig. 21.

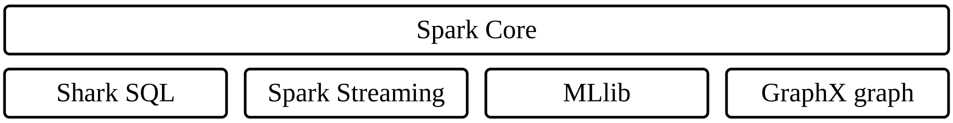


Fig. 21 Architecture of Apache Spark

The *Apache Spark* context is essentially a client of the *Spark* execution environment and acts as the master of *Apache Spark* application. The *Apache Spark* context acts as the master of *Apache Spark* application. As compared *Hadoop MapReduce* returns the result to the disk after the mapping and

[8] Details about the Daytona GraySort contest: http://sortbenchmark.org/

reducing action has been completed.

Table 6 presents some comparison aspects that are important for the thesis. The main goal of the comparison was to identify the current state of algorithm deployment of *Apache Hadoop* and *Spark* for a future experimental analysis. The comparison aspects are as follows: API, data processing architecture (data flow model) and operations, compatibility, machine learning algorithms, hardware provisioning, fault tolerance, supported programming languages and interfaces, and a possibility to run the classification and dimensionality reduction algorithms. The comparison analysis presents that Apache Hadoop and Spark are very similar, but have several essential differences, since *Apache Hadoop* supports a two-stage disk-based data processing architecture, while Spark has to cash in the memory, which means that data partitions are read from RAM instead of the disk [70]. Another difference is based on hardware requirements which technically are the same. The data must be proportionally equal to the memory to accomplish an optimal speedup and performance. Hardware requirements always depend on the size of data and the distance between hardware components. The last and essential difference was found regarding graphical user interface. *Apache Spark* is best prepared for it, but *Apache Hadoop* itself is not. *Apache Spark* and *Apache Hadoop* are as an open-source project offers a license-free solution, including useful documentation, running instructions and examples. On the other hand, usage of *Apache Hadoop* could be a better option regarding costs because it requires less hardware. However, if data-intensive computations must be done not so often, then, using the in-memory access over the distributed machines of a cluster, they will proceed with the entire iterative process. The performance has been evaluated for *Apache Spark* over *Apache Hadoop,* while the memory consumption or other system performance criteria are not deeply analyzed. Some experiments have already shown that, although *Apache Spark* is faster than *Apache Hadoop* in iterative operations [64], it must pay more for memory consumption. Table 6 also presents a comparison of classification algorithms.

Table 6. Comparison of Apache Spark and Apache Hadoop

Apache Spark	Apache Hadoop
Application programming interface supported	
R, Scala, JavaScript, Java and Python, Spark SQL (Shark)	R, Scala, JavaScript, Java, Python, Hive SQL
Operations	
Map, filter, group by, count, collect, reduce, save	Map, filter, group by, count, collect, reduce, save
Data processing architecture	
In-memory	Two-stage disk-based
Deployment possibilities	
Commodity servers, Cloud computing, Single computer	Commodity servers, Cloud computing, Single computer
Hardware provisioning	
Cores 8-16	Cores 4
Memory 8 GB to hundreds of gigabytes	Memory 24 GB
Disks 4-8 one TB disks	Disks 4-6 one TB disks
Network 10 GB or more	Network 1 GB Ethernet all-to-all
Supported file systems	
HDFS, RDD	HDFS
Fault-tolerance	
Yes	Yes
Components	
Tachyon, Mesos	HDFS, YARN
Tools	
Spark native API, Spark SQL, MLlib, Spark Streaming, GraphX, Spark Notebook/Spark	Pig, Hive, Mahout, Storm, Giraph, HUE
Supported classification methods	
Logistic regression, Naive Bayes, Random forest, Decision Tree, Support Vector Machine, Multilayer perceptron classifier	Naive Bayes, Random forest

Apache Hadoop and *Spark* have a common execution engine, and similar libraries (*Mahout* and *MLlib,* respectively) and both could be considerably consolidated to deal with machine learning algorithms. *Mahout* library was previously used only by *Hadoop MapReduce,* and now it switched to *Apache Spark*. *Mahout* library was modified and now supports *Apache Spark*, while *MLlib* only is supportable by *Apache Spark*. In this analysis, only the classification algorithms were used.

2.6. Conclusions of Chapter 2

The analysis in the field of text classification using machine learning is presented in this section.

1. The analysis has shown that the multiclass text data research classification methods can be successfully transferred to the cloud-based large-scale data processing platforms, and could be provided as a service. The number of solutions by machine learning is therefore slowly increasing. One way in which textual data can be transformed into the knowledge is by applying classification methods using machine learning.
2. The existing data analytics frameworks adapted to big data are increasing the solutions, but the transfer of data classification algorithms is a slow and very complex process so far. The existing classification algorithms and parallel strategies cannot be easily applied directly to the cloud computing technology platform and require specific customization because they must be prepared for horizontal scalability over the multimode clusters to deal with large multidimensional data.
3. The findings indicate that the cloud computing technology with the data analytics framework *Apache Spark* is most attractive and provides essential technological opportunities for big data classification using machine learning. Based on the comparison study that has concluded that *Apache Spark* is one of the best data analytics framework that has interoperability with widely known classical classification methods. *Apache Spark* data analytics framework with the *MLlib* library has been taken and prepared for in-memory intensive operations for data classification using machine learning experiments.

4. The theoretical results have shown that large-scale textual classification using machine learning algorithms and natural language processing techniques, such as noise reduction by using word stemma, rule-based word grouping by *n*-grams and part-of-speech tagging, calculation of (inverse) term frequency can be constructed and implemented using *Apache Spark* data analytics framework.

Chapter 3 Research Methodology

The research methodology of natural language processing methods for multi-class classification, based on modern cloud computing technology solutions and data analytics frameworks, is presented in this chapter. This methodology consists mainly of three stages:

1) data extraction and selection;
2) noise reduction by applying various pre-processing techniques to filter the useful text that can be used for the classification stage, and
3) application of the selected classification methods that will classify product-review data into five classes.

Several propositions and considerations are made in this chapter how to apply data feature selection techniques using multi-class classification methods for large-scale product-review data that allow us to determinate the classification problem with a higher classification accuracy. A modified workflow model also including the corresponding natural language processing methods and techniques for a multi-class classification allow us to compare and evaluate the performance criteria, i.e., the measurement of classification accuracy. This modified workflow model was used to classify short-text messages and assisted in performing experiments presented in Chapter 4.

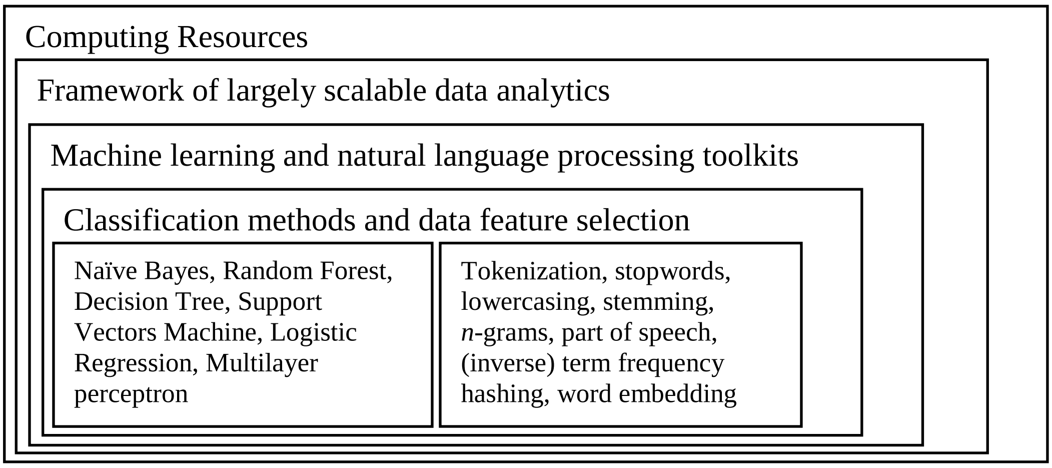

Computing Resources
Framework of largely scalable data analytics
Machine learning and natural language processing toolkits
Classification methods and data feature selection
Naïve Bayes, Random Forest, Decision Tree, Support Vectors Machine, Logistic Regression, Multilayer perceptron
Tokenization, stopwords, lowercasing, stemming, *n*-grams, part of speech, (inverse) term frequency hashing, word embedding

Fig. 22 Research environment

The defined research environment consists of computing resources where all the experiments will take place (Fig. 22). The framework *Apache Spark* of largely scalable data analytics with the *MLlib* library enables us to run big data computation tasks in the cloud computing environment.

3.1. Information of dataset

Two independent and open datasets have been selected that validate the presented experimental results and scientific conclusions. The *Amazon* customers' product-review dataset for *Android* apps (dataset A), and *Amazon* customers' product-review dataset for movies and *TV* (television) (dataset B) are selected for investigating [71]. The total number of records in *Amazon* customers' product-review data for *Android* apps is given by $n = 2{,}638{,}274$. The total number of records in *Amazon* customers' product-review data for movies and *TV* is given by $n = 4{,}607{,}047$. Both of them do not contain duplicate items. Respectively, the description of Apps for *Android*, and movies and *TV* datasets is provided in Table 7.

Table 7. The description of apps for Android, and movies and TV datasets

Dataset Object	Apps for Android (A)	Movies and TV (B)
Number of reviews	2,638,273	4,607,047
Number of classes	5	5
Number of users	94,5148	1,633,591
Number of users with > 50 reviews	501	4319
Average length of a review (words)	115	145
Timespan	2011-2016	2012-2016

The data consist of more than six- and five-year files for individual product categories. For both datasets, the customers' product-review fields were extracted. An example of the review text is presented below:

```
["reviewerID": "AUI0OLXAB3KKT", "asin": "B004A9SDD8",
"reviewerName": "A Customer", "helpful": [0, 0], "reviewText": "Glad
to finally see this app on the android market. My wife has it on her
iPhone and iPad and my son (15 months) loves it! Hopefully more apps
like this are on the way!", "overall": 5.0, "summary": "Great
app!!!", "unixReviewTime": 1301184000, "reviewTime": "03 27, 2011"],
```

where *reviewerID* – id of the user; *asin* – identification number; *reviewerName* – name of the user; *helpful* – fraction of users who found the review helpful, present users feedback about the quality and helpfulness of review; *reviewText* – a written customer review about the product; *overall* – a rating given by the customer for the product (ratings from 1 to 5 are used in this research: 1 is the lowest evaluation, and 5 is the best; the review meaning is presented in Table 8); *summary* – gives an abridged version of the customer's review or subject matter; *(unix)ReviewTime* – (*Unix*) time of the review. Only *overall* and *reviewText* (review text) data fields were used in the experimentation.

Table 8. Review meaning

Rating	Meaning
1 star	I hate it
2 stars	I do not like it
3 stars	It's OK
4 stars	I like it
5 stars	I love it

The data consist of different customer reviews given by $D = \{d_1, d_2, d_3 \dots d_n\}$, where n is the total number of reviews. These reviews are classified by different customers, having a certain category assigned to the review with a rating numerical value of $C = \{c_1, c_2, c_i \dots c_5\}$, where C_i ($C_i = i$, where i is a class index), *m* is the total number of classes ($m = 5$) considered as a label or class.

3.2. Statistics of datasets

These datasets represent some product and services related to brand, movie, and TV episode names. One can argue that these two datasets are very similar and related to some categories, another can argue that the meaning of a word depends on the context, e.g., words are derived from the context in which they are used. For instance, *Android* mobile software applications are designed to run on mobile devices such as a smartphone or tablet computers. According to the Statista website, there are around 600.000 *Android* apps in the *Amazon* App store[9] today. In comparison, nowadays there are around 3.5 million *Android* apps in the market[10]. On the other hand, a movie and TV as media are expressed through a video stream content delivered via Internet, television, or the cinema. In *Amazon* store, there are subcategories of Movies, TV Shows, *Blue-ray*, 4K Ultra HD, Best Sellers, Today's Deals, New Releases, Pre-orders, Kids and Family, *Amazon* Video, and Trade-In. According to the IMDb database statistics, currently, there are approximately 4.3 million titles that contain subcategories of Movies and TV worldwide[11]. The *Global Internet Phenomena* report has concluded that *Amazon* Video is now the third-ranked downstream application in North America and contains more than 80 million of video users in the United States[9]. As to 2017, *Amazon* offered more than 22 thousand movies and TV shows[9]. For both review datasets, the given words represent opinions that exist in the given context. So, for further investigation, expectedly the word population covers the meaning of words that represent not only commonly used words, but also special words in the given context in which they are used, e.g., game, app, play, movie, fun, trailer, etc. So, most probably, another review set will contain specific words that hold the meaning of the context in which they are used. These datasets can also be considered as categories in domains such as entertainment, food and drinks, health and beauty, retails, travel and vocations, or miscellaneous, etc. For instance, the movie and TV dataset fall into the category of entertainment. Thus the *Android* app dataset falls into the category retail as an electronic device that includes smartphone or tablet devices and all the related applications. On the other hand, applications themselves can also be a part of entertainment category, because some apps are related to the video games and other similar activities. Nevertheless, according to the given data, some overlapping categories can imply a model and the classification accuracy thereby. This is because some

[9] More details: https://www.statista.com

[10] More details: https://www.appbrain.com

[11] More details: http://www.imdb.com/stats

words that represent the context very well in one category will have no or other meaning in another category. If possible, for better classification results and less confusion in the model, disjointed categories should be considered. Otherwise, overlapping categories will create a negative effect on the predictions of classification accuracy. Summing up, the selected datasets are theoretically well categorized, and the meaning of words is derived from the context in which they are used. Such a conclusion can be argued by TOP10 words that are nearly constant for the given datasets A and B. TOP10 words are presented in Table 9.

Table 9. TOP10 words per class

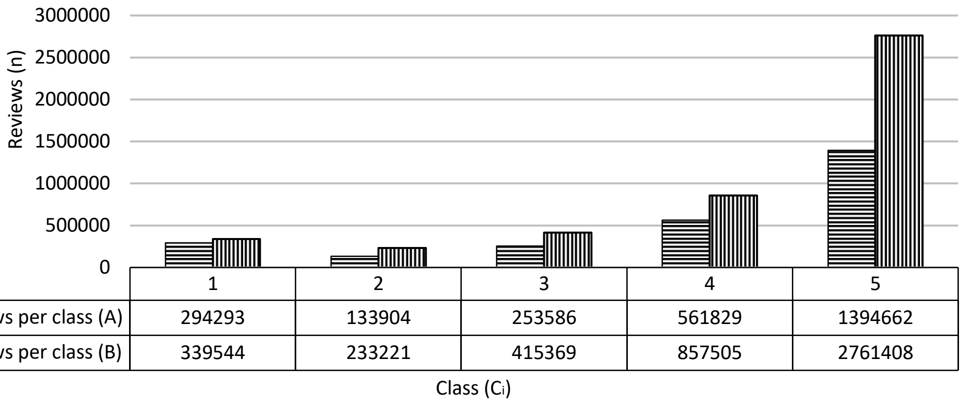
Dataset A

No	Class1	Class2	Class3	Class4	Class5
1	game	game	game	game	game
2	not	not	not	fun	love
3	app	app	play	play	play
4	get	play	like	like	app
5	play	get	app	app	fun
6	would	like	get	not	great
7	time	would	fun	get	like
8	work	time	would	good	get
9	kindle	work	time	time	not
10	like	kindle	good	great	time

Dataset B

No	Class1	Class2	Class3	Class4	Class5
1	movie	movie	movie	movie	movie
2	not	not	film	film	one
3	film	film	not	not	love
4	one	one	one	one	film
5	like	like	like	like	not
6	dvd	would	good	good	great
7	watch	good	would	great	watch
8	would	get	get	watch	like
9	time	watch	time	time	dvd
10	get	time	watch	get	time

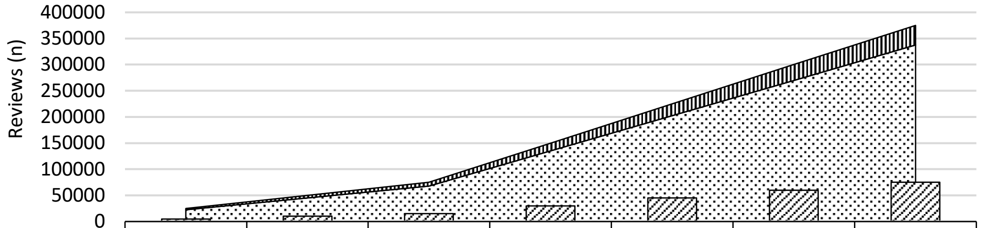
Only a representative part of text corpus was used. Nevertheless, the size of data can be scaled and the infrastructure, adjusted horizontally as *Apache Spark*, can perform classification tasks in hundreds of nodes with memory intense computing resource allocation. Therefore, the data class distribution C_i in a dataset is presented in Fig. 23.



	1	2	3	4	5
▤ Reviews per class (A)	294293	133904	253586	561829	1394662
▥ Reviews per class (B)	339544	233221	415369	857505	2761408

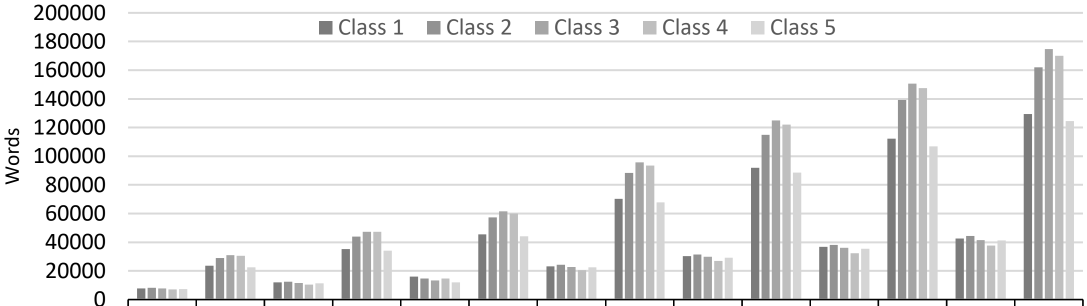
Fig. 23. Distribution of customers' product-reviews by classes

To improve the classification, it was decided to split the data to equally distributed sets per each class and to use the method for measuring the skewness of data [72], so that each class would collect an equal number of customer product-review records. Seven equally distributed datasets *DS1, DS2, DS3, DS4, DS5, DS6, DS7* of various sizes were used in the experiments. A composition of a dataset for training and testing is distributed as follows: 90% for training and 10% for testing, and the equal number of reviews per class. The composition of datasets for training and testing is illustrated in Fig. 24.



	DS1	DS2	DS3	DS4	DS5	DS6	DS7
▥ Testing	2500	5000	7500	15000	22500	30000	37500
⊡ Training	22500	45000	67500	135000	202500	270000	337500
▨ Reviews per class	5000	10000	15000	30000	45000	60000	75000

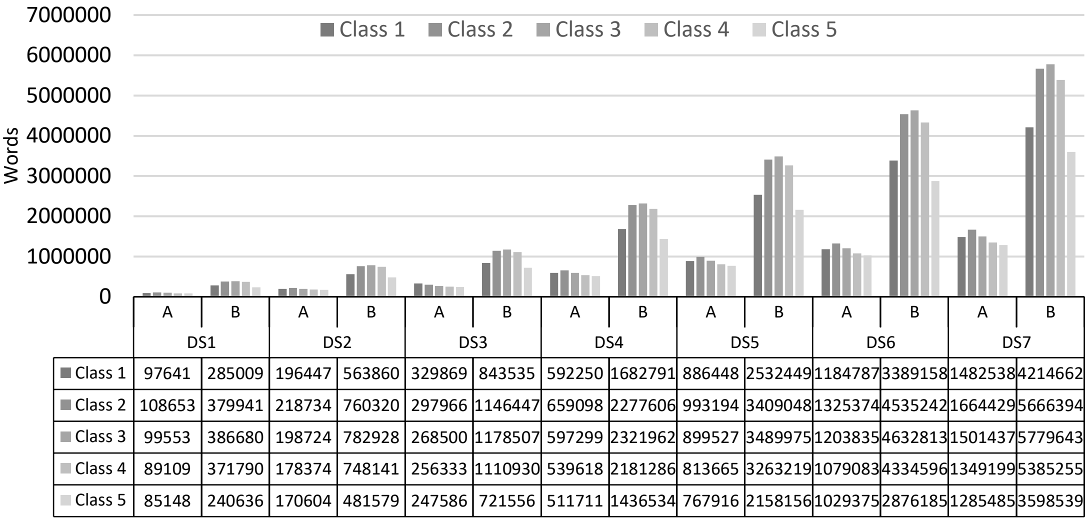
Fig. 24. Composition of datasets for training and testing



	DS1		DS2		DS3		DS4		DS5		DS6		DS7	
	A	B	A	B	A	B	A	B	A	B	A	B	A	B
■ Class 1	7788	23569	12088	35261	15933	45596	23267	70243	30284	91972	36696	112173	42527	129496
■ Class 2	8151	29026	12439	43996	14773	57282	24380	88302	31492	114902	38099	139156	44344	161914
■ Class 3	7758	30882	11564	47336	13332	61573	22766	95607	29897	124892	36009	150617	41515	174809
■ Class 4	7019	30473	10492	47248	14578	60071	20589	93449	27026	122090	32365	147558	37740	170072
■ Class 5	7373	22491	11244	34131	12049	44090	22474	67906	29135	88643	35424	106901	41194	124471

Fig. 25. Unique words (terms) per class in A and B datasets

Correspondingly, Fig. 25 and Fig. 26 present the statistics of the unique and total words (terms). All unique words are counted in comparison to all the words, existing in the given dataset per each class. In general, the selected text corpus has unique words that consist of less than 10% of total words and the distribution of unique words have higher values in class 2 and lower in class 4. Usually, the unique words represent the given class very well, and reasonable similarities exist between 1 and 2, 4 and five classes.



	DS1		DS2		DS3		DS4		DS5		DS6		DS7	
	A	B	A	B	A	B	A	B	A	B	A	B	A	B
■ Class 1	97641	285009	196447	563860	329869	843535	592250	1682791	886448	2532449	1184787	3389158	1482538	4214662
■ Class 2	108653	379941	218734	760320	297966	1146447	659098	2277606	993194	3409048	1325374	4535242	1664429	5666394
■ Class 3	99553	386680	198724	782928	268500	1178507	597299	2321962	899527	3489975	1203835	4632813	1501437	5779643
■ Class 4	89109	371790	178374	748141	256333	1110930	539618	2181286	813665	3263219	1079083	4334596	1349199	5385255
■ Class 5	85148	240636	170604	481579	247586	721556	511711	1436534	767916	2158156	1029375	2876185	1285485	3598539

Fig. 26. Total words (terms) per class in A and B datasets

Fig. 27 illustrates the ratio (the ratio of unique words to total words per class) of total words and unique words per class in A and B datasets. Absolute rates are similar; however, datasets are increasing.

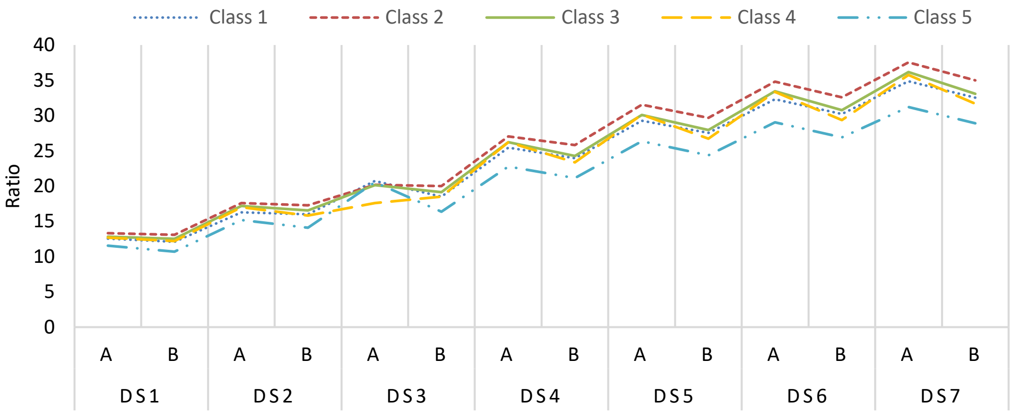


Fig. 27. The rates of total words and unique words (terms) per class in A and B datasets

How is the sampling of used datasets good and reliable? Assuming that a set of comments represents all viable options. The selected population should cover the existing context in the defined population as widely as possible. The different sets represent different contexts and product categories. However, they represent the same object – product-reviews. Also, there are other sources of short-text message types, e.g., *Twitter*. *Twitter* messages are also used very often for sentiment analysis; however, they are enriched with hashtags or emoticons, more formal, formative, very short, and less emotional. Based on this discussion, the experimentation and findings can be expanded by using other datasets.

3.3. A framework of data analysis for text classification

In research, the infrastructure of a data processing cluster (Fig. 28) has been used during the experimentation. Data-classification tasks will be completed by using the *MLlib* library on the *Apache Spark* [69] computing platform with the configuration as follows:

- The infrastructure was created on the *Google Cloud* Platform. Experiments were done using *Apache Spark v1.6.2* [69], *Python v2.7.6* [73] and *NLTK* v3.0.0 [43].

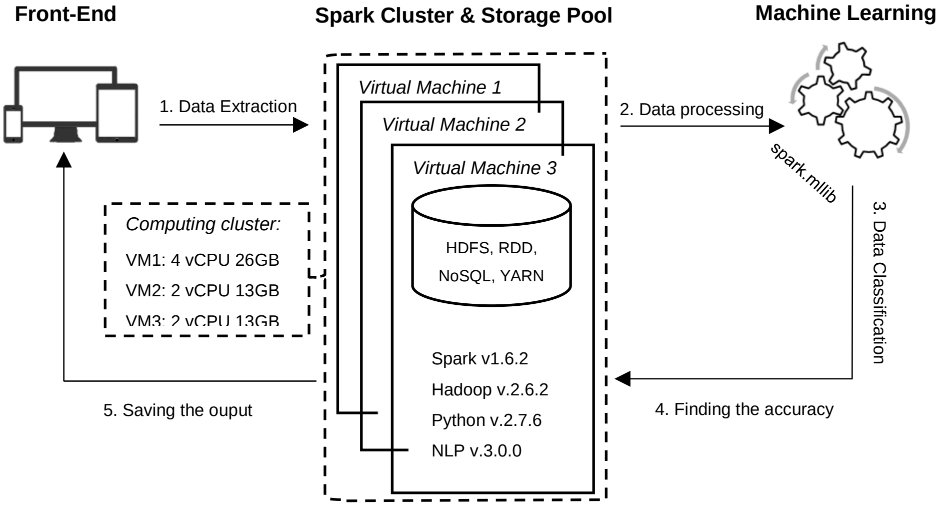


Fig. 28. Infrastructure of the Apache Spark data-processing cluster

- Fig. 28 presents the *Apache Spark* model infrastructure of data processing cluster including the machine learning toolkit *MLlib* that was installed on *Google Cloud* platform. The master node consists of 2 vCPUs (virtual central processing unit) and 26 GB of memory, and two worker nodes have 2 vCPUs, with 13 GB of memory on each, and 3 clusters of the same level have been used to process the tasks faster.

- Apache Spark is an in-memory computing platform designed to be one of the fastest computing frameworks able to run various kinds of computing tasks. Such intensive in-memory computations will allow us to solve big data multi-class text-classification tasks.
- The fixed size computing cluster was used, but technical capabilities of the cloud computing technology can scale the cluster proportionally to the size of data, including the costs for the given services. All the experiments were carried out using the infrastructure that has been described in this section.

3.4. Workflow model

The workflow model for product-review processing in Fig. 29 was established to compare *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine,* and *Logistic Regression, Multilayer Perceptron* methods. This model is a modified version of that presented by *Seddon* [74].

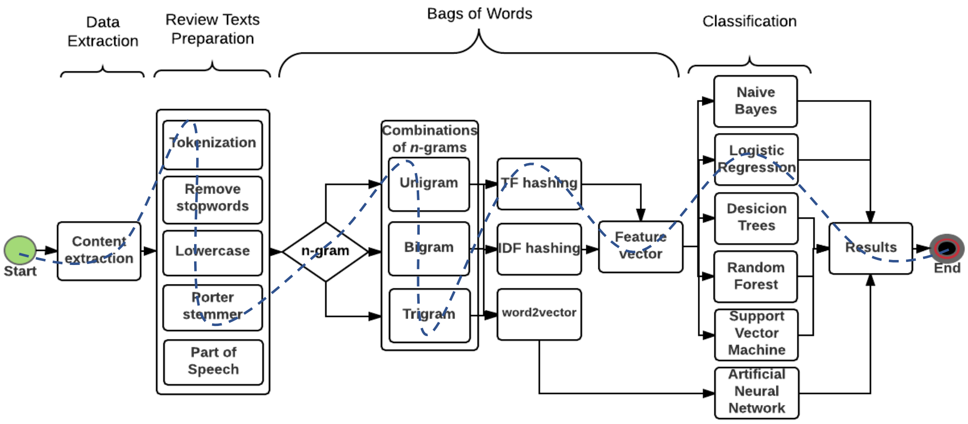


Fig. 29. Workflow model for product-review processing

The workflow model for product-review processing has been used in this research, and the highlighted path will show the best-performed classification method with used data feature selection. In the following sections, the workflow model is described for review processing. The workflow consists of four key stages: data extraction, review text preparation (noise reduction), a bag of words, the transformation of the text to a sparse vector and application of classification algorithms. This workflow consists of stages as follows:

- Firstly, the main goal of the data extraction stage is to take only the necessary and related data from the data source.
- Secondly, to prepare data for the machine learning stage. This stage is carried out by data conversion that includes transformation and application of data feature selection (features are explained in the following sections). The key point is to collect major data fields the most important one of which is reflecting in the required data science investigation.
- Thirdly, the main goal of the machine learning stage is to implement the machine learning process, based on the investigated data. This stage enables us to apply classification methods by training the model and by making the following predictions on the new data. This stage can be enriched with an additional visualized solution that helps tell the story behind the data, i.e., data statistics or machine learning results, etc.
- Data processing for all the presented algorithms is the same, so, in this chapter, a unified data pre-processing process is defined. Data pre-processing is the first and initial stage that must be considered at the beginning of engineering of the data analytics solution.

All the steps of this model will be explained in the following parts of this chapter.

3.4.1. Data extraction

The main goal of this stage is to select only the necessary and related data fields to process the data and optimize memory usage. This stage was carried out as follows:

- Access to the dataset file that includes product-reviews.
- Datasets are containing a label, author, id, text_review, etc.
- Selecting only the required *overall* and *review text* fields from input dataset.
- Collecting the equal number of customer product-review records in each class, e.g., *skewness method*.

3.4.2. Preprocessing of review texts

The main goal of this stage is to prepare *review text* fields for extraction of features. This stage was fulfilled by with applying data feature selection as follows:

- Removing punctuations and tokenizing each single word by white space.
- Removing stop-words (stop-words corpus was taken from the *NLTK* website [43]), such as *a* and *the.* Stop-words *a* and *the* have often been in use in any text, but not include specific information required to train this data model. The removal of stop-words is not entirely necessary, because the product-reviews are short, and using techniques such as TF-IDF their influence could be reduced.
- Converting all capital letters to a lower case.
- Stemming and reducing inflectional forms to the common base form of stemma. The *Porter* stemming algorithm is applied [46]. Stemming not necessarily improves the quality of the classifier, so its impact should be experimentally tested.

3.4.3. Bags of words

The *n*-gram method as a sequence of written words of length n is applied to construct bags of words.

- It splits a sentence into words and groups them using a predefined combination of *n*-grams. Bags of words (unigrams, bigrams, trigrams, and the combination) are created from review texts that have passed the previous stages, based on the selected *n*-gram model. *N*-grams are a contiguous sequence of n items (tokens) from a given sequence of text.
- Also, the application of combined *n*-grams is used, and an example is presented in Table 10 (an example of *n*-gram compositions: *better chance enjoy good weather*). For instance, the meaning of bigram "not good" can change sentimentally the meaning of two separated individual unigrams "not" and "good".
- Instead of building *n*-grams from the sentences, a continuous text flow is in use. This is because the task of a classifier is not to attempt to understand the meaning of a sentence, it creates the input to the classifier with all the features (tokenized terms, and term groups), the classifier creates a model that assigns the class as accurately as possible.
- Applying part-of-speech tagging, every term gets a tag whether it is an adjective, noun, verb, etc.

Table 10. An example of n-gram compositions

Model	Example of *n*-grams
unigrams (n = 1)	*['better', 'chance', 'enjoy', 'good', 'weather']*
bigrams (n = 2)	*['better chance', 'chance enjoy', 'enjoy good', 'good weather']*
trigrams (n = 3)	*['better chance enjoy', 'chance enjoy good,' 'enjoy good weather']*
Combination of *n*-grams	
unigrams (n = 1) and bigrams (n = 2)	*['better', 'chance', 'enjoy', 'good', 'weather', 'better chance', 'chance enjoy', 'enjoy good', 'good weather']*
unigrams (n = 1)	*['better', 'chance', 'enjoy', 'good', 'weather', 'better chance', 'chance enjoy', 'enjoy good',*

Model	Example of *n*-grams
and bigrams (n = 2) and trigrams (n = 3)	*'good weather', 'better chance enjoy', 'chance enjoy good', 'enjoy good weather']*

3.4.4. Construction of a feature vector

This stage was accomplished by transforming a bag of words into a feature vector as follows:

- Encoding every single word into a numerical value. These words are imported to a specially created term frequency hashing or an inverse document frequency hashing vectorizer with the length of $x = 2000$, which counts the frequency in the set and assigns a unique numerical value to the classification stage.
- The term frequency counting identifies how important a word is to a review in a corpus, i.e. the key as a word and value as the number of frequency in the given review set. Each term gets a unique frequency parametrical value.
- In the case of word embedding a skip-gram model is applied to *Multilayer perceptron classifier.*

3.4.5. Text classification

This stage was fulfilled as follows:

- The proposed block diagram for text classification has two main sections. The first one contains an algorithm for feature selection with *n*-grams, part of speech, term frequency, etc. The second one evaluates the performance of the selected classification method.
- The block diagram of the algorithm for text data classification, which is based on the previous descriptions is illustrated in Fig. 30.
- Dividing a dataset into two groups roughly 90% for training and 10% for testing data.

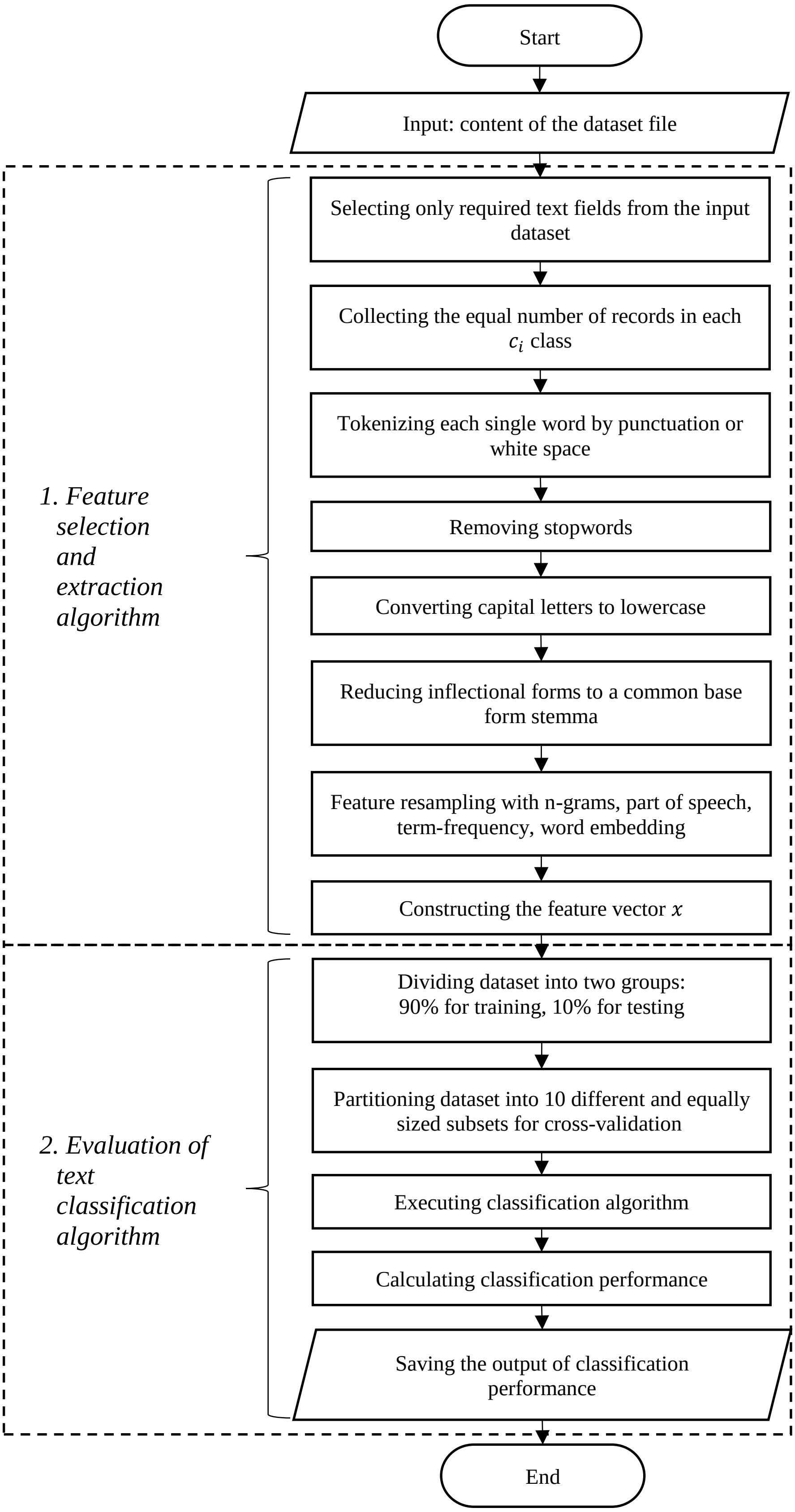

Start
Input: content of the dataset file
1. Feature selection and extraction algorithm
Selecting only required text fields from the input dataset
Collecting the equal number of records in each c_i class
Tokenizing each single word by punctuation or white space
Removing stopwords
Converting capital letters to lowercase
Reducing inflectional forms to a common base form stemma
Feature resampling with n-grams, part of speech, term-frequency, word embedding
Constructing the feature vector x
2. Evaluation of text classification algorithm
Dividing dataset into two groups: 90% for training, 10% for testing
Partitioning dataset into 10 different and equally sized subsets for cross-validation
Executing classification algorithm
Calculating classification performance
Saving the output of classification performance
End

Fig. 30. Block diagram for evaluating classification including feature selection

- The *n*-fold cross-validation method is used. It partitions data into *n* different and equally-sized subsets. Later, every subset is used in the testing phase. The rest of subsets are combined for the training phase to learn how to classify. Textual data training and testing are performed by the selected classification method using 10-fold cross-validation.

Evaluation of text classification algorithm procedure is presented below:

- Executing classifiers: *multinomial Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine* with a linear kernel and the *Stochastic Gradient Descent* optimization algorithm [75], *Logistic Regression* with a limited memory used with the *Broyden–Fletcher–Goldfarb–Shanno* optimization algorithm [76], and the *Multilayer Perceptron* algorithm.
- The classification methods were used mostly with their standard hyperparameters [77] (Table 11) that are configured in the *Apache Spark v1.6.2 MLlib* library, except the number of features (3000), trees (50) and depth (30) – these were customized according to the size of the data and limitations associated with the use of computing resources with Random Forest. The number of features as input (200), hidden layer 1 (20), hidden layer 2 (10), and output (5) for the *Multilayer perceptron classifier* were in use.

Table 11. Hyperparameters of the used classification methods

Methods Parameters	LR	SVM	RF	DR	NB	MP
Training data	*RDD of LabeledPoint*	*RDD of LabeledPoint*	*RDD of LabeledPoint*	*RDD of LabeledPoint*	*RDD of LabeledPoint*	*RDD of LabeledPoint*
Dimension of the features (inputs)	*2000*	*2000*	*2000*	*2000*	*2000*	*2000*
Number of classes (outputs)	*5*	*5*	*5*	*5*	*5*	*5*
Number of iterations	*100*	*100*	*100*	*100*	*100*	*100*
Initial weights	*None*	*None*	*None*	*None*	*None*	*None*
Smoothing (lambda)	-	-	-	-	*1,0*	-
Step size	-	*1,0*	-	-	-	*0,03*
Regularizer parameter	*0,01*	*0,01*	-	-	-	-

Methods Parameters	LR	SVM	RF	DR	NB	MP
Fraction of data to be used for each SGD iteration	-	*1,0*	-	-	-	-
Type of regularizer	*"l2" for L2 regularization*	*"l2" for L2 regularization*	-	-	-	-
Boolean (for interception)	*False*	*False*	-	-	-	-
Number of corrections	*10*	-	-	-	-	-
Convergence tolerance of iterations	*1e-4*	-	-	-	-	*1e-6*
Boolean (for validation)	*True*	*True*	-	-	-	-
Condition that determines iteration termination (convergence)	-	*0,001*	-	-	-	-
Number of trees	-	-	*50*	*50*	-	-
Maximum depth of the tree	-	-	*30*	*30*	-	-
Number of features to consider for splits at each node	-	-	*sqrt*	-	-	-
Maximum number of bins used for splitting features	-	-	*32*	*32*	-	-
Internal layer activation function	-	-	-	-	-	*sigmoid*
Output layer activation function	-	-	-	-	-	*softmax*
Block size	-	-	-	-	-	*128*
Hidden layer	-	-	-	-	-	*2*

3.5. Evaluation of the classification performance

The average classification accuracy formula for test data is presented in the section.

- The classification accuracy is calculated by the actual class that is equal to the predicted class, divided according to by the total corpus size into test data. The purpose of creating a model or classifier is not to classify the training set, but to classify the data whose class of which is not known. The data must be classified correctly, but often there is no way of to find out whether the model acts like that. If the nature of data changes over time, for instance, if we are trying to detect negative product-reviews, then we need to measure up the performance over time. For example, in the case of negative product-reviews, the rate of negative product-reviews that were not classified as negative. The accuracy depends on the number of *n*-grams, part-of-speech data feature selection, and the number of product-review sets is presented in the sections of Chapter 4.

- The classification accuracy is calculated by actual labels that are the same as to the predicted label divided by the total corpus size into test data. The selected average accuracy formula for multi-class classification measurement is defined in formula (31). Other measurements such as Error rate, Precision, Recall and F-score are also formally defined in section 2.4.

3.6. Conclusions of Chapter 3

The research methodology in the field of machine learning with textual data classification by the cloud computing technology is presented in this section.

1. Collected data statistics represent the given data features, and a modified version of the data workflow model can be adjusted for classifying short text messages.
2. For a statistical measure of prepared classification experiments, the classification accuracy and additional measurements are selected to examine how precise a multi-class classification assigns the class to the number of instances. Classification accuracy is mainly used as the main control as a depended variable for similar experimental analysis.

Chapter 4 Experimentation and Results

In this chapter, a comparison of research with planning, experimentation, and results is presented. The comparison is based on a combination of data feature selection (*n*-gram, part of speech, (inverse) term frequency), and classifiers (*Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, Multilayer Perceptron*) for multi-class text classification that determines a multi-class classification problem by measuring the classification accuracy of two independent large-scale product-review datasets.

Support Vector Machine with the linear kernel is a very fast method, but it does not always give the best classification accuracy comparing to *Support Vector Machine* with the nonlinear kernels. The training process of *Support Vector Machine* with the nonlinear kernels is difficult to distribute, and therefore, these methods are not yet implemented in the *Apache Spark* machine learning library, used in this experiment. In Chapter 3, an already defined methodology for data-intensive technologies, multi-class classification algorithms, and natural language processing methods are applied.

4.1. Planning an experiment

The planning an experiment is based on an evaluation of selected *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, and Multilayer Perceptron* methods for multi-class text classification including data feature selection: *n*-grams, part of speech, (inverse) term frequency, word embedding.

4.1.1. Formulating a hypothesis and selecting variables

The hypothesis and questions usually assist in defining and planning the experimental analysis properly. The questions and tasks, which have been stated and formulated at the beginning of the dissertation, require experimenting.

- The *hypothesis* is: yes, a combination of data feature selection (combination of *n*-grams) can increase the classification accuracy.
- Based on this hypothesis, I predict the use of a combination of data feature selection will result in the highest classification accuracy with the selected classification methods for the given product-review data in comparison to a baseline classification method.
- In a scientific experiment, it is important to choose some independent variables as the factor that will be changed during the experiment. Thus, dependent variables as the factor will change predictably. Let us define the function f, and the output variable A_i as a classification accuracy formally:

$$A_i \ = f\ (C_{methods}, Features, NLP_{technics}, D_i,), \tag{36}$$

where independent variables are defined for this experiment as follows:

- Classification methods *($C_{methods}$): Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, Multilayer perceptron.*
- Data feature selection *($Features$):* *n*-grams, part of speech, term frequency, inverse document frequency, word embedding.
- Noise reduction ($NLP_{technics}$): tokenization, stop-words, lowercasing, term-normalization (stemming), etc.
- Datasets *(D_i):* classification experiments are performed on two independent datasets (A and B) and statistically measured according to the classification accuracy, and in addition with Error rate, Precision, Recall, F1 measurement.

This experiment is done in 4 experimental cycles and each cycle the independent variables, e.g., classification methods, and data feature selection is used so as defined in Table 12. For the 2nd, 3rd, and 4th experimentation cycles only 3-4 best-performed classification methods are selected.

Table 12. Experimental cycles

Variables Cycles	$C_{methods}$	$Features$	$NLP_{techniques}$
1st	*Baseline,* *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, Multilayer Perceptron*	1. Term frequency (except *Multilayer Perceptron* with word2vect) 2. Unigrams, Bigrams, Trigrams 3. Combinations of *n*-grams	Tokenization, lowercasing, stop-words, stemming
2nd	*Naïve Bayes,* *Support Vector Machine, Logistic Regression*	1. Inverse document frequency 2. Unigrams and combinations of *n*-grams	Tokenization, lowercasing, stop-words, stemming
3rd	*Naïve Bayes,* *Support Vector Machine,* *Logistic Regression* *Multilayer Perceptron*	1. Inverse document frequency (except *Multilayer Perceptron* with word2vect) 2. Unigrams and combinations of *n*-grams 4. Part of speech tagging	Tokenization, lowercasing, stop-words, stemming
4th	*Naïve Bayes,* *Support Vector Machine,* *Logistic Regression*	1. Inverse document frequency 2. Unigrams and combinations of *n*-grams	Tokenization, lowercasing, w/o stop-words removal, stemming

In the 4th experimental cycle, it was considered to keep high-frequency words without applying the stop-word removal function to perform the usefulness of stop-words with the inverse document frequency experimentally. As it is not necessarily useful to remove stop-words when an inverse document frequency data feature selection is in use. Stop-word removal will delete some words that might be very relevant to the classification task.

4.1.2. Controlling an experiment and collecting data

In a scientific experiment, it is important to define control standards of comparison that will treat all experiments the same way as:

- For this experiment, the classification accuracy is used as the main control of a dependent variable. Also, in the experiments, the parameters of Error rate, Precision, Recall, and F1-measurement were measured.

- The comparison of the classification accuracy of multinomial *Naïve Bayes*, *Random Forest*, *Decision Tree*, *Support Vector Machine* with the linear kernel and the *Stochastic Gradient Descent* optimization algorithm [75], and *Logistic Regression* with the limited memory *Broyden–Fletcher–Goldfarb–Shanno* optimization algorithm [76], *Multilayer Perceptron* classification methods, related to the classification accuracy, the number of product reviews, and a combination of *n*-grams, respectively, for both datasets.
- Datasets are divided into two groups, roughly 90% in training and 10% in test data.
- The classification methods were used with their default parameters that are configured in the *Apache Spark v1.6.2 MLlib* library.

The experiment is conducted in the same manner to ensure that the collected data meets the satisfy conditions for all compared experimental groups and provide the same conditions for all test posts:

- Collections of equally distributed number of customer product-review records in each class were in use.
- 10-fold cross-validation technique was in use to validate the evaluation of classification experiments, results, and conclusions.
- Hardware and software environment, defined in section 3.3, was used as well.
- The value of classification accuracy and other values of predefined measurements for each of the test pots were calculated at the end of each experiment.
- Details on the performance of classification accuracy and other predefined measurements of experiments are presented in charts and tables.
- Summarized results are presented in the following section 4.3 of this chapter.

4.2. Analyzing results

The results of the experiment are presented in the bar charts, graphs, and tables so that they can be easily analyzed:

- On the x-axis (the horizontal axis), a combination of n-gram or classification methods as an independent variable is presented.
- On the y-axis (the vertical axis) the classification accuracy as a dependent variable is presented. This is the main factor that is measured during the experimental analysis.
- In advance, Error, Precision, Recall, and F1-measurement classification performance metrics are used to measure the classification performance.

4.2.1. Comparing against a baseline

Comparing the classification methods, there is a need to include a baseline classifier which will show that the selected advanced classification methods function significantly better in comparison to the baseline. The basic model is a chance baseline that assigns a classification label randomly. If having a certain category assigned to the product-review with a numerical rating value of $C = \{C_1, C_2, C_i \dots C_5\}$ where C_i ($C_i = i$, where i is a class index), m is the total number of classes ($m = 5$) and considered as a class. The equal number of product-review records per each class is collected. So, theoretically randomly classifying each product-review instance as either 1 or 5, after performing the n-fold cross-validation method, in the end the value of classification accuracy will be around 20% right just by chance of any out of 5 classes.

4.2.2. Decision Tree

The 1st experimental cycle: *term frequency, and n-grams (unigrams, bigrams, trigrams, combinations)*

A more advanced classification method as compared to the *baseline method* is *Decision Tree*. The classification accuracy results (Fig. 31, Table 13) of *Decision Tree* are presented in this section. The results of *Decision Tree* classification accuracy were the lowest (min in trigram: 24.10%, max in uni, bi, tri-gram: 34.58%) as compared to the classifiers analyzed.

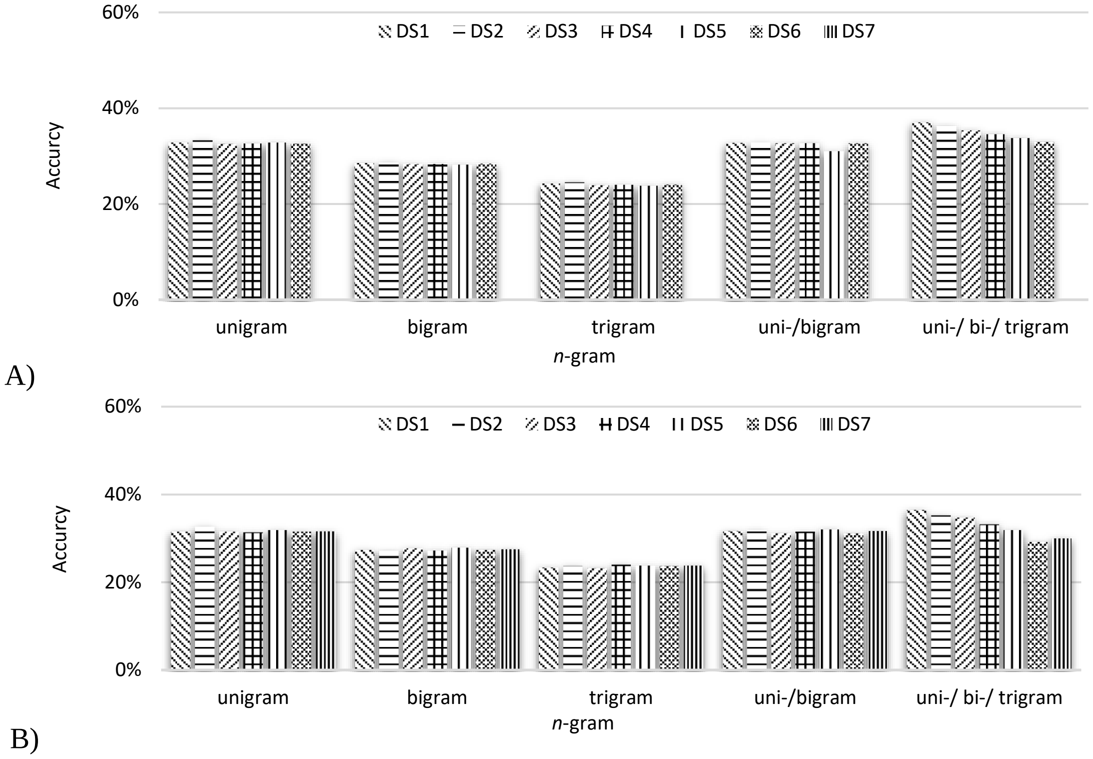


Fig. 31. Classification accuracy of DT in the 1st experimental cycle

Table 13. Average performance measurements of DT in the 1st experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	32,75	67,25	32,02	32,32	32,17
bigram	28,40	71,60	27,77	28,03	27,90
trigram	24,10	75,90	23,56	23,79	23,67
uni-/bigram	32,48	67,52	31,75	32,05	31,90
uni-/bi-/trigram	**34,58**	65,42	33,81	34,13	33,97
Dataset B					
unigram	31,73	68,27	32,44	32,75	32,60
bigram	27,45	72,55	28,07	28,34	28,20
trigram	23,65	76,35	24,18	24,41	24,30
uni-/bigram	31,56	68,44	32,27	32,58	32,42
uni-/bi-/trigram	**32,95**	67,05	33,70	34,01	33,85

More cycles by the *Decision Tree* classifier were not tested, because the existing results have shown that classifier is not functioning very well comparing to other methods, even though the results are higher (min in trigram: 4.10%, max in uni, bi, tri-gram: 14.58%) in comparison to the baseline classifier.

4.2.3. Random Forest

The 1st experimental cycle: *term frequency, and n-grams (unigrams, bigrams, trigrams, combinations)*

In this section, experimental results (Fig. 32, Table 14) of *Random Forest* are presented. As already described previously, *Random Forest* is based on the same idea as *Decision Tree*, but contrarily *Random Forest* does not create a large and deep tree. *Random Forest* creates random feature subsets and builds smaller trees inside itself and calculates the votes.

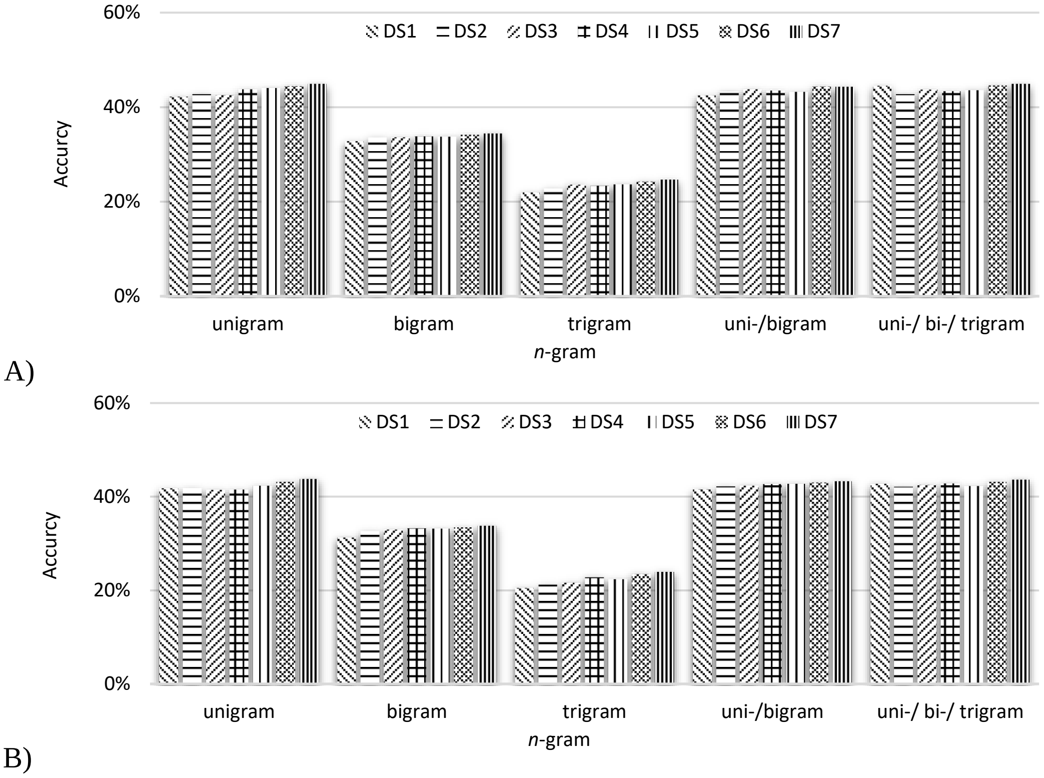


Fig. 32. Classification accuracy of RF in the 1st experimental cycle

The highest importance variable in *Random Forest* becomes the root node in that tree. The performance of classification accuracy of *Random Forest* is higher in comparison to *Decision Tree* and the baseline classifier (min in trigram: 22,26%, max in uni, bi, tri-gram: 43.93%).

Table 14. Average performance measurements of RF in the 1st experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					

n-gram	Accuracy	Error	Precision	Recall	F1
unigram	43,53	56,47	42,56	42,96	42,76
bigram	33,75	66,25	33,00	33,31	33,16
trigram	23,47	76,53	22,95	23,16	23,06
uni-/bigram	43,58	56,42	42,61	43,01	42,81
uni-/bi-/trigram	**43,93**	56,07	42,95	43,35	43,15
Dataset B					
unigram	42,27	57,73	43,23	43,64	43,43
bigram	32,92	67,08	33,67	33,99	33,83
trigram	22,26	77,74	22,77	22,98	22,88
uni-/bigram	42,55	57,45	43,51	43,92	43,72
uni-/bi-/trigram	**42,74**	57,26	43,71	44,12	43,92

Random Forest is one of the slowest classifiers because the performance decreases dramatically if higher parameters of trees and depth are configured higher than 50 and 30, respectively. Increasing these parameters, the objective should usually lead to an increase in the capacity of the classifier and classification accuracy. Nevertheless, more experimental cycles were not experimented, because the existing results have shown that the *Random Forest* classifier is not functioning very well compared to other methods. Thus, this method requires a lot of resources of computing memory and, without providing additional computing resources, the experiments usually take much more time in comparison to other classification methods.

4.2.4. Naïve Bayes

The 1st experimental cycle: *term frequency, and n-grams (unigrams, bigrams, trigrams, combinations)*

The experimental results (Fig. 33, Table 15) have shown that the *Naïve Bayes* classification method for product-review data in the 1st experimental cycle only with dataset A achieves the average of classification accuracy 1 – 2% higher than the *Random Forest* and *Support Vector Machine* method, but the difference is not statistically significant.

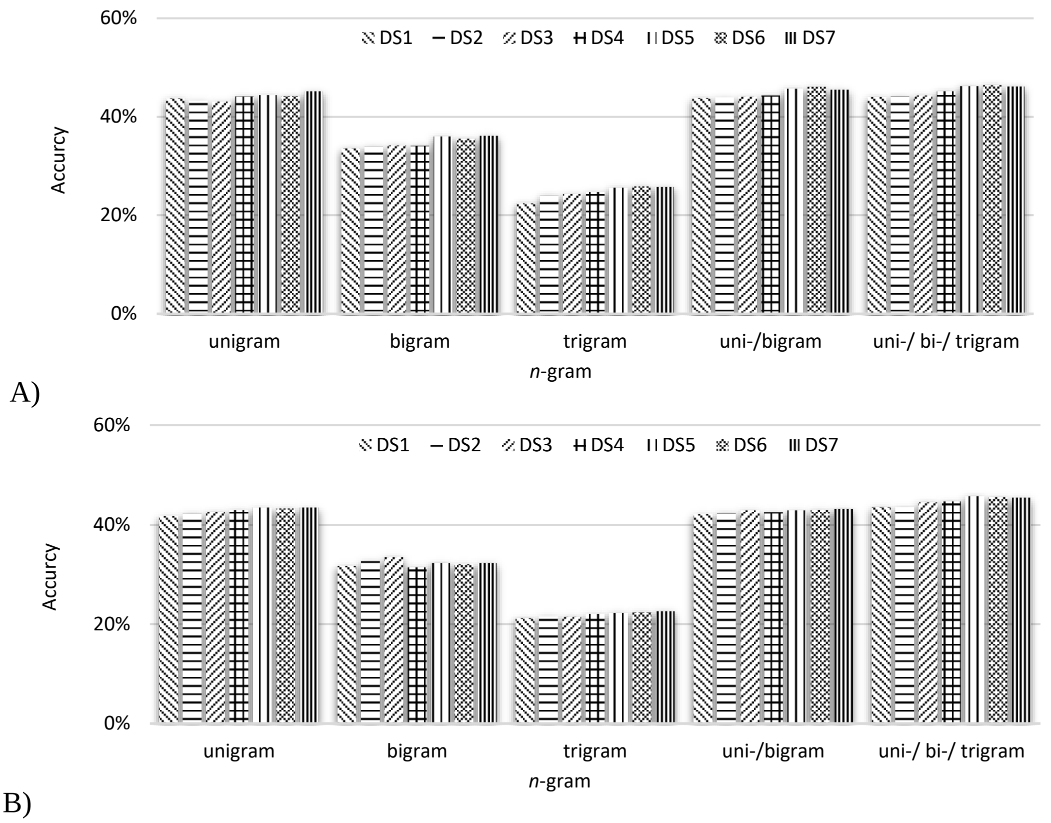


Fig. 33. Classification accuracy of NB in the 1st experimental cycle

Table 15. Average performance measurements of NB in the 1st experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	44,01	55,99	43,03	43,43	43,19
bigram	34,82	65,18	34,04	34,36	34,17
trigram	24,66	75,34	24,11	24,34	24,20
uni-/bigram	44,78	55,22	43,78	44,19	43,95
uni-/bi-/trigram	**45,22**	54,78	44,21	44,63	44,38
Dataset B					
unigram	42,82	57,18	43,79	44,20	43,99
bigram	32,29	67,71	33,02	33,33	33,18
trigram	21,94	78,06	22,44	22,65	22,55
uni-/bigram	42,70	57,30	43,66	44,07	43,87
uni-/bi-/trigram	**44,69**	55,31	45,70	46,13	45,92

The 2nd experimental cycle: *inverse document frequency, and n-grams (unigrams and a combination of n-grams)*

The experimental results (Fig. 34, Table 16) have shown that the *Naïve Bayes* classification method for product-review data in the 2nd experimental cycle does not improve the classification accuracy in comparison to the results collected in the 1st experimental cycle.

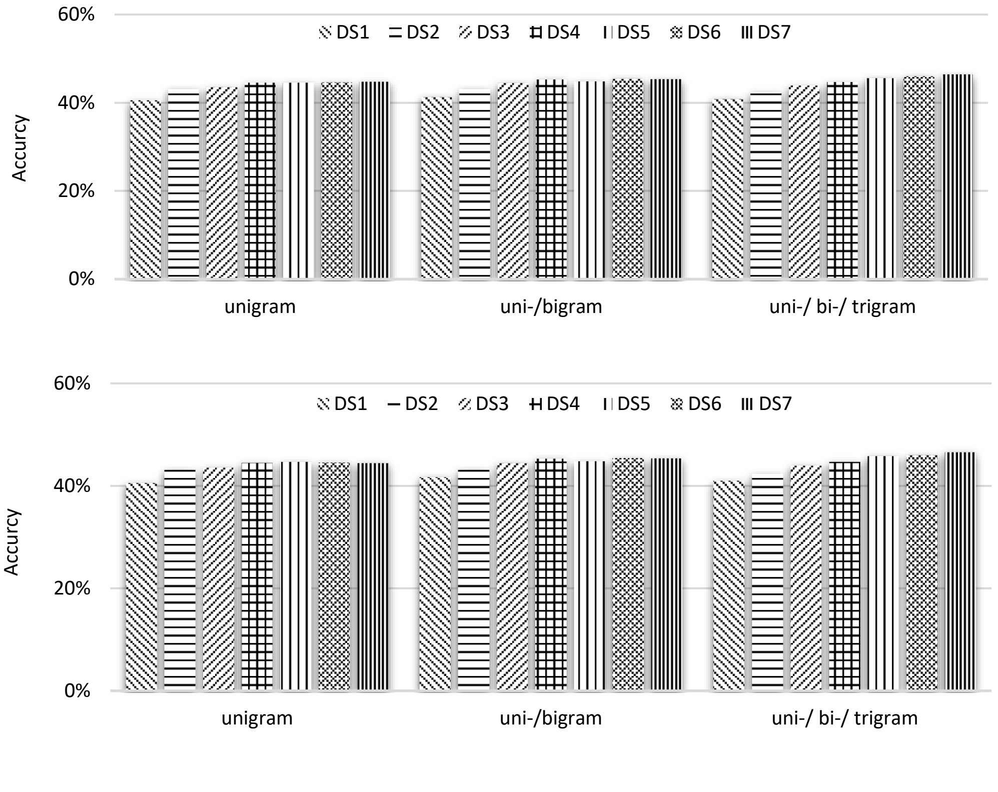


Fig. 34. Classification accuracy of NB in the 2nd experimental cycle

Table 16. Average performance measurements of NB in the 2nd experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	43,67	56,33	42,90	43,67	43,28
uni-/bigram	44,25	55,75	43,64	44,18	43,91
uni-/bi-/trigram	**44,26**	55,74	43,70	44,26	43,98
Dataset B					
unigram	43,63	56,37	43,49	43,65	43,57
uni-/bigram	44,31	55,69	43,71	43,93	43,82
uni-/bi-/trigram	**44,33**	55,67	44,39	44,19	44,29

The average results of uni, bi, tri-gram are higher (44,26-44,33%) by approximately 1,3% in comparison to the results using only unigrams (43,67-44,63%), but these results are not statistically significant.

The 3rd experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams), part of speech*

The experiments and results (Fig. 35, Table 17) by applying part-of-speech tagging are presented. As already defined in the previous sections (2.3.3), with the help of the feature part-of-speech tagging, special labels are used, corresponding to the part-of-speech of every given term in this context.

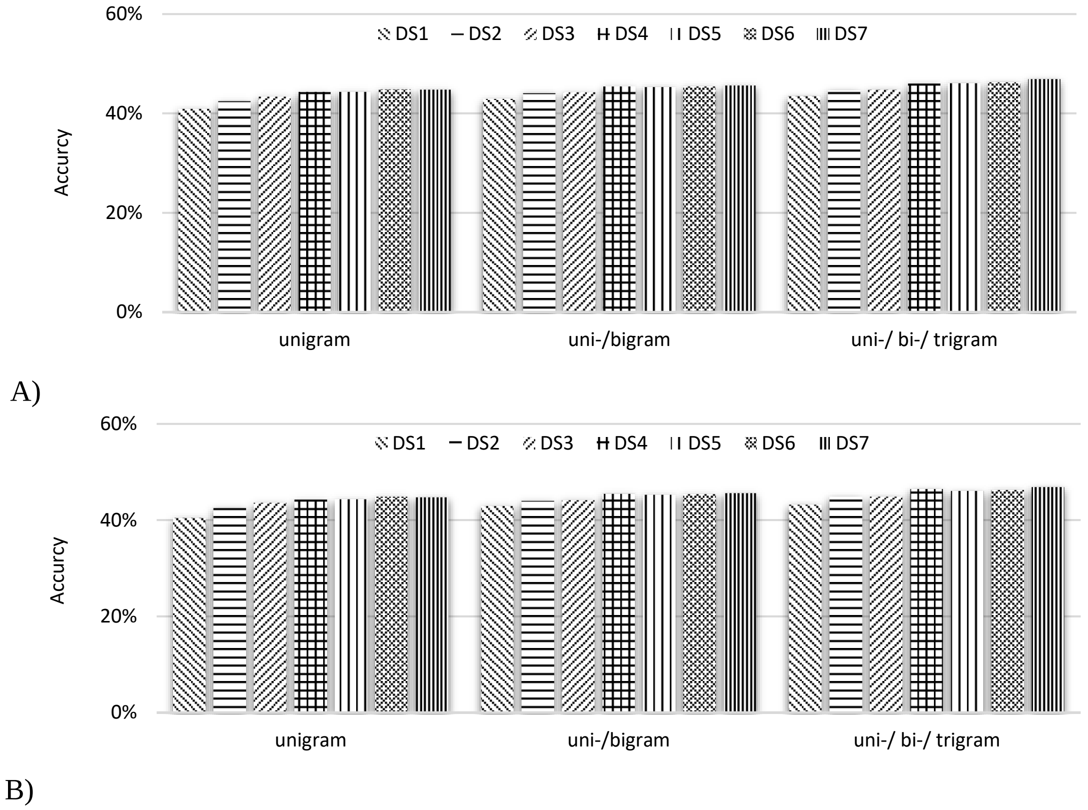


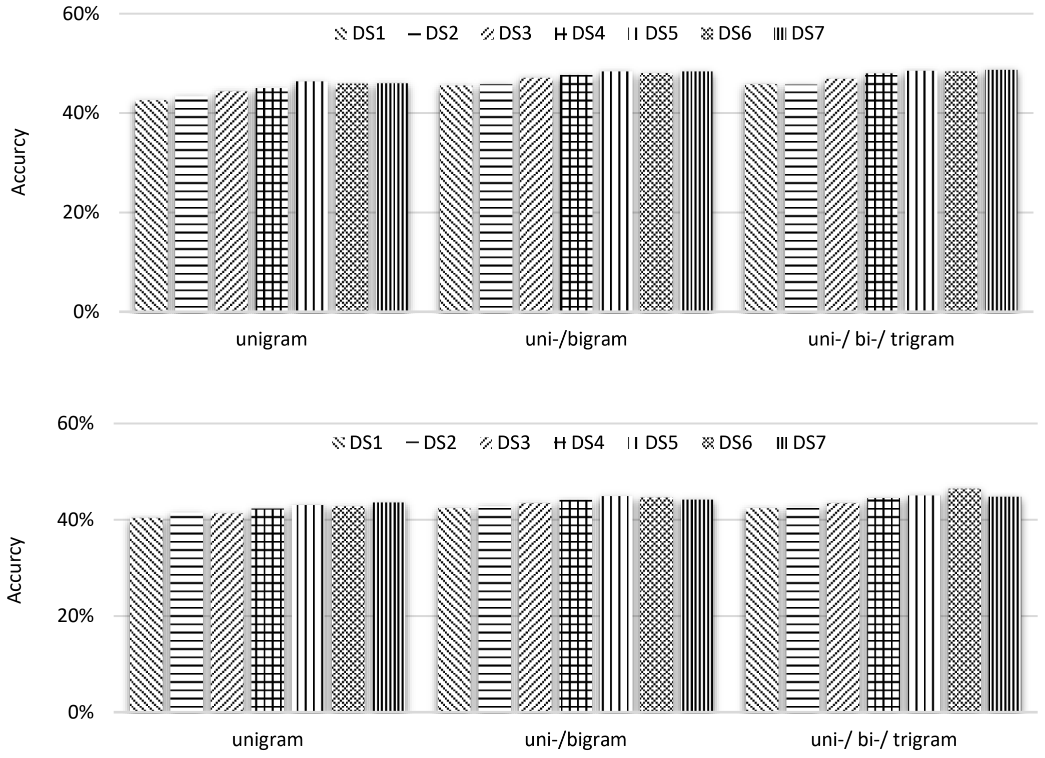
Fig. 35. Classification accuracy of NB in the 3rd experimental cycle

Table 17. Average performance measurements of NB in the 3rd experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	43,58	56,42	42,82	43,59	43,20
uni-/bigram	44,71	55,29	43,94	44,71	44,32
uni-/bi-/trigram	**45,46**	54,54	44,62	45,75	45,18
Dataset B					
unigram	43,55	56,45	43,41	43,57	43,49
uni-/bigram	44,61	55,39	44,01	44,22	44,12
uni-/bi-/trigram	**45,52**	54,48	45,58	45,37	45,47

Part-of-speech tagging identifies all terms as nouns, verbs, adjectives, adverbs. The experimental results have shown that the *Naïve Bayes* classification method for product-review data in the 3rd experimental cycle improves the classification accuracy in comparison to the results, collected in the 1st and 2nd experimental cycles. The average results of uni, bi, tri-gram are higher (45,46-45,52%) by approximately 2% in comparison to the results using only unigrams (43,58-43,55%).

The 4th experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams)*



A)

B)

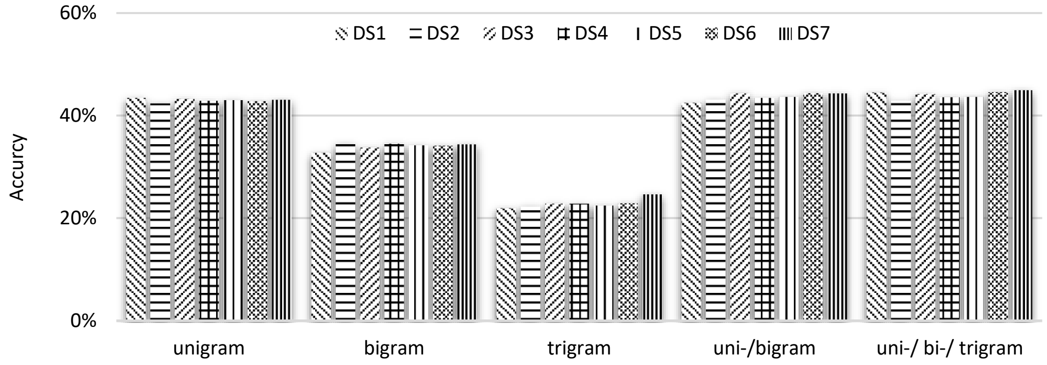
Fig. 36. Classification accuracy of NB in the 4th experimental cycle

Table 18. Average performance measurements of NB in the 4th experimental cycle

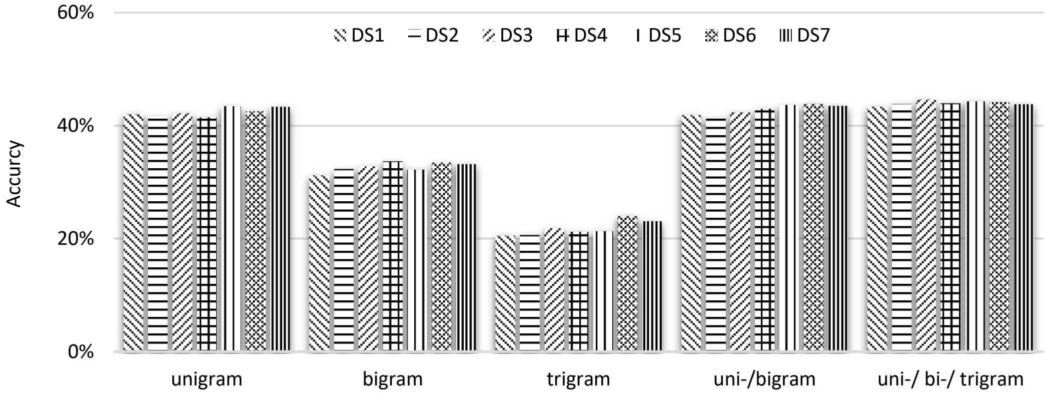
n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	44,79	55,21	44,01	44,79	44,40
uni-/bigram	47,27	52,73	46,59	47,27	46,93
uni-/bi-/trigram	**47,44**	52,56	46,81	47,41	47,11
Dataset B					
unigram	42,13	57,87	40,75	42,13	41,43
uni-/bigram	43,79	56,21	42,42	43,79	43,09
uni-/bi-/trigram	**44,19**	55,81	42,53	43,86	43,18

4.2.5. Support Vector Machine

The 1st experimental cycle: *term frequency, and n-grams (unigrams, bigrams, trigrams, combinations)*



A)



B)

Fig. 37. Classification accuracy of SVM in the 1st experimental cycle

Support Vector Machine with the linear kernel is a very fast method, but it does not always yield the best classification accuracy, comparing to *Support Vector Machine* with the nonlinear kernels. It is difficult to distribute the training process of *Support Vector Machine* with the nonlinear kernels, and therefore, these methods are not yet implemented in the *Apache Spark* machine learning library.

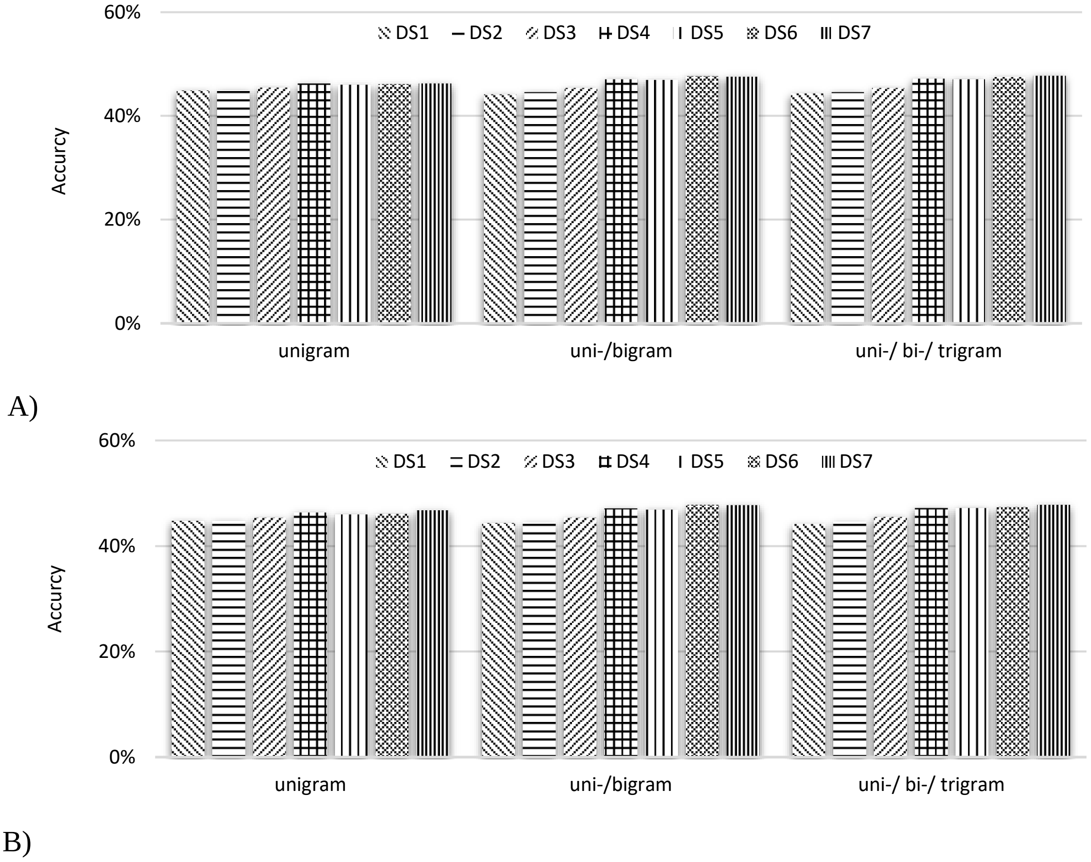
Table 19. Average performance measurements of SVM in the 1st experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	42,99	57,01	42,04	42,43	42,23

bigram	34,06	65,94	33,30	33,62	33,46
trigram	22,84	77,16	22,34	22,55	22,44
uni-/bigram	43,70	56,30	42,72	43,13	42,92
uni-/bi-/trigram	**44,06**	55,94	43,08	43,49	43,28
Dataset B					
unigram	42,37	57,63	43,33	43,74	43,53
bigram	32,69	67,31	33,43	33,74	33,59
trigram	21,81	78,19	22,30	22,51	22,41
uni-/bigram	42,79	57,21	43,76	44,17	43,96
uni-/bi-/trigram	**44,00**	56,00	45,00	45,42	45,21

The experimental results (Fig. 37) have shown that the *Support Vector Machine* classification method for product-review data in the 1st experimental cycle has not outperformed only the *Logistic Regression* method (min 30,43%, and max 58,48%) applying the term frequency data feature selection and achieved min in trigram: 21,81%, max in uni, bi, tri-gram: 44,06% average classification accuracy.

The 2nd experimental cycle: *inverse document frequency, and n-grams (unigrams and combination of n-grams)*



A)

B)

Fig. 38. Classification accuracy of SVM in the 2nd experimental cycle

Table 20. Average performance measurements of SVM in the 2nd experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	45,71	54,29	45,03	45,71	45,37
uni-/bigram	46,19	53,80	45,60	46,02	45,81
uni-/bi-/trigram	**46,25**	53,75	45,60	46,25	45,92
Dataset B					
unigram	45,74	54,26	45,47	45,65	45,56
uni-/bigram	**46,26**	53,74	45,82	45,93	45,88
uni-/bi-/trigram	46,24	53,76	46,21	46,17	46,19

The experimental results (Fig. 38, Table 21) have shown that the *Support Vector Machine* classification method for product-review data in the 2nd experimental cycle has outperformed all the compared methods (*Naïve Bayes, Logistic Regression, Support Vector Machine*) using the inverse document frequency feature, n-gram properties for both datasets and achieved min in unigram: 45,71%, max in uni/bigram: 46,26% and uni/bi/trigram: 46,24% average classification accuracy.

The 3rd experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams), part of speech*

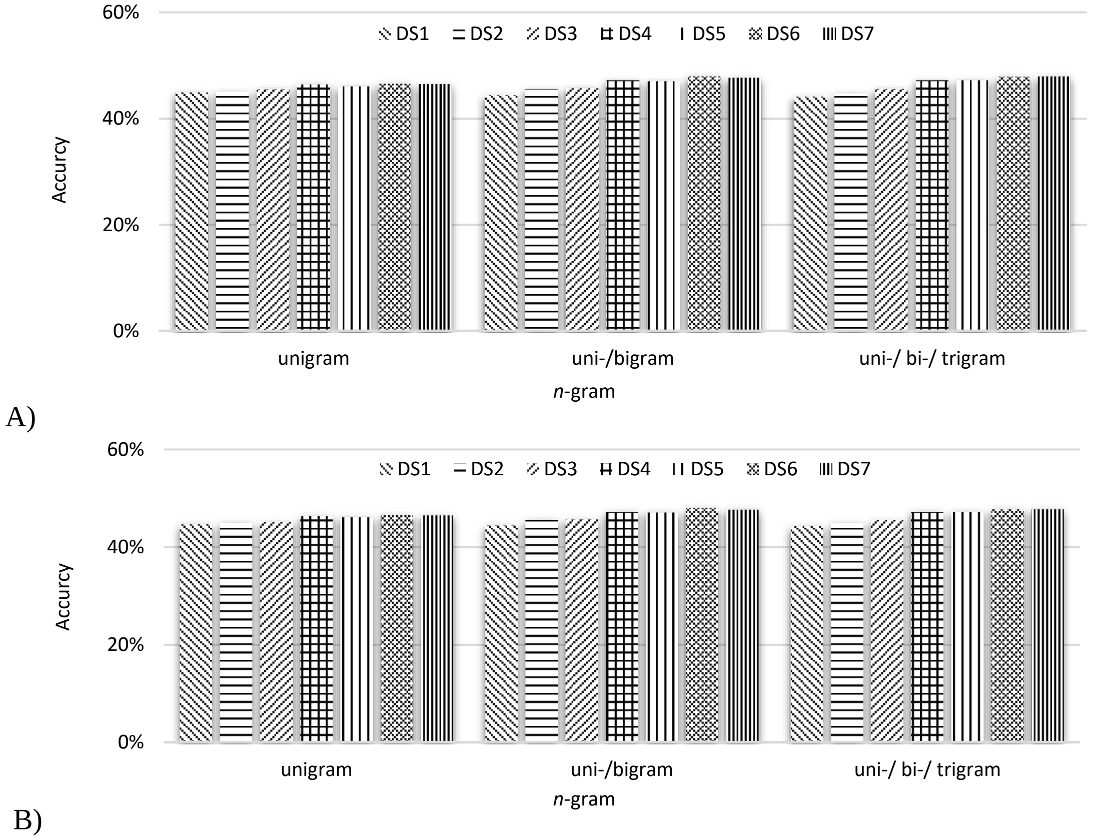


Fig. 39. Classification accuracy of SVM in the 3rd experimental cycle

Table 21. Average performance measurements of SVM in the 3rd experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	45,74	54,26	45,47	45,65	45,26
uni-/bigram	**46,50**	53,50	45,82	45,93	45,89
uni-/bi-/trigram	46,38	53,72	46,21	46,17	45,82
Dataset B					
unigram	44,24	54,26	45,60	45,77	45,68
uni-/bigram	**46,53**	53,47	45,90	46,12	46,01
uni-/bi-/trigram	46,37	53,63	46,44	46,23	46,33

The 4th experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams)*

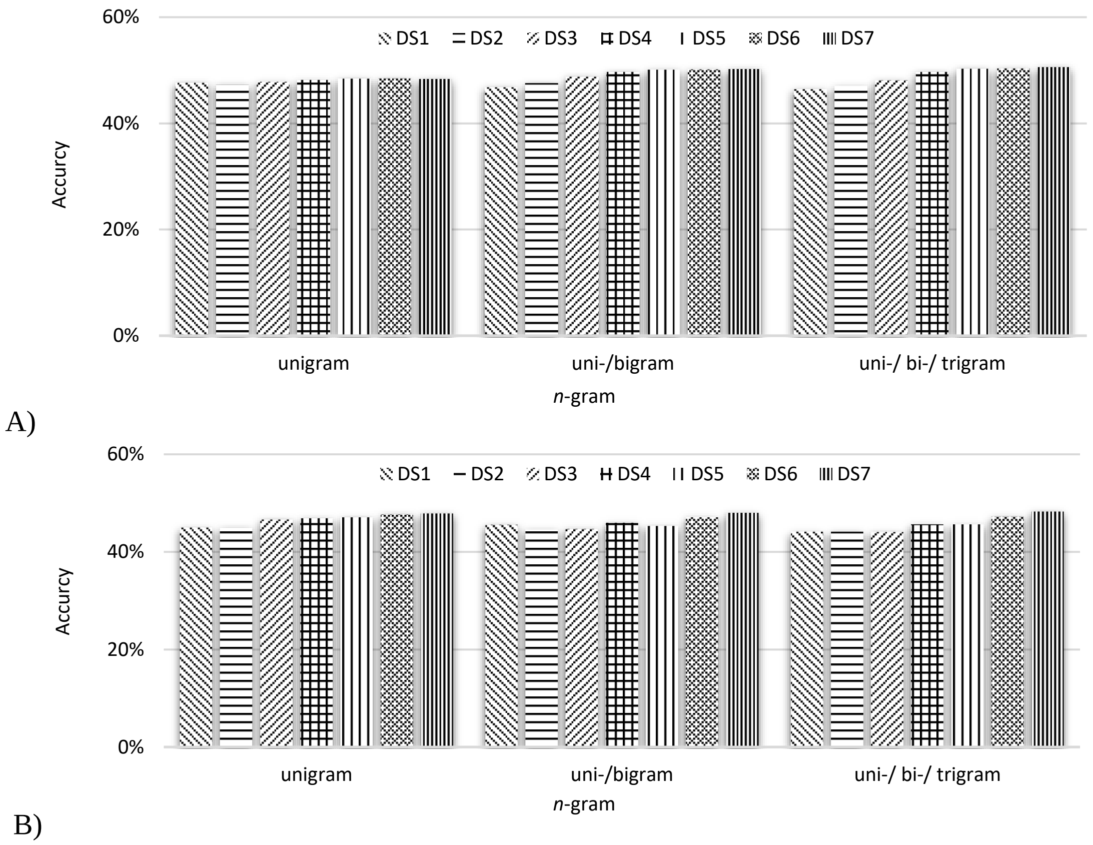


Fig. 40. Classification accuracy of SVM in the 4th experimental cycle

Table 22. Average performance measurements of SVM in the 4th experimental cycle

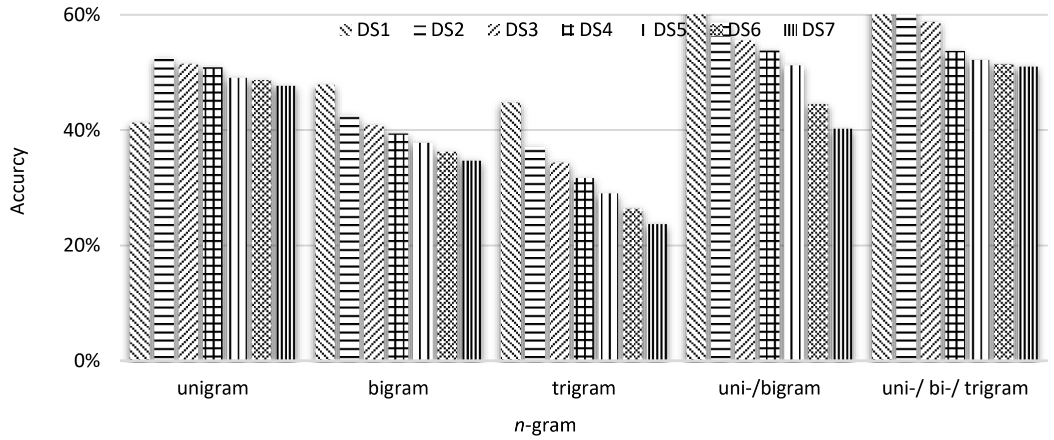
n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	48,02	51,98	46,72	48,02	47,36
uni-/bigram	**49,07**	50,93	47,89	49,07	48,47
uni-/bi-/trigram	48,93	51,07	48,02	48,93	48,47
Dataset B					
unigram	**46,58**	53,42	46,38	46,58	46,48

n-gram	Accuracy	Error	Precision	Recall	F1
uni-/bigram	45,89	54,11	48,99	45,61	47,21
uni-/bi-/trigram	45,58	54,42	48,15	45,58	46,81

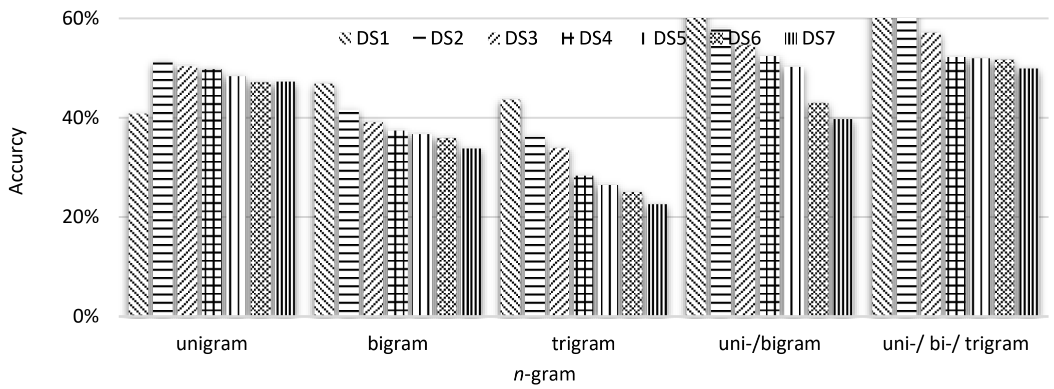
4.2.6. Logistic Regression

The 1st experimental cycle: *term frequency, and n-grams (unigrams, bigrams, trigrams, combinations)*

Currently, *PySpark* API does not yet officially support a multi-class classification by *Logistic Regression*. Nevertheless, the multi-class regression was implemented during this experiment using the *PySpark* API optional multi-class property. The findings indicate that the *Logistic Regression* multi-class classification method with the given data of product-reviews in the 1st experimental cycle has achieved min 30,43%, and max 58,48% average classification accuracy in comparison to the analyzed classifiers (Fig. 41, Table 23). Comparing unigram results, the classification accuracy is decreasing overall, when increasing the size of a dataset and a combination of uni, bi, tri-gram models increases the average accuracy but decreases when the bigram and trigram models are applied.



A)



B)

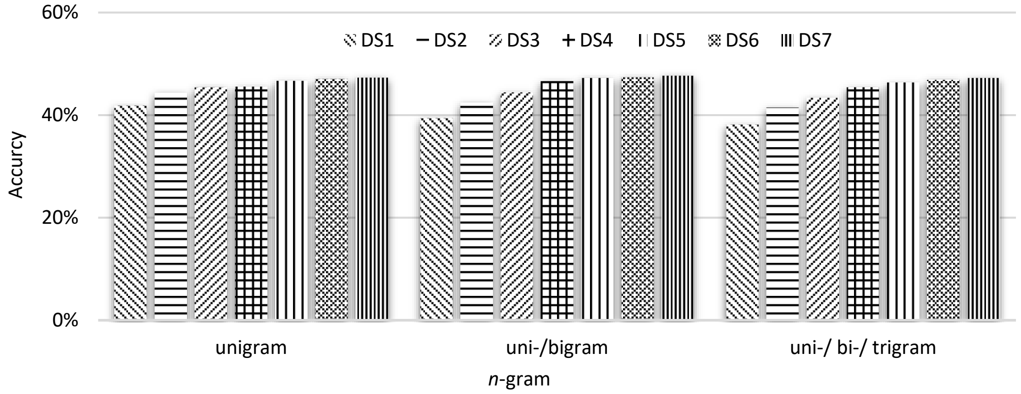
Fig. 41. Classification accuracy of LR in the 1st experimental cycle

Table 23. Average performance measurements of LR in the 1st experimental cycle

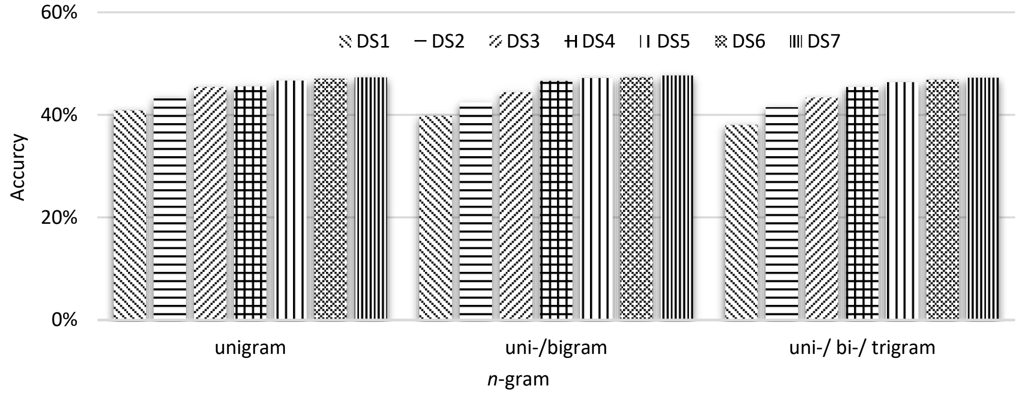
n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	48,78	51,22	47,70	48,15	47,92
bigram	39,93	60,07	39,04	39,41	39,22
trigram	32,43	67,57	31,71	32,01	31,86
uni-/bigram	53,30	46,70	52,11	52,60	52,36
uni-/bi-/trigram	**58,48**	41,52	57,18	57,72	57,45
Dataset B					
unigram	47,87	52,13	48,95	49,42	49,18
bigram	38,78	61,22	39,66	40,03	39,84
trigram	30,88	69,12	31,58	31,88	31,73
uni-/bigram	52,31	47,69	53,49	54,00	53,74
uni-/bi-/trigram	**57,62**	42,38	58,93	59,48	59,20

The *Logistic Regression* multi-class classification method is a less stable method as the values of average classification accuracy are spaciously distributed in comparison to other methods. The results (Fig. 41, Table 23) indicate several possible scenarios on how such a decrease in the classification accuracy can be explained. Firstly, the increased size of the dataset leads to the decrease in the classification accuracy, in case the classification model is not trained long enough since usually, the increased dataset should lead to more accurate results. On the other hand, the result could be influenced by the data feature selection, such as term frequency or combinations of *n*-grams. As the results have shown, the use of the inverse document frequency makes another impact on the classification model. The results of the experiment will be described in the next sections.

The 2nd experimental cycle: *inverse document frequency, and n-grams (unigrams and combination of n-grams)*



A)



B)

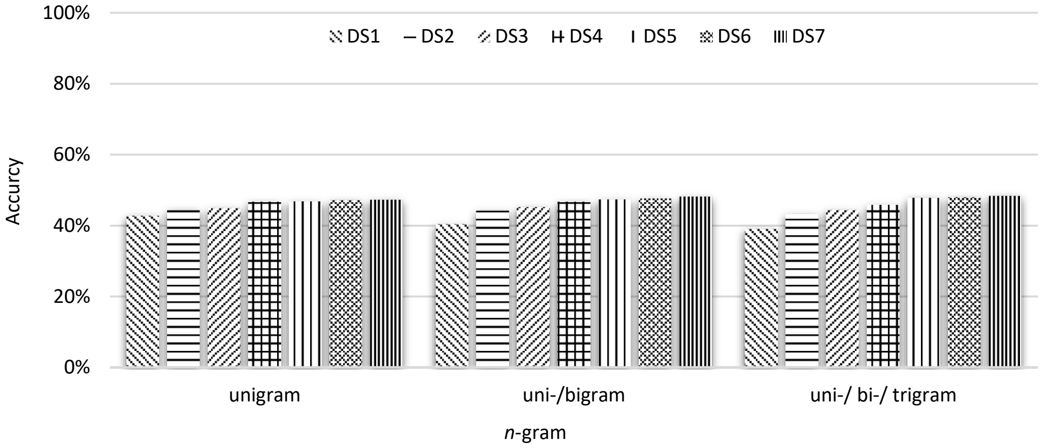
Fig. 42. Classification accuracy of LR in the 2nd experimental cycle

Table 24. Average performance measurements of LR in the 2nd experimental cycle

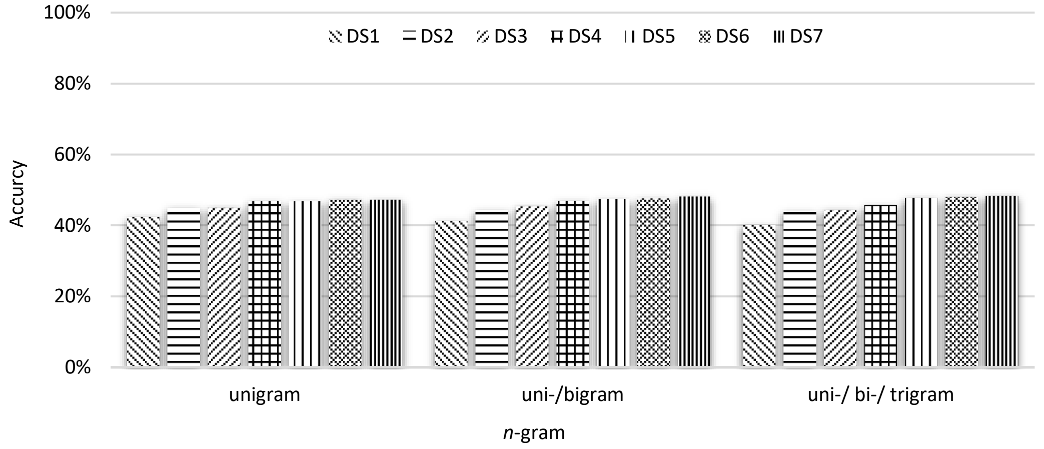
n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	**45,46**	54,54	44,78	45,46	45,12
uni-/bigram	45,03	54,97	44,42	45,03	44,73
uni-/bi-/trigram	44,15	55,85	43,64	44,15	43,90
Dataset B					
unigram	**45,17**	54,83	45,18	45,59	45,38
uni-/bigram	45,08	54,92	44,36	44,92	44,64
uni-/bi-/trigram	44,13	55,87	43,48	44,54	44,00

The results of the *Logistic Regression* multi-class classification method with the given data of product-reviews in the 2nd experimental cycle (Fig. 42, Table 24) indicate that using the inverse document frequency, the results are far less distributed in comparison to the result presented in the 1st experimental cycle.

The 3rd experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams), part of speech*



A)



B)

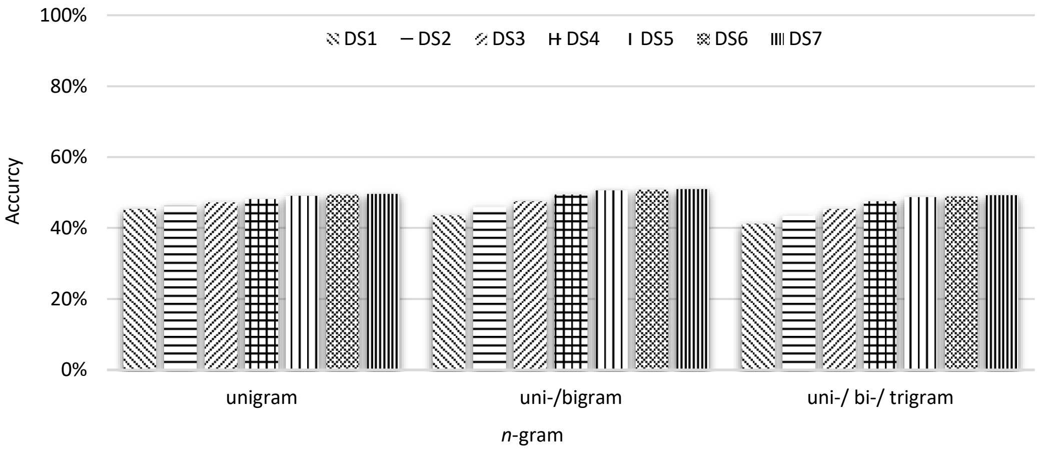
Fig. 43. Classification accuracy of LR in the 3rd experimental cycle

Table 25. Average performance measurements of LR in the 3rd experimental cycle

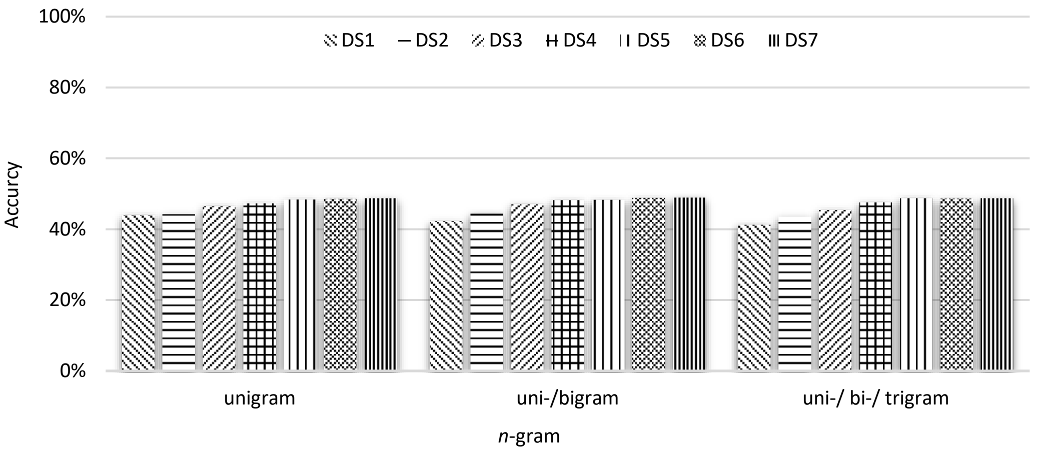
n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	**45,73**	54,27	45,05	45,73	45,39
uni-/bigram	45,68	54,32	45,02	45,68	45,34
uni-/bi-/trigram	45,25	54,75	44,67	45,25	44,96
Dataset B					
unigram	45,78	54,22	45,49	45,91	45,70
uni-/bigram	**45,86**	54,14	45,13	45,70	45,41
uni-/bi-/trigram	45,50	54,50	44,83	45,92	45,37

Fig. 43 and Table 25 illustrates the classification accuracy of the *Logistic Regression* classifier using part-of-speech tagging. In comparison to the unigram results (min 45,25%, max 45,73%), the classification accuracy results are insignificant to all the presented *n*-gram combinations by *Logistic Regression.* Comparing the unigram results, the classification accuracy is increasing overall when increasing the size of the dataset and a combination of uni, bi, tri-gram models increase the average accuracy but decrease when bigram and trigram models are applied. Meanwhile, the results of the average classification accuracy of the *Logistic Regression* classifier without part-of-speech tagging and with part-of-speech tagging as well as using the inverse document frequency are very similar. The results have shown that the *Logistic Regression* multi-class classification method applied to product-review data with part-of-speech tagging has a little higher classification accuracy, but the difference statistically is insignificant. The findings indicate that the *Logistic Regression* classifier with part-of-speech tagging has a higher classification accuracy (~1%) than using uni/bi/trigram as compared to the Logistic Regression classifier without part-of-speech tagging.

The 4th experimental cycle: *tf-idf, and n-grams (unigrams and combination of n-grams)*



A)



B)

Fig. 44. Classification accuracy of LR in the 4th experimental cycle

Table 26. Average performance measurements of LR in the 4th experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	47,90	52,10	47,22	47,87	47,54
uni-/bigram	**48,41**	51,59	47,77	48,26	48,01
uni-/bi-/trigram	46,37	53,63	45,67	46,35	46,00
Dataset B					
unigram	46,82	53,18	45,93	46,96	46,44
uni-/bigram	**46,87**	53,13	46,19	47,44	46,81
uni-/bi-/trigram	46,27	53,73	45,49	46,27	45,88

4.2.7. Multilayer Perceptron

Multilayer Perceptron does not fit very well between the 1st and 2nd experimental cycles as it does not use the inverse document frequency or the term frequency data feature selection. In this case, *Multilayer Perceptron* is applied using Word2Vector that is defined in section 2.3.5.

The 1st experimental cycle: *n-grams (unigrams, bigrams, trigrams, combinations)*

The results (Fig. 45, Table 27) of the 1st classification cycle have shown that a two-layer perceptron classifier had got the best classification accuracy using unigram in average 45,18% and 45,80% for both datasets. The minimum values in average 22,27% and 22,55% of classification accuracy were obtained using trigrams.

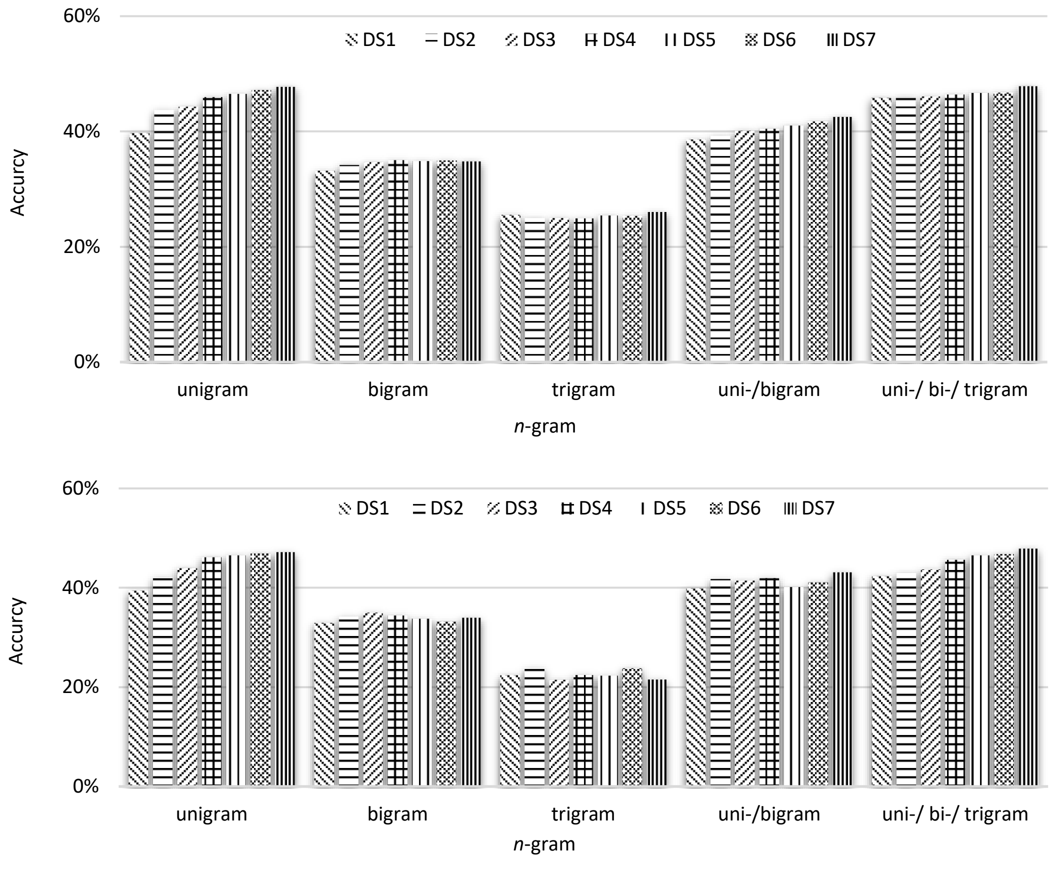


Fig. 45. Classification accuracy of MP in the 1st experimental cycle

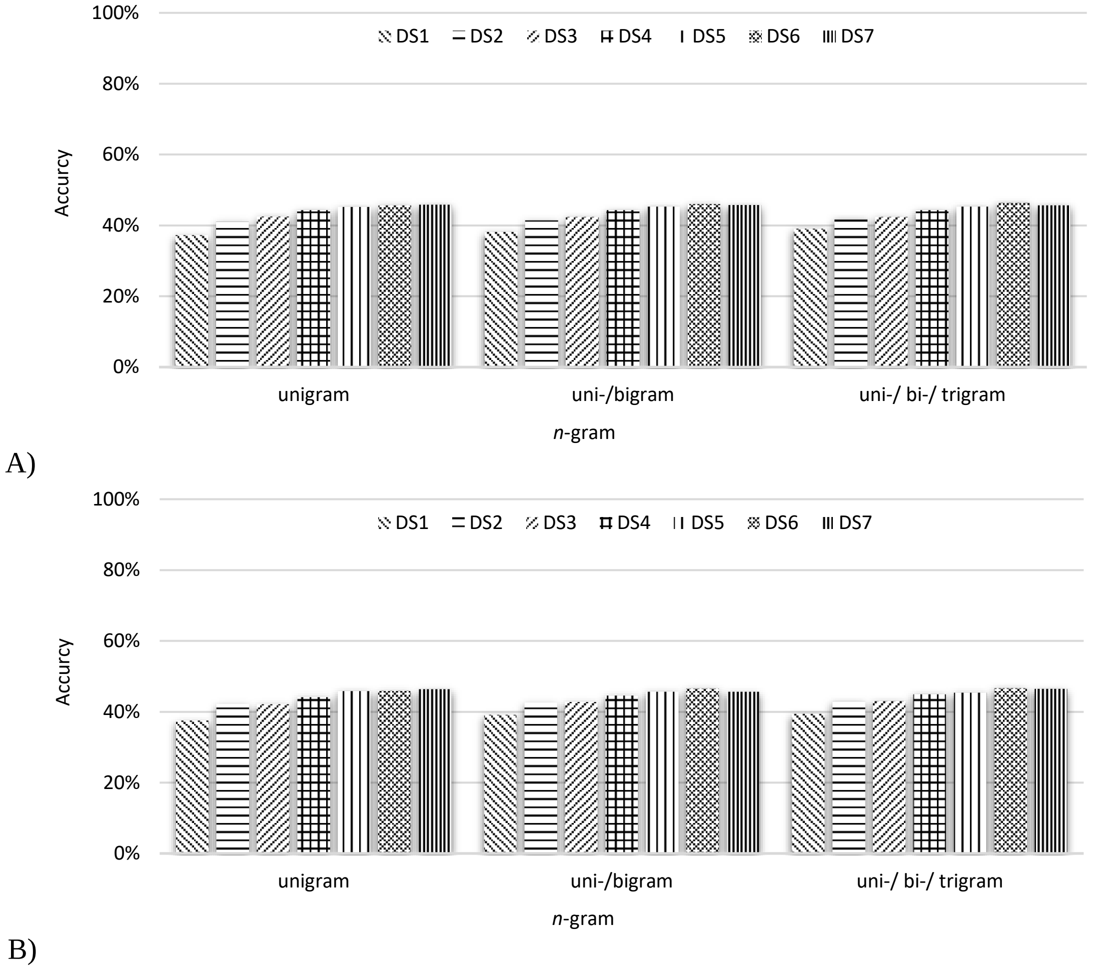
Table 27. Average performance measurements of MP in the 1st experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	**45,04**	54,96	44,25	44,93	44,59
bigram	34,56	65,44	34,43	34,56	33,65
trigram	22,27	77,73	22,18	22,27	21,68
uni-/bigram	41,77	58,23	41,61	41,77	40,66
uni-/bi-/trigram	44,65	55,35	44,48	44,65	43,47
Dataset B					
unigram	44,61	55,39	45,62	46,05	45,49
bigram	33,92	66,08	34,68	35,01	34,85
trigram	22,55	77,45	23,06	23,28	23,17
uni-/bigram	41,35	58,65	42,29	42,69	42,49
uni-/bi-/trigram	**45,13**	54,87	46,15	46,58	46,36

Except for the *Logistic Regression* classifier, the two-layer perceptron has outperformed *Decision Tree, Random Forest, Support Vector Machine*, and *Naïve Bayes* methods using the dataset B in the 1st experimental cycle. Dataset B consists of Movie and TV reviews data that contain higher number unique and total words in comparison to the dataset A.

The 3rd experimental cycle: *n-grams (unigrams and combination of n-grams), and part of speech (except inverse document frequency)*

The results (Fig. 46, Table 28) using *Multilayer Perceptron* (using 3-8 layers, with tens, hundreds, and thousands of neurons depending on the size of data), have showed that overall classification accuracy was only 22-23% using all words (including stop-words, removing punctuation, without spelling correction, word2vec, skewness).



A)

B)

Fig. 46. Classification accuracy of MP in the 3rd experimental cycle

Table 28. Average performance measurements of MP in the 3rd experimental cycle

n-gram	Accuracy	Error	Precision	Recall	F1
Dataset A					
unigram	43,12	56,88	42,22	43,12	42,67
uni-/bigram	43,35	56,65	42,70	43,35	43,02
uni-/bi-/trigram	**43,59**	56,41	43,13	43,59	43,34
Dataset B					

n-gram	Accuracy	Error	Precision	Recall	F1
unigram	43,44	56,56	42,85	43,06	42,96
uni-/bigram	**43,85**	56,23	43,91	43,71	43,81
uni-/bi-/trigram	44,11	55,89	43,84	44,24	44,04

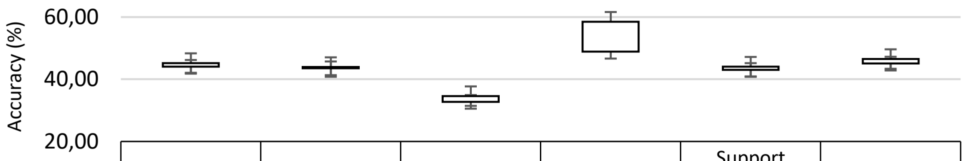
The experiments were carried out using various multilayer parameters, and for all cases, the output of classification accuracy was the same (22-23%). This is a considerably low result in comparison to other used methods since the most recent results (within the last 2-3 years) indicate that artificial neural networks work very effectively. Unfortunately, experimentation with various numbers of hidden layers was not successful in finding the best ones during this experimentation.

4.3. Overview of the results

The experimental results have shown that even if these classification methods are widely known and used for text and another type of data classification using machine learning, their classification accuracy correlates with the changes in the input to the classification model, e.g., using data feature selection and applying natural language processing techniques such as noise reduction. Thus, the classification accuracy performance mostly depends on the used features and combinations of *n*-grams. In most cases, the finding indicates that the used combinations of *n*-grams allow us to increase the classification by 1-5%, but this increase is insignificant. Also, the methods such as skewness, and techniques such as cross-validation are classical and universal for all the cycles in this experimentation. As described before, the skewness method helps us to collect the equally distributed number of customer product-review records in each class. Thus the cross-validation technique helps to evaluate the classification experiments, results, and conclusions. For all the methods and experimental cycles, the following noise reduction techniques were used: tokenization, stop-word removal (except in the 4th cycle), dropping out punctuation characters, reducing all capital letters to a lower-case form, and word stemming.

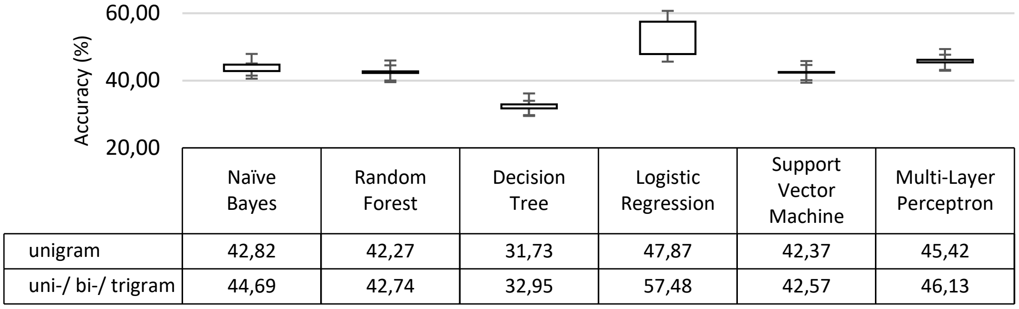
During the experimental cycles, neither the quality of reviews nor their grammatical correctness by a spellchecker or emoticons was assessed. Word corrections by the spellchecker, due to a number of words, in general, takes much more time than expected when checking the correctness of the words (experimentally tested), so it was decided to skip it in experimental cycles. Some tests were performed, and the results have shown that the influence of spellchecker has increased the classification accuracy by 1-2 % of the *Naïve Bayes* classification method. However, stylistic or lexical errors are more difficult to correct because it is a written form of spoken language; it is not possible every time because there is no single case of exact correction of the language. However, nowadays the trend is to make as few changes as possible in the given text by using an artificial neural network, so the classifier can better learn the data in the form as they are. In an ideal way, when using an artificial neural network, the textual data should not be processed by the natural language methods and techniques.

Fig. 39 indicates the best performed *n*-gram (comparing unigrams vs. uni-/bigrams vs. uni-/bi-/trigrams) features in the average classification accuracy results summaries in the 1st experimental cycle that the average values of the classification accuracy of *Naïve Bayes, Random Forest,* and *Support Vector Machine* are similar (dataset A: min in unigram: 42,99 – 44,01%, max in uni, bi, tri-gram: 43,93 – 45,22%; dataset B: min in unigram: 42,27 – 42,82%, max in uni, bi, tri-gram: 42,57 – 44,69%), and *Naïve Bayes* has achieved 1 – 2% higher average classification accuracy results in comparison to *Random Forest* and *Support Vector Machine*, but the difference is not statistically significant.



	Naïve Bayes	Random Forest	Decision Tree	Logistic Regression	Support Vector Machine	Multi-Layer Perceptron
unigram	44,01	43,53	32,74	48,86	42,99	45,04
uni-/ bi-/ trigram	45,22	43,93	34,58	58,5	44,06	46,51

A)



	Naïve Bayes	Random Forest	Decision Tree	Logistic Regression	Support Vector Machine	Multi-Layer Perceptron
unigram	42,82	42,27	31,73	47,87	42,37	45,42
uni-/ bi-/ trigram	44,69	42,74	32,95	57,48	42,57	46,13

B)

Fig. 47. Average classification accuracy of the 1st experimental cycle

The findings of the 1st experimental cycle by using *n*-grams and term-frequency indicate that the *Logistic Regression* multi-class classification method for product-reviews has achieved the highest (min 32,43%, max 58,50%) average classification accuracy in comparison with *Naïve Bayes, Random Forest, Decision Tree,* and *Support Vector Machine* classification methods. On the contrary, the *Baseline classifier* (~20%), and *Decision Tree* have obtained the lowest average accuracy values (min in trigram: 24,10%, max in uni, bi, tri-gram: 34,58%). The highest average accuracy was accomplished by the *Logistic Regression* (Fig. 41) method that outperformed using the term frequency and combinations of *n*-grams (min: 57% max:58%) in comparison to *Logistic Regression* method and unigram (min: 47% max:48%). Both datasets have presented equivalent results. The results of the 1st experimental cycle indicate that an increase in the size of the training dataset from 5000 to 75000 reviews per class leads to the insignificant growth of the classification accuracy (1 – 2%) of *Naïve Bayes, Random Forest, Support Vector Machine,* and *Logistic Regression* classifiers. These results show that the training set size of 5000 reviews per class is sufficient. Only *Logistic Regression* applied in the 2nd, and 3rd experimental cycles with the size of the training dataset from 5000 to 45000 reviews per class lead to the growth of the classification accuracy (8-9%), comparing to the accuracy of the training dataset with the size 5000. Such a growth leads to a situation where an increased size of the dataset helps to perform better regarding the classification accuracy. For other methods, the growth is insignificant, and the classification accuracy relates more to the data feature selection such as *n*-gram, (inverse) term frequency, and part-of-speech tagging.

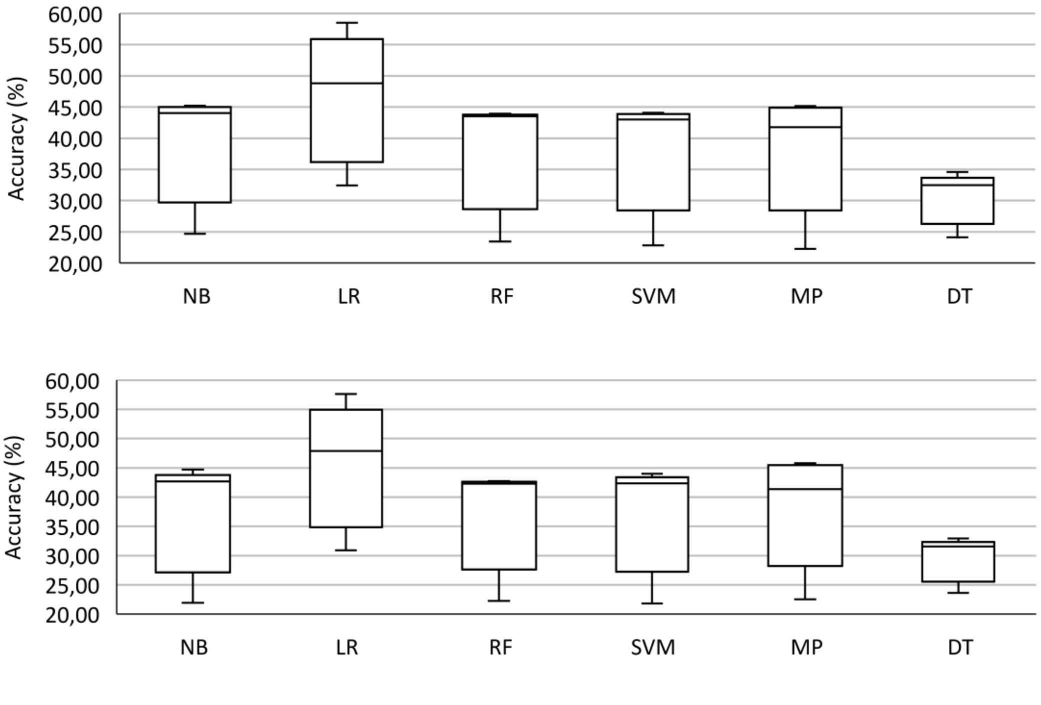
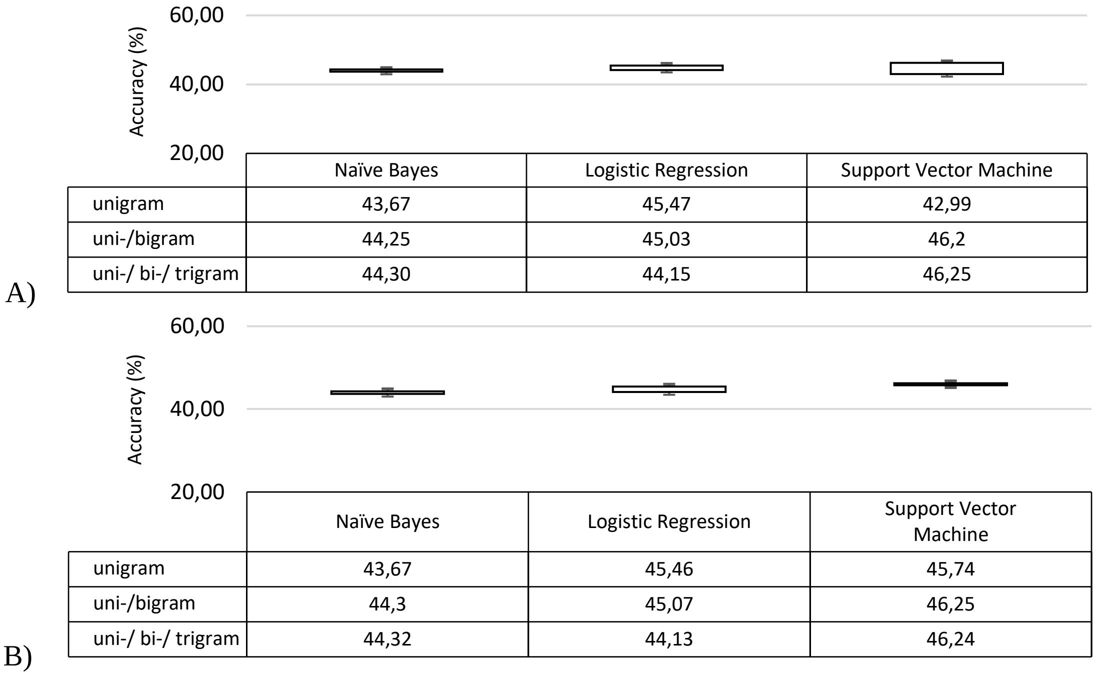


Fig. 48. Absolute values of the average classification accuracy of the 1st experimental cycle

Absolute values illustrated in Fig. 48, demonstrate that Logistic Regression achieved the highest accuracy with the proposed combination of *n*-grams (uni-/bi-/trigrams) as compared to unigrams or other classification methods experimentally analyzed in the 1st cycle. Thus, the linear Logistic Regression method contains less stability, and the values of classification accuracy are more distributed. Except for *Logistic Regression*, the performance of analyzed classification methods in the 1st experimental cycle contains more stability, and the values of the average classification accuracy are less distributed according to the results presented in Fig. 47.



A)

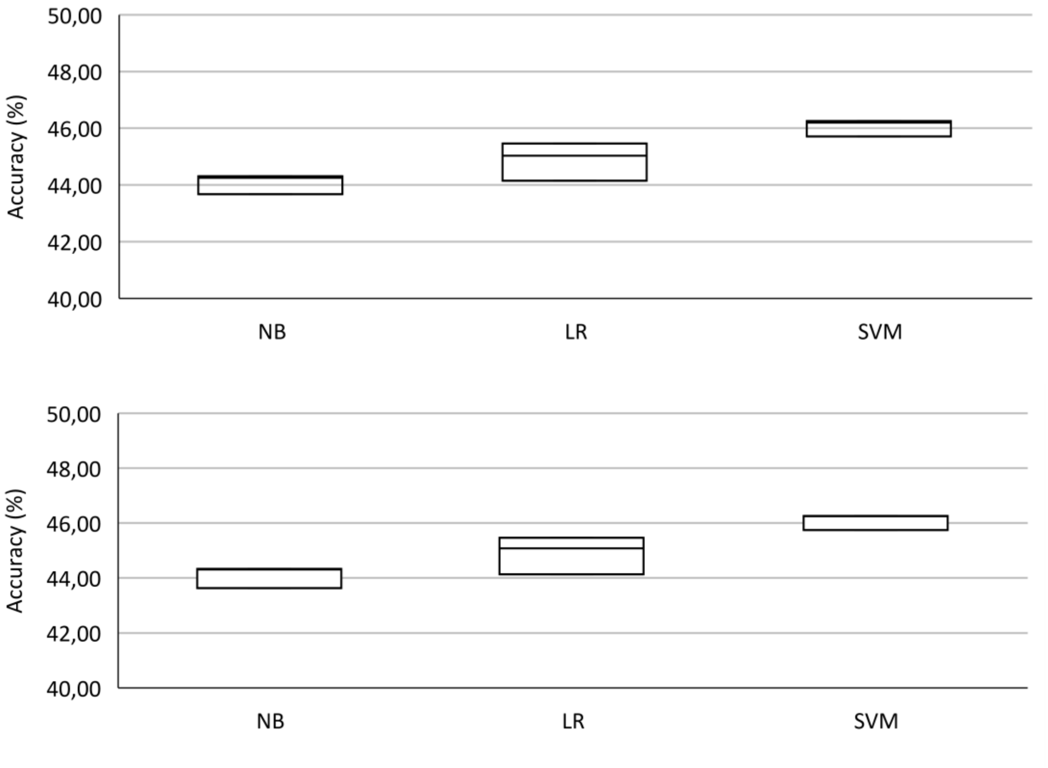
	Naïve Bayes	Logistic Regression	Support Vector Machine
unigram	43,67	45,47	42,99
uni-/bigram	44,25	45,03	46,2
uni-/ bi-/ trigram	44,30	44,15	46,25

B)

	Naïve Bayes	Logistic Regression	Support Vector Machine
unigram	43,67	45,46	45,74
uni-/bigram	44,3	45,07	46,25
uni-/ bi-/ trigram	44,32	44,13	46,24

Fig. 49. Average classification accuracy of the 2nd experimental cycle

Fig. 49 indicates the classification accuracy values of the 2nd experimental cycle. *Naïve Bayes* (dataset A: min in unigram: 43,67%, max in uni, bi, tri-gram: 44,30%; dataset B: min in unigram: 43,67%, max in uni, bi, tri-gram: 44,32%) and *Support Vector Machine* (dataset A: min in unigram: 42,99%, max in uni, bi, tri-gram: 46,25%; dataset B: min in unigram: 45,74%, max in uni, bi, tri-gram: 46,24%) classifiers with the proposed combination of *n*-grams (uni-/bi-/trigrams) have got higher values in comparison to unigram values. Therefore, *Logistic Regression* has performed with a higher classification accuracy by applying the unigrams (dataset A: max in unigram: 45,47%, A: min in uni, bi, tri-gram: 44,15%; dataset B: max in unigram: 45,46%, min in uni, bi, tri-gram: 44,13%) in comparison to a combination of *n*-grams (uni-/bi-/trigrams). The highest classification accuracy of the 2nd experimental cycle was accomplished by *Support Vector Machine* (dataset A: max in uni-/bi-/trigrams 46,25%; dataset B: max in uni-/bi-/trigrams 46,24%) with a combination of *n*-grams (uni-/bi-/trigrams).

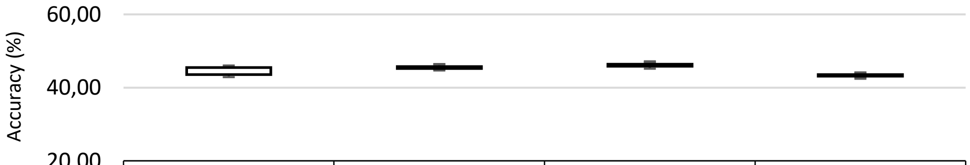


A)

B)

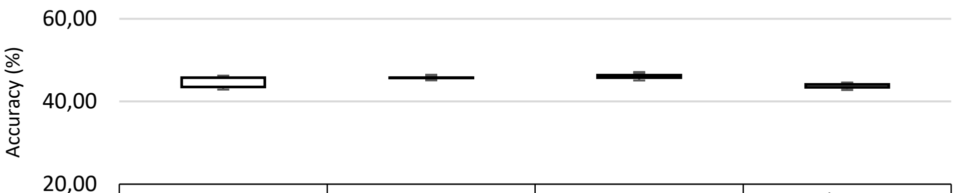
Fig. 50. Absolute values of the average classification accuracy in the 2nd experimental cycle

The findings of the 2nd experimental cycle, using *n*-grams and the inverse term-frequency, indicate that the *Support Vector Machine* multi-class classification method for product-reviews has achieved the highest (min: 45% max: 46%) classification accuracy in comparison with *Naïve Bayes*, and *Logistic Regression* classification methods. Thus, *Logistic Regression* with the size of data 5000 has presented a decreased classification accuracy (min: 38% max: 41%), but with an increased size of the dataset starting from 10000 reviews, the accuracy is increasing (46-47%). Both datasets have presented equivalent results.



	Naïve Bayes	Logistic Regression	Support Vector Machine	Multi-Layer Parceptron
unigram	43,57	45,73	45,86	43,12
uni-/bigram	44,71	45,68	46,51	43,35
uni-/ bi-/ trigram	45,46	45,25	46,39	43,58

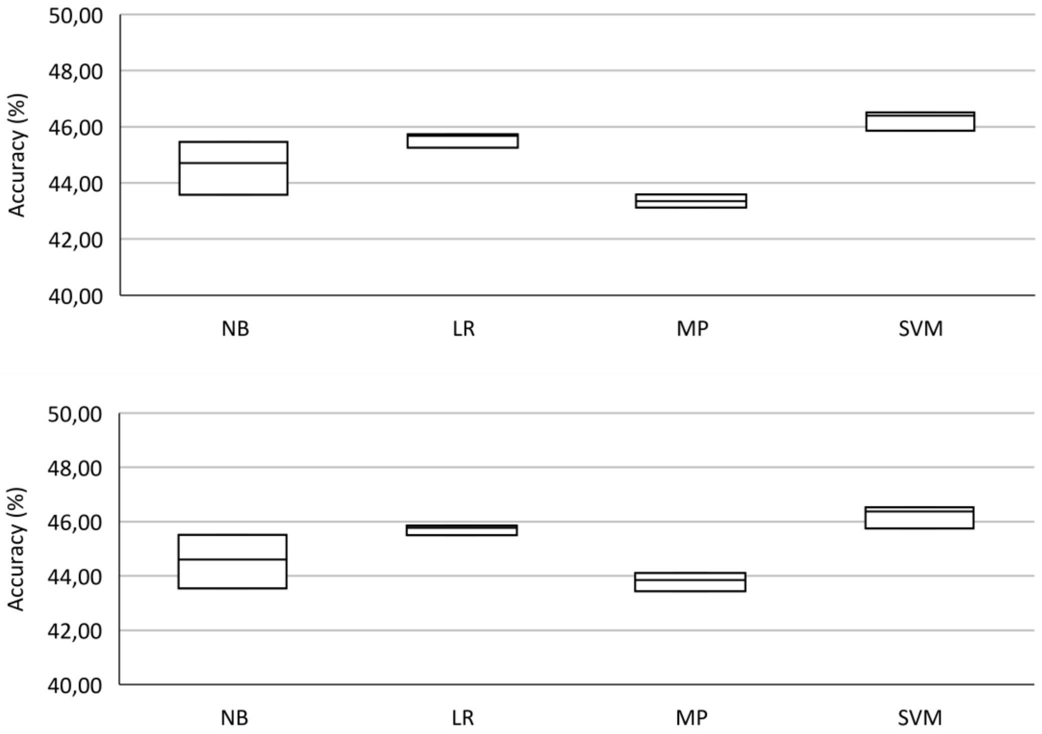
A)



	Naïve Bayes	Logistic Regression	Support Vector Machine	Multi-Layer Parceptron
unigram	43,54	45,77	45,74	43,43
uni-/bigram	44,6	45,85	46,52	43,85
uni-/ bi-/ trigram	45,77	45,74	46,37	44,11

B)

Fig. 51. Average classification accuracy in the 3rd experimental cycle

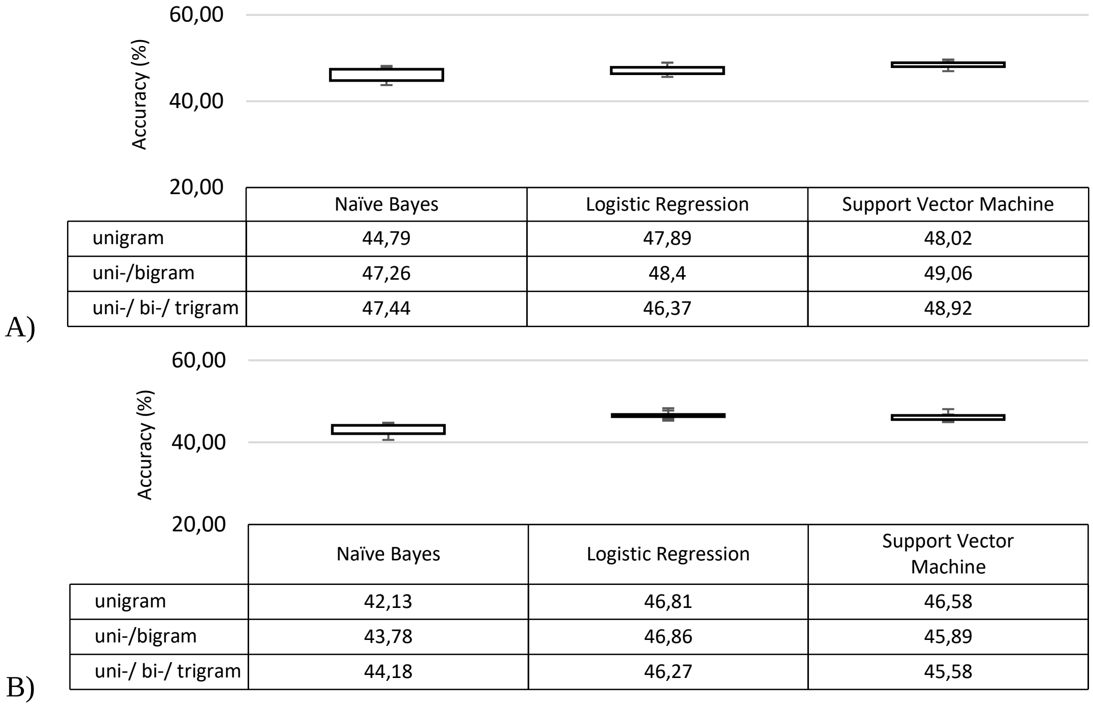


A)

B)

Fig. 52. Absolute values of the average classification accuracy of the 3rd experimental cycle

The findings of the 3rd experimental cycle, using *n*-grams, inverse term-frequency and part-of-speech tagging, illustrate that the *Support Vector Machine* multi-class classification method for product-reviews has achieved the highest (min: 45% max: 47%) classification accuracy in comparison with *Naïve Bayes*, and *Logistic Regression* classification methods. Both datasets have obtained equivalent results.



A)

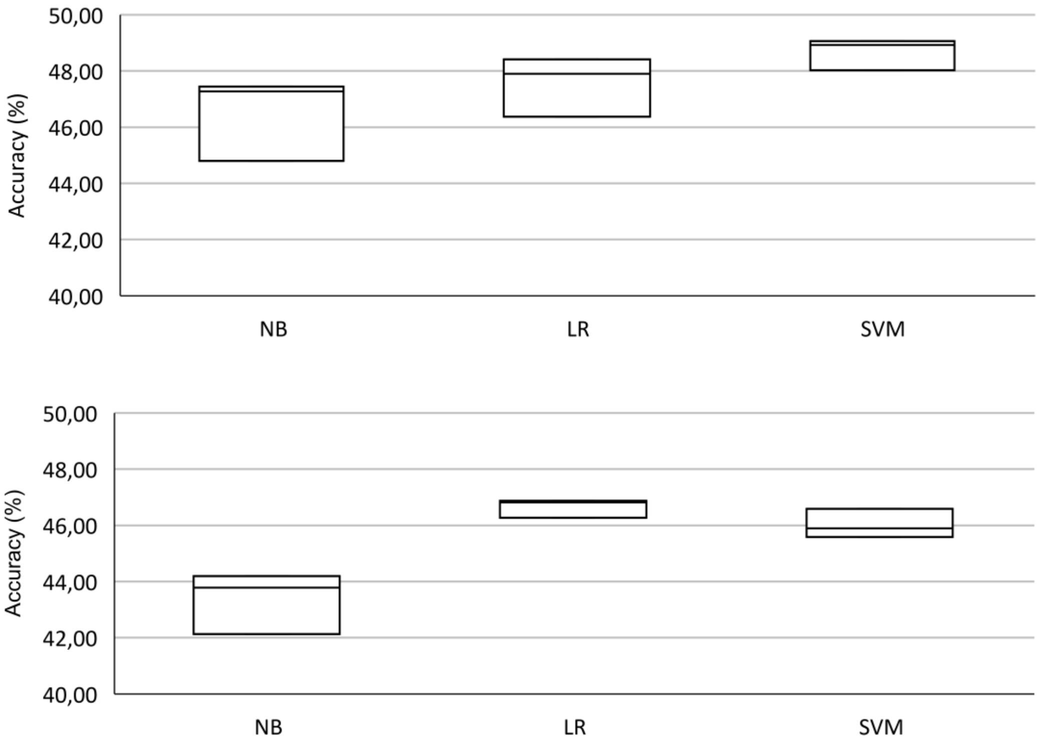
	Naïve Bayes	Logistic Regression	Support Vector Machine
unigram	44,79	47,89	48,02
uni-/bigram	47,26	48,4	49,06
uni-/ bi-/ trigram	47,44	46,37	48,92

B)

	Naïve Bayes	Logistic Regression	Support Vector Machine
unigram	42,13	46,81	46,58
uni-/bigram	43,78	46,86	45,89
uni-/ bi-/ trigram	44,18	46,27	45,58

Fig. 53. Average classification accuracy in the 4th experimental cycle

Fig. 53 indicate the classification accuracy values of the 4th experimental cycle using inverse document frequency, unigrams and combinations of *n*-grams, tokenization, lowercasing, stemming, but without w/o stop-words removal. *Naïve Bayes* (dataset A: min unigram: 44,79 %, max unigram, bigram and trigram: 47,44 %; dataset B: min unigram: 42,13 %, max unigram, bigram and trigram: 44,18 %) and *Support Vector Machine* (dataset A: min unigram: 48,02 %, max unigram and bigram: 49,06 %; dataset B: min unigram, bigram and trigram: 45,58 %, max unigram: 46,58 %) classifiers with unigrams and the proposed combination of *n*-grams (uni-/bi-/trigrams) has achieved a higher classification accuracy in average by 2 % as compared to the classification accuracy calculated in the 2nd experimental cycle (difference: applying stop-words removal). Whereas, the classification accuracy of *logistic regression* remained at the same level (dataset A: min unigram, bigram and trigram: 46,37 %, max unigram and bigram: 48,4 %; dataset B: min unigram, bigram and trigram: 46,27 %, max unigram and bigram: 46,86 %) level as compared to the results calculated in the 2nd experimental cycle.



A)

B)

Fig. 54. Absolute values of the average classification accuracy in the 4th experimental cycle

As some authors have concluded that stop-words and stemming do not necessarily improve the quality of classification. Therefore their use was experimentally examined in the 4th experimental cycle. However, the results have shown that using the inverse-term frequency without removing stop-words, the classification accuracy using *Naïve Bayes, Logistic Regression* and *Support Vector Machine* with a unigram and a combination of *n*-grams was higher by approximately 2% in comparison to the results that were obtained in the 2nd experimental cycle. The results without stemming have been collected outside the predefined experimental cycles. Therefore, the experiments were done only using DS4 (30.000 per class) and have shown that classification accuracy by *Naïve Bayes, Logistic Regression* and *Support Vector Machine* with a unigram and a combination of *n*-grams were lower by approximately 2% in comparison to the results obtained in the 2nd experimental cycle.

Table 29. The best methods performed using selected features for the product-review classification

Cycle	Highest Avg. Accuracy (Dataset A)	Highest Avg. Accuracy (Dataset B)	Classifier
1st	**58,48%**	**57,62%**	**Logistic Regression**
	45,18%	45,80%	Multilayer Perceptron
	44,06%	44,00%	Support Vector Machine
2nd	46,25%	46,26%	Support Vector Machine
	45,46%	45,46%	Logistic Regression
3rd	45,73%	46,53%	Support Vector Machine
	46,51%	45,58%	Logistic Regression
4th	49,06 %	46,58 %	Support Vector Machine
	48,4%	46,86%	Logistic Regression

Summarizing the results, the best-performed classification method for multiclass classification of short text messages (product-reviews) is *Logistic Regression* method applied with term-frequency combinations of *n*-grams (uni, bi, tri-gram), data feature selection. The summary of best-performed methods including data feature selection in every experimental cycle is summarized in Table 29.

4.4. The classification method proposed

The main idea of this dissertation was to analyze the following classical methods and propose data feature selection such as *n*-grams combination, and term-frequency (sections 0, 2.3.1, 2.3.4, 3.4.3), for especially applying multi-class classification for short-text messages. Fig. 55 illustrates a proposed method, including the best-performed data feature selection.

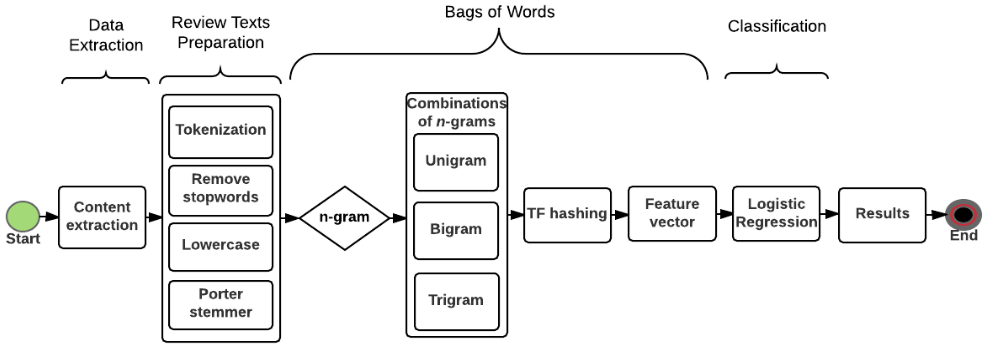


Fig. 55. Proposed method scheme

Fig. 55 contains stages that are as proposed as a method with a combination of data feature selection with *n*-grams combination and term-frequency. After data is extracted, punctuations must be removed and each word tokenized by white space. Then, stop-words removal should be performed. Words, such as "an" and "the", are often in use in any text, but do not hold specific information required to train this data model. As well, we in the 4th experimental analysis we saw that removing of stop-words is not entirely necessary. Their influence could be reduced by using techniques such as inverse document frequency. Converting all capital letters to lowercase comes just after the stop-word removal is finished. Word stemming allowed us to reduce inflectional forms to a common base form stemma. Also, stemming does not necessarily improve the quality of the classifier, but during the experimental analysis, we have indicated that stemming has insignificant influence on classification accuracy.

As we have seen from the results, the classification performance of some methods depends on the size of data, whereas the cloud computing technology and data analytics frameworks, such as *Apache Spark*, are capable of utilizing the computing resources for solving large-scale and data-intensive classification tasks within thousands of computer nodes.

4.5. Conclusions of Chapter 4

The comparison of *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine*, *Logistic Regression,* and *Multilayer Perceptron* methods for the multi-class text classification is analyzed.

1. The findings indicate that the *Logistic Regression* multi-class classification method for product-review data in the 1st experimental cycle has achieved the highest (min 32-33%, max 57-58%) classification accuracy in comparison with *Naïve Bayes, Random Forest, Decision Tree,* and *Support Vector Machine* classification methods. On the contrary, *Decision Tree* has

got the lowest average accuracy values (min in trigram: 24.10%, max in uni, bi, tri-gram: 34.58%).

2. The experimental results have shown that the *Naïve Bayes* classification method for product-review data in the 1st experimental cycle achieves 1 – 2% higher average of classification accuracy than the *Random Forest* and *Support Vector Machine* method, but the difference is not statistically significant.
3. Following a comparative analysis, it can be stated that the overall classification accuracy in combination with uni, bi, tri-gram models increases the average of classification accuracy (avg. ~10%), but these values are insignificant as compared to the unigram model of all classification methods.
4. The investigation indicates that the increase in the size of the training dataset from 5.000 to 75.000 product-reviews per class leads to the insignificant growth of the classification accuracy (1 – 2%) of *Naïve Bayes*, *Random Forest*, and *Support Vector Machine* classifiers. These results show that a training set size of 5.000 product-reviews per class is sufficient for all the analyzed classification methods, except the *Logistic Regression* method applied with the inverse document frequency (in the 2nd experimental cycle). In the sense of classification accuracy for all the analyzed classification methods, the classification accuracy relates more to the *n*-gram properties, while *Logistic Regression* and *Multilayer Perceptron* classifiers also relate to the size of the dataset.
5. The results have shown that *Logistic Regression* has outperformed *Support Vector Machine*, and *Multilayer Perceptron* by applying term frequency, but both methods can be recommended for the short text classification tasks. However, it would be reasonable to optimize the parameters or apply other kernel methods using *Support Vector Machine* for the better classification accuracy.

Chapter 5 General Conclusions

1. The findings and comparison accomplished in this thesis have proved that the cloud computing technology integrates the big data analytics frameworks that can be successfully used for in-memory intensive operations in the classification using machine learning algorithms.
2. Following the compared performance of all classification algorithms, it can be indicated that the overall classification accuracy based on a combination of *n*-grams (uni, bi, tri-gram), term frequency, increases the average of classification accuracy (12-15%) with short-texts such as product-review messages, but these values are insignificant as compared with the unigram model of all classification methods.
3. The *Logistic Regression* method with the proposed combination of n-grams (uni, bi, tri-gram) has outperformed, by the highest multi-class classification average accuracy (12-15%), other classification methods such as *Naïve Bayes, Random Forest, Decision Tree, Support Vector Machine,* and *Multilayer Perceptron* with the given large-scale multi-class data of short text messages, and natural language processing, term frequency, and other noise reduction features such as tokenization, stop-word removal, lowercasing, and term normalization.
4. The proposed method that combines *Logistic regression* with a combination of *n*-grams (uni, bi, tri-gram) and term frequency has established the highest classification accuracy (avg. 57-58%) for short-text classification tasks, e.g., product-review classification.

References

[1] Alpaydin E., Introduction to Machine Learning, Second Edition ed., Cambridge: The MIT Press, 2010, pp. 1 - 3.

[2] Mell P., Grance T., "The NIST Definition of Cloud Computing," U.S. Department of Commerce, Gaithersburg,, 2011.

[3] Hashem I. A. T., Yaqoob I., Anuar N. B., Mokhtar S., Gani A., Khan S. U., "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems,* vol. 47, pp. 98-115, 2014.

[4] University of Jaen, Department of Computer Science, Jean.

[5] Fernández A., Río S., López V., Bawakid A., Jesus M. J., Benítez J. M., Herrera F., "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks," *Wiley interdisciplinary reviews: data mining and knowledge discovery,* 2014.

[6] Wu C., Buyya R., Ramamohanarao K., "Big Data Analytics = Machine Learning + Cloud Computing," in *Big Data: Principles and Paradigms*, Burlington, Cornell University, 2016.

[7] Goodfellow I., Bengio Y., Courville A., Deep Learning, MIT Press, 2016.

[8] T. Mitchell, Machine Learning, McGraw Hill, 1997, p. 2.

[9] Hastie T., Tibshirani R., Friedman J., The Elements of Statistical Learning, Second Edition ed., New York: Springer-Verlag, 2009.

[10] Ouda K., "Distributed Machine Learning: A Review of current progress," 2015.

[11] Grefenstette G., Tapanainen P., "What is A Word, What is a Sentence? Problems of Tokenization," in *Proceedings of the 3rd Conference on Computational Lexicography and Text Research (COMPLEX'94),* Budapest.

[12] Manning C. D., Raghavan P., Schütze H., Introduction to Information Retrieval, Cambridge University Press, 2008, pp. 191-192.

[13] Konar A., Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain, CRC Press, 2000, pp. 401-431.

[14] Reid S. R., Model Combination in Multiclass Classification, University of Colorado, 2010.

[15] Drucker H., Wu D., Vapnik V. N., "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks,* vol. 10, no. 5, p. 1048, September 1999.

[16] Mitchell T., Machine Learning, 1997.

[17] Provost F., Fawcett T., Data Science for Business, vol. First Edition, Sebastopol: O'Reilly Media, Inc., 2013, pp. 83-18.

[18] Russel S., Norvig P., Artificial Intelligence: a modern approcach (third edition), Pearson, 2010, pp. 860-867.

[19] Pang B., Lee L., Vaithyanathan S., "Thumbs up? Sentiment Classification using Machine Learning Techniques," in *Proceedings of EMNLP-02*, 2002.

[20] Tausczik Y. R., Pennebaker J. W., "The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods," *Journal of Language and Social Psychology,* vol. 29, no. 1, p. 24–54, 2010.

[21] McNair D. M., Lorr M., Droppleman L. F., "Profile of Mood States (POMS) Manual," *San Diego, CA: Education and Industrial Testing Service,* 1971.

[22] Bollen J., Mao H., Zeng X.-J., "Twitter mood predicts the stock market," *Journal of Computational Science,* vol. 2, no. 1, pp. 1-8, 2011.

[23] Manning Ch.D., Raghavan P., Schütze H., Introduction to Information Retrieval, Online Edition ed., Cambridge: Cambridge University Press, 2008, p. 258.

[24] Agrawal R., Gupta A., Prabhu Y., Varma M., "Multi-Label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages," *WWW '13 Proceedings of the 22nd international conference on World Wide Web,* pp. 13 - 24, 2013.

[25] Rokach L., Maimon O., "Top-down induction of decision trees classifiers-a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 35, no. 4, p. 476–487, November 2005.

[26] Flannery B. P., Teukolsky S., Press W. H., Vetterling W. T., "Section 16.5. Support Vector Machines," in *Numerical Recipes: The Art of Scientific Computing,* 3rd ed., New York, Cambridge University Press, 2007.

[27] Caraciolo M., "Machine Learning with Python - Logistic Regression," Artificial Intelligence in Motion, 2011. [Online]. Available: http://aimotion.blogspot.lt/2011/11/machine-learning-with-python-logistic.html. [Accessed 5 September 2016].

[28] Pang B., Lee L., Opinion mining and sentiment analysis, vol. 2, Now Publishers Inc, 2008, pp. pp.21-22.

[29] Joachims T., "Text categorization with support vector machines: learning with many relevant features," p. pp. 137–142, 1998.

[30] Dumais, J. Platt, D. Heckerman, M. Sahami, "Inductive learning algorithms and representations for text categorization," p. pp. 148–155, 1998.

[31] Joachims T., “Text categorization with support vector machines: learn-ing with many relevant features,” in *Proceedings of ECML-98, 10th Euro-pean Conference on Machine Learning*, 1998.

[32] Dumais, J. Platt, D. Heckerman, M. Sahami, “Inductive learning algorithms and representations for text categorization,” in *Proceedings of the seventh international conference on Information and knowledge manage-ment*, Bethesda, Maryland, USA, 1998.

[33] McCallum A., Nigam K., “A comparison of event models for Naive Bayes text classification,” in *Proceedings of AAAI-98 workshop on learning for text categorization*, 1998.

[34] Pak A., Paroubek P., “Twitter for Sentiment Analysis: When Lan-guage Resources are Not Available,” in *Proceedings of Database and Ex-pert Systems Applications (DEXA 2011)*, 2011.

[35] Markievicz I., Kapočiūtė-Dzikienė J., Tamošiūnaitė M., Vitkutė-Adžgauskienė D., “Action Classification in Action Ontology Building Using Robot-Specific Texts,” *Information Technology And Control*, vol. Vol 44, no. No 2, pp. 155-164 pp., 2015.

[36] Burges C. J.C., “A Tutorial on Support Vector Machines for Pattern,” *Data Mining and Knowledge Discovery*, p. 121–167, 1998.

[37] Hosmer D. W. Jr., Lemeshow S., “Applied Logistic Regression,” A Wiley-Interscience Publication, 2000.

[38] Gelžinis A., Verikas A., “Neuroniniai tinklai ir neuroniniai skaičiavimai,” Kaunas, KTU, 2008.

[39] Cambria E., White B., “Jumping NLP Curves: A Review of Natural Language Processing Research,” vol. 9, IEEE, 2014, pp. 48 - 57.

[40] McEnery, A. and Wilson, A., Corpus Linguistics, 2nd ed., Edinburgh: Edinburgh University Press, 2001.

[41] Asghar M.Z., Ahmad S., Qasim M., Zahra S.R., Kundi F.M., “SentiHealth: Creating health-related sentiment lexicon using hybrid approach,” *SpringerPlus*, vol. 5, 2016.

[42] Daudaravičius V., Collocation segmentation for text chunking, Kaunas: Vytautas Magnus University, 2012.

[43] Natural Language Toolkit Project, “Natural Language Toolkit,” [Online]. Available: http://www.nltk.org/. [Accessed 1 March 2016].

[44] DL L.,, “A Machine Learning Approach For Emotion Classification Using Document Semantics,” *International Journal of Computational Intelligence Research*, vol. 13, no.

2, pp. 175-182, 2017.

[45] Frakes W., Baeza-Yates R., Information Retrieval: Data Structures and Algorithms, Chapter 8: Stemming Algorithms ed., Prentice Hall, 1992, p. Information Retrieval: Data Structures and Algorithms .

[46] Porter M., “The Porter Stemming Algorithm,” [Online]. Available: https://tartarus.org/martin/PorterStemmer/. [Accessed 23 Sep 2016].

[47] Kapočiūtė-Dzikienė J., Nøklestad A., Johannessen J. B., Krupavičius A., “Exploring Features for Named Entity Recognition in Lithuanian Text Corpus,” in *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, 2012.

[48] Sang E. F. T. K., “Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition,” in *Proceedings of CoNLL-2002,* Taipei, Taiwan, 2002.

[49] Bird S., Klein E., Loper E., Natural Language Processing with Python, O'Reilly Media, 2009.

[50] Benamara F., Cesarano C., Picariello A., Reforgiato D., Subrahmanian V. S., “Sentiment analysis: Adjectives and adverbs are better than adjectives alone,” in *In Proceedings of the International Conference on Weblogs and Social Media*, 2007.

[51] Wiebe J. M., Wilson T., Bruce R., Bell M., Martin M., “Learning subjective language,” in *Computational Linguistics,* 2004, p. pp. 277–308.

[52] Wilks Y., Stevenson M., “The grammar of sense: Using part-of-speech tags as a first step in,” *Journal of Natural Language Engineering,* vol. 4, no. 2, p. pp. 135–144, 1998.

[53] Bengio Y., Schwenk H., Sen J-S., Morin F., Gauvain J.-L., “Neural probabilistic language models,” *Innovations in Machine Learning,* pp. 137-186, 2006.

[54] Mikolov T., Corrado G., Sutskever I., Chen K., Corrado G., Dean J., “Distributed Representations of Words and Phrases and their Compositionality,” 2013.

[55] Sokolova M., Lapalme G., “A systematic analysis of performance measures for classification tasks,” vol. 45, p. 427–437, 2009.

[56] Mell P., Grance T., “The NIST Definition of Cloud Computing,” National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, September 2011.

[57] Parkhill D. F., The challenge of the computer utility, Addison-Wesley, 1966, p. 207.

[58] Zhang Q., Cheng L., Boutaba R., “Cloud computing: state of the art and research challanges,” *Internet Service Applications,* pp. 7-18, 2010.

[59] Agrawal D., Bernstein P., Bertino E., Davidson S., Dayal U., “Challenges and Opportunities with Big Data,” Purdue e-Pubs, West Lafayette, 2011.

[60] Mell P., Grance T., “The NIST Definition of Cloud Computing,” *Computer security: Special Publication 800-145,* 2011.

[61] Popek G. J., Goldberg R. P., “Formal Requirements for Virtualizable Third Generation Architectures,” *Communications of the ACM,* vol. 17, no. 7, p. 412–421, 1974.

[62] Karau H., Konwinski A., Wendell P., Zaharia M., Learning Spark, O’Reilly Media, Inc, 2015.

[63] Zaharia M., Chowdhury M., Franklin M. J., Shenker S., Stoica I. , “Spark: cluster computing with working sets,” in *USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10),* Berkeley, CA, 2010.

[64] Gu L., Li H., "Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark," *High Performance Computing and Communications & 2013 IEEE International Conference,* pp. 725-727, 2013.

[65] J. Jackson, “IBM Watson Vanquishes Human Jeopardy Foes,” 2011.

[66] Lally A., Prager J. M., McCord M. C., Boguraev B. K., Patwardhan S., Fan J., Fodor P., Chu-Carroll J., “Question analysis: How Watson reads a clue,” *IBM J. RES. & DEV.,* vol. 56, no. NO. 3/4, 2012.

[67] Chen Philip C.L., Zhang C.-Y., “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,” *Information Sciences,* vol. 275, p. 314–347, 2014.

[68] White T., Hadoop: The Definitive Guide, 3rd Edition, O’Reilly, 2012.

[69] The Apache Software Foundation, “MLlib is Apache Spark's scalable machine learning library,” [Online]. Available: http://spark.apache.org/mllib/. [Accessed 8 March 2016].

[70] Neumann S., “Apache Spark vs Hadoop MapReduce,” 2014.

[71] McAuley J., Targett C., Shi J., Hengel A., “Image-based recommendations on styles and substitutes,” *SIGIR,* 2015.

[72] Rennie M., Shih L., Teevan J., Karger D., “Tackling the Poor Assumptions of Naive Bayes Text Classifiers,” 2003.

[73] Python Software Foundation, “Python programming language,” 2017. [Online]. Available: www.python.org.

[74] Seddon M., “Natural Language Processing with Apache Spark ML and Amazon Reviews,” 2015. [Online]. Available: https://mike.seddon.ca/natural-language-processing-with-apache-spark-ml-and-amazon-reviews-part-1/. [Accessed 10 March 2016].

[75] Gupta M. R., Bengio S., Weston J., “Training Highly Multiclass Classifiers,” *Journal of Machine Learning Research,* vol. 15, 2014.

[76] Mokhtari A., Ribeiro A., “Global Convergence of Online Limited Memory BFGS,” vol. 16, December 2015.

[77] Apache Software Foundation, “Apache Spark,” [Online]. Available: https://spark.apache.org/. [Accessed 5 March 2015].

[78] Hortonworks, “Cluster planning guide”.

[79] Apache Software Foundation, “Mahout 0.10.0 Features”.

[80] “Types of Cloud Computing,” Electronic Privacy Information Center, 2015. [Online]. Available: https://epic.org/privacy/cloudcomputing/. [Accessed 1 Oktober 2015].

[81] Choi F. Y. Y. , “Advances in domain independent linear text segmentation,” in *Proceeding NAACL 2000 Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, Stroudsburg, 2000.

[82] Cheng J., Mizil C., Leskovec J., “Antisocial Behavior in Online Discussion Communities,” 2014. [Online]. Available: http://cs.stanford.edu/people/jure/pubs/trolls-icwsm15.pdf.

[83] Triantaphyllou E., Felici G., Data mining and knowledge discovery approaches based on rule induction techniques, New York: Spinger, 2006, p. 748.

[84] Dzemyda G., Kurasova K., Žilinskas J., Daugiamačių duomenų vizualizavimo metodai, Vilnius: Mokslo aidai, 2008.

[85] Chen P. C. L., Zhang C.-Y., “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,” *Information Sciences,* vol. 275, p. 314–347, 2014.

[86] Hammoud M., Sakr M. F., “Distributed Programming for the Cloud,” in *Large Scale and Big Data: Processing and Management, 1st Edition*, Auerbach Publications, 2014, pp. pp.1-34.

[87] Bi W., Kwok J.T., “Efficient Multi-label Classification with Many Labels,” *Proceedings of the 30th International Conference on Machine Learning (ICML-13),* vol. vol. 28, pp. 405-413, 2013.

[88] Nagel W. E., Müller-Pfefferkorn R., Kluge M., Hackenberg D., “Execution Environments for Big Data: Challenges for Storage Architectures and Software,” 2012. [Online]. Available: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/BDEC-Position-Paper-Nagel_0.pdf. [Accessed 11 January 2015].

[89] Lanford J., Nykodym T., Rao A., Wang A., "Generalized Linear Modeling with H2O’s R," 2015.

[90] Indurkhya N., Damerau F. J., Handbook of Natural Language Processing, 2nd edition,

2010.

[91] Rivera J., Meulen R., “Hype Cycle for Emerging Technologies, 2015,” Gartner, Inc., 2015.

[92] Shirakawa M., Hara T., Nishio S., “N-gram IDF: A Global Term Weighting Scheme Based on Information Distance,” in *WWW 2015*, Florence, 2015.

[93] Cavnar W.B., Trenkle J. M., “n-Gram-Based Text Categorization,” 1994.

[94] Zhou L., Wang H., Wang W., “Parallel Implementation of Classification Algorithms Based on Cloud Computing Environment,” *TELKOMNIKA*, vol. 10, pp. 1087-1092, 2012.

[95] Anderson J., Rainie L., “Pew Research Center,” Washington, 2012.

[96] Ivanov K., “Quality-control of information: On the concept of accuracy of information in data banks and in management information systems,” Stockholm, 1972.

[97] Nasukawa T., Yi J., “Sentiment analysis: Capturing favorability using natural language processing,” in *In Proceedings of the Conference on Knowledge Capture*, 2003.

[98] Dipper S., “Theory-driven and corpus-driven computational linguistics, and the use of corpora,” in *Corpus Linguistics: An International Handbook, Volume 1*, Berlin, Mouton de Gruyter, 2009, pp. pp. 68-96.

[99] Zhang Y., Jin R., Zhou ZH., “Understanding Bag-of-Words Model: A Statistical Framework,” *International Journal of Machine Learning and Cybernetics*, vol. Volume 1, no. 1, p. pp 43–52, December 2010.

[100] Blaise B., "Introduction to Parallel Computing," 12 06 2012. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Accessed 18 06 2012].

[101] Phyu T. N., “Survey of Classification Techniques in Data Mining,” vol. Vol. 1, 2009.

[102] Trim C., “The Art of Tokenization,” IBM, 2013. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en. [Accessed 28 August 2016].

[103] Sokolova M., Lapalme G., “A systematic analysis of performance measures for classification tasks,” *Information Processing and Management*, vol. 45, p. 427–437, 2009.

[104] Dean J., Ghemawat S., “MapReduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, pp. 107-113, January 2008.

[105] Curry E., “The Big Data Value Chain: Definitions, Concepts, and Theoretical Approaches,” in *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*, Springer Open, 2016, pp. 29-38.

[106] Bengio Y, Ducharme R., Vincent P., Jauvin Ch. , “A Neural Probabilistic Language Model,” *Journal of Machine Learning Research,* pp. 1137-1155, 2003.

[107] McCormick Ch., “Word2Vec Tutorial - The Skip-Gram Model,” 2016.

Tomas Pranckevičius

INVESTIGATION OF MULTI-CLASS CLASSIFICATION METHODS FOR TEXTUAL DATA

Doctoral Dissertation

Technological Sciences, Informatics Engineering (07 T)

Editor Janina Kazlauskaitė