DE GRUYTER OPEN

Information Technology and Management Science

doi: 10.1515/itms-2014-0018

2014 / 17

Markov Chains in the Task of Author's Writing Style Profile Construction

Pavels Osipovs [1], Andrejs Rinkevics [2], Galina Kuleshova [3], Arkady Borisov [4], [1-4] *Riga Technical University*

Abstract – **This paper examines the possibility of using Markov chains when constructing a profile of author's writing style. Thus, the constructed profile can be then used to analyze other texts and calculate their level of similarity. The extraction of the unique profile of text writing style that is characteristic of a specific human can be a topical task in many spheres of human activity. As an example, the task of detecting authorship for scientific and fiction texts can be mentioned. The paper describes a basic theoretical apparatus used for profile construction, software implementation of the experimental system as well as the experiments made and provides experimental results and their analysis.**

Keywords – **Formalization of author's writing style, level of texts similarity, Markov chain.**

I. Introduction

Nowadays the task of accurate determination of text authorship has not been solved with the accuracy required to guarantee correct identification yet. Studies can be mentioned that show up to 80 % correct results. Since the topicality of this kind of task is constantly growing, comparatively many studies in the field are conducted [3].

Different specialists are interested in finding a solution to the task above: literary critics, mathematicians, philologists, lawyers, criminalists, historians etc. [10]. To identify the authorship of a text, inquiries are frequently sent to experts, literary critics or historians who are able to identify the author of the unknown text or determine authorship by characteristic language peculiarities and stylistic techniques. Previously, to identify plagiarism and determine authorship of a literary work, handwriting expertise was used, which is completely unaccepted for the printed text. Automatic identification of the authorship makes it possible to get rid of some essential drawbacks observed in identifying authorship by experts such as time consumption and biases of expert point of view.

Various researchers use different approaches to solve the task of author style detection, e. g.,

- fuzzy area based methods have shown good results [4] on test data sets;
- top-k elements approach [5];
- the ability to use classical statistical approaches in the area under consideration has been studied in [6];
- support vector machine approach has been used in text classification for authorship attribution analysis [7];
- probabilistic approach has been used by Microsoft researchers to unite text classifiers under common meta-class for better effectiveness [8].

This research is focused on studying the possibility of using Markov chains in the task of text authorship identification [9]. The technique used in the study is based on the ideas suggested in paper [1] that discusses the system of anomalous action detection using Markov chains. By using this kind of approach, a model of author's writing style is created, which represents a Markov chain. Chain nodes are an unchanged sequence of words of the specified quantity that is picked up from the text. The weight of the edge characterizes the probability of prolonging the sequence of words of the node, in other words, the connected node of the given arc.

II. Description of the Technique

This study is aimed at checking the possibility of constructing a model of author's writing style using Markov chains.

General scheme of the implemented technique is shown in Fig. 1.

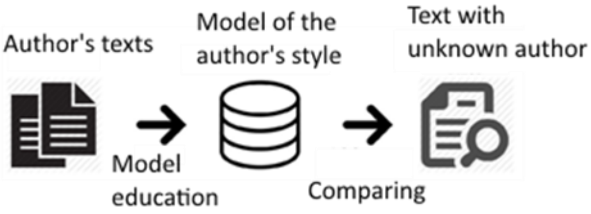


Fig. 1. General scheme of the technique.

A model of author's writing style is constructed using the texts for which the authorship of the targeted author is known for sure. The model is then used to analyze another text, whose authorship is unknown.

The final result will be Boolean value: **True** or **False**, corresponding, respectively, whether the text under consideration belongs to the target author or not.

A. Peculiarities of Subject Domain

It is worth noting that in some cases an author changes his style as a specific literary technique. Also modern authors make use of assistance of other people, who write some pieces of their works.

The volume of the analyzed text and texts used to train the model is also important. It has to be large enough to contain plenty of information about the specifics of style of its author. It must be stated that as of today texts of small volume are difficult to classify. There is empirically discovered lower limit of text size that contains enough information – it is

26000 words. It is clear that the size differs depending on the author.

Another special feature of the domain of the task under consideration is translated texts. In general case, it is highly not recommendable to analyze a text in the original language. Maximum that can be done is to compare texts translated by a single translator because literary translation is a complicated task and in the process of translation the author contributes to the text parts of his own author's style.

B. Model Training

Model training is conducted on the basis of all the texts collected for which there is a confidence in their authorship. Over each text an analyzer is passed, and by applying to it the iterative procedure a summary graph of the author's style is constructed. In what follows, the procedure is described in more detail.

C. Model Use

The use of the model consists in applying the already trained model to the analysis of the target text. In addition, an iterative procedure is explored that calculates a consequence of values of level metrics of the authorship for a large number of text pieces. The overall result will be the summary mean value of metrics for the whole text. It is also possible to use characteristics that differ from a simple mean value.

The procedure of using the existing model to analyze a new text is described in more detail below.

III. Model Training

The construction of the model of author's style includes the following steps:

1. Collection of original texts used to train the model;
2. Pre-treatment of the collected texts;
3. Pre-treatment setting configuration:
   a. Should punctuation be removed?
   b. Specify the size of words that will be named as "*short*".
   c. Should "*short*" words be deleted?
   d. Should the word form be normalized?
4. Specification of the parameter of model learning, ***w*** – window size (number of words per node of the graph).
5. Directing the learning process based on the application requirements.

When choosing a window size ***w***, there is a dilemma: its small values contain too little information, but a big size too precisely adapts to the training set (the classic problem of over-education).

The process of navigating a text consists in an iterative application of several operations.

At the beginning of the creation of the model, a window is initialized to an empty set of symbols – **∅**. The initial value of the window ***w*** is specified.

Two operations are defined on paths:

- *shift*(***σ***,*x*) which shifts the trace ***σ*** to the left and adds atomic element ***x*** at the end of the track, for example: *shift*(*aba*, *c*) = *bac*.
- *next*(***σ***) returns the first character of a trace ***σ*** and moves one position to the left, for example, ***next***(*abcd*) = *a* and updates path to the state *bcd*.

The initial state of the Markov chain is defined as the trace with length = ***w***, consisting of null characters. For example, if the window size is ***w*** = 3, the initial state will have the form [∅,∅,∅].

The process of building the Markov chain consists of the following steps that are iteratively repeated. To each track of the current set these operations are applied:

- let *c* = *next*(***σ***);
- setting *next state*= *shift*(*current state*, *c*);
- increase counter for state *current state and* arc between *current state* and *next state*;
- update *current state* to having value *next state*.

That is, at each iteration two counter values are formed: the value for the *current state* and the value for the transition from the *current state* to the *next state*. There is also the set or updated value of the transition probability for the transition between the *current state* and *next state* nodes.

As a result, when the operations described above are applied to all of the analyzed text, it will create a graph containing all present in it combination of words of a given length, their frequency of use and communication with other combinations.

IV. Model Use

By the term "*model using*" we mean its application in the analysis of a new text. The final result will be the use of the vector identity values for each level of the anomalous part of the target text size ***w***.

The process of calculating each metric anomaly is to perform a sequence of operations described below.

At the beginning of the analysis, additional global variables ***X*** and ***Y*** are introduced, which are used throughout all analysis iterations.

Initially, the variables ***X*** and ***Y*** are equal to 0, the next step changes the current state by adding an atomic element to the end and removing the initial description of the current state.

On the basis of the previously developed Markov chain that contains an author's style template for each transition between the atomic elements of the text written by an unknown author, metric μ (a) is calculated, which denotes status of the current value of the metric and new values for variables ***X*** and ***Y***.

At each step, there are two ways of calculating the value of the metric by the following algorithm. If the current model graph contains transition arc from the *previous state* to the *new state* $\beta_i \rightarrow \beta_{i+1}$, then ***X*** and ***Y*** are updated using the following function-parameters:

$Y = Y + F(s, (s, s`));$

$X = X + G(s, (s, s`));$

Else, if in the current model graph an arc from the *previous state* to the *current state* is not presented, then these functions-parameters are used:

$Y = Y + Z;$

$X = X + 1$.

And finally, the value of metric $\boldsymbol{\mu}$(a) is calculated, which is equal to Y/X.

Metric $\boldsymbol{\mu}$(a) shows how well the Markov chain predicts trace a, that is, the smaller its value is the better the model predicts the author's style.

Since $\boldsymbol{\mu}$ is parameterized by functions $\boldsymbol{F}$, $\boldsymbol{G}$ and the number $\boldsymbol{Z}$, a different selection of $\boldsymbol{F}$ and $\boldsymbol{G}$ will affect the final value of the classifier, which adds the ability to customize the fine specifics of the problem domain.

A. Metrics of the Difference

Functions $\boldsymbol{F}$ and $\boldsymbol{G}$ can be implemented in different ways, depending on the characteristics of the analyzed text. Nowadays, there may be used the following approaches:

- Probabilistic metric;
- Local minimal entropy metric;
- Frequency-based metric.

The summary results do not differ very strongly, but depending on the characteristics of the analyzed text, the best results can be shown by different metrics.

V. Experimental Software

For the experiments, a software package was designed that allows proceeding of the analysis of the texts with different kinds of settings.

As the main programming language in the development of experimental software system Python [11], [12] was used, as a universal tool having readable syntax, non-strict typing and often used in scientific research.

Since for large texts the final graph of the author's style may have a very large number of nodes and interconnection arcs, a 64-bit platform was used to store such big data structures in RAM. The established platform of experiments consists of several main software modules.

For easy experiment management and evaluation, a graphic user interface has been created that allows managing all experiment settings, saving/loading and visualizing experiments results (see Fig. 2).

A. A Module for Text Pre-processing

The main task of this module is to pre-process the text.

Global constants defined in the module:

• SHORT_WORD_LENGTH – the number that characterizes the length of the word, which will be declared as a small or insignificant word. It is assumed that most interjections, unions and other insignificant words fall under this characterization.

• PUNCTUATION – a character set defined as punctuation.

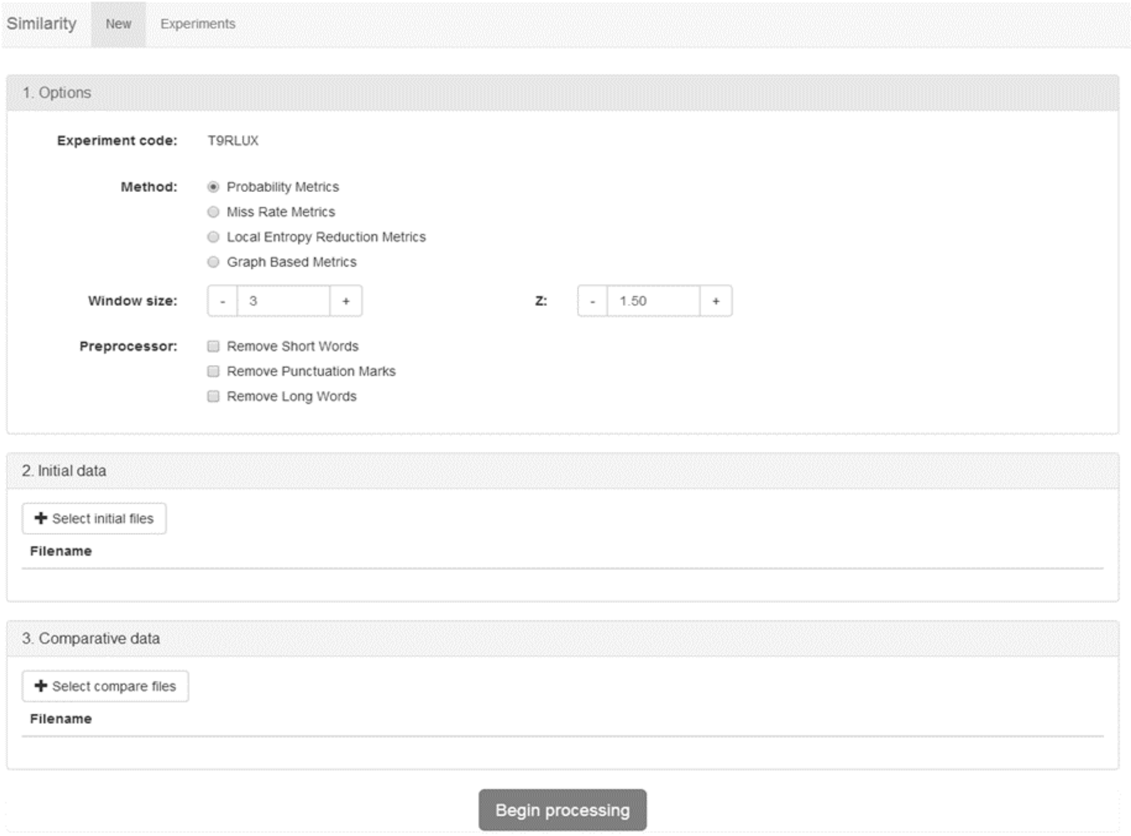
• REGEX_PUNCTUATION – a compiled set of punctuation marks for use in regular expressions.

There are two classes presented in the module:

• *SplitterFlags* – a class containing all the flags that define details and methods for text pre-processing;

• *Splitter* – base class for text pre-processing.

The overall structure of the module is shown in Fig. 3.



Experiment code: T9RLUX

Method:
● Probability Metrics
○ Miss Rate Metrics
○ Local Entropy Reduction Metrics
○ Graph Based Metrics

Window size: - 3 +

Z: - 1.50 +

Preprocessor:
□ Remove Short Words
□ Remove Punctuation Marks
□ Remove Long Words

Fig. 2. Graphic user interface for experiment management.

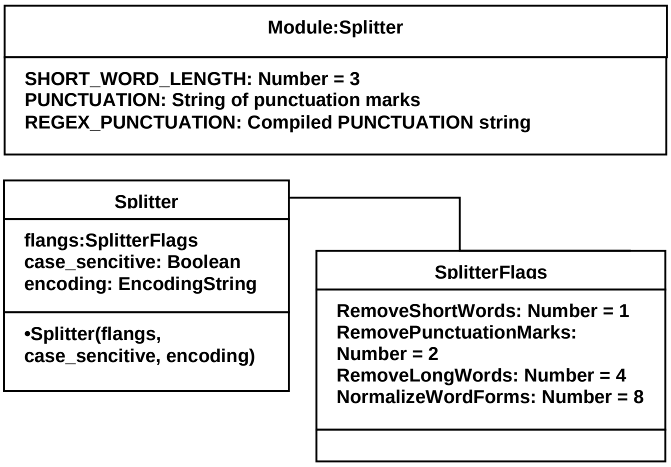

Module:Splitter
SHORT_WORD_LENGTH: Number = 3
PUNCTUATION: String of punctuation marks
REGEX_PUNCTUATION: Compiled PUNCTUATION string
Splitter
flangs:SplitterFlags
case_sencitive: Boolean
encoding: EncodingString
•Splitter(flangs, case_sencitive, encoding)
SplitterFlags
RemoveShortWords: Number = 1
RemovePunctuationMarks: Number = 2
RemoveLongWords: Number = 4
NormalizeWordForms: Number = 8

Fig. 3. The structure of the text pre-processing module.

B. Module to Create and Use Models of the Author's Style

The main module that combines all the functionality is "*Similarity*" module; it includes mechanisms for the construction and use of models of the author's style, based on a directed graph represented as a Markov chain.

The overall structure of the module is shown in Fig. 4.

The module includes the following classes:

- SimilarityMatcherFactory – a base class, which includes all necessary for work functions;
- ProbabilitySimilarity – a class that extends the base class and calculates the difference in author's style of the unknown author's text and current style model, on the basis of probability metric;
- LocalEntropyReductionSimilarity – a class that extends the base class and calculates the difference in author's style of the unknown author's text and current style model on the basis of local entropy metric.
- MissRateSimilarity – a class that extends the base class and calculates the difference in author's style of the unknown author's text and current style model on the basis of frequency-based metric.
- GraphSimilarity – a class that extends the base class and calculates the difference in author's style of the unknown author's text and current style model on the basis of graph similarity based metric.

C. Module for Directed Graph Creation and Manipulation

The module for graph manipulation is used to create and store the model of the author's style. Additionally, the module has an ability to save the graph in GraphML [2] file format.

The module contains two classes:

- WindowNode – this class is a basic data type representing the node of the graph, which includes all the necessary information in order to represent the graph as a Markov chain;
- WindowGraph – this class is used to store all the nodes of the graph and adding new ones.

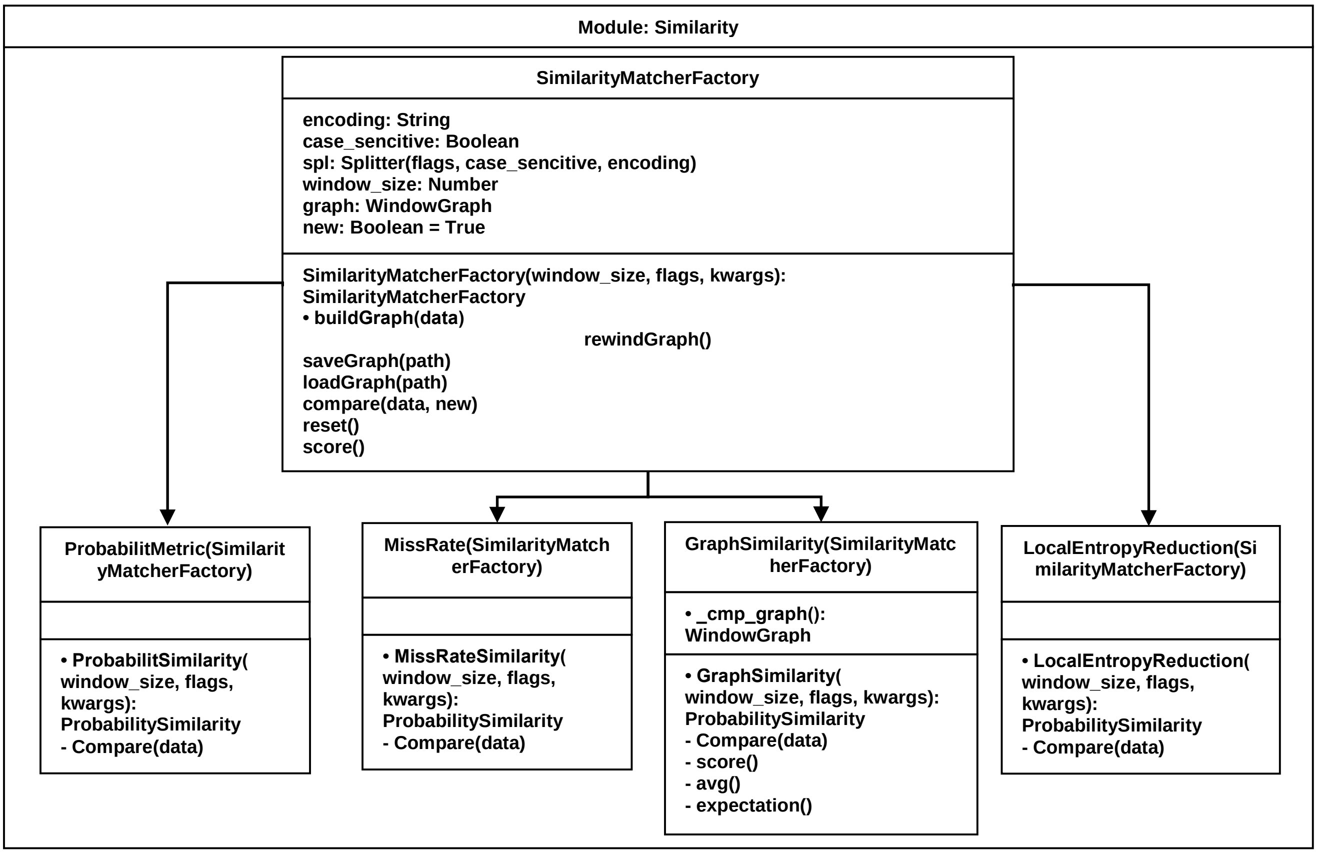

Module: Similarity
SimilarityMatcherFactory
encoding: String
case_sencitive: Boolean
spl: Splitter(flags, case_sencitive, encoding)
window_size: Number
graph: WindowGraph
new: Boolean = True
SimilarityMatcherFactory(window_size, flags, kwargs): SimilarityMatcherFactory
• buildGraph(data)
rewindGraph()
saveGraph(path)
loadGraph(path)
compare(data, new)
reset()
score()
ProbabilitMetric(SimilarityMatcherFactory)
• ProbabilitSimilarity(window_size, flags, kwargs): ProbabilitySimilarity
- Compare(data)
MissRate(SimilarityMatcherFactory)
• MissRateSimilarity(window_size, flags, kwargs): ProbabilitySimilarity
- Compare(data)
GraphSimilarity(SimilarityMatcherFactory)
• _cmp_graph(): WindowGraph
• GraphSimilarity(window_size, flags, kwargs): ProbabilitySimilarity
- Compare(data)
- score()
- avg()
- expectation()
LocalEntropyReduction(SimilarityMatcherFactory)
• LocalEntropyReduction(window_size, flags, kwargs): ProbabilitySimilarity
- Compare(data)

Fig. 4. The program structure of the module to create and use models of the author's style.

The overall structure of the module is shown in Fig. 6.

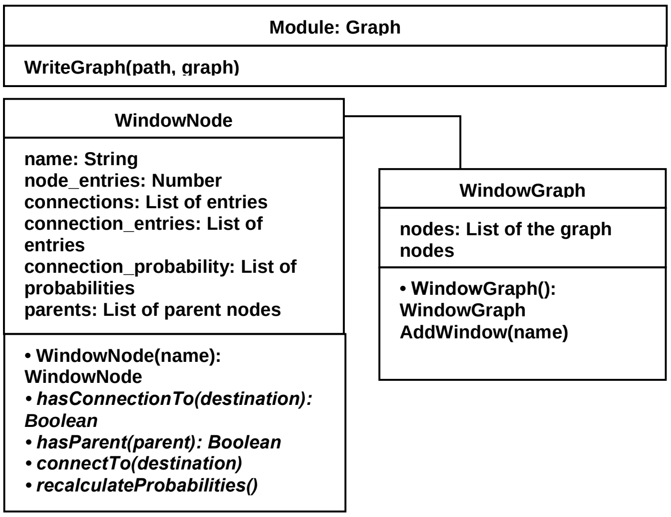

Module: Graph
WriteGraph(path, graph)
WindowNode
name: String
node_entries: Number
connections: List of entries
connection_entries: List of entries
connection_probability: List of probabilities
parents: List of parent nodes
• WindowNode(name): WindowNode
• hasConnectionTo(destination): Boolean
• hasParent(parent): Boolean
• connectTo(destination)
• recalculateProbabilities()
WindowGraph
nodes: List of the graph nodes
• WindowGraph(): WindowGraph
AddWindow(name)

Fig. 5. The program structure of the module creation of a directed graph.

D. *The General Scheme of the Component Relationship*

The main component that connects all the modules is "*Similarity*". By importing this class, child classes get access to all the necessary functionality to create a model of the author's style.

The block diagram that shows the interconnection of components is shown in Fig. 6.

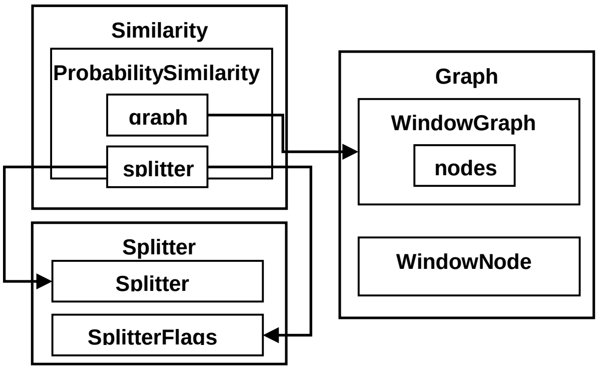


Fig. 6. Interconnections of the main components.

VI. Experiments and Results

Experiments were carried out on the basis of the works of 10 authors, each of which presented from 10 to 40 texts. The number of words in each of the texts used was more than 26000.

The basic process of the experiments was accomplished as follows:

1. Cross-comparisons to establish the authorship were performed. It was assumed that there were sufficiently long fragments belonging to a number of authors that use phonological writing. For the works of each author, the model of the author's style was built.
2. At the next step of the analysis, a text with a known author was selected, but the author was marked as unknown.
3. Upon receiving the result, it was compared with the expected one that characterized the accuracy of the determination of authorship.

A. *Comparison of the Effectiveness for Different Values of the Window Size*

Here, the results of text classification using different values of the window size are compared.

As a result of measurement, it was ascertained that the length of the windows affects the performance. If the lowest possible window length equal to one atomic element is applied, modeling takes a lot of time. When the dimension of the window is increased, there is a growth rate of construction of models of author's style. This is due to the lessening of linkages between nodes, as the same windows in the construction are rarely observed.

The parameters of one of the conducted experiments are given in Table I.

TABLE I
Experiment Plan

Parameter	Value/State
Window size (W)	1, 2, 3, 5, 10, 25, 50, 100
Z	1.5
G	1.0
Pre-processing	No
Case normalization	Yes
Metric used	*Probability metric*

According to the results of the experiment, the method of step-by-step metric was able to determine the authorship with different sizes of windows in 60–80 % of the total work. The accuracy of author detection with different values of window size is illustrated in Fig. 7.

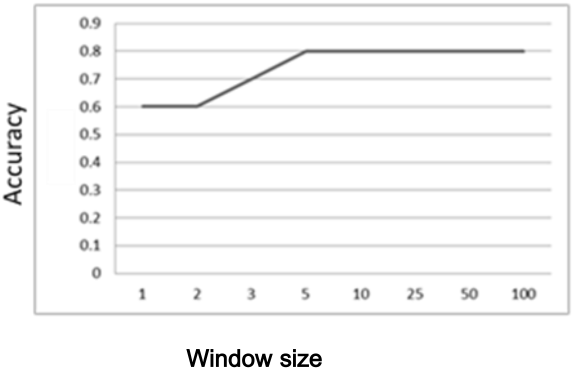


Fig. 7. The success rate for different window sizes.

The impact of different types of pre-processing on the effectiveness of text classification has been studied experimentally. This experiment is based on the results of an experiment investigating the dependence of recognition on the dimension of the window size of the author's style model. The continuation of the study foresees to elucidate the effect of pre-processing of text on final classification results. The main objective of this experiment is to increase the detection of stylistic factors, accumulating only the information required in the model of the author's style.

The applied variants of text pre-processing are as follows:

1. remove short (irrelevant) words;
2. remove punctuation;
3. remove short (insignificant), the words and punctuation marks;
4. remove long words;
5. normalize word forms;
6. remove long words and punctuation marks;
7. case-sensitivity, remove punctuation;
8. case-sensitivity, remove short (irrelevant) words;
9. case-sensitivity, remove short (insignificant), the words and punctuation marks;
10. case-sensitivity, remove long words.

The chart with experimental results is shown in Fig. 8.

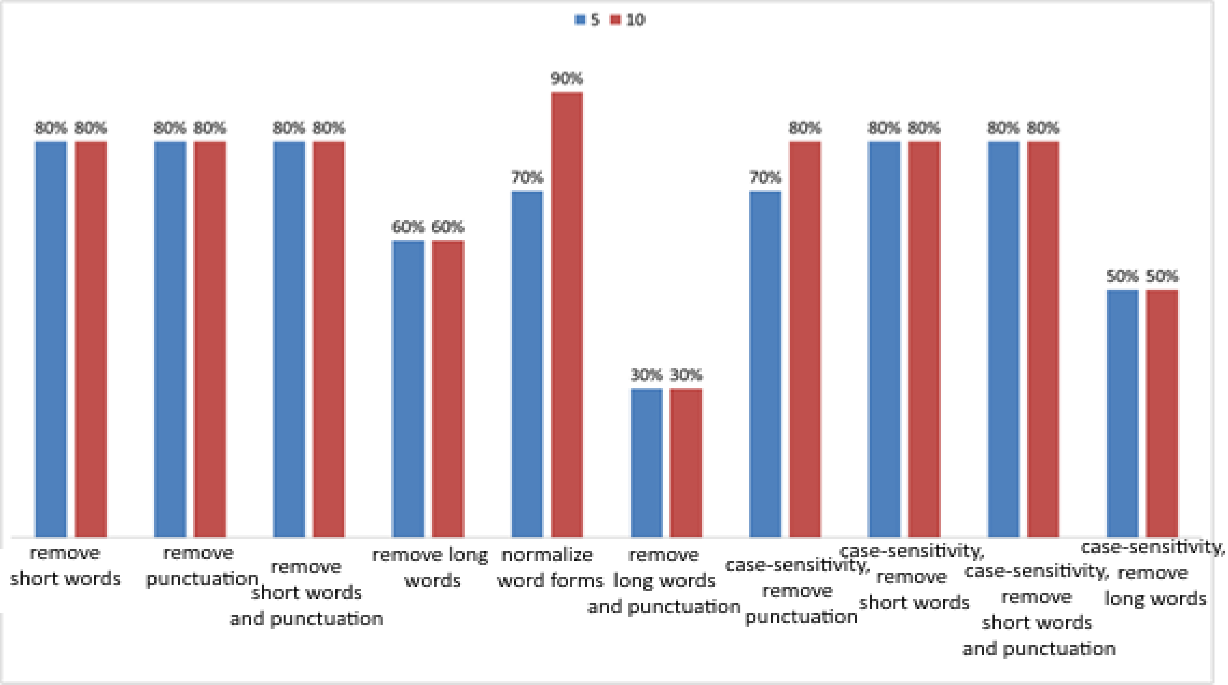


Fig. 8. Results for different types of text normalization.

According to the results of the experiment, it can be assumed that short words and punctuation marks do not affect the final classification effectiveness. By using the prior removal of short (insignificant) words, together with the removal of punctuation mark, or separately, the total recognition of the author's works does not change and is held at the level of 8 recognized works out of 10.

In turn, in the case of prior removal of long words, a lot of information about the author's style is lost. The accuracy of determination falls by 20 – 30%. This observation confirms the results of experiments 4, 6, and 10. Together with the removal of punctuation, the recognizability drops to 3 words out of 10, which is the worst indicator of all the experimentally verifiable changes. Presumably, long words and punctuation accumulate most information about the author's style.

As a result, we can point to the possibility of a pre-removal of short words and punctuation marks, as they do not affect the accuracy of the determination. This modification will increase the processing speed and reduce memory consumption by the construction of the graph. Supposedly, one can use the normalization of the word forms to improve recognizability. But this approach has a weakness in the transformation of the forms of words. As of today, the conversion algorithm is based on the grammar rules of the studied language, which can distort the author's invented speech turns, losing the important elements of the author's style.

VII. Conclusion

According to the results of the conducted research, it can be stated that the use of this method makes it possible to formalize the specifics of the author's style as a software object. The main source of data for this object is objects that are multi-dimensional in their attributes, such as literary texts written by professional writers. It is assumed that the writer in his work adheres to a certain manner of writing, which makes it possible to use different methods for determining the authorship of his texts.

Summary results of the made experiments show the possibility of determining the authorship of the text with a probability of 60–80 %. In the process of setting up a pilot system, some new ideas how to increase the effectiveness of the developed approach, were obtained. These ideas require software implementation and new experiments to assess their impact on the effectiveness of text classification.

References

[1] P. A. Osipov and A. N. Borisov, "Abnormal action detection based on Markov models", in *Automatic Control and Computer Sciences*, vol. 45, no. 2. 2011, pp. 94–105. http://dx.doi.org/10.3103/S0146411611020052

[2] The GraphML File Format. [Online]. Available: http://graphml.graphdrawing.org. [Accessed 05 July, 2014].

[3] M. S. Elayidom, C. Jose et al, "Text classification for authorship attribution analysis", in *Advanced Computing: An International Journal*, ACIJ, vol. 4, no. 5, Sep. 2013, 10 p.

[4] N. Homem and J. P. Carvalho, "Authorship Identification and Author Fuzzy Fingerprints" in *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, 978-1-61284-968-3/11/2011 IEEE, 2011, pp. 1–6.

[5] A. Metwally, D. Agrawal and A. Abbadi "Efficient Computation of Frequent and Top-k Elements in Data Streams", University of California, Santa Barbara, USA, Tech. Rep. 2005–23, September, 2005.

[6] R. M. Dabagh "Authorship attribution and statistical text analysis", in *Metodološki zvezki*, vol. 4, no. 2, 2007, pp. 149–163.

[7] R. Zheng, Yi Qin, Z. Huang, H. Chen, "Authorship analysis in cybercrime investigation", H. Chen et al. (Eds.): *ISI 2003*, LNCS 2665, Springer-Verlag Berlin Heidelberg, 2003, pp. 59–73.

[8] P. N. Bennett, S. T. Dumais and E. Horvitz. "The combination of text classifiers using reliability indicators", Information Retrieval, vol. 8, no. 1, pp. 67–100, 2005.

[9] C. Sanderson and S. Guenter, "On Authorship Attribution via Markov Chains and Sequence Kernels," *18th International Conference on Pattern Recognition*, ICPR 2006, Aug. 20–24, 2006, Hong Kong, China. http://dx.doi.org/10.1109/ICPR.2006.899

[10] E. Stamatatos, W. Daelemans et al., "Overview of the Author Identification Task at PAN 2014", *CLEF Conference, PAN part*, Sheffield, UK, Sep. 15–18, 2014.

[11] H. P. Langtangen, "A Primer on Scientific Programming with Python", in *Texts in Computational Science and Engineering*, vol. 6. 4th ed. 2014, XXXI, 872 p. ISBN 978-3-642-54959-5.

[12] J. R. Johansson, P.D. Nation and F. Nori, "QuTiP: An open-source Python framework for the dynamics of open quantum systems", in *Computer Physics Communications*, vol. 183, Issue 8, 2012, pp. 1760–1772. http://dx.doi.org/10.1016/j.cpc.2012.02.021

Pavel Osipov, *Mg. sc. comp.* He is a Doctoral Student of the Institute of Information Technology, Riga Technical University. He received his Master Degree from the Transport and Telecommunications Institute, Latvia. His research interests include web data mining, machine learning and knowledge extraction. A large part of research is focused on different aspects of user behavior modeling. Another new area of interests is to explore using Python programming language at all steps of scientific research, from initial idea brainstorm support, through all steps, till final article preparation in Text format, to provide for young researchers the common uniform research envelopment.
E-mail: pavels.osipovs@rtu.lv.

Andrejs Rinkevics, B. Sc. (Eng)., Institute of Information Technology, Riga Technical University. His interests include computer vision, machine learning, business process improvement with modern technologies, implementation of solutions, research of wireless technologies and data security issues.
E-mail: andrejs.rinkevics@rtu.lv.

Galina Kuleshova is a Researcher with the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). She received her M. Sc. degree in *Decision Support Systems* from Riga Technical University. Current research interests include artificial neural networks, data mining, classification methods and bioinformatics.
Address: Institute of Information Technology, Riga Technical University, 1 Kalku Str., Riga LV-1658, Latvia.
E-mail: galina.kulesova@cs.rtu.lv.

Arkady Borisov holds the Doctoral degree of Technical Sciences in Control of Technical Systems and the *Dr. habil. sc. comp*. degree. He is a Professor of Computer Science with the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 205 publications in the area. He has supervised a number of national research grants and participated in the European research project ECLIPS.
E-mail: arkadijs.borisovs@cs.rtu.lv

Pāvels Osipovs, Andrejs Rinkevičs, Gaļina Kuļešova, Arkādijs Borisovs. Markova ķēžu pielietošanas iespēju izpēte autora stila identifikācijai
Rakstā aprakstīts pētījums par Markova ķēžu pielietošanu autora stila modeļu būvēšanai. Autora stila modelēšana ir aktuāls uzdevums. Pielietojot šādu modeli, to var salīdzināt ar dažādiem citiem tekstiem, kuriem autorība nav zināma. Šī salīdzinājuma rezultāts ir līdzības līmenis starp diviem tekstiem; ja tas ir pietiekami augsts, tad mēs varam teikt, ka abus tekstus rakstījis viens un tas pats cilvēks. Autora stila modeļa izmantošanas process ir iedalīts divās daļās: modeļa apmācība un tā izmantošana teksta analīzei. Modeļa apmācība balstīta uz tekstiem, kuriem autorība ir zināma. Tā rezultātā apmācīts modelis saglabā indivīdu teikumus un frāzes būvniecības iezīmes. Svarīga pieejas iezīme šajā stadijā ir prasība izmantot liela apjoma tekstu apmācības procesā. Izmantošanas posmā, apmācīts modelis tiek izmantots, lai aprēķinātu stila līdzības līmeni ar analizēto tekstu. Apskatīts teorētiskais pamats, kā būvēt Markova ķēdes grafu, balstoties uz autora tekstu, kā arī apskatīta iespēja turpmākai teksta attīrīšanai, pirms tas tiek izmantots, lai apmācītu modeli, un tā ietekme uz gala klasifikācijas rezultātu. Veikti dažādi eksperimenti, lai novērtētu parametru ietekmi uz izmantoto algoritmu klasifikācijas efektivitāti. Galīgais līmenis ir tad, ja pareizie rezultāti sasniedz 60–80 %, kas ir samērā labi. Tālākai izpētei vajadzētu palielināt klasifikācijas precizitāti.

Павел Осипов, Андрей Ринкевич, Галина Кулешова, Аркадий Борисов. Исследование возможностей применения Марковских цепей для идентификации авторского стиля
В статье описано исследование возможностей применения Марковских цепей в задаче построения модели авторского стиля. Моделирование особенностей стилистики человека является актуальной задачей. Имея такую модель, возможно сравнивать с ней различные тексты, авторство которых не установлено. Итогом такого сравнения будет уровень сходства авторских стилей двух текстов. Если он достаточно высок, то можно говорить о том, что оба текста написал один и тот же человек. Использование модели авторского стиля делится на две условные части: обучение модели и непосредственно ее использование для анализа текста. Построение модели происходит на наборах текстов, для которых заведомо известно авторство. В итоге созданная модель хранит в себе особенности построения фраз и словосочетаний конкретного человека. Важной особенностью подхода на данном этапе является требование использовать для обучения тексты большого объёма. На этапе использования обученная модель применяется для вычисления уровня сходства стиля с анализируемым текстом. Рассмотрена основная теоретическая база построения графа Марковской цепи, основываясь на авторском тексте. Рассматривается возможность дополнительной очистки текста перед его использованием для обучения модели и влияние этого на итоговый результат классификации. Произведены различные эксперименты для оценки влияния параметров используемого алгоритма на эффективность классификации. Итоговый уровень корректных результатов находится в районе 60–80 %, что сравнительно неплохо. Дальнейшие исследования должны увеличить уровень распознавания.