Proceedings of the 8th Student Computing Research Symposium (SCORES'22)

Ljubljana, Slovenia
October 6, 2022

Uroš Čibej
Luka Fürst
Lovro Šubelj
Jure Žabkar
(Eds.)

SCORES
https://www.scores.si

University *of Ljubljana*
Faculty *of Computer and Information Science*

Proceedings of the 8th Student Computing Research Symposium (SCORES'22)

Ljubljana, Slovenia
October 6, 2022

Uroš Čibej
Luka Fürst
Lovro Šubelj
Jure Žabkar
(Eds.)

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID=126673155

ISBN 978-961-7059-11-3 (Fakulteta za računalništvo in informatiko, PDF)

Digitalna izdaja je prosto dostopna
This digital publication is freely available
http://zalozba.fri.uni-lj.si/SCORES2022.pdf
DOI: https://doi.org/10.51939/scores22

Založnik: Založba UL FRI, Ljubljana
Izdajatelj: UL Fakulteta za računalništvo in informatiko, Ljubljana
Urednik: prof. dr. Franc Solina

Title **Proceedings of the 8th Student Computing Research Symposium (SCORES'22)**

Editors Uroš Čibej
(University of Ljubljana, Faculty of Computer and Information Science)

Luka Fürst
(University of Ljubljana, Faculty of Computer and Information Science)

Lovro Šubelj
(University of Ljubljana, Faculty of Computer and Information Science)

Jure Žabkar
(University of Ljubljana, Faculty of Computer and Information Science)

Conference **8th Student Computing Research Symposium (SCORES'22)**

Venue University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113, SI-1000 Ljubljana, Slovenia

Date October 6, 2022

Published by **University of Ljubljana**
Faculty of Computer and Information Science
Večna pot 113, SI-1000 Ljubljana, Slovenia
https://www.fri.uni-lj.si/en, dekanat@fri.uni-lj.si

Co-published by **University of Maribor**
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, SI-2000 Maribor, Slovenia
https://feri.um.si/en/, feri@um.si

Co-published by **University of Primorska**
Faculty of Mathematics, Natural Sciences and Information Technologies
Glagoljaška ulica 8, SI-6000 Koper, Slovenia
https://www.famnit.upr.si/en, info@famnit.upr.si

Edition 1st

Publication type E-book

Published Ljubljana, Slovenia, October 2022

Univerza *v Ljubljani*

Editors' Foreword

In computer science, there is certainly no lack of scientific gatherings. Even in a comparatively small country such as Slovenia, multiple conferences are held annually. However, a vast majority of those events are aimed at a general scientific community, which predominantly consists of professional researchers and PhD students. Students at the BsC or MsC level seldom engage in potentially publishable research, and even those who do rarely decide to put their work in print, be it fear of rejection or unfamiliarity with publication venues. However, if they were offered an opportunity to present their work at a conference that took into account their relative lack of experience but nevertheless upheld well-established criteria of acceptance (scientific novelty, referencing prior work, quality of presentation, etc.), they would be more likely to take up research during their BsC and MsC years, to publish the results of their work, and, ultimately, to pursue a scientific career. Furthermore, a student-centered conference might entice the 'uninitiated' audience to follow the suit. Needless to say, bringing students with different interests and from different backgrounds together would create a melting pot for all sorts of ideas, including those that can be potentially developed into future research papers.

Formerly known as StuCoSReC (Student Computer Science Research Conference), SCORES is one of the few conferences explicitly targeting the BsC and MsC student population. Even more, each paper *must* be authored by at least one BsC or MsC student. Ever since its inception in 2014, the conference has been jointly organized by the computer science departments of the University of Ljubljana, the University of Maribor, and the University of Primorska, although it is customary that every year one of them is selected as the primary organizer. In the year 2022, this role was played by the University of Ljubljana (Faculty of Computer and Information Science).

At SCORES'22, we accepted 12 papers and arranged them into three sessions of four papers each: Artifical Intelligence and Machine Learning, Algorithmics and Theoretical Computer Science, and Applications of Computer Science. Each paper had been examined by at least two reviewers. The authors come from a diverse set of academic and non-academic institutions, namely University of Ljubljana, University of Maribor, University of Primorska, Hongik University (Republic of Korea), Rhodes College (USA), University Medical Center Ljubljana, and the Ministry of the Interior of the Republic of Slovenia. The Best Paper Award was given to Benjamin Džubur, Žiga Trojer, and Urša Zrimšek, for their paper *Semantic analysis of Russo-Ukrainian war tweet networks*. Ina Bašić, Eric Gottlieb, and Matjaž Krnc, the authors of the paper *Some observations on the column-row game*, received the Best Presentation Award. Congratulations!

The papers that we received via the submission system, the reviews prepared by the members of our Program Committee, and the presentations given at the conference itself not only dispelled any concerns regarding quality but actually thoroughly exceeded our expectations. Having attended many conferences in our local environment and abroad, we dare say that quite a few of them were rivaled, if not surpassed, by what we saw at SCORES'22. Nevertheless, we are certain that there is still room for improvement, and we have no doubts that SCORES'23 will push the envelope even further.

We are immensely grateful to our reviewers, the members of the Program Committee; to David Nabergoj and Vitjan Zavrtanik, our PhD students who gave engaging talks on two cutting-edge research topics; to Prof. Franc Solina, who took care of everything that was necessary to have these proceedings published; to the people from Communication Office and Computer Center at our Faculty, who were extremely helpful throughout the process; to Iztok Fister from the University of Maribor, who told us all we had to know to get things started; and, last but not least, to Dewesoft and Wisdom Labs, Slovenian companies that generously funded not only the prizes awarded at SCORES'22 but … quite possibly those to be given at SCORES'23 and SCORES'24, too.

We sincerely apologize to all those whom we forgot to mention. It was certainly not done on purpose.

Uroš Čibej
Luka Fürst
Lovro Šubelj
Jure Žabkar

Conference Program

Use of network features to improve 3D object classification

Gal Petkovšek
gp19194@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

Janez Božič
jb1236@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

Marija Marolt
mm7522@student.uni-lj.si
Faculty of computer science,
University of Ljubljana Večna pot 113
SI-1000 Ljubljana, Slovenia

ABSTRACT

In our work we tackle a problem of 3D object classification, which is a task traditionally closer to computer graphics rather than network analysis. There are several different approaches for solving this problem including deep learning, topological data analysis, graph theory *etc.* We use the Mapper algorithm that transforms the point cloud into a graph, which simplifies the data while obtaining the key properties of the structure. This algorithm is already used to solve such a task however, the features extracted from the graph were very limited. The novelty that we introduce is the calculation of some network properties on such graphs which are used for classification. The results show that the models have better classification accuracy scores when using network analysis attributes in addition to attributes from topological analysis however, it is challenging to determine exactly which features will perform well for a classification of objects.

KEYWORDS

network analysis, 3D object classification, Mapper algorithm, feature selection

1 INTRODUCTION

3D object recognition is one of the most challenging fields in object recognition community. The goal of the task is for a model to differentiate between different 3D objects that are presented to it in a certain format.

In 3D computer graphics a polygon mesh is a collection of vertices, edges and faces that defines a polyhedral object [1] (however in this work we consider only vertices positioned in a 3D space - from now on this will be referred to as point cloud). Because the process of scanning objects and saving them in such format has become very common, a need has developed to be able to automatically classify and visualise such data. An example can be seen on Figure 1. Given points together can human easily discern silhouette as a chair. It takes a computer quite some time to process all the points and connect them intuitively, so we need to extract the necessary data from the point cloud, train a model, and then make a prediction.

The first and one of the most fundamental challenges of this task is choosing the best representation of such data to be inputted into the models for classification. One approach is to transform the data into a set of 2D images and feed them into the model [11] and another would be to just use the whole set of points [8]. In this work we used a Mapper algorithm [10] to transform the set of points into a compact network. An example of such transformation can be seen on Figure 1, where we can on the left side observe the original point cloud and the extracted network on the right. The idea is to represent the object in the simplest way possible, while still keeping the essential information required for the classification. From this relatively simple representation of an object we can compute some topological properties of the structure or some properties from network analysis that would be useful for classification. The options for attributes are quite extensive and in this work we limit ourselves to three topology features and 14 network analysis features. We then test different subsets of features to determine which combination is most effective for classification.

Figure 1: Chair point cloud and its network made with Mapper algorithm

There are a lot of different approaches of either describing the data or doing the actual classification of 3D objects. A paper from 2016 [8] describes an approach that takes all the data points and feeds them to a neural network. Another approach [11] consider the data as a set of 2D images. Another research [4] uses a technique called topology matching on graphs. Some authors [3, 6, 11, 13] also describe the use of Convolutional Neural Networks (CNNs) for solving such problems. Instead of CNN we will be using Support vector machine (SVM), as in the article [5] Anupama *et. al.* The Mapper algorithm that we are using to transform the data was first described in the article by Singh *et. al* [10] however, for classification they limited themselves to the outputs of the Mapper algorithm while we explore the network further.

2 METHODS

2.1 Problem definition

We have tested our approach on a dataset that consists of four classes (table, chairs, octopuses and spiders). We tackle this problem as a binary classification (deducing whether a sample is *e.g.* a table or not). More precisely we compare four different classification tasks (one binary classifier for every one of the four classes) in order to determine how the use of network analysis features affect the performance of models on different target classes.

For every one of the four models we test every possible subset of features that are described in Section 2.4.

New models (with network theory features) are compared to models with one homology feature.

https://doi.org/10.51939/scores22.01

2.2 Data

The data is obtained from McGill 3D Shape Benchmark. [14] It is a repository of point clouds for testing 3D shape retrieval algorithms. We use the mesh format where each object's surface is provided in triangulated form. To provide versatility but also enough models in every class, one class is containing goal object models and second class containing three other objects models. There are point clouds of 21 tables, 23 chairs, 25 spiders and 31 octopuses taken from McGill 3D Shape Benchmark repository (together 100 samples). Examples of the referred samples can be seen on Figure 2.

Figure 2: Examples of four types of objects in the database

2.3 Network construction

To create a network from data point cloud we use the Mapper algorithm. It is composed of three main steps: the filtering function, the cover function and the clustering function [12]. After some initial testing we chose the best working components. Example of the transformation is shown on figure 1.

The main purpose of the filtering function is to reduce the dimensionality of the data. In our case this function projects data points defined in a 3D space onto the xy plane.

The next step is to calculate groups of overlapping points with the help of the covering function. This function takes as input the filtered space, calculates overlapping groups with dividing and transform points from each group back to the original space. Overlapped groups are computed with division of each input dimension by 3 equal-length intervals with fractional overlap of 0.15.

Finally, we apply a clustering algorithm on each group and calculate an undirected graph. This is done with a clustering algorithm that is able to determine the number of clusters in a dataset automatically or by manually defining it. For this purpose we use the DBSCAN [7] algorithm where the maximum distance between two points for one to be considered as in the neighborhood of the other is 10 using Chebyshev distance. Each cluster generates a node and the edges between them are created if the two clusters share some points.

The resulting network contains a lot less nodes than there are points in the point cloud and is therefore more easily handled than the original data representation. An example of such network can be seen in Figure 1.

2.4 Attribute calculation

A novel practice in point cloud object recognition is applying different homology based attributes to the output of the mapper. A "continuous" shape is built on the top of the data in order to highlight the underlying topology or geometry. By using homology, we observe numbers of components, holes, and voids while adding connections between the points. We measure when connected components appear and when they merge, representing their lifetime. In our case we use persistence entropy (entropy of the points in a persistence diagram later referred to as h1), and amplitudes of persistent diagrams, which plot lifetimes of connected components, computed with Wasserstein or Bottleneck distance as a classification parameters (later referred to as h2 and h3). From network theory we included the following features:

(0) **Number of articulation points** (NA) - nodes that separate the notwork into multiple sub-networks.
(1) **Average degree of node in a network** ($\langle k \rangle$)
(2) **Network density** (ρ)
(3) **Average clustering coefficient** ($\langle C \rangle$)
(4) **Normalised number of nodes in top 10% closeness centrality** ***(number of nodes having 90% or more of maximum node closeness centrality divided by number of nodes)*** ($top10\_l^{-1}$)
(5) **Normalised number of nodes in top 10% betweenness centrality** ***(number of nodes having 90% or more of maximum node betweeness centrality divided by number of nodes)*** ($top10\_\sigma$)
(6) **Normalised number of cliques with 4 nodes** ***(number of cliques containing 4 nodes divided by number of nodes)*** (NClique)
(7) **Degree assortativity coefficient** (DAC)
(8) **Normalised number of leaves** ***(number of nodes with degree 1 divided by number of nodes)*** (Nl)
(9) **Maximum node closeness centrality** ($Max l^{-1}$)
(10) **Maximum node betweenness centrality** ($Max\sigma$)
(11) **Average node closeness centrality** ($Avg l^{-1}$)
(12) **Average node betweenness centrality** ($Avg\sigma$)
(13) **Number of Louvain Communities** (NLC)

Attributes such as number of articulation points, number of leafs or number of louvain communities tell us a lot about the articulation parts of the structure which could be a very valuable information for the model. Average degree of a node and network density provide information on how many edges formed in the network which in other words is how many of clusters obtained from the mapper algorithm described in Section 2.3 share some points. Features including node centralities contain information about the importance of specific nodes (either on average, maximum or just what proportion of nodes achieves the higher centralities). The idea behind number of cliques and average clustering coefficient is that they could indicate if an object would contain some larger even surface that would be more fully connected.

Our goal is to determine the best combination of features, however since we only have a very limited number of samples, we do not consider too large subsets of features (on which the model would only overfit and produce meaningless results). Therefore we restrict the number of features to either 4 (if we also include some homology) or 6 otherwise.

We extract most of the measures from Python library NetworkX [9], and calculate the remaining with custom code using data from the

library. Specifically average network clustering, number of articulation points, assortativity coefficient, number of louvain communities, betweenes and closeness centralities were calculated with NetworkX functions.

2.5 Learning

For modeling we use Support Vector Machines (SVM) implementation from SciKit Learn [2] library for Python with a linear kernel setting. Other hyperparameters of the model are set to their default values as specified in the SciKit Learn [2] documentation.

We evaluate our models with 10-fold cross validation to improve stability of the results.

For a metric we use AUC (area under the ROC curve) which is a metric ranging from 0 to 1 (in practice from 0.5 to 1 as any classifier scoring lower than 0.5 can be inverted, transitioning to AUC>0.5) with 1 being the most and 0 (in reality 0.5) being the best score. This metric was chosen since it shows how well the model can separate the samples into correct classes. It is calculated by first constructing the ROC curve, which shows the relation between true positive rate (TPR) and false positive rate (FPR). It is obtained by moving the threshold and for each value calculating the TPR and FPR. After all points are gathered, the AUC can be calculated.

2.6 Experimetal setup

Experiments are conducted on the data set of labeled point clouds, described in Section 2.2. As we have 4 different labels we conduct 4 independent experimets, each time choosing one label as positive (object belongs to that group) and the other labels as negative (object does not belong to the chosen group).

For each of those 4 experimets we first transform all the cloud points to smaller networks using the mapper algorithm. For each one of those we then extract the features (both topological and features form network analysis). Once we have the labeled feature vectors we perform a grid search over all feature subsets (that suffices the size limitations presented in Section 2.4). For each of these subsets we performe a 10-fold cross validation with the SVM model (each fold using 90 samples for training and 10 for testing). The results are then saved and analysed.

3 RESULTS

The goal of our research is to determine if the additional network features contribute to the scores and if so which features are the most beneficial to include.

To determine whether adding additional features is even beneficial for the predictions, we first take a look at the results of the best models (models built on subsets of features that performed best according to the AUC score). Combinations of attributes that achieve the highest score in our test cases are displayed below. In addition to the best performing subset of features and the corresponding scores we included the results for best performing homology for each use case (same results as can be seen in Table 1).

- **Tables:** The best features are $Maxl^{-1}$, $Avgl^{-1}$ and h3. AUC score is 0.96 (AUC using only h3 was 0.84).
- **Chairs** The best features are NA, $\langle C \rangle$ and NLC in combination with either $top10\_l^{-1}$ and $Max\sigma$ or h2. AUC score is 0.96 (AUC using only h1 was 0.9).

	tables	chairs	octopus	spiders	Average
h1	0.62	0.91	0.66	0.71	0.73
h2	0.48	0.80	0.48	0.77	0.63
h3	0.84	0.60	0.62	0.81	0.72
NA	0.50	0.48	0.50	0.48	0.49
$\langle k \rangle$	0.86	0.43	0.45	**0.79**	**0.63**
ρ	0.43	0.37	0.56	0.41	0.44
$\langle C \rangle$	0.24	0.36	0.27	0.33	0.30
$top10\_l^{-1}$	0.59	0.44	0.56	0.30	0.47
$top10\_\sigma$	0.74	**0.77**	**0.71**	0.37	**0.65**
NClique	0.35	0.42	0.46	**0.79**	0.51
DAC	**0.90**	0.44	0.46	0.39	0.55
Nl	0.62	0.35	0.48	0.49	0.49
$Maxl^{-1}$	**0.92**	0.43	0.37	**0.67**	0.60
$Max\sigma$	0.43	**0.58**	**0.57**	0.50	0.52
$Avgl^{-1}$	**0.92**	0.40	0.56	0.50	0.60
$Avg\sigma$	0.82	0.42	**0.69**	0.60	**0.63**
NLC	0.47	**0.49**	0.52	0.50	0.50

Table 1: The results of the model for each individual feature (only 1 feature used for classification).

- **Octopuses** The best features are NA, DAC and h3. AUC score is 0.85 (AUC using only h1 was 0.66).
- **Spiders** The best features are $top10\_\sigma$, Nl, $Maxl^{-1}$ and h2. AUC score is 0.89 (AUC using only h3 was 0.81).

The best performing features are greatly dependent on the domain (specific object we want to classify), which is also seen in Table 1. From our results of combination of best performing features we can see improvements over the baseline (using only homology for classification). Addition of network analysis attributes to the feature vector improves classification. However, based on the results above we cannot determine which features in general perform best.

To determine this we first viewed the scores of models using only single features to predict the class. Each column (classification problem) has the three best results bolded. As we can see, we could not point out a few features that would be significantly better than the others in all cases. The quality of each attribute is greatly dependant on the class that we are trying to predict. This can easily be explained as different objects tend to produce different networks and each has different properties that separate it from the rest. Some of the best performing features are $top10\_\sigma$, $Max\sigma$ and $Maxl^{-1}$ however, while most of them perform quite well on two classes they perform worse on the other two. More consistent features that perform relatively well on most classes are for example $top10\_\sigma$ and $Avg\sigma$.

To further test the importance of features we set a cutoff point at the score of the model using the best performing homology feature (for each classification task) and only consider models with feature subsets that scored higher. For each feature we calculate in what percent of those models it appears. Those results are presented in Figure 3. We can observe again that the results mostly differ when it comes to the observed class. More preferable are features that perform well on all or most of the use cases. Such example would

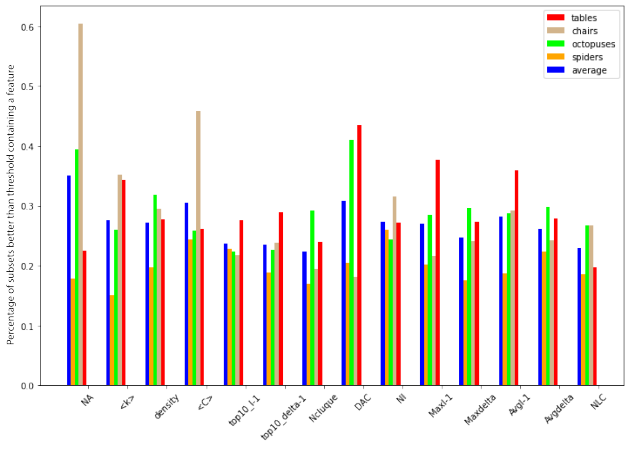


Figure 3: Chair point cloud and its network made with mapper algorithm

be $\langle C \rangle$, Nl, $Avgl^{-1}$ or $Avg\sigma$. The latter two are by themselves a relatively good features. $\langle C \rangle$ is not as successful on its own but is often present in the most successful subsets. It indicates how connected a network is therefore an objects with a more homogeneous body might be more connected than the ones with more articulation parts. The number of louvain communities could also be a good measure to determine out of how many parts the object consists since it also appears in many of the best subsets for all use cases.

NA is one of extreme examples that perform very well on two classes but quite worse on the others. Number of articulation points is greatly dependant on how many nodes there are in general and while in some cases it might be a very good indicator of how many articulation parts there are in others it might be deceiving as for example a long leg of a spider might have multiple articulation points. Such features (DAC is another such example) might also be taken into consideration when building a model however it is not guaranteed that they will perform well on all use cases.

4 CONCLUSION

Our experiments have shown that it is beneficial for a model for 3D object classification to use (in addition to topology features) network analysis features. However, it is difficult to determine which will perform best for a certain use case. We have tested our models on four different classification problems and the results have shown that while some attributes are indeed in general better than the others, for the majority of objects it is hard to determine which combination will perform best.

In future work we could expand our research by including a larger dataset (include more objects). We would also like to test more network features, and get an even better insight into what are the most beneficial attributes.

REFERENCES

[1] 2022. Polygon Mesh. https://en.wikipedia.org/wiki/Polygon_mesh
[2] David Cournapeau. 2022. SciKit Learn: Machine learning in Python.
[3] Kan Guo, Dongqing Zou, and Xiaowu Chen. 2016. 3D Mesh Labeling via Deep Convolutional Neural Networks. *ACM Trans. Graph.* 35, 1, Article 3 (dec 2016), 12 pages. https://doi.org/10.1145/2835487
[4] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. 2001. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 203–212. https://doi.org/10.1145/383259.383282
[5] Anupama Jawale and Ganesh Magar. 2019. Comparison of Image Classification Techniques : Binary and Multiclass using Convolutional Neural Network and Support Vector Machines. (12 2019).
[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[7] Jiirg Sander Martin Ester, Hans-Peter Kriegel and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.
[8] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR* abs/1612.00593 (2016). arXiv:1612.00593 http://arxiv.org/abs/1612.00593
[9] Aric Hagberg Pieter Swart Dan Schult. 2022. NetworkX: Network analysis library.
[10] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. 2007. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Eurographics Symposium on Point-Based Graphics*, M. Botsch, R. Pajarola, B. Chen, and M. Zwicker (Eds.). The Eurographics Association. https://doi.org/10.2312/SPBG/SPBG07/091-100
[11] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *CoRR* abs/1505.00880 (2015). arXiv:1505.00880 http://arxiv.org/abs/1505.00880
[12] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Wojciech Reise, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. 2021. giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration. arXiv:2004.02551 [cs.LG]
[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.
[14] Juan Zhang, Kaleem Siddiqi, Diego Macrini, Ali Shokoufandeh, and Sven Dickinson. 2005. Retrieving Articulated 3-D Models Using Medial Surfaces and Their Graph Spectra. 285–300. https://doi.org/10.1007/11585978_19

Efficient machine learning based graph layout calculation for investigation platform

Alen Granda
alen.granda@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Niko Lukač
niko.lukac@um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Aleksander Pur
aleksander.pur@policija.si
Ministry of the Interior
Štefanova ulica 2
SI-1000 Ljubljana, Slovenia

Štefan Kohek
stefan.kohek@um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Abstract

Modern web technologies enable interactive visualization of graphs in a web browser. However, larger graphs, which are common in investigation data, bring numerous technical limitations in terms of transfer bandwidth and application responsiveness. In this paper we propose a machine learning based method to efficiently transfer graphs' data between the client and the server. Graphs are stored in the investigation platform database on the server and are transferred to web browser on the client for interactive visualization and manipulation. For this reason, we utilize a concept called graph embedding. The main aim is to offer responsive web application due to less complex layout calculation algorithm on the client, less bandwidth requirements and incremental visualization of nodes during graph transfer. We performed distinct experiments, which demonstrate faster graph visualization on the client. We perceived up to 130 times faster layout calculation on the client compared to standard force graph algorithms with maximum 1000 nodes, while preserving adequate visualization accuracy.

Keywords

machine learning, graph embedding, graph visualization, investigation platform, web application

1 Introduction

Data representation using graphs is nowadays widely adopted standard at any form of applications [4]. Graphs provide natural and intuitive method of presenting enormous amounts of research data (e.g. bank transactions between entities, relations on social media), which daily increase in size. In this paper, the main focus is on modern web applications enriched with graphs, which are the basis of research platforms. Example of such application is an advanced investigation platform [13], which facilitates investigations by visualizing nodes with their relations using graphs. We extend the platform by utilizing suggested approach to facilitate several drawbacks, which are discussed in succeeding paragraph.

For graph visualization, most common approach in web applications is D3.js library [2], which uses Dwyer's proposition of incremental graph layout calculation. It computes incremental layout with complexity of $O(|V|\log|V| + |E|)$ per iteration where V is a set of vertices and E a set of edges [3]. This algorithm is computationally demanding, which is noticeable especially at the side of low-power clients (e.g. mobile phones). Therefore, web application may suffer from unresponsiveness. Another difficulty is also lower bandwidth in comparison to local desktop applications. To construct a layout, algorithm expects complete graph data to be available, which prevents incremental visualization of graphs and therefore responsiveness of the web application. By application of our proposition, aforementioned problems can be alleviated due to simultaneous data transfer and graph construction.

This problem has been extensively studied in the literature in terms of time and space complexity. Gove [6] recently presented random vertex sampling algorithm running at $O(|V|)$ time and $O(|V|^{\frac{3}{4}})$ auxilliary space. With combination of parallelization, speedup ratio of 26.7% may be achieved [17]. In addition, by using specialized method for graph representation termed Log (graph) [1], high compression ratios with minimal cost decompression can be realized.

In this paper we propose machine learning-based graph layout calculation for web applications. Given nodes and edges, the server application trains neural network based on expected node coordinates calculated in advance. For the visualization in the web application on the client, only the weights of the neural network, graph nodes and graph edges are transferred from the server. The client is provided with the same neural network as the server. After having received the weights, the client is capable of incrementally constructing the layout by simultaneously receiving and predicting received node's coordinates, resulting in improved responsiveness compared to the standard layout calculation algorithms, which require complete graphs to be available in advance.

https://doi.org/10.51939/scores22.02

The experiments were performed on an advanced investigation platform data [13]. Results were compared to baseline visualization algorithms and assessed using graphs of accuracy and time complexity. We indicated lower bandwidth requirements with acceptable accuracy. In addition, faster graph construction on the client-side was demonstrated.

Below, the main contributions of the paper are outlined.

- Efficient procedure of conveying graph data from the server to the client is provided.
- Graph visualization workflow is designed, which breaks dependency between the nodes for the layout calculation and therefore enables incremental visualization of graphs.
- A methodology is delivered, which increases responsiveness of web applications on the client when visualizing enormous amounts of nodes and connections because of reduced visualizaton algorithm complexity.
- An approach of node coordinate prediction based on the node embedding method called random walk is introduced.

2 Layout construction

Proposed algorithm takes advantage of machine learning (ML) to predict node coordinates in the graph. However, graphs are known to hardly incorporate with ML algorithms due to their diversity [10]. As a consequence, they have to be transformed to an input whose neural network is familiar with. For this reason, a technique called graph embedding [7] is utilized. Graph embedding is a transformation of graphs, which preserves as much graph property information as possible. The main motivation of proposed algorithm is to suggest a concept of establishing a machine learning model for layout calculation of nodes on the server and transferring it to the client for execution while preserving original graph appearance.

Figure 1 depicts the proposed methodology. The powerful server is responsible for graph feature learning, while the client manages graph visualization. Below, each segment is described in detail.

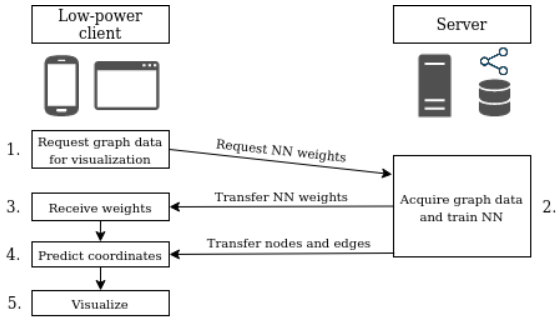


Figure 1: Proposed methodology workflow scheme.

2.1 Building machine learning model

ML model construction happens on the server-side of web application. The server is responsible to predict node coordinates and to teach the neural network. Figure 2 indicates suggested configuration of neural network based on the multilayer perceptron architecture (MLP) [14] with 2 dense hidden layers, each including q neurons. The output layer consists of 2 neurons, each representing coordinate in the final graph visualization. Nodes in the hidden layer are provided with the hyperbolic tangent activation function, whereas the output layer possesses simple linear activation function.

As an input data, MLP accepts n-dimensional array of node embedding, where n denotes the size of an embedding. The main aim of a node embedding is to transform a node into a vector space while preserving its properties [16]. Consequently, it retains node relatedness, resulting in clustering coupled nodes, while distancing unrelated nodes. Let c_i denote the i-th node in the graph. To construct its embedding, random walk-based node embedding *node2vec* [8] has been chosen. Random walk is a procedure of joining similar nodes in the complex graph to an embedding. As an input it accepts the starting node and produces a vector of specified length. Each element in the vector represents a characteristic of the visited node. With the help of extra parameters, which guide the walk, node2vec switches between breadth-first search (BFS) and depth-first search (DFS) to simulate the random walk. Correspondingly, it traverses local clusters as well as distant ones.

Before training the MLP, expected coordinates are computed for every node using an implementation of Fruchterman-Reingold force-directed algorithm [5] from NetworkX Python library [9]. As a learning algorithm backpropagation [11] in combination with Adam optimizer [12] is consumed. After having optimized the weights, the server sends them back to the client accompanied by the graph nodes and the edges. The client is subsequently responsible for graph layout calculation based on the received machine learning model and graph data.

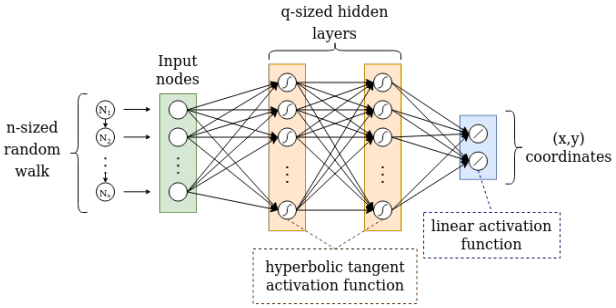


Figure 2: Proposed neural network scheme.

2.2 Graph visualization

The client is provided with the neural network from the server. As a consequence, it generates almost identical graph layout after acquiring the MLP weights. For this reason, the only task left for the client is to generate the random walks for an individual received node and predict its coordinates. After this stage, the client is prepared to visualize the graph.

3 Results and discussion

The aim of the experiments was to demonstrate improvements in speed compared to the standard visualization algorithms (e.g. Fruchterman-Reingold force-directed algorithm [5]) while maintaining the accuracy of graph visualization. Furthermore, the impact of neural network parameters was inspected by altering the number

of neurons q in the hidden layers and by modifying the number of random walks r per individual node. Having small number of random walks might result in many different graph layouts as the random walks are arbitrarily generated.

Experiments were examined on hardware with following specifications: CPU: Intel Core i7-8700, RAM: 16 GB, GPU: GeForce GTX 1070. In addition, Python programming language was selected with TensorFlow [15] library of version 2.4.0 for neural network construction and NetworkX [9] library of version 2.5 to execute force-directed algorithm.

Input data was acquired from an advanced investigation platform [13]. Graphs are directed, containing from 10 to 1000 nodes. Figure 3 depicts an example of a graph with 12 related nodes. Before learning the model, the training set was prepared using output coordinates of Fruchterman-Reingold force-directed algorithm. As a consequence, ground truth for results is the latter algorithm, which was executed using NetworkX library. The root mean squared error (RMSE) metric were applied to analyze the correctness of the predicted node coordinates. RMSE indicates how distant are node coordinates between the suggested algorithm and node coordinates from the training set. Individual node coordinate was presented as floating-point numbers ranging between -10 and 10 units. Training period was limited to 100 epochs and learning rate was set to 0.001.

Three experiments were performed. The aim of the first was to assess an accuracy of proposed algorithm against the number of neurons q in the hidden layers of the MLP neural network. The number of random walks r for every node was arranged to 50, each having the length $n = 10$. Hence, the size of the input layer was fixed to 10. Such configuration has appeared as a good compromise between efficiency and time consumption when running experiments. Beginning with 10 neurons in each hidden layer, number of neurons was increased by 10 until 200. MLP learned the weights and RMSE was calculated in every iteration. Procedure was repeated 10 times for graphs with 10, 50, 100 and 1000 nodes. Figure 4 indicates average RMSE of all recurrences. Improvement of accuracy might be observed as the number of neurons increased, regardless of the initial graph size. However, starting with 100 neurons per hidden layer neural network exhibited no significant enhancement.

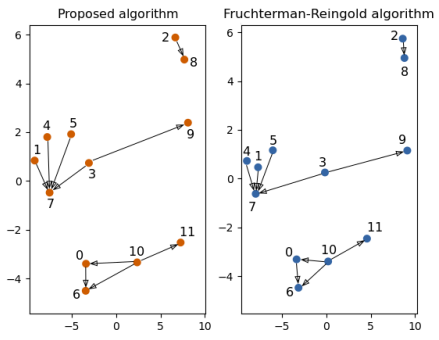


Figure 3: Example layout construction of proposed method against Fruchterman-Reingold algorithm.

The focus of the second experiment was to evaluate precision of proposed algorithm against the number of random walks r per node. Strategy was related to the initial experiment. The number of random walks was manipulated with presupposition of having $q = 100$ neurons in each hidden layer of MLP and the length of each random walk n to be 50. Firstly, MLP was trained with 5 random walks per node and RMSE was calculated. Afterwards, 5 random walks were added and procedure was repeated. Iteration stopped when the number of random walks reached 100. Presented process was rerun 10 times for graphs having 10, 50, 100 and 1000 nodes. Figure 5 manifests average RMSE. It is noticeable how accuracy increased as more random walks were included for each node. Correspondingly to Figure 4, exactness started converging to some value as the number of random walks exceeded 60.

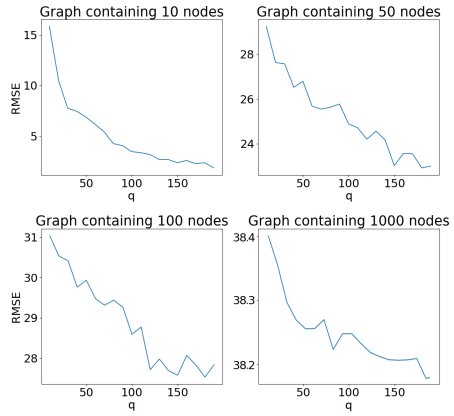


Figure 4: Graphs of accuracy against number of neurons in hidden layer of MLP.

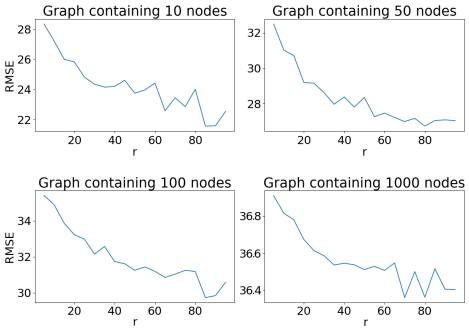


Figure 5: Graphs of accuracy against number of walks per node.

Finally, runtimes were examined. We experimented time consumption for node coordinate prediction with given neural network on the client for graphs of distinct size and discovered nearly linear increment with the number of nodes. Compared to an implementation of Fruchterman-Reingold algorithm in NetworkX library, Figure 6 indicates remarkable time-consumption reduction. In combination with the parallel layout construction while receiving the graph data, a considerable potential for minimizing time necessary to compute coordinates is perceived. Standard force-directed algorithms namely construct node coordinates with a time complexity of $O(n^2)$ per individual iteration of the simulation, which might

be time-consuming, especially for graphs extending 400 nodes as shown in Figure 6. Considering that the execution of algorithms occurs on devices with lower hardware specifications, the differences in time consumption should escalate.

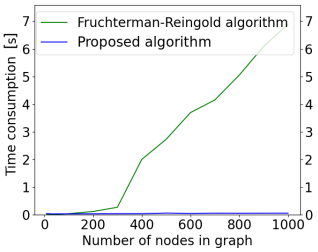


Figure 6: Time consumption for node coordinate calculation.

As mentioned, the purpose of the presented methodology was also to decrease bandwidth requirements. Let us presume our proposition of MLP architecture with $q = 100$ neurons in each hidden layer and an input layer of size $n = 10$. Hence, neural network consists of 11200 weights. On condition that each weight is presented as 32-bit floating point, the amount of data required to be transferred between the server and the client is 44800 bytes, which is considerably compact regarding today's technology. After the weights are sent, the client is able to gradually construct the layout while receiving the nodes and the edges.

A perspective to expose is the resistance of the proposed algorithm to the number of edges in the graph. Being the graph fully connected or not, the algorithm requires to generate r random walks of length n for every node. Random walk construction of a node without connections would result in a vector containing the node itself n-times.

4 Conclusion

A method for graph visualization in modern web applications using MLP neural network has been presented. Based on the experiments, time consumption of proposed procedure is promising compared to the Fruchterman-Reingold algorithm. In addition, visualization accuracy has been preserved. For this reason, the experiments have manifested that the purpose of proposed algorithm is fulfilled.

One of the concerns might be managing multiple big data requests from clients. However, as modern server infrastructures are getting more powerful each day, we believe our concept has practical value. Instead of executing complex calculations on the client and transferring huge amounts of data between the server and the client, we conceive our method as a replacement to aforementioned techniques. In addition, proposed methodology is able to construct layout incrementally at the time of receiving the nodes from the server, resulting in improved user experience. The only limitation is the requirement for transferring edges in an appropriate order for a random walk on the client. The problem can be alleviated by limiting the length of the walk. Moreover, the server could take advantage of caching to avoid neural network learning process duplication.

In this paper, the main focus was on node coordinate prediction. Nonetheless, our contribution might be reproduced to predict other graph parameters (e.g. graph coloring). In addition, presented methodology could be improved by selecting other neural network architectures. Having inspected the experiments, we have perceived how RMSE increased with graph size. Nevertheless, results have improved as the complexity of MLP increased. For this reason, we propose future research to focus on selecting a more complex neural network for given task. Aditionally, we suggest:

- Testing an impact of node2vec random walk parameters on accuracy,
- Sensitivity of the results to the length of node embeddings,
- Testing other graph embedding techniques,
- An evaluation of server performance when receiving multiple requests from clients and an evaluation of distributed computation.

Acknowledgments

The authors acknowledge joint financial support from the Slovenian Research Agency and Slovenian Ministry of the Interior (target research programme No. V2-2117).

References

[1] Maciej Besta, Dimitri Stanojevic, Tijana Zivic, Jagpreet Singh, Maurice Hoerold, and Torsten Hoefler. 2018. Log (graph) a near-optimal high-performance graph representation. In *Proceedings of the 27th international conference on parallel architectures and compilation techniques*. 1–13.

[2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.

[3] Tim Dwyer. 2009. Scalable, versatile and simple constrained graph layout. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 991–998.

[4] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*. 1433–1445.

[5] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11 (1991), 1129–1164.

[6] Robert Gove. 2019. A Random Sampling O (n) Force-calculation Algorithm for Graph Layouts. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 739–751.

[7] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[9] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74. http://sites.computer.org/debull/A17sept/p52.pdf

[11] Henry J. Kelley. 1960. Gradient theory of optimal flight paths. *Ars Journal* 30, 10 (1960), 947–954.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization.. In *ICLR (Poster)*, Yoshua Bengio and Yann LeCun (Eds.). http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14

[13] Niko Lukač, Borut Žalik, Matej Brumen, David Jesenko, Štefan Kohek, Andrej Nerat, and Marko Bizjak. 2019. *Zasnova napredne preiskovalne platforme (NNP): zaključno poročilo.* UM FERI.

[14] Leonardo Noriega. 2005. Multilayer perceptron tutorial. *School of Computing. Staffordshire University* (2005).

[15] TensorFlow Developers. 2022. TensorFlow. https://doi.org/10.5281/ZENODO.4724125 Accessed: 2022-06-19.

[16] Mengjia Xu. 2021. Understanding graph embedding methods and their applications. *SIAM Rev.* 63, 4 (2021), 825–853.

[17] Abenov Zhansultan, Aubakirov Sanzhar, and Paulo Trigo. 2021. Parallel implementation of force algorithms for graph visualization. *Journal of Theoretical and Applied Information Technology* 99, 2 (2021), 503–515.

Music genre classification based on spectrograms of the sound recording using an ensemble of CNNs

Tadej Lahovnik
tadej.lahovnik@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Vili Podgorelec
vili.podgorelec@um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

ABSTRACT

Several papers have attempted to classify music genres based on the features extracted from sound recordings. However, none have implemented an ensemble classifier of different CNNs for various types of spectrograms.

One thousand sound recordings from the GTZAN database were used for classification by the authors. Each sound recording was converted into three different spectrogram types, resulting in 3000 spectrograms. 85% of the spectrograms were used to train three CNN models, and the remaining 15% were used for testing. The individual CNN models formed a classifier ensemble, which combined the predictions of respective models into a single prediction based on the sum of the scores of respective genres.

Since the accuracy of the classifier ensemble (54.67%) is higher than the accuracy of the individual classification models (44.00%, 53.33%, 26.67%), it was beneficial to combine the CNN models into one. The confusion matrix revealed some common errors in genre prediction. The somewhat low accuracy is likely a consequence of the truncated sound recordings. Although the classifier ensemble did not achieve high accuracy, it predicted the genre based on the spectrograms of the sound recording more accurately than a human. Weighting the individual CNN models could significantly improve the results.

KEYWORDS

classification, spectrogram, machine learning, convolutional neural network, music genre, sound recording

1 INTRODUCTION

During the last two decades, we have seen an increase in AI-driven recommendation systems on audio streaming platforms. There are two general approaches to music recommendation - collaborative and content-based recommendation. While the former recommends objects that the user group of similar preference has liked, the latter analyses the content of objects that a user has previously preferred and recommends the ones with relevant content [8]. For content-based music recommendation, automatic music genre identification plays an important role.

Various machine learning (ML) methods have been developed for accurate music genre classification [7]. The algorithms behind these systems are often based on metadata and features extracted from sound recordings using audio processing techniques. Audio signal processing algorithms generally involve analysis of a signal, extracting its properties, predicting its behaviour, recognising if any pattern is present in the signal, and how a particular signal is correlated to other similar signals [16]. The extracted audio signal features represent training data for a selected ML algorithm.

While such ML approaches have shown some excellent results, there are other ways of representing and processing audio signals. An audio signal can be visually represented with spectrograms, potentially revealing patterns characteristic of different types of music. Existing works [3, 12] have already approached the task using spectrograms or features extracted from sound recordings. Different spectrogram types can be used to represent an audio signal. For example, Mel spectrograms[1,2] can be used instead of typical spectrograms [3, 9, 17]. As spectrograms are visual representations of music, the most advanced deep neural network-based image classification techniques can be applied to music genre classification.

In this paper, we propose a new ensemble classification method that predicts the music genre of a sound recording based on several individual convolutional neural network (CNN) models, each trained on its type of spectrogram. To the best of our knowledge, this is the first report on music genre classification using an ensemble classifier combining different types of spectrograms. An essential advantage of the proposed method is the direct use of spectrogram images for training the CNN models, which does not require the often demanding process of extracting and selecting features from sound recordings.

2 BACKGROUND

2.1 Music genre

A music genre is a category of music characterised by a particular style. A genre can also be influenced by social conventions, marketing, association with a particular artist, and other external influences [2].

Repetition is the foundation of genres. A genre codifies past repetitions and encourages new repetitions [14].

In this paper, genres represent classes for classification. A class is a set of things that can be grouped meaningfully. We often think of a class in terms of the common properties of its members, especially those that distinguish them from other things that are similar in many ways [13].

[1] a method of representing audio visually [17]
[2] substitutes the frequency on the y-axis with the mel scale and indicates colours using the decibel scale instead of the amplitude [6]

https://doi.org/10.51939/scores22.03

2.2 Spectrograms

A spectrogram represents a signal's intensity or 'loudness' over time at the different frequencies present in a particular waveform. Changes in energy levels over time are displayed in a spectrogram. [1].

Spectrograms are two-dimensional graphs, where colours represent the third dimension [1]. Time is represented on the horizontal axis. The vertical axis represents frequency, which can also be interpreted as pitch or tone. The lowest frequencies are located at the bottom and the highest at the top.

2.3 Spectrogram generation process

The Librosa[3] library allows us to generate a simple spectrogram from a sound recording. The sound recording is converted into a floating point time series during the upload process. The resulting time series must be converted from the square of the amplitude to decibels before the spectrogram can be displayed.

Figure 1 shows an example of the first type of spectrograms we used for classification. The spectrogram shows the presence of specific frequencies over time. Orange represents a high presence of a particular frequency, and blue represents a low presence.

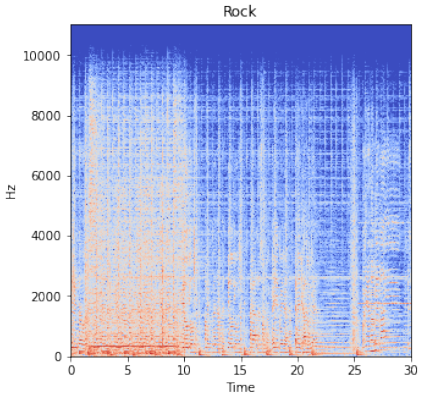


Figure 1: Spectrogram (frequency/time)

Figure 2 shows an example of the second type of spectrograms, which shows the presence of certain tones over time. Orange colour represents a high presence of a particular tone, and blue represents a low presence. The markings on the y-axis indicate the individual octaves (C1-C9).

Figure 3 shows an example of the third type of spectrograms, which shows the presence of tones across all octaves over time. Orange represents a high presence of a particular tone, and blue represents a low presence. Individual marking on the y-axis includes all matching tones from different octaves.

[3]https://librosa.org/doc/latest/index.html

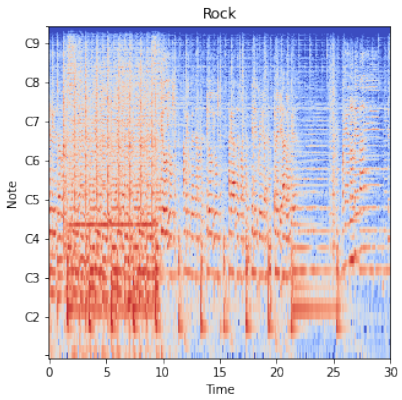


Figure 2: Spectrogram (tone/time)

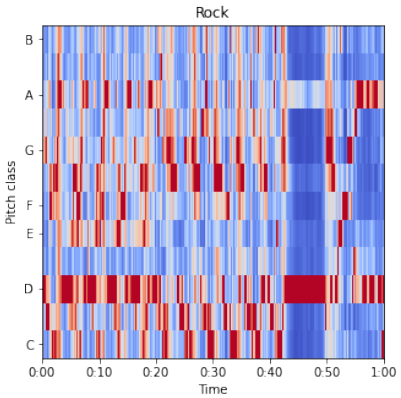


Figure 3: Spectrogram (chroma/time)

2.4 Classification

Classification occurs in many human activities. When used in its broadest sense, the term can refer to any situation in which a prediction or decision is made based on currently available information [10].

Numerous classification algorithms exist, including decision trees, rule-based learning, support vectors, Bayesian networks, and convolutional neural networks (CNNs). If necessary, classification algorithms may also be combined into ensembles (e.g., boosting, bagging, stacking, tree forests, and more) [12].

2.4.1 Neural network. A neural network is a set of algorithms that seek to identify the underlying connections in a set of data through a process that mimics the human brain [5].

Neural networks have three main components: an input layer, a processing or hidden layer, and an output layer [5]. Inputs can be weighted based on different criteria.

In neural networks, learning is accomplished by altering the weights across connections in response to new input data or learning patterns [15].

A convolutional neural network is adapted to analyse and recognise visual data such as digital images or photographs [5].

3 THE PROPOSED METHOD

The GTZAN database [11], which contains 1000 sound recordings, was used for the development. Each sound recording is 30 seconds long and belongs to one of the ten genres[4] provided by the database. The sound recordings were later converted into spectrograms using the Librosa library.

Three different types of spectrograms were created for each sound recording in the dataset. 85% of the spectrograms were used to train the CNN models, and the remaining 15% were used for testing. The training set was partitioned into training and validation sets using the Keras[5] library. The validation set consisted of 30% of the training data.

The generated images were classified using CNNs provided by the Sklearn[6] library. Three separate CNN models were created, trained, and combined into a classifier ensemble. The predictions of the individual models were combined into a single prediction based on the sum of the predictions.

The classifier ensemble was evaluated with several metrics[7]. Additionally, we displayed the results with a confusion matrix, revealing common errors of the implemented classifier ensemble.

3.1 The ensemble of CNN models

3.1.1 The CNN model. For classification, we used the sequential model from the Keras[8] library. Figure 4 shows the visualisation of the model.

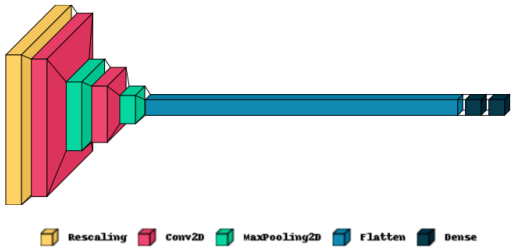


Figure 4: CNN model

The *Rescaling* layer standardises the input data. *Conv2D* creates a 5x5 convolution kernel that is convolved with the layer input to produce a tensor of outputs. *MaxPooling2D* downsamples the input along its spatial dimensions (height and width) using a 2x2 pooling window. The *Flatten* layer flattens the input. The *Dense* layer implements an operation that returns a vector with a length equal to the number of classes, providing the classification scores for each respective class (music genre in our case).

3.1.2 Training a CNN model. Each CNN classification model has been trained separately on its type of spectrogram images (each CNN model is trained from scratch with random initialisation of weights). Fifty epochs were used to train the model. Each batch contained 128 samples. Stochastic gradient descent was used to minimise the loss function of the CNN model. The CNN model outputs a vector with the same length as the number of classes (music genres). This vector represents the scores of individual classes - the higher the score for a particular class, the higher the expectation that the recording belongs to the corresponding music genre. Since these values can be arbitrary, we used a function called Softmax, which ensures that the sum of all the values is 1, thus constraining the individual scores between 0 and 1 [4].

3.1.3 Combining CNN models into an ensemble. After each CNN model is trained on its type of spectrogram from sound recordings, the individual CNN classifications are combined into an ensemble, comprising the classification results of all three specific CNN models. The instances of three specific CNN models have been combined in a single ensemble model by averaging the outputs of the corresponding Softmax layers. Figure 5 showcases the proposed ensemble method.

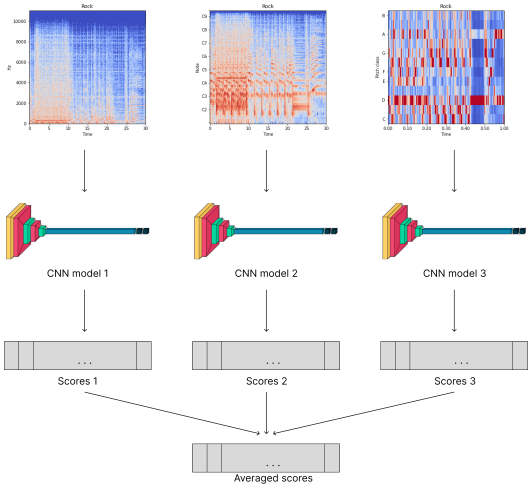


Figure 5: The proposed ensemble classifier

4 RESULTS

Three different CNN classification models were used to perform the classification. We used 150 test instances or 15% of the total dataset for prediction.

The first classification model (which predicted the genre based on classical spectrograms) correctly classified 66 test instances. The accuracy of the first classification model was 44.00%.

The second classification model (which predicted the genre based on spectrograms showing the presence of certain tones over time) correctly classified 80 test samples. The accuracy of the second classification model was 53.33%.

The third classification model (which predicted genre based on spectrograms showing the presence of individual tones across all octaves over time) correctly classified merely 40 test samples. The accuracy of the third classification model was 26.67%.

After combining the three individual CNN classifiers into an ensemble, the ensemble classifier correctly predicted 82 test instances. The accuracy of the ensemble classifier is 54.67%, which is higher than the accuracy of each CNN classification model. In this manner,

[4] blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock
[5] https://keras.io
[6] https://scikit-learn.org/stable
[7] accuracy, recall, precision, F-score
[8] https://keras.io/guides/sequential_model

merging the individual CNN classification models into an ensemble classifier was worthwhile.

Figure 6 shows the final confusion matrix of the proposed ensemble classifier. The labels of the individual rows show the actual genres of the test instances, while column labels show the genre predicted by the classification ensemble. Values in the cells show the number of test instances belonging to the genre shown in the row, classified as the genre shown in the column. In addition to the numerical labels, we can use the colour scale found next to the confusion matrix.

The confusion matrix shows that there are some common errors in genre prediction. The ensemble often predicted blues as rock, country as jazz and rock, and disco as hip hop, pop, and rock. The most incorrect predictions occurred for the disco genre. On the other hand, classical music seems to be the easiest to distinguish from other genres, as it was only misclassified on a few occasions with jazz.

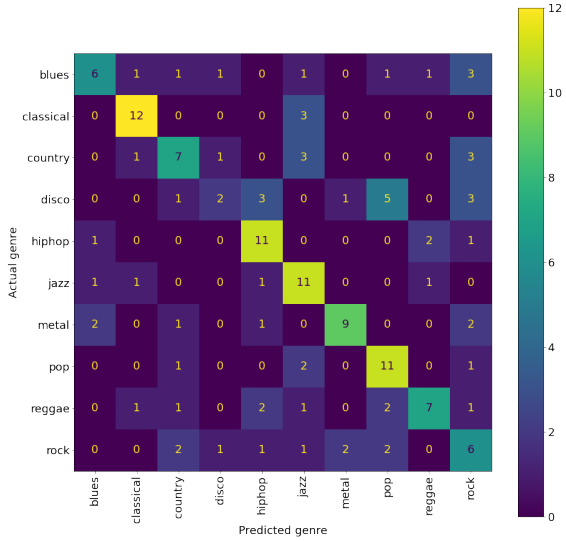


Figure 6: Confusion matrix

5 CONCLUSIONS

Although the achieved classification accuracy (54.67%) does not seem to be very high, we have to consider that it is difficult to distinguish between 10 different music genres, even for a human. Thus, we are satisfied with the result. However, there is still plenty of room for improvement.

Librosa produces wide white margins around the spectrogram, which are useless for classification. The spectrograms could be cropped to ensure that only the spectrogram is present in each image.

Classification of a music genre based on spectrograms is not the most accurate. The low accuracy is most likely also due to the truncated music files. Each sound recording is only 30 seconds long. If the dataset contained full tracks, we would have a more representative sample, which would most likely improve the results.

A single track may belong to several different genres. Therefore, performing a multi-label classification and comparing the results would be reasonable. Alternatively, the multi-label classification model could be used within the classifier ensemble.

Features could be extracted from the sound recordings, and another classification model could be added to the classifier ensemble. The classification would likely be more accurate as these features contain additional information.

It might also be worth considering weighing the individual classifiers in the ensemble. As a result, each classifier would not necessarily contribute the same amount to the final ensemble prediction.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).

REFERENCES

[1] [n. d.]. Spectrogram. https://en.wikipedia.org/wiki/Spectrogram Accessed: 2022-07-28.

[2] Shlomo Argamon, Kevin Burns, Shlomo Dubnov, and Roger B Dannenberg. 2010. Preprint from The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning Style in Music. , 45-58 pages. http://www.cs.cmu.edu/~rbd/

[3] Hareesh Bahuleyan. 2018. Music Genre Classification using Machine Learning Techniques. (4 2018). http://arxiv.org/abs/1804.01149

[4] Hmrishav Bandyopadhyay. 2022. Image Classification Explained. https://www.v7labs.com/blog/image-classification-guide

[5] James Chen. 2021. Neural Network. https://www.investopedia.com/terms/n/neuralnetwork.asp

[6] Ketan Doshi. 2021. Audio Deep Learning Made Simple - Why Mel Spectrograms perform better. https://ketanhdoshi.github.io/Audio-Mel/

[7] Ahmet Elbir, Hilmi Bilal Çam, Mehmet Emre Iyican, Berkay Öztürk, and Nizamettin Aydin. 2018. Music genre classification and recommendation by using machine learning techniques. In *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 1–5.

[8] Dongmoon Kim, Kun-su Kim, Kyo-Hyun Park, Jee-Hyong Lee, and Keon Myung Lee. 2007. A music recommendation system with a dynamic k-means clustering algorithm. In *Sixth international conference on machine learning and applications (ICMLA 2007)*. IEEE, 399–403.

[9] Jash Mehta, Deep Gandhi, Govind Thakur, and Pratik Kanani. 2021. Music Genre Classification using Transfer Learning on log-based MEL Spectrogram. *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*, 1101–1107. https://doi.org/10.1109/ICCMC51019.2021.9418035

[10] Donald Michie, David J Spiegelhalter, and Charles C Taylor. 1994. Machine learning, neural and statistical classification. (1994).

[11] Andrada Olteanu. 2020. GTZAN Dataset - Music Genre Classification. https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[12] Nikki Pelchat and Craig M. Gelowitz. 2020. Neural Network Music Genre Classification. *Canadian Journal of Electrical and Computer Engineering* 43 (6 2020), 170–173. Issue 3. https://doi.org/10.1109/CJECE.2020.2970144

[13] Claude Sammut and Geoffrey I Webb. 2011. *Encyclopedia of machine learning*. Springer Science & Business Media. https://books.google.si/books?id=i8hQhp1a62UC

[14] Jim Samson. 2001. *Genre*. Vol. 1. Oxford University Press. https://doi.org/10.1093/gmo/9781561592630.article.40599

[15] Yi Shang and Benjamin W Wah. 1996. Global optimization for neural network training. *Computer* 29 (1996), 45–54. Issue 3.

[16] Garima Sharma, Kartikeyan Umapathy, and Sridhar Krishnan. 2020. Trends in audio signal feature extraction methods. *Applied Acoustics* 158 (2020), 107020.

[17] Sugianto Sugianto and Suyanto Suyanto. 2019. Voting-based music genre classification using melspectogram and convolutional neural network. *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 330–333.

Super-Resolution Method for Reconstructing Street Images from Surveillance System based on Real-ESRGAN

Nguyen Quoc Toan
quoctoann3@gmail.com
Department of Electronic and Electrical Engineering,
Hongik University,
94 Wausan-ro, Mapo-gu,
Seoul, Republic of Korea

ABSTRACT

In many cities around the world, large sums of money are invested in surveillance camera systems, but few optimize the benefits and costs of those investments, and thus the overall impact of surveillance cameras on crime rates. In this paper, based on a technique named Real-ESRGAN applied to a practical restoration application that has been enhanced by the efficient ESRGAN. It is a super-resolution method that was developed in blind super-resolution to reinstate low-resolution street images with unknown and complicated degradations. It can be applied for security purposes in surveillance systems. Since video surveillance systems typically capture low-resolution images in many areas, the detection and identification of objects are sometimes required. This task's super-resolution is tough because image appearances vary depending on a variety of factors. The low resolution combined with poor optics is completely insufficient for identifying the subject of interest on the street, from a distance, in bad weather, or under any other limitations. Furthermore, to strengthen discriminator capability and create stable training dynamics, the U-Net discriminator was employed with spectral normalization. Hence, when compared to other experimental techniques, it can be demonstrated that this method delivers the best result. Experiment results show that super-resolution recovery of street images taken from a surveillance system is attainable with the following results: PSNR: 30.36dB and SSIM: 0.86.

KEYWORDS

computer vision, super-resolution, GAN, U-Net, Real-ESRGAN

1 INTRODUCTION

Image resolution is a significant factor in calculating image quality. The better the resolution, the more detailed the information in the image, making it more robust for objects on street recognition tasks. Improving image resolution has always been an unstoppable pursuit of industry and academia. Real-ESRGAN [11] has been applied due to its significant improvements compared to other experimental methods. The ESRGAN [12] was extended by Real-ESRGAN, to restore general real-world LR images by combining training sets with a more practical degradation process.

Simply put, Real-ESRGAN extends the classical "first-order" degradation model to "high-order" degradation modeling techniques, i.e., the degradations are modeled with multiple repeated degradation processes, each of which is the classical degradation model. A second-order degradation process is used empirically to obtain a great harmony of simplicity and effectiveness. A recent paper [14] assumes a random shuffling strategy for synthesizing more practical degradations. But even so, it still includes a set amount of degradation processes, and it is unclear whether all the shuffled degradations are useful. High-order degradation modeling, on the other hand, is more adaptable and aims to imitate the real degradation generation process, *sinc* filters in the synthesis procedure are employed to simulate ringing and over-shoot artifacts. Because the degradation space can be much bigger than ESRGAN, training becomes tough. First, the discriminator must be more powerful to distinguish realness from complicated training output. Secondly, the discriminator's gradient feedback should be more precise for local information improvement. In Real-ESRGAN, a VGG-style discriminator was upgraded to a U-Net design [7][8][13]. Thirdly, the U-Net architecture and complex degradations increase training instability. To balance the training dynamics, spectral normalization (SN) regularization [6][8] was used. It is simple to train Real-ESRGAN and obtain stability of local detail advancement and artifact suppression with the dedicated improvements.

2 REAL-ESRGAN

2.1 Classical Degradation Model

Blind SR recovering high-resolution images from low-resolution images that have undergone unknown and intricate degradations. Based on the underlying degradation process, existing techniques can be divided into two categories: explicit modeling and implicit modeling. The classic degradation model [1][4] is widely used in explicit approaches [2][3][15] and includes blur, downsampling, noise, and JPEG compression. To generate the low-resolution input, the classical degradation architecture [1][4] is commonly utilized. In general, the ground-truth image y can be convolved with the blur kernel k first. Afterward, with scale factor r, a downsampling operation is applied. Adding noise n yields the low-resolution x. Lastly, JPEG compression is used because it is ubiquitous in real-world images. In general, D represents the process of degradation.

$$x = D(y) = [(y * k) \downarrow_r + n]_{JPEG}, \tag{1}$$

- **Blur.** Gaussian filters, both isotropic and anisotropic, are selected. Although Gaussian blur kernels are typically utilized to model blur degradation, they may still not accurately represent real camera blur. Gaussian blur kernels [5] and a plateau-shaped allocation implement more diverse kernel shapes are generalized as well.

https://doi.org/10.51939/scores22.04

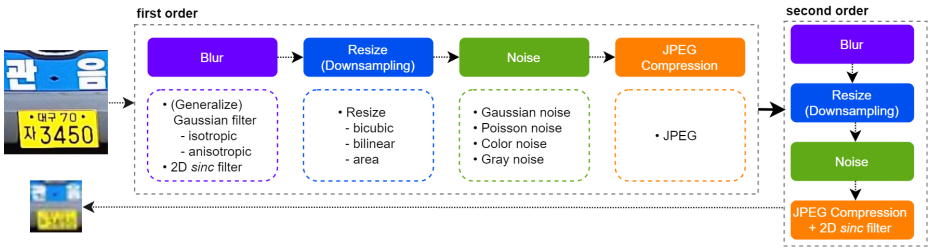


Figure 1: Illustration of High-order data generation Real-ESRGAN

- **Noise.** The additive Gaussian and Poisson noise types are applied. The probability density function of additive Gaussian noise is the same as the Gaussian distribution. The noise intensity is governed by the Gaussian distribution's standard deviation. Color noise occurs when unbiased sampled noise is present for each channel of RGB images. The Poisson distribution is initiated by Poisson noise. It is frequently used to estimate sensor noise caused by statistical quantum fluctuations, or variations in the photon flux sensed at a given exposure level. Poisson noise has a value proportional to image intensity, and noises at the pixel level are self - reliant.
- **Resize (Downsampling).** Downsampling is known as the resize operation. Nearest-neighbor interpolation, area resize, bilinear interpolation, and bicubic interpolation are all resize algorithms. Different resize operations yield different outcomes, some produce blurry images, while others may produce over-sharp ones with overshoot artifacts. To include more diverse and complex resize effects, a random resize operation from the options listed above. Nearest-neighbor interpolation is included because it exposes the misalignment issue and only deems the area, bilinear, and bicubic operations.
- **JPEG compression.** It is a prevalent lossy compression methodology for digital images. It decodes images to the YCbCr color space first, then downsamples the chroma channels. After that, the features are extracted into 8 x 8 blocks, and each block is converted with a 2D discrete cosine transform (DCT). [10] provides more information on JPEG compression algorithms. JPEG compression frequently presents unappealing block artifacts. A quality factor $q \in [0, 100]$ reflects the quality of compressed images, with a lower q indicating a higher compression ratio and lower quality.

2.2 High-order Degradation Model

When we use the above classical degradation model to generate training pairs, the trained model can handle some real-world samples. Nevertheless, it is still unable to handle some complex degradations in the real world, particularly unknown noises, and complex artifacts. This is due to the fact that synthesized low-resolution images still have a significant difference from realistic degraded images. To model more practical degradations, the classical degradation model was extended to a high-order degradation process. The classical degradation model only involves a limited number of fundamental degradations, which can be thought of as first-order modeling. Notwithstanding, real-world degradation is quite various and typically consists of a series of mergers such as imaging systems of cameras, image collection quality from video, and so on. In particular, the original image could be a very limited pixel image that the surveillance camera can get far away from the set up system, which inevitably contains degradations such as camera blur, sensor noise, and low resolution.

The classical first-order model could not model such a complex deterioration process. Therefore, a high-order degradation model is employed. An n-order model consists of n repeated degradation processes (as shown in Eq.2), in which each degradation process precedes the classical degradation model Eq.1 but with different hyperparameters. It should be highlighted that the term "high-order" is deployed differently here than it is in mathematical operations. It primarily refers to the time required to complete the same operation. [14] suggests that the random shuffling strategy encompasses repetitive degradation processes (e.g., double blur or JPEG). The key is the high-order degradation process which indicates that not all of the shuffled degradations are intended. To maintain a reasonable image resolution, the downsampling equation is altered with a random resize execution. Therefore, a second-order degradation process is employed, as it can remedy most real-world problems while remaining simple. The general flow of data generation stream is represented in Fig.1.

$$x = D^n(y) = (D_n \circ ... \circ D_2 \circ D_1)(y) \tag{2}$$

2.3 Ringing and overshoot artifact

Ringing artifacts commonly occur as spurious edges near sharp image transitions. They appear as bands near the edges. Overshoot artifacts are frequently combined with ringing artifacts, which manifest as a higher jump at the edge transition. The primary source of these artifacts is that signal is bandlimited and lacks high frequencies. These artifacts are very common and are typically caused by a sharpening algorithm, JPEG compression, and so on. *sinc* filter assists in cutting off high frequencies and synthesizing ringing and overshoot artifacts for training sets, its filters are deployed twice: during the blurring process and at the end step of the synthesis. The arrangement of the last *sinc* filter and JPEG compression is randomized transferred to cover a larger degradation space because some images may be over-sharpened (with overshoot artifacts) and then JPEG compressed, while others may be JPEG compressed first

and then sharpened. The equation of *sinc* is represented in Eq.3, where (i,j) represents the kernel coordinate, ω_c denotes the cutoff frequency, and J_1 means the first order Bessel operation of the first kind:

$$k(i,j) = \frac{\omega_c}{2\pi\sqrt{i^2+j^2}} J_1(\omega_c\sqrt{i^2+j^2}) \tag{3}$$

2.4 Networks and Training

2.4.1 ESRGAN. Firstly, the same generator (SR network) as ESRGAN [12] Fig.2 is used, i.e., a deep network with residual-in-residual dense blocks (RRDB). In conducting super-resolution with scale factors of x2 and x1, the extension with x4 ESRGAN architecture was represented. Since ESRGAN is a large network, the pixel-unshuffle (an inverse function of pixelshuffle [9]) was used before continuing to feed inputs into the main ESRGAN architecture to lower spatial size and increase channel size. As a result, most calculations are conducted in a smaller resolution space, which relieves GPU memory and the computational consumption of resources.

2.4.2 U-Net discriminator with spectral normalization (SN). Since Real-ESRGAN tackles a much bigger degradation space than ESRGAN, the existing discriminator architecture in ESRGAN is no longer appropriate. For complex training outputs, the discriminator in Real-ESRGAN necessitates more discriminative ability. It must ensure an accurate gradient responses for local textures in addition to discriminating global styles. The VGG-style discriminator in ESRGAN was enhanced to a U-Net architecture with skip connections, inspired by [8][13]. The U-Net generates realness values for each pixel that can provide the generator with detailed per-pixel responses. Meanwhile, the U-Net architecture and complicated degradations increase training instability. Regularization of spectral normalization [6] can aid in the stabilization of training dynamics. Furthermore, spectral normalization can help to reduce the oversharpness and annoyance caused by GAN training. Real-ESRGAN training can easily reach a better balance of local detail improvement and artifact suppression with these adjustments.

3 EXPERIMENT

3.1 Dataset

A brand-new dataset collected by HAIL was used for the experiment. It was collected by recording videos on streets in Seoul, South Korea with a Hanwha Techwin PNO-A9081RLP camera, then frames were extracted into images. There are 4500 images in total for use. The resolution is 4K (4096 x 2160). And 3500 crop images in the test set only included license plates, car brands, and text on the car for model evaluation. Fig.3 is samples from the proposed dataset.

3.2 Result

The luminance-based evaluation and comparison criteria, SSIM (Structural Similarity) and PNSR (Peak Signal-to-Noise Ratio) are used. To illustrate, two images were used for these evaluations. We refer to them as image 1 and image 2. Image 1 is the original degraded image from the test dataset, while image 2 is a reconstruction of image 1. SSIM is a method for calculating the similarity of two images. The SSIM values range from -1 to 1. PSNR is calculated to compare the quality of image 1 to image 2 which is calculated by

Table 1: Results comparison between Real-ESRGAN, BSRGAN and Bicubic

Method	Bicubic	BSRGAN	Real-ESRGAN
PSNR (dB)	25.51	28.09	30.36
SSIM	0.71	0.76	0.86

using the mean squared error (MSE). MSE is a statistical concept that means that an estimator's mean square error is the mean of the squares of the errors, or the difference between the estimate and what is evaluated, lower value means better performance. On the contrary, the greater the value of SSIM and PSNR, the higher the quality of the reconstructed image followed by Eq.4. Y represents the ground truth (reference) and Y^* describes reconstructed images:

$$PSNR(Y, Y^*) = 10\log_{10}\frac{255^2}{MSE} \tag{4}$$

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(Y^*(i,j) - Y(i,j))^2 \tag{5}$$

The SSIM evaluation between patches P_{Y^*} and P_Y is formulated as:

$$SSIM(P_{Y^*}, P_Y) = \frac{(2\mu_{PY^*}\mu_{PY} + c_1)(2\sigma_{Y^*}\sigma_{PY} + c_2)}{(\mu^2_{PY^*} + \mu^2_{PY} + c_1)(\sigma^2_{PY^*} + \sigma^2_{PY} + c_2)} \tag{6}$$

where $\mu_{PY^*}(\mu_{PY})$ and $\sigma_{PY}(\sigma_{PY^*})$ are the knowing and standard deviation of patch $P_{Y^*}(P_Y)$. $c1$ and $c2$ are minor constants. So, the mean score of patch-based SSIM over the image is SSIM (Y^*,Y).

4 CONCLUSION

In this paper, by applying Real-ESRGAN, a method for reconstructing low-resolution street images into recognizable images acceptable for recognition information tasks. The model achieved outstanding results (SSIM:0.86, PSNR:30.36dB). It proved that generating degraded real-life scenarios input play an extremely vital role in super-resolution models for street image recognition tasks. Real-ESRGAN performs greatly in the removal of artifacts and the restoration of texture details.

ACKNOWLEDGMENTS

I would like to express my heartfelt appreciation to HAIL (Hongik University - Artificial Intelligence Laboratory, Seoul, Republic of Korea), which is advised by Professor Seongwon Cho, for facilitating me in carrying out this research.

REFERENCES

[1] Michael Elad and Arie Feuer. 1997. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE transactions on image processing* 6, 12 (1997), 1646–1658.

[2] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. 2019. Blind super-resolution with iterative kernel correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1604–1613.

[3] Yan Huang, Shang Li, Liang Wang, Tieniu Tan, et al. 2020. Unfolding the alternating optimization for blind super resolution. *Advances in Neural Information Processing Systems* 33 (2020), 5632–5643.

[4] Ce Liu and Deqing Sun. 2013. On Bayesian adaptive video super resolution. *IEEE transactions on pattern analysis and machine intelligence* 36, 2 (2013), 346–360.

[5] Yu-Qi Liu, Xin Du, Hui-Liang Shen, and Shu-Jie Chen. 2020. Estimating generalized Gaussian blur kernels for out-of-focus image deblurring. *IEEE Transactions on circuits and systems for video technology* 31, 3 (2020), 829–843.

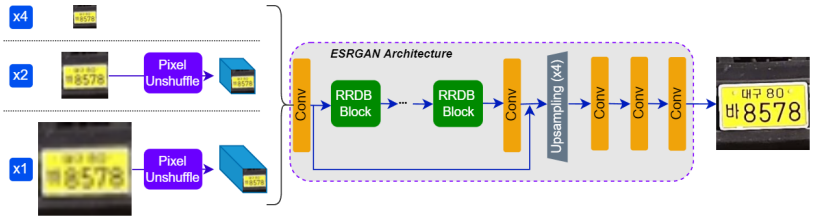
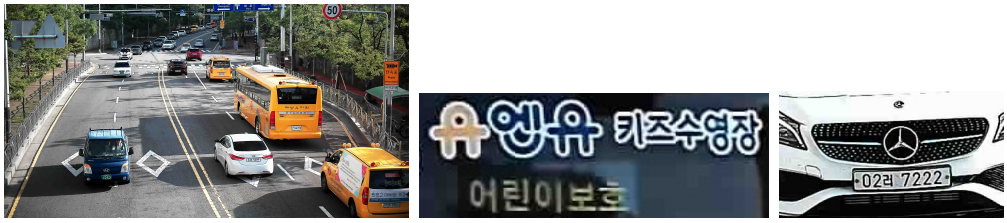


Figure 2: ESRGAN generator network. It first uses a pixel-unshuffle operation to decrease the spatial size and re-arrange information to the channel dimension for scale factors of x1 and x2.



(a) (b) (c)

Figure 3: a is sample from the train set, b and c are samples from the test set

Figure 4: Reconstructed image result comparing between Real-ESRGAN vs BSRGAN and Bicubic

[6] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[8] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. 2020. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8207–8216.

[9] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1874–1883.

[10] Richard Shin and Dawn Song. 2017. Jpeg-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, Vol. 1. 8.

[11] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. 2021. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1905–1914.

[12] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. 2018. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*.

[13] Yitong Yan, Chuangchuang Liu, Changyou Chen, Xianfang Sun, Longcun Jin, Xinyi Peng, and Xiang Zhou. 2021. Fine-grained attention and feature-sharing generative adversarial networks for single image super-resolution. *IEEE Transactions on Multimedia* 24 (2021), 1473–1487.

[14] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. 2021. Designing a practical degradation model for deep blind image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4791–4800.

[15] Kai Zhang, Wangmeng Zuo, and Lei Zhang. 2018. Learning a single convolutional super-resolution network for multiple degradations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3262–3271.

An analysis of time complexity and a discussion on interpretability for two methods of constructing social network graphs

Žan Jonke
zj0527@student.uni-lj.si
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
1000 Ljubljana, Slovenia
Munich Innovation Labs GmbH
Pettenkoferstr. 24
80336 Munich, Germany

ABSTRACT

Gathering useful information from user interactions on social media is a challenging task but has several important use cases. For example, law enforcement agencies monitor social media for threats to national security, marketers use them for launching marketing campaigns, etc. Since most social media platforms do not provide a standardized way of monitoring their data, most analyses are carried out manually. We aim to expedite this process by constructing social network graphs, where analysts can visually determine what users and contents are important. In this paper we compare two different approaches for constructing such graphs (path-weighted and degree-weighted). We analyze the time complexity of graph construction and discuss the usefulness of their visualization. In order to empirically evaluate both approaches, a method was developed, which stochastically generates data adhering to rules that govern the generation of data on a social media platform. We found that constructing degree-weighted graphs is faster, although the visualization of a path-weighted graph can answer more questions about the dataset.

KEYWORDS

Social media, network analysis, graph construction complexity

1 INTRODUCTION

Social media platforms have grown in popularity over the last two decades, manifesting several new ways of how humans interact with one another. The central idea is that users post contents and other users interact with them. For example, on Facebook [9] users post photos, write texts, create events etc. and other users like, comment or re-post the contents.

Law enforcement agencies monitor social media platforms for extremist groups, which might use them for spreading misinformation, incitement of violence or other forms of threat to national security [8, 14].

The application of social network analysis in marketing can provide marketers with valuable insights for developing communication and branding strategies by building up social capital in social networking sites [1, 5].

How data is collected from such platforms and what information can be extracted is of great utility. Being able to visually consider a network and to investigate it manually can be of great importance to analysts. In order for such a visualization to be beneficial, appropriate parts of the network graph (i.e. important users and content) must visually stand out. Our aim is to provide two ways of doing so and comparing them to one another from a technical perspective i.e. analyzing how much time is needed to construct such graphs and comparing them in terms of interpretability of their visualization.

2 RELATED WORK

Social network graphs can be constructed based on direct or inferred relations, including re-posting, replying or mentioning, through the shared use of hashtags or URLs, reciprocation or minimum levels of interaction activity, or friend/follower connections [4]. Karunasekera et al. [12] constructed networks of Twitter [10] accounts based on re-posts and mentions to discover communities active during the 2017 German election, valuing mentions and re-posts equally. URL sharing behaviour is often studied in the detection and classification of spam and political campaigns [2, 7, 17].

Edwards et al. [6] evaluated several different approaches of extracting social network graphs from datasets, which included linking two people if they were detected at the same event. Nasim et al. [13] introduce an approach of how to detect content polluting bots on Twitter. Their approach was to construct a two-mode user-event network linking two users if they had posted contents on the same day. Nguyen Vo et al. [16] constructed social network graphs which helped them evaluate an algorithm for revealing and detecting malicious re-posting groups. In their approach they considered only re-posts between users and for each pair determined re-post similarity and connected them if it was high enough.

3 METHODS

3.1 Constructing graphs

In this section we propose two different methods of constructing discrete graphs given a dataset which can be obtained from a social media platform. The dataset contains entities called users and the

https://doi.org/10.51939/scores22.05

content that they generated. Contents can also be reactions to one another i.e. a comment or a share.

Both methods have nodes of classes "user", "content", "comment" and "share". The methods differ in the way how edges are formed between the nodes, in which direction they are oriented and in the way the node weight is calculated.

The first method calculates node weights based on their degree (degree-weighted). We present this method in graph theoretic terms as follows:

Let $G_d(V, E)$ be a directed (degree-weighted) graph. Let $U \subset V$ be the set of users, $C \subset V$ the set of contents, $C_c \subset V$ the set of comments and $C_s \subset V$ the set of shares. Let $u \in U$, $c \in C$, $c_c \in C_c$ and $c_s \in C_s$. Edges $\{(u, c), (u, c_c), (u, c_s)\} \subset E$ only if u is the author of c, c_c or c_s. In this method there are no edges linking directly comments to contents or shares. Such a relation is represented as two edges $\{(u, c), (u, c_c)\} \subset E$ where u is the author of c_c (the same holds for c_s should c_c be a comment on a share). Shares are modelled in the same manner. The weights of edges are equal to 1, however the weights of nodes are equal to the node degrees.

The second method is based on calculating the node weight based on the number of simple paths that lead to that node (path-weighted). We present this method in graph theoretic terms as follows:

Let $G_p(V, E)$ be a directed (path-weighted) graph. Sets and nodes are defined as above. Edges $\{(c, u), (c_c, u), (c_s, u)\} \subset E$ only if u is the author of c, c_c or c_s and $\{(c_c, c), (c_c, c_s), (c_c, c_c), (c_s, c), (c_s, c_s)\} \subset E$ only if c_c is a comment to c, c_s or c_c or if c_s is share of c or c_s. The weights of all edges are equal to 1, but the weights of all nodes are equal to the number of directed simple paths ending in that node.

Cycles in the network would prevent the calculation of node weights from converging. The proposed connection rules guarantee that no cycles exist in the resulting networks. This can be easily seen in the following way: a node in a cycle must have both incoming and outgoing edges. As such U nodes can not be part of a cycle (they only have incoming edges). C nodes may have both incoming (from comments or shares) and outgoing (to the authors) edges. In order for a cycle to be formed there would need to be a directed edge from an author of the content to its comment, this is however not possible since such an edge is not defined. The same proof can be applied to both C_c and C_s nodes.

The degree-weighted graph can be easily understood and offers quick means for analysis, while the path-weighted graph is a bit more complex, albeit offers more in-depth information. In figure 1 we illustrate examples for both methods on the same dataset.

3.2 Generating random data

We want to empirically evaluate the time complexity of construction of both graphs and as such need many datasets with different numbers of contents and comments. Datasets from social media platforms varying in size might however be governed by different social dynamics, which have an effect on the network topology. Sampling from those datasets hence carries the risk of introducing an uncontrolled selection bias. Therefore we propose a way to artificially generate data (only systematically biased by the assumptions being made) in a stochastic manner governed by two parameters:

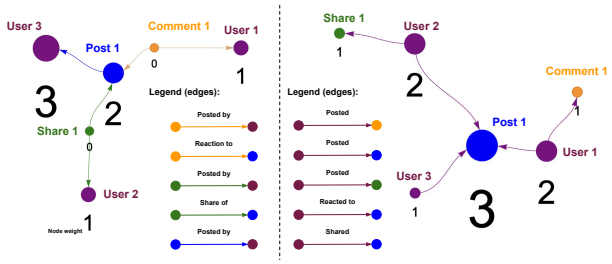


Figure 1: Example of a path-weighted (left) and degree-weighed graph (right) on the same dataset.

- the number of contents generated, denoted as N
- the probability that the new content generated is a comment, denoted as p_c

Our approach makes the assumptions that very little content on social media gains very high attention and that most users either post content or react to it but rarely both. To reflect this, each random content node that got generated got assigned a weight, which was sampled from a Pareto distribution [3], which is a power-law probability distribution that is used in description of social, quality control, scientific, geophysical, actuarial, and many other types of observable phenomena. This weight was used for weighed sampling, when assigning user reactions i.e. comment and shares to contents. Each user also got assigned a weight upon creation also sampled from a Pareto distribution, which got used when sampling users for authors of contents. A second weight is generated, which is equal to the inverse of the previous one and is used when sampling users for authors of comments. Using an inverse and weighed sampling manifests unsymmetrical behaviour of users in regards to their posting habits of contents and comments. We observed that most users who post high impact content are more likely to do so more often than others. To reflect this, the weights of all contents got multiplied by the weight of its author (i.e. the user). We also wanted to capture the observation that new users are more likely to enter a discourse, when a popular post was made. As such, when a new random content got generated with a high enough weight, the probability of a new user spawning in their respective pool, got increased and linearly decayed with each new content generated. We also assumed that content shares are more similar to content than they are to comments and therefore introduced a transformation from content to share, which was dependent only on an individual user and their probability to post a share. For each share, a content was sampled using their respective weights. To take into account nested comments, these can also be sampled when assigning user reactions, however we observed these are not as frequent and as such their weights are sampled from the uniform distribution.

4 RESULTS

4.1 Time complexity of construction

We assume a dictionary representation of a discrete graph. Computing a degree-weighted graph takes $O(|V|)$ time. This can be achieved by iterating over all contents, comments and shares and adding corresponding nodes and edges. Calculating node weights

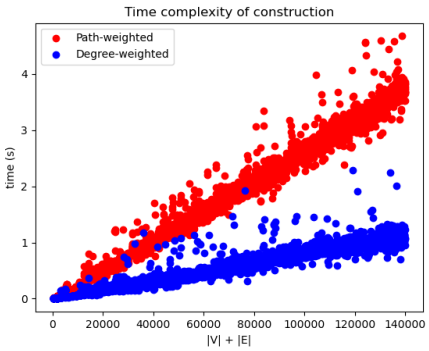


Figure 2: Time complexities of graph construction for both methods. Both depend linearly on $|V| + |E|$.

takes $O(|V|)$ time because all that is needed is looping over all nodes and calculating the node degree, which can be done in constant time.

Constructing a path-weighted graph as described above takes $O(|V|)$ time since the same concept of looping over all contents, comments and shares can be used. Now we consider calculating node weights. Calculating one simple directed path between two nodes using depth first search takes $O(|V| + |E|)$ time [15]. Naively doing so for all nodes and all possible paths results in a combinatorial explosion in terms of time complexity. This analysis assumes that lengths of paths are comparable to $|V|$. However in our case this assumption can not be made since long paths require nested comments or shares i.e. comments to comments or shares of shares. For simplicity we exclude such paths from our analysis since we observed those are usually not found very frequently in interactions on social media. We estimate that calculating all simple paths for all nodes to take $O(|V| + |A|)$ time, where A is a list of all ancestors of all nodes. We assume that calculating all ancestors of a node to be constant since we are not considering nested comments or shares. The following observations can be made:

- $|A| \approx |V| + |E|$ since every edge roughly introduces one new ancestor,
- a node might have a large number of ancestors, however each ancestor has at most two outgoing edges and is at most at a distance 2 from the source, meaning that depth first search finds all simple paths between two nodes in constant time.

When $|V|$ gets large enough, most of the time needed for the computation of a path-weighted graph is used for calculating node weights. Therefore calculating a path-weighted graph takes $O(|V|+|E|)$ time.

An empirical evaluation of time complexity can be seen in figure 2. Datasets of different sizes were constructed with N ranging from 0 to 45000 with a step of 500. At every step we generated 30 random datasets using the above described method and measured time of construction for both approaches.

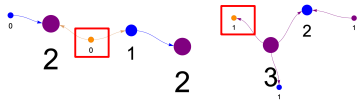


Figure 3: Comparison of interpretability of the path-weighted (left) and degree-weighted (right) graph. It is not possible to determine to what content does the comment belong to in the degree-weighed graph.

4.2 Interpretability

Here we demonstrate a way how one could interpret the node weights and degrees of both types of graphs. In a degree-weighted graph a content's weight reflects the sum of all users that reacted to it, with either a comment or a share. The same interpretation can be made for both comment and share nodes. A user's weight reflects the sum of all their activities. Conversely, in a path-weighted graph a content's weight reflects the sum of all comments and shares written as a reaction to it, hence its "impact". As before, the same interpretation can be made for both comments and shares. A user's weight reflects the sum of the contents written by the user (the "activity") plus the sum of the their impacts (total impact), hence their (relative) importance. Additionally the degrees of nodes in a path-weighted graph can be interpreted as weights of degree-weighted graphs.

As an edge is missing from either a comment or a share to the content in a degree-weighted graph, this results in an ambiguity. Consider figure 3. It shows two visualizations of the same data. For the degree-weighted graph it can not be determined by visualization alone where the comment belongs to, however this is not the case for the path-weighted graph.

4.3 Visualization

Here we show some visualizations of networks using the random data generating model described above. We generated two datasets with values of N equal to 1000 and 2000 and p_c equal to 0.8. In figures 4 and 5 two visualizations of path-weighted and degree-weighted graphs are shown. The visualizations were done using a Javascript library called vis.js where graph physics enable for a more flexible visualization. The physics solver ForceAtlas2 was used [11]. The tool allows for zooming and makes even larger networks still manageable to analyze manually. However, these visualizations take a lot of computing power and it becomes very time consuming to render networks which contain more than 5000 nodes. This issue can be mitigated by using alternative libraries with GPU support.

5 DISCUSSION

We have presented and analyzed two different approaches (path-weighted and degree-weighted) for constructing directed graphs from data modelling social media networks. We have shown that in terms of time complexity the degree-weighted graphs are favourable, since they require less time to construct, although the difference only becomes noticeable when the dataset is very large.

The advantages of degree-weighted graphs are that they are fast to construct and their visualizations are easy to interpret regarding questions about the most popular posts and which users post most

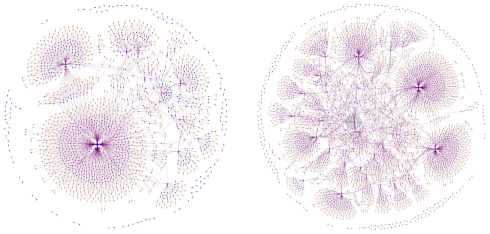


Figure 4: Visualizations of degree-weighted graphs with N equal to 1000 (left) and 2000 (right).

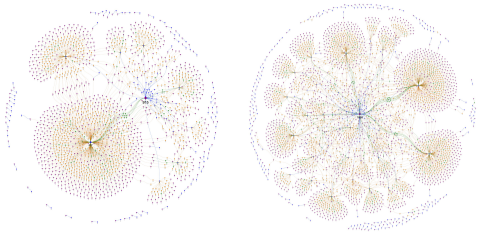


Figure 5: Visualizations of path-weighted graphs with N equal to 1000 (left) and 2000 (right). The same datasets were used as in figure 4.

frequently. The corresponding nodes will have a high weight and therefore visually stand out. A disadvantage of this method is that further analysis is harder to conduct i.e. it is harder to answer questions about the most important users, the impact of individual posts and about the users most people interact with.

The advantage of path-weighted graphs is for analysts to be able to answer the above questions more easily since all relations are unambiguously reflected by graph edges and more meaningful weights are given to content and user nodes. However, the disadvantage of this method is that network graph construction takes a longer time.

Analysing our model for generating random social media data, we find that it lacks modelling of phenomena where users are more likely to react with only a small pool of users (their friends and family), rather than with users who post the most popular content.

As a next step we plan to test the validity of our findings on real social media data and analyse the time complexities of adding new nodes to an existing graph in a realistic monitoring scenario.

ACKNOWLEDGMENTS

We would like to express our very great appreciation to Mathias Uhlenbrock and Oussama Jarrousse for their valuable and constructive suggestions during the planning and development of this research project.

REFERENCES

[1] Ioannis Antoniadis and Anna Charmantzi. 2016. Social network analysis and social capital in marketing: theory and practical implementation. *International Journal of Technology Marketing* 11 (01 2016), 344. https://doi.org/10.1504/IJTMKT.2016.077387

[2] Cheng Cao, James Caverlee, Kyumin Lee, Hancheng Ge, and Jinwook Chung. 2015. Organic or Organized? Exploring URL Sharing Behavior. (2015), 513–522. https://doi.org/10.1145/2806416.2806572

[3] Henry T. Davis and Michael L. Feldstein. 1979. The Generalized Pareto Law as a Model for Progressively Censored Survival Data. *Biometrika* 66, 2 (2022/09/27/1979), 299–306. https://doi.org/10.2307/2335662

[4] Lewis Mitchell Derek Weber, Mehwish Nasim and Lucia Falzon. 2021. Exploring the effect of streamed social media data variations on social network analysis. *CoRR* abs/2103.03424 (2021). arXiv:2103.03424 https://arxiv.org/abs/2103.03424

[5] Shaun Doyle. 2007. The role of social networks in marketing. *Journal of Database Marketing Customer Strategy Management* 15, 1 (2007), 60–64. https://doi.org/10.1057/palgrave.dbm.3250070

[6] Michelle Edwards, Jonathan Tuke, Matthew Roughan, and Lewis Mitchell. 2020. The one comparing narrative social network extraction techniques. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 905–913. https://doi.org/10.1109/ASONAM49781.2020.9381346

[7] Fabio Giglietto, Nicola Righetti, Luca Rossi, and Giada Marino. 2020. It takes a village to manipulate the media: coordinated link sharing behavior during 2018 and 2019 Italian elections. *Information, Communication & Society* 23, 6 (2020), 867–891. https://doi.org/10.1080/1369118X.2020.1739732 arXiv:https://doi.org/10.1080/1369118X.2020.1739732

[8] John S. Hollywood, Michael J. D. Vermeer, Dulani Woods, Sean E. Goodison, and Brian A. Jackson. 2018. *Using Social Media and Social Network Analysis in Law Enforcement: Creating a Research Agenda, Including Business Cases, Protections, and Technology Needs.* RAND Corporation, Santa Monica, CA. https://doi.org/10.7249/RR2301

[9] Meta Platforms Inc. 2004. Facebook. https://www.facebook.com. Accessed: 2022-07-29.

[10] Twitter Inc. 2006. Twitter. https://www.twitter.com. Accessed: 2022-07-29.

[11] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. 2014. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE* 9, 6 (06 2014), 1–12. https://doi.org/10.1371/journal.pone.0098679

[12] Fred Morstatter, Yunqiu Shao, Aram Galstyan, and Shanika Karunasekera. 2018. From Alt-Right to Alt-Rechts: Twitter Analysis of the 2017 German Federal Election. (04 2018), 621–628. https://doi.org/10.1145/3184558.3188733

[13] Mehwish Nasim, Andrew Nguyen, Nick Lothian, Robert Cope, and Lewis Mitchell. 2018. Real-Time Detection of Content Polluters in Partially Observable Twitter Networks. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1331–1339. https://doi.org/10.1145/3184558.3191574

[14] International Association of Crime Analysts. 2018. Social Network Analysis for Law Enforcement [White paper]. (2018).

[15] Robert Sedgewick. 2001. *Algorithms in C, Part 5: Graph Algorithms, Third Edition* (third ed.). Addison-Wesley Professional.

[16] Nguyen Vo, Kyumin Lee, Cheng Cao, Thanh Tran, and Hongkyu Choi. 2017. Revealing and Detecting Malicious Retweeter Groups *(ASONAM '17)*. Association for Computing Machinery, New York, NY, USA, 363–368. https://doi.org/10.1145/3110025.3110068

[17] Tingmin Wu, Sheng Wen, Yang Xiang, and Wanlei Zhou. 2018. Twitter spam detection: Survey of new approaches and comparative study. *Computers Security* 76 (2018), 265–284. https://doi.org/10.1016/j.cose.2017.11.013

Empirical evaluation of sequential, parallel and distributed implementations of k-means clustering

Andrej Perković
89201045@student.upr.si
Faculty of Mathematics, Natural
Sciences and Information Technologies
University of Primorska
Glagoljška ulica 8
SI-6000 Koper, Slovenia

Aleksandar Tošić
aleksandar.tosic@upr.si
Faculty of Mathematics, Natural
Sciences and Information Technologies
University of Primorska
Glagoljška ulica 8
SI-6000 Koper, Slovenia

ABSTRACT

In this paper we present a sequential, parallel and distributed implementation of the infamous k-means clustering algorithm. We perform extensive testing of all three implementations on state the art hardware, and show the performance benefits of paralellization. The research was inspired by a use-case of reverse logistics optimisation of wood in Germany, which translates to a facility location problem. K-means is an heuristic approach that renders surprisingly good results compared to mathematical modelling approaches, which are usually not feasible in large inputs as they belong to the class of NP-hard problems.

KEYWORDS

k-means, Clustering, MPI, Parallel computing

1 INTRODUCTION

The concept of reusing waste wood is not a new concept [4]. Over the years, the concept of reusing or recycling waste wood has been gaining increasing attention both from academia and industry participates. The potential of reusing waste wood has multiple benefits such as positive environmental effects as less trees need to be cut in order to meet the demand of raw materials. Currently, in most countries waste wood is collected but rarely sorted or decontaminated. Both of these processes are required before reusing waste wood. The process of sorting is important to filter out wood not suitable for reuse, which is mainly due to size constraints, and type of wood (hardwood/softwood). The decontamination process involves some mechanical cutting and grinding of parts of the suitable waste wood to remove unwanted objects (nails, screws, etc..) and chemical compounds such as adhesives. However, both of these processes are inherently costly, and the capital and operational expenses need to be justified by the added value obtained by reusing waste wood as oppose to buying new raw material [1]. To achieve this goal, legislation must both subsidise the transition to circular economy and at the same time impose restrictions or taxes on excessive CO_2 emissions [14].

Due to lack of investment and motivation by market participants, most of the waste wood is burned for energy and put into landfill where burning is not an option due to heavy contamination. Sometimes, this is done by accumulation sites directly.

For a successful implementation, new facilities for sorting and decontamination must be built. These facilities would then offer recycled wood to the market to fund their operational expenditure. The placement of such facilities is a logistics optimisation problem commonly known as the Facility Location Problem (FLP). Existing research is mostly focused on mathematical modeling and linear programming to find the near optimal positioning of facilities considering all constraints [1, 3]. However, due to the computational complexity of the problem, such solutions are not scalable for large logistic networks such as the entire EU zone.

K-means clustering has been heavily explored as a heuristic approach to solving large problems where linear programming solvers become infeasible [9]. Clustering is a simple and powerful principle for making sense of large swats of data. It is becoming more and more important in today's data-intensive and data-driven society [7]. *K*-means clustering is a rudimentary algorithm for achieving this goal. It has been one of the researchers' favorite, with a plethora of variations and tweaks [16]. It is widely studied, with the most notable work coming from Lloyd [10], Forgey [5], Friedman and Rubin [6] and MacQueen [11].

The crux of the algorithm are it's two steps - the *assignment* (binding) *step* and the *update step*. In the former, we assign each point to the "closest" cluster centroid. In the latter, we recalculate the centroid of the cluster. Definition of closeness depends on the choice of the algorithm. The algorithm stops once there are no changes in the binding. Recently, clustering has been used to improve runtime of Mixed-Integer Linear Models (MILP) to give the solver a better initial state then random [2].

In this paper we present an open-source parallel and distributed implementation of the k-means algorithm. We evaluate our implementation on the aforementioned use case using data obtained from the statistical office in Germany. Our initial data-set contains 10.000 accumulation sites, which accumulate used wood. The dataset was prepared by statistically estimating the amount of wood that should accumulate in an area based on the population density, and average waste wood created per inhabitant. Both of the aforementioned statistics were obtained from the statistical office of Germany. Each site is located using GPS coordinates and distances are computed using the Cosine-Haversine Formula [13].

2 IMPLEMENTATION

To tackle the problem at hand, we decide to use the weighted *k*-means clustering algorithm. It's different form the classical version in that it takes into consideration the capacity of the facility, not just the Euclidean distance. This way the calculation of the centroid of a cluster is biased towards the larger collection facilities of that cluster, which reflects practical needs of placing the treatment facilities

https://doi.org/10.51939/scores22.06

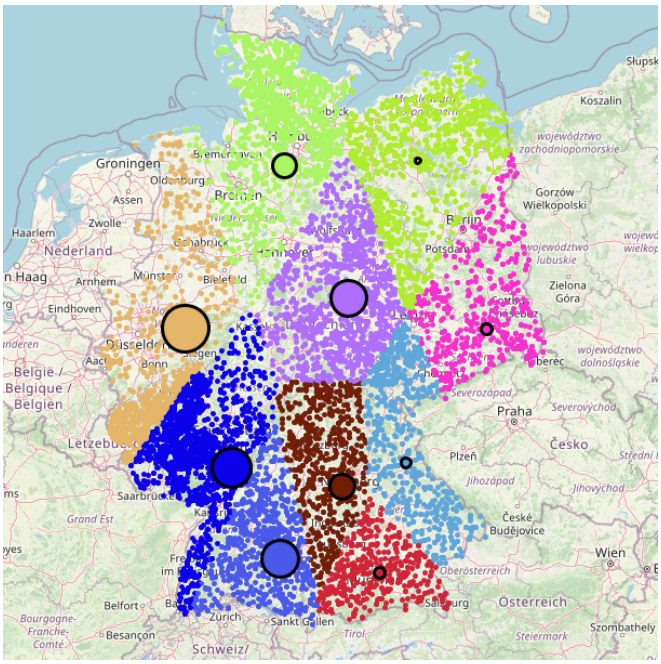


Figure 1: An example of the output results using MPI with 10 clusters of 10 000 sites.

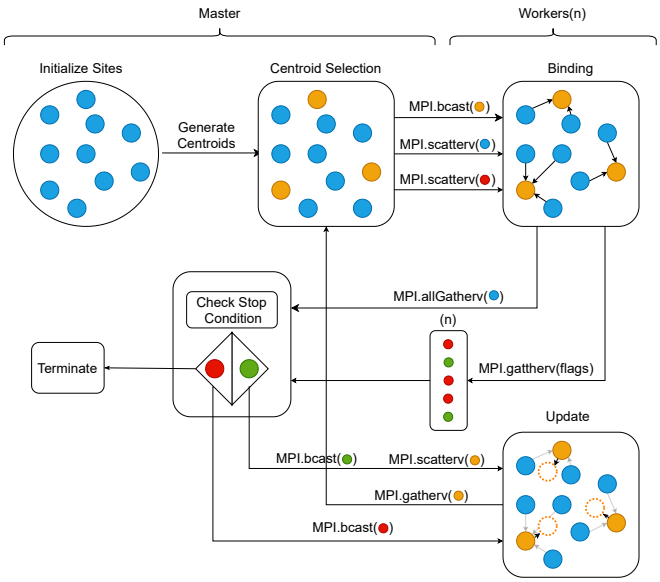


Figure 2: An outline of the distributed algorithm's workflow using MPI.

closer to places with more significant accumulation of wood. Hence, we get the coordinates of the new centroid in the following way:

$$C_j(x) = \frac{\sum\limits_{x_i \in C_j} x_i \cdot w_i}{\sum\limits_{x_j \in C_j} w_j}$$

The new y coordinate of the centroid is calculated analogously.

In the assignment step, we assign to each data point the closest centroid. The Euclidean distance formula for calculating the distance is used here. We assign point p_i to cluster C_j if it holds that:

$$j = \arg\min_l \{\sqrt{(p_i(x) - C_l(x))^2 + (p_i(y) - C_l(y))^2}\}$$

For the initialization, we decided to go with the Frogy method of randomly choosing k points as initial cluster centorids from the given data set, which is proven to be the best one for the simple k-means [8].

2.1 Parallel mode

This mode was implemented using the thread pool principle with the help of `Executors` class [12]. This allows for dynamic control of threads, so there is less possibility for human error. Synchronization was done with the `CountDownLatch` instances called `barrierBind` and `barrierUpdate`, reinitialized before each assignment and update step, respectively.

For the binding step, each thread gets an approximately equally sized chunk of `Site` collection to process. To do this, all threads need to know location of current centroids. They can safely share this since they only read the values in this step. There is no critical sections here regarding sites, since threads are accessing partitions of the `Site` list, i.e. disjoint sets. On the other hand, the parallel program can get into the race condition in the part of the code where a `Site` object is appended to a cluster's list of the given objects. For this reason, we performed the insertion of sites to appropriate lists sequentially after the computations are done. One would argue that the `ArrayList.add()` method could have been made synchronous, but there are situations where even this can fail. Moreover, doing this sequentially has a negligible influence on the performance.

In this mode, the stopping condition is returned as the result value of the function `bindCluster()`. If it is `true`, only then do we enter the block of code that initiates the update step.

For the update step, we call the method `updateCentroid()`, which very simply creates a `Runnable` for each cluster and sends it to the executor.

2.2 Distributed mode

This mode is more "independent", or better put, "self-contained" [12]. We implemented it with the `MPJ express` library [15]. Due to the nature of message passing and `MPJ`'s underlying implementation in `C`, it is not possible to pass complex structures between processes, like `Cluster` and `Site`. We can only natively send the primitive types. For this reason, we decided to "serialize" the `Cluster` and `Site` arrays. We transform them into `double` arrays, `centroidBuffer`, represented with orange circles in Figure 2, and `siteBuffer`, represented with blue circles, respectively. For `Cluster` transformation, we extract the *id*, *latitude* and *longitude* of each instance. Hence, for cluster i, we have its *id* at position $3i$, its *latitude* at position $3i+1$ and its *longitude* at position $3i+2$ in `centroidBuffer`. Similarly for

the sites, we store the *id* at position $5i$, *latitude* at $5i+1$, *longitude* at $5i+2$, *weight* at $5i+3$ and *clusterID* at $5i+4$ in the `siteBuffer` for the site i. Process 0 acts as the master and completes the setup, that is the transformation into `double` arrays. After that, we start looping. Processes loop as long as the stopping condition is not satisfied. To check this, there is a separate buffer for flags, represented with red and green circles in Figure 2. Each process gets a flag with the help of the `Scatter` function to signalize whether it registered a change in its binding step.

Since the first step in the algorithm is assignment, the coordinator first broadcasts `centroidBuffer` and then scatters the `siteBuffer`. We used `Scatterv` for the latter, since the number of sites need not be divisible with the number of processes running.

After this, the change flags are returned to the coordinator. It decides whether the stopping condition has been reached and broadcasts that to the workers, which is represented with the *Check Stop Condition* box on the Figure. Based on that, we either terminate the calculations or proceed to the update step.

For the update step, we decided to implement it in the way that all the processes loop through the entire `siteBuffer`, but only perform calculations on instances whose *clusterID* corresponds to the cluster centroid instance the given process is assigned. For this reason, we used the `Allgatherv` on the `siteBuffer` right after the assignment step, but then we used `Scatterv` on `centroidBuffer` to assign approximately equal number of clusters to each process.

3 RESULTS

To evaluate our implementation, all three versions were implemented and tested for performance. All tests were performed on the same hardware namely, two AMD Epyc CPU's with 64 compute cores each, 512GB of RAM running Linux. To test the performance we conduct two separate tests with 20 workers for the concurrent versions. Firstly, we test the impact the number of sites has on performance by fixing the number of centroids and increasing the number of sites. Secondly, we test the impact centroids have on performance by fixing the number of sites.

In Figure 3 we show the scalability of all three implementations. As expected, we observe a reduction in run-time in both parallel and distributed over the serial implementation. In Figure 3, we can see that the communication overhead in distributed computation pays off only for the heavier half of the test cases. In general, shared memory should perform much faster then buffered IO used by the loopback interface.

Figure 4 shows the impact of centroids on performance. As expected, the sequential version scales linearly, while both parallel and distributed see a marginal hit on performance.

4 CONCLUSIONS AND FUTURE WORK

What we did expect is for the parallel and the distributed to perform much better than the sequential. But what was unexpected is the difference between `MPI`'s performance in Figure 3 and 4. It is almost on par with the performance of the parallel version for the case of fixed number of sites, yet far from it in the case of fixed number of centroids. We attribute this to caching. Array of sites is much bigger than the latter, so having them fixed could be the reason for the performance kick.

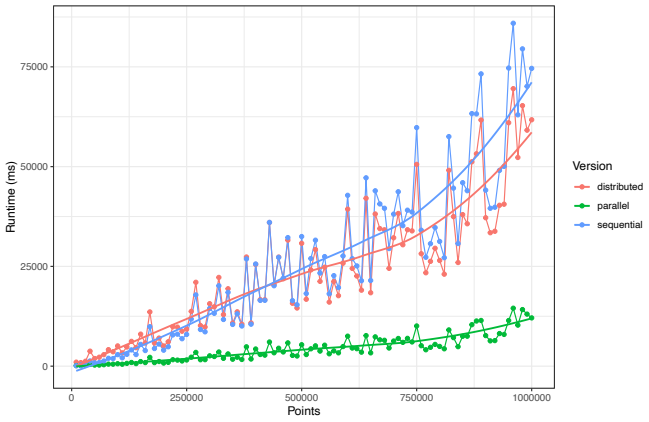


Figure 3: Performance evaluation of all three implementations. The results were obtained by increasing the number of points for each test while keeping the number of centroids at 100

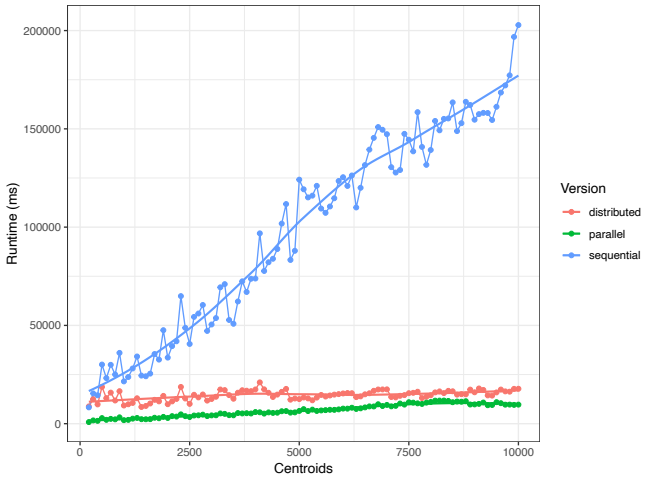


Figure 4: Performance evaluation of all three implementations. The results were obtained by increasing the number of centroids for each test while keeping the number of points at 100.000,00

How different caching strategies and the Cluster Configuration of the distributed part, better reflecting the real-world performance of the aforementioned computing, influence on the comparisons made in this paper; additionally, pushing the boundary of the test data size further, beyond what can fit in a single computer's or server's memory, and how does that influence the performance of the two would be the subject of our future work.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the European Commission for funding the InnoRenew CoE project (H2020 Grant Agreement #739574) and the PHArA-ON project (H2020 Grant Agreement

#857188) and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund) as well as the Slovenian Research Agency (ARRS) for supporting project number J2-2504.

REFERENCES

[1] Michael David Burnard, Črtomir Tavzes, Aleksandar Tošić, Andrej Brodnik, and Andreja Kutnar. 2015. The role of reverse logistics in recycling of wood products. In *Environmental implications of recycling and recycled products*. Singapore : Springer, cop. 2015, 1–30.

[2] Jean-Thomas Camino, Christian Artigues, Laurent Houssin, and Stéphane Mourgues. 2021. MILP formulation improvement with k-means clustering for the beam layout optimization in multibeam satellite systems. *Computers & Industrial Engineering* 158 (2021), 107228.

[3] Péter Egri, Balázs Dávid, Tamás Kis, and Miklós Krész. 2021. Robust facility location in reverse logistics. *Annals of Operations Research* (2021), 1–26.

[4] Bob Falk et al. 1997. Opportunities for the wood waste resource. *Forest Products Journal* 47, 6 (1997), 17–22.

[5] Edward W Forgy. 1965. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics* 21 (1965), 768–769.

[6] Herman P Friedman and Jerrold Rubin. 1967. On some invariant criteria for grouping data. *J. Amer. Statist. Assoc.* 62, 320 (1967), 1159–1178.

[7] Attri Ghosal, Arunima Nandy, Amit Kumar Das, Saptarsi Goswami, and Mrityunjoy Panday. 2020. A short review on different clustering techniques and their applications. *Emerging technology in modelling and graphics* (2020), 69–83.

[8] Greg Hamerly and Charles Elkan. 2002. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*. 600–607.

[9] Ke Liao and Diansheng Guo. 2008. A clustering-based approach to the capacitated facility location problem 1. *Transactions in GIS* 12, 3 (2008), 323–339.

[10] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[11] James McQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967*. 281–297.

[12] Andrej Perkovic. 2022. *K Means Clustering*. https://github.com/AndrejPer/k_means_clustering

[13] C Carl Robusto. 1957. The cosine-haversine formula. *The American Mathematical Monthly* 64, 1 (1957), 38–40.

[14] Erwin M Schau, Črtomir Tavzes, Igor Gavrić, Iztok Šušteršič, Eva Prelovšek Niemelä, Balázs Dávid, Jaka Gašper Pečnik, and David DeVallance. 2022. Environmental and economic assessment of using wood to meet Paris Agreement greenhouse gas emission reductions in Slovenia. In *E3S Web of Conferences*, Vol. 349. EDP Sciences, 03005.

[15] Aamir Shafi, Bryan Carpenter, and Mark Baker. 2009. Nested parallelism for multi-core HPC systems using Java. *J. Parallel Distributed Comput.* 69, 6 (2009), 532–545. https://doi.org/10.1016/j.jpdc.2009.02.006

[16] Junjie Wu. 2012. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media.

Exact Exponential Algorithms for the Center Closing Problem

Samo Metličar
samo.metlicar@hotmail.com
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

Jurij Mihelič
jurij.mihelic@fri.uni-lj.si
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

ABSTRACT

In this paper, we focus on the center closing problem which is similar to the well-known k-center problem. Both problems are defined on a network with the goal of optimizing the worst-case service time for the clients, but with the difference that in the center closing problem several existing centers are closed in order to optimize total cost of operation. First, we show the $\mathcal{NP}$-hardness of the problem. Afterwards, we describe several exact exponential algorithms for solving the problem. Finally, experimentally evaluate these algorithm on two test scenarios.

KEYWORDS

center closing problem, exact algorithm, combinatorial optimization, experimental evaluation, set cover

1 INTRODUCTION

Facility location problems deal with the optimality of the placement of objects into space. Its roots stretch at least back to the 17th century in the form of Fermat's problem [5] and it has since branched out significantly. In this paper, we discuss one such problem – the center closing problem. We define it and its main properties and propose several exact algorithms for solving it. Those are also implemented and tested using different benchmarks.

The problem deals with closing a set number of centers (e.g. post offices) that offer some service to consumers (e.g. towns) and is represented by a network (e.g. road network).

A lot of other prominent facility location problems can also be found in the literature. One such example is the k-suppliers problem, analyzed by Hochbaum and Shmoys [6]. A similar problem where the metric is limited to the L_2 space is examined by the authors of [11]. There are also more complex variations of the problem where objects are more dynamic and can move or change over time [7, 8]. Such problems are often solved with integer linear programs instead of algorithmically.

2 PRELIMINARIES

We begin by introducing basic concepts used in the paper. The center closing problem first requires a network, which we define as a graph with weighted edges. For each pair of nodes u and v the length of the shortest path between them is denoted by $d(u, v)$. For each node v in the network and an arbitrary bound B we also define its bounded open neighbourhood $N_B(v)$ as a set of nodes connected to v by a path of length at most B, excluding v itself. Finally, let S be a set. We say that the family of sets $\mathcal{P}$ forms a partition of set S if the following statements hold:

- $\emptyset \notin \mathcal{P}$,
- $\bigcup_{A \in \mathcal{P}} A = S$ in
- $\forall A, B \in \mathcal{P} : A \neq B \implies A \cap B = \emptyset$.

From this the formal definition of the problem can follow.

Definition 2.1 (Center closing problem). Let $G = (V, E)$ be a complete undirected network, sets of centers S and consumers C form a partition of V, $h : C \to \mathbb{R}_0^+$ be a non-negative weight function and k be a positive integer not greater than $|S|$. For every subset $R \subseteq S$ define the cost as

$$\text{cost}(R) = \max_{c \in C} \min_{s \in S \setminus R} h(c) d(c, s).$$

The problem is to find such a set $R \subseteq S$, where $|R| \geq k$, which minimizes the cost.

The problem can also be presented in the decision version by adding a bound B and asking if there exists a solution of cost at most B.

3 HARDNESS OF THE PROBLEM

Because of the similarity to the k-center [10] problem and other facility location problems we assume that it is also $\mathcal{NP}$-hard. We prove the assumption by reducing the k-center problem to the center closing problem in polynomial time. Let $G' = (V, E)$ be a complete undirected network and k' be a non-negative integer not greater than $|V|$. The k-center problem is to find a set $P \subseteq V$, where $|P| \leq k'$, which minimizes the cost

$$\text{cost}(P) = \max_{v \in V} \min_{p \in P} d(v, p).$$

We reduce the k-center problem to the center closing problem as follows:

- network G contains 2 copies of each node in V and sets the weight of the edge connecting them to 0. The rest of the edges are copied from E,
- $S = C = V$, ie., the set of nodes represents both the set of centers and the set of consumers,
- $h = \mathbb{1}_C$, ie., all consumer weights are 1, and
- $k = |S| - k'$, ie., the number of closed centers as opposed to the number of opened centers.

Lemma 3.1. *Set P^* solves the k-center problem if and only if set $R^* = S \setminus P^*$ solves the center closing problem.*

Proof. Let us denote by cost_1 and cost_2 the cost functions of the k-center problem and the center closing problem. Let P be some (not necesserily optimal) solution to the k-center problem and $R = S \setminus P$.

https://doi.org/10.51939/scores22.07

Then it holds that $|P| \leq k'$ and $|R| \geq k$ and R is thus also a solution to the center closing problem.[1] We now observe the cost functions:

$$\begin{aligned}\text{cost}_2(R) &= \max_{c \in V} \min_{s \in V \setminus R} \mathbb{1}_V(c) d(c,s) \\ &= \max_{v \in V} \min_{p \in P} d(v,p) \\ &= \text{cost}_1(P).\end{aligned}$$

Lets say there exists $R' \neq R^*$ s.t. $\text{cost}_2(R') < \text{cost}_2(R^*)$. Since the costs are equivalent it follows that $\text{cost}_1(V \setminus R') < \text{cost}_1(P^*)$. Therefore P^* does not solve the k-center problem – a contradiction. □

We have shown how to reduce the k-center problem to the center closing problem in polynomial time. We have also shown that the cost functions match in both problems, therefore the decision versions of them can use the same boundary and will return equivalent results. From [9] it then follows that since the decision version of the k-center problem is $\mathcal{NP}$-complete, this must also hold for the decision version of the center closing problem. Further it follows from [12] that since the decision version of the center closing problem is $\mathcal{NP}$-complete, its optimization version must be $\mathcal{NP}$-hard.

4 EXACT EXPONENTIAL ALGORITHMS

4.1 Brute-force Enumeration

The most straightforward approach to solving the problem is generating all subsets R of set S. Let $n = |C|$, $m = |S|$, and $k = |R|$, then there are $\binom{m}{k}$ possible subsets in total. Each has to be evaluated resulting in the time complexity $T_k(n,m) = \binom{m}{k}(m-k)n$. For a fixed k this results in a polynomial time of $O(m^{k+1}n)$, however in the worst case scenario where $k = \frac{m}{2}$ the time complexity is exponential, namely $O(2^m\sqrt{mn})$ (derived using the Stirling formula). This algorithm will later be referred to with the abbreviation `bf`.

4.2 Backtracking

Here, we represent the search space of the problem with a rooted binary tree, where each inner node represents a set of closed centers and outgoing connections represent available actions – the center can be closed (the left branch) or kept open (the right branch). Leaf nodes represent solutions and are, thus, evaluated. The algorithm, denoted with `bt`, traverses the tree using depth-first search introducing two pruning methods:

- if there are not enough centers left in the branch to fulfill the minimum required number of closed centers, the branch is pruned;
- if there are already enough centers closed, the branch below the current node is pruned. The currently visited node becomes a leaf.

The backtracking approach was also implemented in the algorithm, denoted with `btd`, that solves the decision version of the problem allowing for an additional pruning method. In particular, by tracking intermediate costs at each node, we can take advantage of the fact that those costs are non-decreasing and prune the branch if the intermediate cost exceeds the boundary B.

[1] This also holds in reverse where R is a solution to the center closing problem and $P = V \setminus R$.

4.3 Branch and Bound

We upgrade the backtracking algorithm with the *branch and bound* strategy, denoted with `bnb`. During the *branching* part of the algorithm, a priority queue is maintained determining the order in which centers are processed. On each step, the intermediate solution is evaluated enabling the use of the *bound* part of the strategy. It takes advantage of the fact that the cost is non-decreasing with respect to closing additional centers. Therefore, if the intermediate evaluation ever exceeds the current lowest cost found, the whole branch can be pruned.

The algorithm starts without an upper bound, meaning it might not prune much until a decent upper bound is determined. To address this, the algorithm first solves the problem with an approximate algorithm and uses that solution as the starting bound. This alteration of the algorithm will be abbreviated with `bnb+`.

4.4 Reduction to the Set Cover Problem

Here, we develop an algorithm based on the reduction to the decision version of the set cover problem, which is defined as follows. Let U be a universal set, $\mathcal{S}$ be a family of subsets of U, and $\ell \in \mathbb{N}$ a boundary. The question is whether there exists such a family $\mathcal{M} \subseteq \mathcal{S}$ of size $|\mathcal{M}| \leq \ell$, that covers the universal set U, i.e., $\bigcup_{M \in \mathcal{M}} M = U$. We reduce the decision version of the center closing problem to the set cover problem as follows:

- $U = C$, i.e., the set of suppliers represents the universal set,
- $\mathcal{S} = \{N_B(s) \mid s \in S\}$, i.e., the family consists of sets of neighbors of the existing centers, and
- $\ell = m - k$, i.e., the bound for the set cover problem.

Lemma 4.1. *Family $\mathcal{M}^* \subseteq \mathcal{S}$ solves the set cover problem if and only if subset $R^* \subseteq S$, where $\{N_B(s) \mid s \in S \setminus R^*\} = \mathcal{M}^*$, solves the center closing problem.*

Proof. ($\Rightarrow$) Let $S^* = \{s \in S \mid N_B(s) \in \mathcal{M}^*\}$. Clearly, $S^* \subseteq S$ and by the definition of the set cover neighborhoods of selected centers cover the whole set C, meaning the cost is at most B. From $R^* = S \setminus S^*$ it also follows that $|R^*| \geq k$ and thus R^* is a valid solution.

($\Leftarrow$) Let $S^* = S \setminus R^*$. For every $c \in C$ it holds that there exists a center $s \in S^*$ s.t. $h(c)d(c,s) \leq B$, meaning $c \in N_B(s)$. Because $|S^*| \leq m - k = \ell$, the family $\mathcal{M}^* = \{N_B(s) \mid s \in S^*\}$ is a valid solution. □

Now, we introduce the set cover algorithm, see Algorithm 1, that is derived from [13] where its correctness as well as its time complexity of order $O(1{,}5169^m)$ is shown.

The algorithm can easily be expanded to solve the optimization version of the problem. To do this, we take into account that the cost of the center closing problem must always be equal to the cost between some consumer and center. All unique costs are therefore ordered allowing for bisection in the ordered list to be used to find the lowest cost for which the solution of the problem exists.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setting

Algorithms were implemented in `C` programming language and compiled using `gcc 10.2.0` with flags `-O3 -std=gnu17`. Tests were

Algorithm 1 Set Cover

Input: Universal set U, family $\mathcal{S}$ and bound ℓ
Output: YES or NO
procedure SetCover($U, \mathcal{S}, \ell$)
if $\ell = 0$ **then**
if $\mathcal{S} = \emptyset$ and $U = \emptyset$ **then return** YES
else return NO
end if
end if
$S \leftarrow \arg\max \{|S'| \mid S' \in \mathcal{S}\}$.
if $\exists u \in U$ s.t. it is contained in exactly one set $R \in \mathcal{S}$ **then**
return SetCover($U \setminus R,\ \{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}, \ell - 1$)
else if $\exists Q, R \in \mathcal{S}$ s.t. $R \subseteq Q$ **then**
return SetCover($U,\ \mathcal{S} \setminus \{R\}, \ell$)
else if $|S| \leq 2$ **then**
Let C be the set cover computed in polynomial time by using maximum matching
if $|C| \leq \ell$ **then return** YES
else return NO
end if
end if
return SetCover($U \setminus S,\ \{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}, \ell - 1$) **or** SetCover($U, \mathcal{S} \setminus \{S\}, \ell$)
end procedure

run on a computer running `Windows 10` on an `AMD Ryzen 5 2600x` cpu with a frequency of 3,6 GHz. It has 6 cores with a total of 576 KB L1 cache, 3 MB of L2 cache and 16 MB of L3 cache. The system has 16 GB of available DDR4 RAM with a frequency of 2966 MHz. In the next two sections, we present results of the experimental evaluation on two different test scenarios.

5.2 Scale-free Graphs

Here, we consider a test scenario consisting of scale-free networks which are common in the real world. Here, the Barabási–Albert model [1] is used to generate random base graphs. The m nodes with the highest degree are selected as the centers while the rest become consumers. Distances between consumers and centers are set as the length of the shortest path between the two. Additionally, the consumer weight is determined by its degree. Generating random networks allowed us to isolate input parameters and test them independently of each other.

First, we focus on the the number of closed centers k in order to determine its effect on the time complexity for each algorithm. For this, 100 random scale-free networks were generated with $n = 100$ and $m = 20$. The results, shown in Figure 1, show that most algorithms are behaving similarly to what we derived in Section 4.1, meaning they are the slowest around $k = \frac{m}{2}$. The only exception to this observation is the `sc` algorithm which performs the best for high values of k and the worst for low values.

Next, we focus on the total number of centers m. For this benchmark 100 networks with $n = 100$ consumers were generated for each value of m. Algorithms were closing $k = 3$ centers and the results can be seen in Figure 2. Observe that several algorithms, such as `bnb`, are outperforming the set cover algorithm quite drastically.

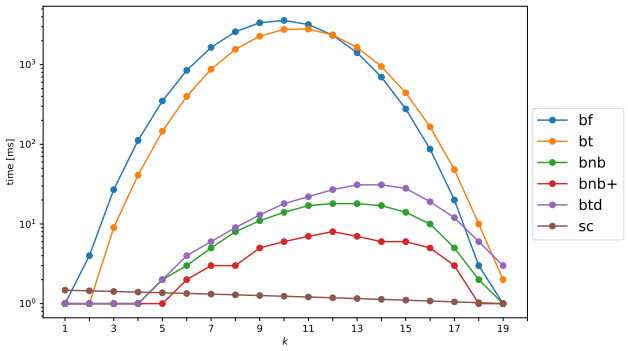


Figure 1: Average execution times for the scale-free networks using exact algorithms in relation to k.

This coincides with the observation from the previous test, where set cover algorithm performs poorly for low values of k while the rest of the algorithms exceed in that range.

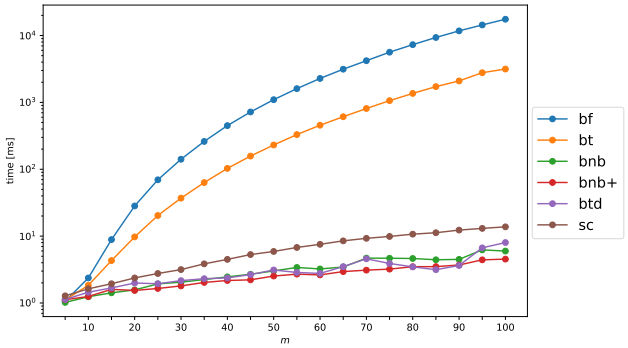


Figure 2: Average execution times for the scale-free networks with $n = 100$ consumers using exact algorithms for parameter $k = 3$ in relation to m.

Finally, the number of consumers n was isolated. 100 networks were generated for each value of n, all with $m = 20$ centers. The number of closed centers was set to $k = 3$. The results, shown in figure 3, indicate that the impact of parameter n on the time complexity is linear for all observed algorithms. This once again coincides with the calculations from 4.1.

5.3 Pmed Benchmark Library

One of the widely used benchmarking libraries to test algorithms for solving facility location problems is `OR-Library` [3]. It contains the set, denoted with `pmed`, of 40 test cases [2] for the p-median problem [4] which also became the standard for testing the algorithms for k-center problem [10], from which the center closing problem originates.

Test cases contain networks with nodes counting from 100 to 900. Graphs were originally generated by adding $|V|^2/50$ edges at random with discrete uniformal distribution of lengths from the interval $[1, 100]$.

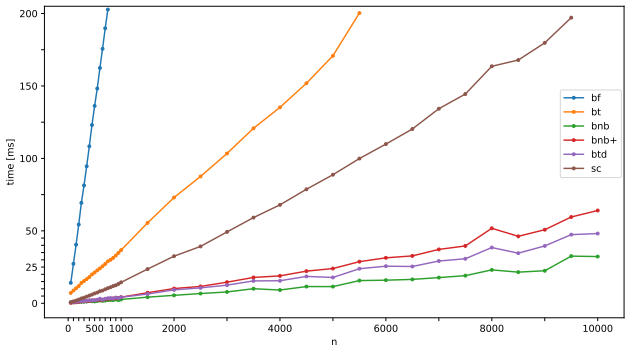


Figure 3: Average execution times for the scale-free networks with $m = 20$ and $k = 3$ using exact algorithms in relation to n.

Algorithms were tested by reducing the k-center problem to the center closing problem. In turn this meant that $C = S = V$ and the high number of centers caused most algorithms to be too slow. This was futher exagarated by the fact that tests required a high amount of those to be closed. This meant that only the set cover algorithm was able to solve some of the tests, which also then serve as an additional test of correctness of the algorithm, since the solutions to all 40 test cases are publicly available.

The algorithm was given 30 minutes for each test case to find the solution. Solve times as well as the solutions can be seen in Table 1. We observe that for large m solutions are rare as expected from Figure 2 as well as from the theoretical time complexity.

#	n, m	k	opt	time	#	n, m	k	opt	time
1	100	95	127	426	21	500	495	40	
2	100	90	98	387	22	500	490	38	
3	100	90	93	2503	23	500	450	22	
4	100	80	74	18	24	500	400	15	948
5	100	67	48	22	25	500	333	11	1351
6	200	195	84	26695	26	600	595	38	
7	200	190	64	43614	27	600	590	32	
8	200	180	55	14321	28	600	540	18	
9	200	160	37	97	29	600	480	13	1377
10	200	133	20	132	30	600	400	9	1857
11	300	295	59	2365	31	700	695	30	
12	300	290	51	563473	32	700	690	29	
13	300	270	36	4930	33	700	630	15	
14	300	240	26	291	34	700	560	11	2923
15	300	200	18	297	35	800	795	30	
16	400	395	47	644	36	800	790	27	
17	400	390	39		37	800	720	15	
18	400	360	28		38	900	895	29	1707288
19	400	320	18	11966	39	900	890	23	
20	400	267	13	748	40	900	810	13	

Table 1: Table showing case parameters, optimal solution and solving time in ms for the set cover algorithm on test cases `pmed` from `OR-Library`.

6 CONCLUSIONS

In this paper we have presented the center closing problem. We proved it is $\mathcal{NP}$-hard and presented several exact algorithms with different approaches to solving the problem. We showed how to reduce a well known k-center problem to center closing problem as well as how to reduce the latter to the set cover problem. Finally, in the paper we have shown the results of experimental evaluation of discussed algorithms tested on different types of networks.

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1. https://doi.org/10.48550/arXiv.cond-mat/0106096

[2] J.E. Beasley. 1985. A note on solving large p-median problems. *European Journal of Operational Research* 21, 2 (1985), 270–273. https://doi.org/10.1016/0377-2217(85)90040-2

[3] J.E. Beasley. 1990. *OR-LIBRARY*. http://people.brunel.ac.uk/~mastjjb/jeb/info.html

[4] Mark S. Daskin and Kayse Lee Maass. 2015. *The p-Median Problem*. Springer International Publishing, Cham, 21–45. https://doi.org/10.1007/978-3-319-13111-5_2

[5] Heinrich Dörrie. 1965. *100 Great Problems of Elementary Mathematics: Their History and Solution*. Dover Publications, Inc., 361–363.

[6] Dorit S. Hochbaum and David B. Shmoys. 1986. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *J. ACM* 33, 3 (may 1986), 533–550. https://doi.org/10.1145/5925.5933

[7] Sanjay Dominik Jena, Jean-François Cordeau, and Bernard Gendron. 2015. Modeling and solving a logging camp location problem. *Annals of Operations Research* 232 (2015), 151–177. https://doi.org/10.1007/s10479-012-1278-z

[8] Sanjay Dominik Jena, Jean-François Cordeau, and Bernard Gendron. 2016. Solving a dynamic facility location problem with partial closing and reopening. *Computers & Operations Research* 67 (2016), 143–154. https://doi.org/10.1016/j.cor.2015.10.011

[9] D. S. Johnson M. R. Garey. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness* (first edition ed.). W. H. Freeman. https://doi.org/10.1137/1024022

[10] Jurij Mihelič and Borut Robic. 2005. Solving the k-center Problem Efficiently with a Dominating Set Algorithm. *CIT* 13 (09 2005), 225–234. https://doi.org/10.2498/cit.2005.03.05

[11] Viswanath Nagarajan, Baruch Schieber, and Hadas Shachnai. 2013. The Euclidean k-Supplier Problem. In *Integer Programming and Combinatorial Optimization*, Michel Goemans and José Correa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 290–301.

[12] B. Robič. 2009. *Aproksimacijski algoritmi*. Fakulteta za računalništvo in informatiko.

[13] Johan M.M. van Rooij and Hans L. Bodlaender. 2011. Exact algorithms for dominating set. *Discrete Applied Mathematics* 159, 17 (2011), 2147–2164. https://doi.org/10.1016/j.dam.2011.07.001

Some observations on the Column-Row game

Ina Bašić
89181003@student.upr.si
UP FAMNIT
Koper, Slovenia

Eric Gottlieb
gottlieb@rhodes.edu
Rhodes College
Memphis, Tennessee, U.S.A.

Matjaž Krnc
matjaz.krnc@upr.si
UP FAMNIT
Koper, Slovenia

ABSTRACT

In this paper we study a new combinatorial game played on Young diagrams, called *Column-Row*. We devise a dynamic-programming algorithm for computing winning positions, or, more generally, Sprague-Grundy values. In turn, we identify winning strategies for several infinite families of starting positions. We prove those results formally, and conclude with a conjecture arising from this work.

KEYWORDS

Sprague-Grundy theory, Young diagrams, Combinatorial games, Dynamic programming

1 INTRODUCTION

Combinatorial game theory is a large and growing field that includes in its scope a wide range of game types, generally focusing on two-player games in which both players have perfect information and there are no moves of chance. The main question in combinatorial game theory is, given some position and optimal play, which player has a winning strategy? Sprague [9] and Grundy [6] introduced a method of quantifying game positions for normal play impartial games. These Sprague-Grundy values are a generalization of winning and losing positions. Furthermore, the Sprague-Grundy values are instrumental in the theory of disjunctive sums of impartial games; see [1, 3, 6, 8–10]. This is why recent research on impartial games is predominantly focused on the Sprague-Grundy values rather than on winning/losing positions. A more detailed introduction to combinatorial game theory, including impartial games, can be found in [2, 3, 8].

In this paper we study winning/losing positions (and more generally, Sprague-Grundy values), of a new game called *Column-Row*. We proceed with some preliminary definitions concerning partitions, and Young diagrams. In Section 2, we describe the game and the methods used to verify the results, which we proceed to present in Sections 3 to 5. Each of these sections presents the Sprague-Grundy values of the *Column-Row* game on a given partition family. We conclude the paper by describing some open problems and discussing their difficulty.

1.1 Preliminaries

We begin by defining some important concepts. A *winning position*, $\mathcal{N}$-position, is a state in which the next player can guarantee a win. A *losing position*, $\mathcal{P}$-position, is a state from which the previous player can guarantee a win. Let S be a set of integers. Then *minimal excluded value* of S, denoted by $\mathrm{mex}(S)$, is the smallest integer which is not in S. Let G be a short impartial combinatorial game under normal play. We say $\widetilde{G}$ is a *subposition* of G if it can be obtained as a result of a single move from G. Let G have k subpositions, $G_1, \ldots, G_k$. Then, we calculate the Sprague-Grundy value x of G recursively as $\mathrm{mex}(\{x_1, \ldots, x_k\})$, where x_i represents the Sprague-Grundy value of G_i, and write $SG(G) = x$. The Sprague-Grundy value of an empty partition is $\mathrm{mex}(\emptyset) = 0$. This definition allows for computing Sprague-Grundy values recursively and represents an essential part of an algorithm presented in Section 2. A Young

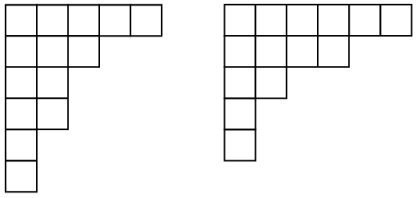


Figure 1: A Young diagram λ and its conjugate λ'.

Diagram is a diagram representing a partition λ of a non-negative integer n, $n = x_1 + \cdots + x_k$ where $x_1 \geq \cdots \geq x_k \geq 0$ is a sequence of non-decreasing integers, as a left-justified array of boxes, such that row i has x_i boxes. We write $\lambda = (x_1, \ldots, x_k)$. We make no distinction between a partition λ and its Young diagram. The only partition of 0 is the empty sum, denoted by $\emptyset$. The *conjugate*, λ' of λ is a partition $\lambda' = (x'_1, \ldots, x'_{x_1})$ where x'_i is the largest integer such that $x_{x'_i} \geq i$. An example of a Young diagram and its corresponding conjugate can be found on Fig. 1. A *subpartition* $\lambda[i, j]$ of λ rooted

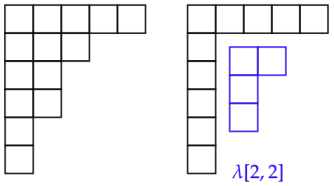


Figure 2: A subpartition $\lambda[2, 2]$ (in blue) of a Young diagram λ.

at the box (i, j) is a partition $(x_i - j,\ x_{i+1} - j,\ \ldots,\ x_1 - j)$, as shown in Fig. 2. We proceed to define families of partitions. All partition parameters are strictly positive. A *rectangle* partition $R_{a,b}$ is a partition of the form (b^a), with a rows and b columns. A *gamma* partition $\Gamma_{a,b}$ is a partition of the form $(b, 1^{a-1})$, with a rows and b columns, such that the first row has b boxes and every subsequent

https://doi.org/10.51939/scores22.08

row consists of only one box. An *almost gamma* partition $\Gamma^{1,d}_{a,b}$, is a gamma partition such that column 0 is repeated $d-1$ times. A *staircase* partition S_n is a partition of the form $(n, n-1, \ldots, 1)$. A *quadrated* partition is a partition of n of the form $(x_1^{m_1}, \ldots, x_k^{m_k})$ where x_t appears m_t times, such that $\sum_{t=1}^{k} x_t \cdot m_t = n$ and all x_t, m_t are even. The examples of those partitions can be found on Fig. 3. For the family of rectangles, the allowable moves correspond to

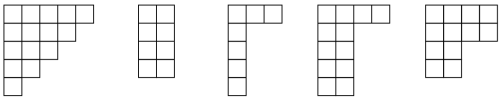


Figure 3: A staircase parition - S_5, rectangle parititon - $R_{4,2}$, gamma parition - $\Gamma_{5,3}$, almost gamma parition - $\Gamma^{1,3}_{5,4}$, and a quadrated partition with $n = 12$.

a simplified variant of the Column-Row game called LCTR (see [4, 5, 7]). In particular, the results by Gottlieb et. al [5] imply the following results for the Column-Row game.

Lemma 1.1. *For positive integers a and b, we have*

$$SG(R_{a,b}) = \begin{cases} 0 & \text{if } b > 1 \text{ and } a > 1 \text{ and } a+b \text{ is even} \\ 2 & \text{if } b \leq 2 \text{ or } a \leq 2 \text{ and } a+b \text{ is odd} \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 1.2 (LCTR [5]). *Let λ be a partition. Then we have*

(1) $\lambda'[i,j] = \lambda[j,i]'$ *and*
(2) $SG(\lambda') = SG(\lambda)$.

2 THE COLUMN-ROW GAME

Column-Row game (CR for short) is an impartial, combinatorial game played on a Young Diagram. A move in CR consists of removing any single row or any single column from the diagram. We do not make the distinction between the CR game and the corresponding Young diagram, thus the notations $\mathrm{CR}(\lambda)$ and λ are equivalent. We denote the row and column moves as i_r and j_c, where i and j are the indices of the row and column which we remove from the diagram, respectively, and write $\lambda \xrightarrow{i_r} \widetilde{\lambda}$ for a row move and $\lambda \xrightarrow{j_c} \widetilde{\lambda}$ for a column move. Refer to Fig. 4 for a visual representation of the moves of CR.

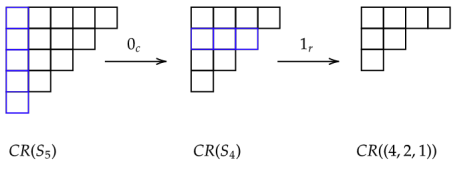


Figure 4: An example of both types of moves.

2.1 Verifying Sprague-Grundy values by aid of computer

As a consequence of the definition of Sprague-Grundy value of a game, we can use a computer to verify the results. By utilizing dynamic programming approach to store the Sprague-Grundy values of subpositions, this can be done efficiently. We outline our approach in Algorithm 1.

```
1 Function SG (λ, D)
      Input: Partition λ, dictionary D.
      Output: SG value of the game CR played on λ.
2     if λ ∉ D then
3         M ← ∅;
4         for i-th column of λ do
5             let λ --i_c--> λ̃ and add SG(λ̃, D) to M;
6         for j-th row of λ do
7             let λ --i_r--> λ̃ and add SG(λ̃, D) to M;
8         D[λ] ← mex(M);
9     return D[λ];
```

Algorithm 1: An outline of the code which computes the Sprague-Grundy value $\mathrm{SG}(\lambda)$, for any input partition λ. In the code, $\mathcal{D}$ is assumed to be a dictionary of partitions as keys, stored together with the corresponding Sprague-Grundy values. $\mathcal{D}$ is initially empty.

Algorithm 1 is particularly useful for identifying (most of the times infinite) patterns of winning positions. As we will see in the subsequent sections, those patterns are then proved formally, or posed as an open conjecture. Without the aid of such an algorithm, computing Sprague-Grundy values would involve exhaustive case-analysis even for a single starting position.

3 SPRAGUE-GRUNDY VALUES OF COLUMN-ROW GAME ON A QUADRATED PARTITION

In this section, we present a solution to the *Column-Row game* on a family of quadrated partitions in terms of traditional $\mathcal{N}/\mathcal{P}$-positions.

Lemma 3.1. *Let $Q = (x_1^{m_1}, \ldots, x_k^{m_k})$ be quadrated, and let G be a subposition of Q. Then*

(1) *G is not quadrated.*
(2) *There exists a quadrated subposition of G.*
(3) *G is an $\mathcal{N}$-position, Q is a $\mathcal{P}$-position.*

Proof. As a consequence of Lemma 1.2, it is enough to consider only the row moves of Q. Suppose G is obtained by $Q \xrightarrow{i_r} G$ for some i. We show each item in turn. Item 1 and Item 2 are an immediate consequence of the properties of quadrated partitions, while Item 3 is shown by induction.

(1) Removing a row reduces the number of occurrences m_t of some x_t in G. Since Q is quadrated, $m_t - 1$ is odd G cannot be quadrated.

(2) Suppose i is odd and consider the move $G \xrightarrow{(i-1)_r} \widetilde{Q}$. It is easy to see that $\widetilde{Q}$ is quadrated. Similarly, suppose i is even. Then the move $G \xrightarrow{(i+1)_r} \widetilde{Q}$ gives us the desired quadrated subposition of G.
(3) We proceed by induction on n. For the base case, we consider $n = 0$. Notice that the only such partition is $\emptyset$, which is quadrated and trivially a $\mathcal{P}$-position. Let Q be a quadrated partition of n, let G be its subposition, and assume the claim holds for any quadrated partition of k for $k < n$. By Item 1, G is not quadrated, and, by Item 2, it has a quadrated subposition $\widetilde{Q}$, a partition of $k < n$. By the induction hypothesis, $\widetilde{Q}$ is a $\mathcal{P}$-position, and therefore G is an $\mathcal{N}$-position. Consequently, since every subposition of Q is an $\mathcal{N}$-position, Q is a $\mathcal{P}$-position.

□

4 SPRAGUE-GRUNDY VALUES OF COLUMN-ROW GAME ON A GAMMA PARTITION

Recall that a gamma partition $\Gamma_{a,b}$ is a partition of the form $(b, 1^{a-1})$, with a rows and b columns, such that the first row has b boxes and every subsequent row consists of only one box. In this section we resolve the Sprague-Grundy values of Column-Row game on any gamma partition.

Theorem 4.1. *The Sprague-Grundy value of a gamma partition $\Gamma_{a,b}$ is given by:*

$$SG(\Gamma_{a,b}) = \begin{cases} (a+b) \bmod 2 + 1 & \text{if } \min(a,b) = 1 \\ 0 & \text{if } a \text{ and } b \text{ have the same parity} \\ 3 & \text{otherwise.} \end{cases}$$

Proof. We proceed by induction on $a + b$. Given $\Gamma_{a,b}$, we have four possible types of moves and thus, four possibly distinct subpositions $G_1, \ldots, G_4$:

G_1: $\Gamma_{a,b} \xrightarrow{0_r} \Gamma_{a-1,1}$,

G_2: $\Gamma_{a,b} \xrightarrow{0_c} \Gamma_{1,b-1}$,

G_3: $\Gamma_{a,b} \xrightarrow{j_c,\ j>0} \Gamma_{a,b-1}$,

G_4: $\Gamma_{a,b} \xrightarrow{i_r,\ i>0} \Gamma_{a-1,b}$.

Notice that G_3 and G_4 are uniquely defined. For the base case we consider $\Gamma_{a,b}$ such that $a + b \leq 4$. We have two possibilities for a and b, namely, either $\min(a, b) = 1$ or $a = b = 2$. In the former case, the Sprague-Grundy values for $\Gamma_{1,3}$ and $\Gamma_{3,1}$ are given directly by Lemma 1, since $\min(a, b) = 1$, thus we have

$$SG(\Gamma_{1,3}) = SG(\Gamma_{3,1}) = 1 = (3+1) \bmod 2 + 1.$$

In the latter case, removing the first row or the first column results in $\Gamma_{1,1}$, while the other two moves result in subpositions $\Gamma_{2,1}$ and $\Gamma_{1,2}$ respectively. In both cases, $\min(a, b) = 1$, hence

$$SG(\Gamma_{1,1}) = 1,$$

while

$$SG(\Gamma_{2,1}) = SG(\Gamma_{1,2}) = 2.$$

Then, the Sprague-Grundy value of $\Gamma_{2,2}$ is given by

$$SG(\Gamma_{2,2}) = \operatorname{mex}\{1, 2\} = 0,$$

thus the claim holds. Assume now, that the claim holds for $\Gamma_{a,b}$ such that $a + b < k$, with $k > 4$ and consider $\Gamma_{a,b}$ such that $a + b = k$. We consider the cases where a and b are of the same and distinct parity separately. Firstly, let a and b have the same parity, and consider the subpositions

$$G_1 = \Gamma_{a-1,1} \qquad G_2 = \Gamma_{1,b-1} \qquad G_3 = \Gamma_{a,b-1} \qquad G_4 = \Gamma_{a-1,b}.$$

We notice that G_1 and G_2 have the same Sprague-Grundy value. In particular, because a and b have the same parity and $\min(a, b) = 1$,

$$\begin{aligned} SG(\Gamma_{a-1,1}) &= SG(\Gamma_{1,b-1}) \\ &= ((a-1)+1) \bmod 2 + 1 \\ &= (1 + (b-1)) \bmod 2 + 1, \end{aligned}$$

that is, the value is 2 if a and b are even and 1 otherwise. Similarly, since both G_3 and G_4 have parameters of distinct parity whose sum is strictly smaller than k, by the induction hypothesis we have

$$SG(\Gamma_{a,b-1}) = SG(\Gamma_{a-1,b}) = 3.$$

Then, the Sprague-Grundy value of $\Gamma_{a,b}$ where a and b are of the same parity is either $\operatorname{mex}\{2, 3\} = 0$, if both are even, or $\operatorname{mex}\{1, 3\} = 0$ if they are odd. Finally, let a and b be of distinct parity. Consequently, as the parities of $a - 1$ and $b - 1$ are distinct as well, the Sprague-Grundy values of G_1 and G_2 cannot be equal, and as they are given by $((a-1)+1) \bmod 2 + 1$ and $(1 + (b-1)) \bmod 2 + 1$, respectively, we are able to reach both a position with Sprague-Grundy value of 2, and a position with Sprague-Grundy value of 1, depending on whether a or b is odd. As for G_3 and G_4, since both a and $b - 1$ as well as $a - 1$ and b have the same parity, the induction hypothesis gives us

$$SG(\Gamma_{a-1,b}) = SG(\Gamma_{a,b-1}) = 0.$$

Thus, the Sprague-Grundy value of $\Gamma_{a,b}$ where a and b are of distinct parity is therefore

$$SG(\Gamma_{a,b}) = \operatorname{mex}\{0, 1, 2\} = 3,$$

as desired. □

5 ALMOST-GAMMA

Recall an *almost gamma* partition is a partition of form (b, d^{a-1}). An example of such a partition $\Gamma^{1,3}_{5,4}$ is shown on Fig. 3. The Sprague-Grundy values of any almost gamma partition is given by the following theorem:

Theorem 5.1. *Let $\Gamma^{1,d}_{a,b}$ be an almost-gamma partition, such that $a \geq 4,\ b \geq 3,\ d \geq 3$ and $d \leq b$. Then*

$$SG(\Gamma^{1,d}_{a,b}) = \begin{cases} a + b \pmod 2 & \text{if } b \text{ and } d \text{ have the same party;} \\ 2 & \text{if } a \text{ and } b \text{ have the same parity} \\ & \text{distinct from } d; \\ 3 & \text{otherwise.} \end{cases}$$

6 CONCLUSION

While analyzing and verifying Sprague-Grundy values for certain partition families is significantly easier with the aid of dynamic programming, doing so in general is far from trivial. In particular, depending on the structure of the partition and the resulting subpositions, the problem could pose a significant computational challenge. Such is the case with the staircase partition defined in Section 1. The following conjecture is given by running Algorithm 1 on S_n for $n \in 1, \cdots 16$

Conjecture 1. *Let S_n be a straircase partition. Then*

$$SG(S_n) = \begin{cases} 0 & \text{if } n \text{ is even} \\ 1 & \text{if } n = 5 \\ 2 & \text{otherwise.} \end{cases}$$

In general, solving a game, like LCTR, see [5, 7] on a staircase partition is, computationally, a relatively easy task. However, when it comes to the *Column-Row*, each staircase partition S_n has exactly n subpositions. Only one of these subpositions is a staircase partition. This significantly limits the usefulness of the algorithm as the remaining $n - 1$ positions will have to be calculated with every increase of n.

ACKNOWLEDGMENTS

This work is supported in part by the Slovenian Research Agency, bilateral project BI-US/22-24-164.

REFERENCES

[1] E.R. Berlekamp, J.H. Conway, and R.K. Guy. 2001. *Winning Ways for Your Mathematical Plays.* Number v. 1. Taylor & Francis. https://books.google.rs/books?id=_1pl8so-qsIC
[2] E.R. Berlekamp, J.H. Conway, and R.K. Guy. 2004. *Winning Ways for Your Mathematical Plays.* Number v. 4. Taylor & Francis.
[3] J.H. Conway. 2000. *On Numbers and Games.* Taylor & Francis.
[4] Eric Gottlieb, Jelena Ilić, and Matjaž Krnc. 2022. Some results on LCTR, an impartial game on partitions. (2022). https://doi.org/10.48550/ARXIV.2207.04990
[5] Eric Gottlieb, Matjaž Krnc, and Peter Muršič. 2022. Sprague-Grundy values and complexity for LCTR. https://doi.org/10.48550/ARXIV.2207.05599
[6] P.M. Grundy. 1939. Mathematics of games. *Eureka* 2 (1939), 6–8.
[7] Jelena Ilić. 2022. Computing Sprague-Grundy values for arbitrary partitions. https://doi.org/10.5281/zenodo.6782383
[8] Aaron N Siegel. 2013. *Combinatorial game theory.* Vol. 146. American Mathematical Soc.
[9] R. Sprague. 1935. Über mathematische Kampfspiele. *Tohoku Mathematical Journal, First Series* 41 (1935), 438–444.
[10] R. Sprague. 1937. Über zwei Abarten von Nim. *Tohoku Mathematical Journal, First Series* 43 (1937), 351–354.

Semantic Analysis of Russo-Ukrainian War Tweet Networks

Benjamin Džubur
Faculty of Computer and Information Science, University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia
bd5830@student.uni-lj.si

Žiga Trojer
Faculty of Computer and Information Science, University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia
zt0006@student.uni-lj.si

Urša Zrimšek
Faculty of Computer and Information Science, University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia
uz2273@student.uni-lj.si

ABSTRACT

Millions of people around the world continue to express their view on various topics on Twitter everyday. Such data is frequently used to generate and analyze networks of users, tweets and hashtags based on specific actions, such as tweets, retweets, mentions etc. In our study we focus on tweets related to the Russo-Ukrainian conflict. We combine sentiment and network analysis approaches to produce various important insights into the discussion of the conflict. We focused on the most influential actors in the debate as well as uncovering communities of users or hashtags which correspond to either side of the conflict. We discovered that the vast majority of users express support for Ukraine, and that the most important accounts belong to political leaders (e.g. Volodymyr Zelenskyy), relevant organizations (NATO) or media outlets, who actively report on the conflict (Kremlin News). Similarly, most of the relevant hashtags are used predominantly in a pro-Ukraine context, while many of them appear in tweets supporting Russia as well (e.g. #war, #Russia). We have identified numerous communities within the networks, which belong to discussions about the conflict being held in various languages or about various aspects, that the war indirectly affects (e.g. finance & cryptocurencies). Apart from a few very evidently pro-Russia communities, all the groups express support for Ukraine to at least some degree. Future research should focus on more thoughtful data collection and consequently thorough analysis of various aspects of the networks.

KEYWORDS

Network analysis, war in Ukraine, sentiment analysis.

INTRODUCTION

Twitter has become one of the most important platforms for public response to current affairs. People join in on debates by tweeting, retweeting, hash-tagging, replying to or mentioning other users. In such ways, people tend to form like-minded groups or communities.

Since the beginning of Russia's invasion on Ukraine at the end of February 2022, the conflict has become the prevailing talking point of many mainstream as well as social media outlets. On Twitter, millions of tweets are still made in regards to the conflict every day. This provides an excellent opportunity for in-depth analysis of various social aspects of the conflict. By constructing networks from Twitter data, e.g. as in Figure 1, based on users, tweets, hashtags etc., this study provides answers to important questions about social aspects of the conflict, namely: who are the most important users and what is their role in the debate? Are there specific users or hashtags that are pro-Russia or pro-Ukraine and do they form clear communities, or do communities correspond to some other semantic property? Are there notable differences in tweets being posted before and after the ban of Twitter in Russia?

With the help of modern sentiment and network analysis approaches, this study provides an important high-level view into the conflict. The questions listed above are answered in the following sections. The results are not only interesting to the general public and social scientists for future research, but also to PR professionals, media outlets, and perhaps even world leaders.

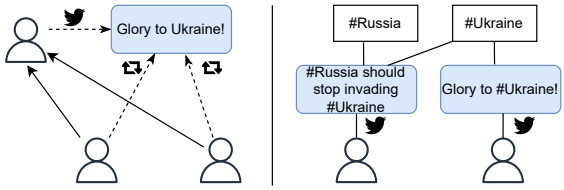


Figure 1: Diagrams of two models of graphs, constructed for our analysis. Left: projected directed user graph based on retweet connections. Right: tri-partite graph between users, tweets and hashtags.

RELATED WORK

Today, social networks play a critical role in online social discourse, particularly during major events such as elections, public affairs, COVID-19 and wars. The authors of [9] and [8] analyzed the elections in the United States and Italy using Twitter data. Study [9] looked at the evolution of the retweet graph over time, focusing on the two main entities. They identified fluctuations in sentiment and measured the volume around each entity. In [8] they used network analysis techniques such as triadic closures, degree-degree correlations, k-core decomposition, and core periphery structure detection to analyze complex semantic structures, prominent topic connections, similar topics within different communities, and how topics animate the discussion over time. They noted that connecting users with tweets using retweets models user preference better than using mentions or replies. In [13] authors analyzed the *who is following who* network to highlight users whose position is particular – important entities. Specifically, they showed that linguistic groups are key factors to explain certain clusters. The authors of [6] used network analysis techniques to visualize different network models on COVID data and used centrality to find the network's most influential hashtags. The authors of [5] used sentiment classification to enhance community

https://doi.org/10.51939/scores22.09

detection and vice-versa. Different supervised techniques for sentiment analysis on Twitter, such as Naive Bayes, max entropy, and SVM, are proposed in [14], but those methods are usually used for positive and negative sentiment. With the rise of deep language models, such as *RoBERTa* [10, 11], more complex data domains can be analyzed, e.g., news texts, where authors typically express their opinion and different topic analysis can be done (e.g. republican vs democratic). Authors of [7] used similar models to investigate target-dependent sentiment classification in news articles. A recent study by [16] looked into tweets about the crisis between Ukraine and Russia. They found that most popular hashtags are `#Ukraine`, `#Russia`, `#StandWithUkraine`, `#Putin`, `#UkraineRussiaWar`, and `#StopPutin`. Additionally, they conducted sentiment analysis and tracked daily positive and negative sentiment between Zelenskyy and Putin and between Ukraine and Russia. They verified that the majority of Twitter users support the Ukrainian side and that Putin is the subject of the majority of negative tweets.

RESULTS

General analysis. Using RoBERTa, we first assign a probability (referred to as **pro-Russia score** or **PRS** in the following) of supporting Russia instead of Ukraine to each tweet. The distribution on both datasets is shown in Figure 2. As expected, we observe significant support for Ukraine in the analyzed tweets. On the 65D sample, data collection is less biased (see Data). However, these models can struggle with sarcasm and produce indecisive scores when content is complex or cannot be directly connected to the predicted classes, which explains the peak at 0.5 and heavier right tail on aforementioned dataset. We also found no noticeable difference in distributions of pre- and post-ban PRS. We constructed 4 main networks for our analysis: a projected hashtag network, two projected user networks based on different types of connections (mentions and retweets) and a tri-partite user-tweet-hashtag network. Basic statistics of the networks are seen in Table 1.

Hashtag network connects hashtags that co-appear, with edge weight representing the number of co-appearances. Each node has two attributes: the number of tweets it appeared in and it's PRS, calculated as the mean of these tweet PRSs. The network is scale free and small-world. Mean PRS in LCC equals to 0.29, while mean PRS of nodes outside of it is 0.42.

Table 1: Network statistics.

network type	data	n	m	$\langle k \rangle$	LCC
hashtag	65D	17k	79k	9.16	84%
user (retweet)	25M	167k	189k	2.26	62%
user (mention)	25M	79k	115k	2.91	76%
tripartite	25M	571k	924k	3.24	83%

n - number of nodes, m - number of edges, $\langle k \rangle$ - average node degree, LCC - largest connected component

We ran PageRank on the network, separated hashtags into pro-Ukraine (PRS below 0.4) and pro-Russia (PRS above 0.6) and selected the most important ones in each group. It turns out that most important pro-Russia hashtags are `#istandwithputin`, `#csto` (post-Soviet intergovernmental military alliance) and `#notmypresident` (references Biden), while the most important pro-Ukraine hashtags are `#ukraine`, `#standwithukraine` and `#nato`.

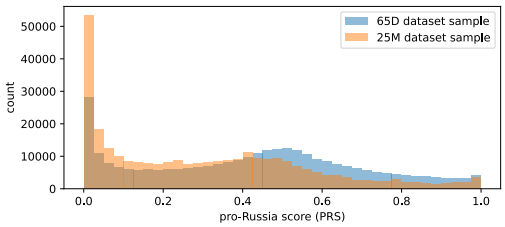


Figure 2: Distribution of tweet pro-Russia score (PRS) on 300k data samples.

To see how hashtags form groups on Twitter, we used Infomap on the LCC, filtered out small clusters (less than 15 hashtags) and calculated their average PRS. The largest cluster includes common hashtags like `#ukraine`, `#russia`, `#nato`, `#putin` and has average pro-Russia probability of 0.38. The most pro-Russia cluster, with mean PRS of 0.58, is clearly representing American republicans. The second one, with similar PRS is connected to Kazakhstan protests. Five out of ten top pro-Russia clusters tweet about different types of investments: stock market, cryptocurrencies, commodities, forex, etc. with average PRS around 0.5. The most pro-Ukraine clusters are heterogeneous, but some talk about transport, helping refugees, or represent different languages (German, Ukrainian) or groups (tech, fashion, art) showing their support.

In Figure 3 we can see the main core of the network, obtained with k-core decomposition, where $k = 51$. All 89 hashtags in it have less than 0.5 PRS, with a mean PRS of 0.29. We conducted a permutation test as we hypothesized that the main core is significantly more pro-Ukraine than a random subgraph of the same size. We estimated the average PRS of 10,000 randomly selected subgraphs and compared it to the PRS of the main core. It was lower in 25%, indicating that it is not completely random, but with a p-value of 0.25 we are unable to confidently claim that it is significantly pro-Ukraine.

User retweet network is a directed, scale free network, which we tried to visualize and find different clusters on (similar as in [13]). Figure 4 shows a sub-network of the retweets network; we filtered out nodes with $k < 5$ before taking only nodes in the largest connected component. There are four well-defined clusters, each of which was annotated based on it's distinguishable topic. The cluster on the left is zoomed in at the bottom. This is the only cluster that is dominantly pro-Russia and it represents a group of Italian-speaking users. After digging deeper into this data, we discovered that it is annotated as pro-Russia, because their tweets are interpreted as slightly pro-Russia by the model, e.g. (translated): *"Of which side should we take sides now? From Italy and the Italians who will pay more..."*, but they do not strongly side with Russia, with mean PRS 0.56. We observe a high density of edges between the "Eastern extremist" and "Journalists and the Western World" clusters, which is a result of the fact, that Eastern sites occasionally post explicit content, which the other cluster frequently retweets. The Tigray cluster is particularly interesting because the main topic of discussion is not the Ukraine war, but rather the Tigray war. Users in this cluster frequently express their belief that the Western world only reports on and aids in the fight against wars in Europe, but is unconcerned about the war in Ethiopia.

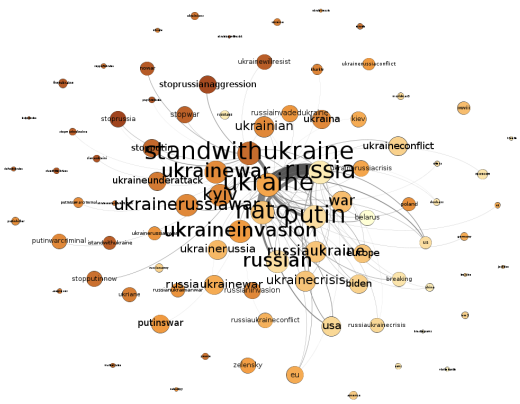


Figure 3: Main core (k=51) of hashtag network. Edge thickness is proportional to hashtag co-appearance and node size to number of appearances. Some thin edges are hidden. Lighter color of nodes represents higher PRS. From `#stoprussia` (0.06) to `#belarus` (0.47). We can observe a strong connection between `#ukraine` and `#russia` that appear together in many tweets and a well defined color gradient over the network.

On the same network, we used the PageRank algorithm to determine the most influential pro-Ukraine users: `lesiavasylenko` (member of the Ukrainian parliament), `HannaLiubakova` (Belorussian journalist), `nexta_tv` (the largest Eastern European media) and pro-Russia users: `MauriceSchleepe` (Russian news), `ejmalrai` (veteran war journalist) and `Charles_Lister` (Syrian reporter).

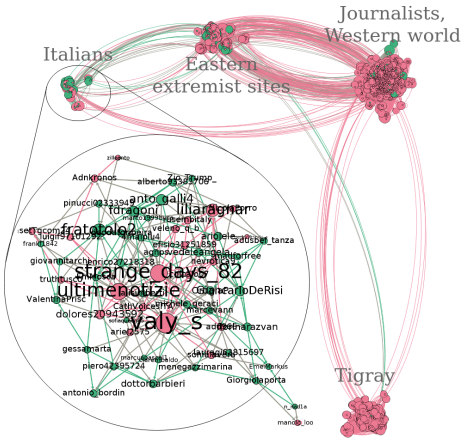


Figure 4: Sub-network of retweets network, consisting of nodes with $k \geq 5$ and nodes within LCC ($n = 8575$, $m = 47851$). Red color represents pro-Ukraine users and green color represents pro-Russia users. Size represents in-degree of the node.

Mentions network is a scale free network, on which we tried to find most influential users. PageRank returns the following pro-Ukraine users: `zelenskyyUa`, `POTUS`, `nato` and pro-Russia users: `createimagic` (suspended account), `kevjmclaughlin` (suspended account) and `needlesineyes` (misclassified by the model as a pro-Russian).

Additional analysis. On above user networks, we were unable to find similar core-periphery structure results as with hashtags, using k-core decomposition. However, we have found significant assortativity based on PRS, with a coefficient of 0.37 on the user (retweet) network.

On the tri-partite network, the 5 largest communities found by Infomap cover 35% of the network and contain the main discourse about the conflict in English. However, the communities which follow by size have an immediately distinguishable property, such as language – Spanish (16k nodes), Thai (11k), French (9k), German (8k), Italian (7k), Indian (6k) etc., or other topics such as hackers (6k), NFTs and Cryptocurrency (6k), etc. These communities are notably pro-Ukraine with average PRS 0.3. Only the 26th largest community with 3k nodes has been found to be noticeably pro-Russia with PRS 0.68 and popular hashtags such as `#istandwithputin`.

DISCUSSION

To summarize, we analyzed two Twitter datasets and found similar results. The majority of tweets are in favor of Ukraine, with many being falsely labeled as pro-Russia as a result of the nature of our process of semantic inference. Here we highlight the community of Italian users, which are mostly labeled as slightly pro-Russia, although this is not apparent from the tweets. However, we have identified a few communities of pro-Russian accounts, many of which have been suspended.

Nevertheless, we observed no differences in the ratio of pro-Russia to pro-Ukraine tweets between pre- and post-ban of Twitter in Russia, as we might have expected. This is likely to be due to the specifics of data collection in our case. When it comes to the most important actors in either side of the debate, we identified important politicians such as Zelenskyy and the US President as well as NATO to fill that role for the pro-Ukrainian side, as we might have expected. Important supporters of the Russian side include local media outlets and influential independent journalists. As expected, we found that most users tend to retweet posts from users with a similar stance on the conflict.

When it comes to hashtags, we found that all the most popular ones are primarily used in a pro-Ukraine setting, even e.g. `#Russia`, which frequently co-appears with `#Ukraine`. Clusters in the hashtag network show that American republicans tend to be pro-Russia. They use `#notmypresident` in reference to Joe Biden, making them side with Trump and consequently (on average) with Russia. While the core hashtags are noticeably pro-Ukraine, others from the periphery are more neutral as they do not directly address the conflict.

Unfortunately, aside from identifying various linguistic communities (most of which were found to be pro-Ukraine), location data was mostly lacking and thus inappropriate for analysis. This could be further looked into in future research alongside a more in depth temporal semantic analysis. A shortcoming of this analysis was the data collection process, which was here bypassed and should be addressed appropriately (considering e.g. balance, completeness).

We suggest using information from various platforms, for example *VKontakte* and *Weibo* (considered as Russian and Chinese Twitter respectively), for a potential future work. In a sense, the data would be less biased in this way, since it would broaden the scope of the research onto those who don't use Twitter.

METHODS

Data. We found two datasets connected to the conflict. The first dataset [2], was gathered from Jan 1, 2022 to Mar 5, 2022, using certain keywords for the search, such as `ukraine war`, `russian troops`, `ukraine troops`, etc. The second one [3], has over 25 million tweets and has been updated on a daily basis since February 27, 2022. It contains tweets whose geolocation was found to be in Ukraine or they contained hashtags such as: `#SlavaUkraini`, `#Russia`, `#RussiaUkraineWar`, `#Putin`, `#ukraineunderattack`, `#StopPutinNow`.

For our analysis, we constructed random subsamples of the two datasets of size 300k, in order to decrease computational complexity while preserving structural properties we were interested in analyzing. We label these two samples as 65D and 25M respectively.

Because information about retweets is only contained in the 25M dataset from Apr 23, 2022 onward, we prepared an additional 300k sub-sample which contains only such data to construct our retweet-based user network.

We argue that the results reported on these subsamples are representative of the full data and that they retain the structural properties of interest. By sampling multiple graphs of the same size and calculating the standard deviation for the average node degree ($\sigma_{\langle k \rangle} = 0.003$) and the largest connected component ($\sigma_{LCC} = 1.5\%$), we validated that different randomly sampled subgraphs retain these properties.

The structure of the graphs was also examined by computing the Jaccard index, which is a statistic used for gauging the similarity and diversity of sample sets. When looking at nodes with degree > 50, we discovered that 80% of nodes overlap between different samples. We also assessed the degree difference for each node that appears in both graphs at the same time and discovered that the difference is modest ($diff = 12.5 \pm 0.2$), implying that local structure is preserved across all samples. Because the standard deviation of the aforementioned statistics across different samples is failry small, we realized that the global and local structure of the graphs matched across samples.

Methods & algorithms. In order to perform semantic analysis, we use a 0-shot pre-trained multilingual model [1], based on BERT. This transformer-based state-of-the-art language model has been successfully used in different fields of Natural Language Processing and especially for the task of Twitter data classification [17]. The model used for our analysis takes as input query text (e.g. a tweet) and a possible set of labels. In our analysis, we used the labels "support Russia" or "support Ukraine". The model semantically interprets the inputs and computes a probability distribution over the set of labels. The probabilities represent the likelihood of the query text being associated with the respective labels. This is a simple but effective technique of labeling our data, which allows us to also assign probabilities to users or hashtags, based on tweets they are directly connected to by averaging. By using a different set of labels for our tweets, namely: "pro Russia" and "pro Ukraine", we observed a high Pearson correlation coefficient of 0.83, showing the model's semantic understanding capabilities.

For the rest of our analysis, we resorted to Infomap for community detection and k-core decomposition for analysis of the networks' core and periphery, PageRank centrality for measuring importance of nodes, Pearson correlation for evaluating assortativity based on PRS and permutation test for evaluating PRS significance of subgraphs. For our flexible visualizations, we relied on Gephi software. We selected Infomap for community detection, because it proved to be a reliable method during our previous work. In [12] they show its consistent performance over synthetic and real datasets. K-core and PageRank were selected because their algorithms (described in [4] and [15]) simulate the behaviour of Twitter users. PageRank simulates how a user would surf over Twitter, by clicking on users mentioned in tweets of the user whose feed he is scrolling (similarly with hashtags). K-core shows the most important users/hashtags by iteratively deleting those that are connected to least others, and thus have lower probability that someone would click on them.

REFERENCES

[1] 2020. xlm-roberta-large-xnli. https://huggingface.co/joeddav/xlm-roberta-large-xnli. Accessed: 2022-05-15.
[2] 2022. Russia-Ukraine war - Tweets Dataset. https://www.kaggle.com/datasets/foklacu/ukraine-war-tweets-dataset-65-days. Accessed: 2022-05-15.
[3] 2022. Ukraine Conflict Twitter Dataset. https://www.kaggle.com/datasets/bwandowando/ukraine-russian-crisis-twitter-dataset-1-2-m-rows. Accessed: 2022-05-15.
[4] José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2005. k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint cs/0504107* (2005).
[5] Deitrick et al. 2013. Mutually Enhancing Community Detection and Sentiment Analysis on Twitter Networks. *Journal of Data Analysis and Information Processing* 01 (01 2013), 19–29. https://doi.org/10.4236/jdaip.2013.13004
[6] Habibi et al. 2021. Hashtag Analysis of Indonesian COVID-19 Tweets Using Social Network Analysis. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* 15 (07 2021), 275–284. https://doi.org/10.22146/ijccs.61626
[7] Hamborg et al. 2021. NewsMTSC: A Dataset for (Multi-)Target-dependent Sentiment Classification in Political News Articles. 1663–1675. https://doi.org/10.18653/v1/2021.eacl-main.142
[8] Radicioni et al. 2021. Analysing Twitter semantic networks: the case of 2018 Italian elections. *Scientific Reports* 11 (06 2021), 13207. https://doi.org/10.1038/s41598-021-92337-2
[9] Shevtsov et al. 2020. Analysis of Twitter and YouTube during USelections 2020. *ArXiv* abs/2010.08183 (2020).
[10] Tan et al. 2022. RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network. *IEEE Access* 10 (2022), 21517–21525. https://doi.org/10.1109/ACCESS.2022.3152828
[11] Wolf et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/ARXIV.1910.03771
[12] R George, K Shujaee, M Kerwat, Z Felfli, D Gelenbe, and K Ukuwu. 2020. A comparative evaluation of community detection algorithms in social networks. *Procedia Computer Science* 171 (2020), 1157–1165.
[13] Grandjean. 2016. A social network analysis of Twitter: Mapping the digital humanities community. *Cogent Arts and Humanities* 3 (04 2016), 1171458. https://doi.org/10.1080/23311983.2016.1171458
[14] Vishal Kharde and Sheetal Sonawane. 2016. Sentiment Analysis of Twitter Data: A Survey of Techniques. *International Journal of Computer Applications* 139 (04 2016), 5–15. https://doi.org/10.5120/ijca2016908625
[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.
[16] Alexander Shevtsov, Christos Tzagkarakis, Despoina Antonakaki, Polyvios Pratikakis, and Sotiris Ioannidis. 2022. Twitter Dataset on the Russo-Ukrainian War.
[17] Hamada M Zahera, Ibrahim A Elgendy, Richa Jalota, and Mohamed Ahmed Sherif. 2019. Fine-tuned BERT Model for Multi-Label Tweets Classification.. In *TREC*. 1–7.

Performance evaluation of the SloBERTa and XML-RoBERTa transformer models on the Slovenian news corpus SentiNews 1.0

Martin Domajnko
martin.domajnko@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Jakob Kordež
jakob.kordez@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

ABSTRACT

Sentiment analysis, also called opinion mining, is a highly restricted natural language processing problem. This paper presents the use of existing SloBERTa and XML-RoBERTa models on the Slovenian news corpus SentiNews 1.0 and compares their performance. The results are further compared to the results achieved by the Multinomial Naive Bayes and Support Vector Machines methods used in the dataset paper. The trained models are also applied to data collected from the social media platform Reddit, in order to analyse the sentiment of posts and comments from the Slovenian community.

KEYWORDS

sentiment analysis, transformers, natural language processing, RoBERTa

1 INTRODUCTION

The rapid growth of digitally recorded opinion data over the past two decades is a key driver of the growing popularity of sentiment analysis. A lot of research is focused on collecting and analysing the data from review websites, forums and social media platforms like Twitter [1, 16], Internet Movie Database (IMDB) [9] and Amazon [11]. The data is collected with the help of web scraping tools and the social media platforms's own public API services.

Sentiment analysis research has been mainly carried out at three levels of granularity: document level, sentence level, and aspect level [7]. The problem can be approached as a binary classification problem, where the text is classified as positive or negative, or as a multi-class classification problem, where the text is classified as positive, negative, or neutral. The latter can also use a different set of three or more classes.

Firstly, a brief overview is given of the different approaches to sentiment analysis and a more exhaustive description of the most popular methods currently. This is followed by an outline of the dataset we used to train and evaluate the selected models and an examination of the data we scraped from the social media platform Reddit. Section 4 describes the training and evaluation pipeline. Results and findings are presented in Section 5 and the paper conludes in Section 6 with possible improvements.

2 RELATED WORK

Reviews of products or services and social media posts are the primary targets for sentiment classification. The employed techniques can be divided into three groups according to the methods they use. The machine learning approach uses machine learning models in combination with linguistic features, and can be broken down into supervised and unsupervised learning methods. The lexicon-based approach utilizes pre-prepared sentiment lexicons and is also divided into dictionary-based methods and corpus-based methods. The final approach is the hybrid approach, which combines the aforementioned methods [12].

Using computational methods, sentiment analysis systematically analyses people's expressed subjective information, such as opinions and emotions, towards different entities. The current state-of-the-art performance is realized with deep learning models using the means of self-attention, also known as transformers [18]. The concept was first presented by the combined team from Google Brain and Google Research in their 2017 paper "Attention is all you need" [18]. The Transformer model solved the recurrent neural networks and long short-term memory networks biggest constraint, that of sequential computation. The model doesn't rely on recurrence, but it insted relies entirely on an attention mechanism to draw global dependecies between input and output [18]. The model achieves faster computation times, because it allows for significantly more parallelization.

The most widely used language representation model is BERT, which stands for Bidirectional Encoder Representations from Transformers. It leverages the benefits of transformers and extends them with the ability to train the models on large unlabelled datasets and then fine-tune them, with just one additional output layer, on a wide range of downstream tasks [5]. One downside of these models is that, because of their size, they are hard to run in constrained computational environments. Authors in [15] proposed a distilled version of the BERT model called DistilBERT, which reduces the model's size by 40% and increases its speed by 60% while still achieving 97% of its performance. Further research on the BERT model has shown it to be considerably under trained [8]. The model RoBERTa, which stands for Robustly optimized BERT approach, proposed some modifications to the pre-training procedure that improved end-task performance and achieved state-of-the-art results on GLUE [19], SQuAD [14] and RACE [6] datasets [8]. Another important mention is the transformer-based multilingual masked language model XLM-R, which was pre-trained on text in 100 languages [4]. In our work, we compare it against the monolingual Slovene RoBERTa model [17] in two and three class sentiment analysis tasks.

https://doi.org/10.51939/scores22.10

Table 1: Number of negative, neutral and positive examples in the dataset based on the level of granularity

Level of granularity	Sentiment			Total
	Negative	Neutral	Positive	
Sentence	26.74%	56.98%	16.28%	168,899
Paragraph	26.36%	57.38%	16.26%	89,999
Document	32.00%	52.03%	15.97%	10,427

3 USED DATA

3.1 Dataset

For the training and evaluation of the models, we used the manually sentiment annotated Slovenian news corpus SentiNews 1.0 [2]. The data is sentiment annotated on three levels of granularity: sentence, paragraph and document level [2]. Our models were only trained on the sentence and paragraph levels. The same corpus is also presented in [3], where it was used to train two classifiers (Multinomial Naive Bayes and Support Vector Machines). We also compare the performance of our models to the results given in [3].

We split the corpus randomly into training, validation and testing sets with the sklearn Python library. The training set contained 90% of the data, while each of the other two contained 5%. As seen in Table 1, the data is heavily imbalanced and favours the neutral class.

3.2 Scraped data

We decided to scrape the r/Slovenia subreddit because the majority of the posts there were in Slovene. We accomplished this with a Python 3 script using the `praw`[1] package, which is a Reddit API wrapper. Because we wanted to scrape the whole subreddit, we also used the package `psaw`[2] which in combination with `praw` enabled us to fetch more than only 1000 posts that we were limited to before.

Using the first script, we were able to obtain the IDs and flairs of all the posts, group them by their flair (category), and save them locally. Our second script then enabled us to individually fetch posts by their IDs. We decided to fetch the post title, body text, score, upvote ratio, a timestamp of when it was created and the comments which we flattened from a tree structure to a list. Each comment has a numerical score and a text entry. After we removed all posts that were simply links to other posts and filtered the remaining posts, we were left with 14,000 posts which combined contained nearly 240,000 comments.

Not all posts and comments were in Slovene, some were in English. Most of the English posts were foreign people asking questions, so we decided to filter only Slovene posts and comments. We accomplished this with the help of N-Gram-Based text categorization with a Slovene and an English corpus. After filtering, we were left with 9,000 posts and 174,000 comments.

4 METHOD

Transformer models follow an encoder-decoder structure. For them to be able to process the text, it is first transformed into numerical vectors, also called word embeddings. Since the models don't use any recurrence, the word embeddings are also combined with positional encodings, which present information about the relative or absolute position of the tokens in the sequence [18]. Our work fine-tuned the XLM-RoBERTa (XLM-R) [4] and SloBERTa [17] models for Sentiment Analysis. PyTorch [13] based implementations of the models from the open-source Transformers library [20] were used. The models and tokenizers were already pre-trained on large datasets.

The architecture and training approach used in the SloBERTa model [17] is the same architecture as the RoBERTa base model [8], but it uses a different tokenization technique. For this, a Sentence Piece[3] model was trained, which splits the text into tokens and encodes it into subword byte-pair encodings. The model has a vocabulary size of 32000 subword tokens [17]. The model is closely related to the French monolingual model CamemBERT [10]. The pre-trained SloBERTa model we used, is the second version of the model, which was trained for 200000 updates in total [17].

XLM-R [4] is a multilingual version of the RoBERTa [8] model. Instead of using language-specific preprocessing and byte-pair encoding, the model uses a Sentence Piece model trained on raw text data for all languages. The pre-trained XLM-R model we used, was trained for 1.5 Million updates on five-hundred 32GB Nvidia V100 GPUs with a batch size of 8192 and has a vocabulary size of 250000 [4].

5 RESULTS

5.1 Experimental setting

The Kaggle environment with an Nvidia Tesla P100 GPU was used for training the models. We used a batch size of 32 and a learning rate of 6×10^{-7} for the SloBERTa model [17] and a batch size of 8 and a learning rate of 1×10^{-7} for the XLM-R model [4]. The hyperparameters were selected empirically. Additionally, to reduce the effects of class imbalance in the training data set, each model was supplied with precomputed class weights. The models were trained for a total of 20 epochs.

5.2 Model performance

The models, trained on the dataset annotated on the paragraph level of granularity, were evaluated on binary (positive and negative) and multi-class (positive, negative, and neutral) sentiment analysis tasks. The results in Table 2 show that on the binary classification task, both the SloBERTa model and XLM-R model achieve similar results with an accuracy above 90%. On the other hand, the multi-class classification task results show that the XLM-R model performs better, with an accuracy of 70.49%, compared to the accuracy of the SLoBERTa model of 66.04%. Additionally, the SloBERTa model trained on the dataset annotated on the sentence level of granularity, was also evaluated on binary and multi-class sentiment analysis tasks. The model's performance was compared to the SVM and NBM models presented in paper [3]. The results are collected in Table 3 and show that the much simpler NVM and NBM models outperform the SloBERTa model on both binary and multi-class classification tasks. Additionally, all the models achieve an accuracy

[1] https://github.com/praw-dev/praw
[2] https://github.com/dmarx/psaw
[3] https://github.com/google/sentencepiece

Table 2: Comparison of results on the dataset annotated on the paragraph level of granularity, for the SloBERTa and XLM-R models, on binary (positive and negative) and multi-class (positive, negative, and neutral) sentiment analysis tasks. The first two columns contain the model name and number of classes. The other four columns present the model accuracy, precision, recall and F1 norm metrics.

Model	No. of classes	Accuracy	Recall	Precision	F1
SloBERTa	2	91.19	90.91	90.67	90.79
XLM-R	2	91.29	90.83	90.89	90.86
SloBERTa	3	66.04	72.19	64.74	65.70
XLM-R	3	70.49	68.93	67.84	68.35

Table 3: Comparison of results on the dataset annotated on the sentence level of granularity, for the SloBERTa model and the two models (SVM and NBM) presented in paper [3], on binary (positive and negative) and multi-class (positive, negative, and neutral) sentiment analysis tasks. The first two columns contain the model name and number of classes. The other two columns present the model accuracy and F1 norm metrics.

Model	No. of classes	Accuracy	F1
SloBERTa	2	90.40	89.87
SVM	2	93.10	94.86
NBM	2	95.21	96.38
SloBERTa	3	65.47	64.64
SVM	3	73.10	55.35
NBM	3	66.46	61.20

above 90% on the binary task and an accuracy above 65% on the multi-class task.

5.3 Reddit analysis

The goal of this experiment was to analyse the sentiment of posts and comments made in the Slovenian community on the social media platform Reddit. We also looked for a potential correlation between the score, flair, and sentiment of the analysed posts and comments.

In the process of fine-tuning, a model trained to perform on a given task is tweaked to perform on a second similar task. We used the SloBERTa and XLM-R models, which were fine-tuned for binary and multi-class sentiment analysis tasks on the dataset annotated at the paragraph level of granularity, to predict the sentiment of the 9,000 posts and 170,000 comments scraped from the social media platform Reddit.

Table 4 summarizes the sentiment classification distribution of posts and comments. We can see that the multi-class SloBERTa model classifies the posts and comments evenly between the three sentiments, while the XLM-R model classifies more posts and comments as neutral.

Table 4: Classification results for posts and comments, made by the SloBERTa and XLM-R models trained on the dataset annotated on the paragraph level of granularity.

Model	No. of classes	Negative	Neutral	Positive
SloBERTa	2	48.66%	/	51.34%
XLM-R	2	56.53%	/	43.47%
SloBERTa	3	32.66%	37.80%	29.54%
XLM-R	3	22.08%	61.22%	16.70%

We found that posts and comments with more downvotes than upvotes tend to have a negative sentiment rather than a positive one. Furthermore, longer comments and comments with a score above 250, are more likely to be classified as negative.

When we grouped posts by their flair, we found that posts tagged with the flair "Question" had a much larger number of neutral posts. The flairs "News" and "Article" had an equal number of posts with positive and negative sentiments, but their comments were heavily leaning to the negative side. We also found that posts and comments tagged with the flair "Discussion" were more negative than positive.

6 CONCLUSION

In this study, we fine-tuned two transformer models, SloBERTa and XLM-R, for binary and multi-class sentiment analysis tasks on the Slovenian news corpus SentiNews 1.0. Both models achieved similar results on the binary classification task, while on the multi-class classification task, the XLM-R model performed better. Additional comparison of the SloBERTa model with the NVM and NBM models has shown that the transformer model achieves slightly worse results on both binary and multi-class classification tasks. The trained models were also applied on data scraped from the social media platform Reddit. The results have shown that the XLM-R model is more likely to classify posts and comments as neutral, while the SloBERTa model classifies all classes evenly.

REFERENCES

[1] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. 2020. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. https://doi.org/10.48550/ARXIV.2010.12421

[2] Jože Bučar. 2017. Manually sentiment annotated Slovenian news corpus SentiNews 1.0. http://hdl.handle.net/11356/1110 Slovenian language resource repository CLARIN.SI.

[3] Jože Bučar, Martin Žnidaršič, and Janez Povh. 2018. Annotated news corpora and a lexicon for sentiment analysis in Slovene. *Language Resources and Evaluation* 52, 3 (Feb. 2018), 895–919. https://doi.org/10.1007/s10579-018-9413-3

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised Cross-lingual Representation Learning at Scale. https://doi.org/10.48550/ARXIV.1911.02116

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[6] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. https://doi.org/10.48550/ARXIV.1704.04683

[7] B. Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions.* 1–367 pages. https://doi.org/10.1017/CBO9781139084789

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa:

A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/ARXIV.1907.11692

[9] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. https://aclanthology.org/P11-1015

[10] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suá rez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a Tasty French Language Model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.645

[11] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *Proceedings of the 7th ACM Conference on Recommender Systems* (Hong Kong, China) *(RecSys '13)*. Association for Computing Machinery, New York, NY, USA, 165–172. https://doi.org/10.1145/2507157.2507163

[12] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5, 4 (2014), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[14] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. https://doi.org/10.48550/ARXIV.1606.05250

[15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. https://doi.org/10.48550/ARXIV.1910.01108

[16] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized Affect Representations for Emotion Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3687–3697. https://doi.org/10.18653/v1/D18-1404

[17] Matej Ulčar and Marko Robnik-Šikonja. 2021. Slovenian RoBERTa contextual embeddings model: SloBERTa 2.0. http://hdl.handle.net/11356/1397 Slovenian language resource repository CLARIN.SI.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. https://doi.org/10.48550/ARXIV.1706.03762

[19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. https://doi.org/10.48550/ARXIV.1804.07461

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

Spletna aplikacija za analizo tapkanja s prsti

Filip Zupančič
fz9889@student.uni-lj.si
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani
Večna pot 113
SI-1000 Ljubljana, Slovenija

Gal Žagar
gz2481@student.uni-lj.si
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani
Večna pot 113
SI-1000 Ljubljana, Slovenija

Dejan Georgiev
dejan.georgiev@kclj.si
UKC Ljubljana, Nevrološka klinika,
Zaloška 2
SI-1000 Ljubljana, Slovenia

Jure Žabkar
jure.zabkar@fri.uni-lj.si
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani
Večna pot 113
SI-1000 Ljubljana, Slovenija

POVZETEK

Diagnoza Parkinsonove bolezni je aktualen raziskovalni problem, s katerim se soočajo zdravniki. Simptomi pri pacientih pogosto niso jasno izraženi, kar poveča možnost za napako pri oceni bolezni. Z uporabo metod strojnega učenja in razvojem namenske uporabniške programske opreme, lahko zdravnik pridobi dodatne informacije, ki zmanjšajo možnost napak pri diagnozi. Prav tako lahko na ta način pacientu ponudimo izbiro, da test opravi doma. Namen članka je predstaviti spletno aplikacijo in zaledni sistem za preprosto analizo posnetka tapkanja s prsti, ki ga izvaja pacient.

KLJUČNE BESEDE

Aplikacija, vaja tapkanja s prsti, Parkinsonova bolezen, video posnetek, MDS-UPDRS.

1 UVOD

Parkinsonova bolezen je počasi napredujoča nevrodegenerativna bolezen, za katero ne poznamo vzroka nastanka. Med najbolj očitne znake oz. simptome bolezni spadajo bradikinezija (upočasnitev gibov), tremor, rigidnost (zvišan mišični tonus), in motnje ravnotežja. Prisotnost omenjenih motoričnih simptomov zdravniki ocenjujejo s pomočjo Združene lestvice za oceno Parkinsonove bolezni Združenja za motnje gibanja (ang. Movement Disorders Society - Unified Parkinson's Disease Rating Scale - MDS-UPDRS) [2]. Ta vsebuje štiri dele vprašanj in motoričnih testov, ki jih pacient izvaja v prisotnosti ocenjevalca/zdravnika. Eden od testov je *tapkanje s prsti*, ki sodi v tretji del lestvice MDS-UPDRS (3.4). S tem testom ocenjujemo upočasnitev gibov, ki je glavna značilnost Parkinsonove bolezni.

Pri tapkanju s prsti se pacient s kazalcem dotika palca, nato pa kazalec kar se da oddalji; ta gib mora čim hitreje ponoviti od 10 do 15-krat. Pri tem se lahko pojavljajo različne motnje (po MDS-UPDRS [2]):

Prekinitev gibanja Ponavadi se kaže v kratkih tresljajih, v katerih se dlan za trenutek odmakne od pričakovane poti.

Obotavljanje je najkrajša motnja, pri kateri ima pacient težave z iniciacijo gibanja. Lahko se pojavlja tudi med gibanjem, zato jo v praksi težko ločimo od prekinitve.

Zamrznitev gibanja je stanje, v katerem se pacientova roka popolnoma ustavi in je nekaj sekund negibna. Zaustavitev traja dlje kot prekinitev in obotavljanje.

Upočasnitev ali zmanjšanje amplitude gibanja je s prostim očesom težko prepoznati, saj se amplituda znižuje postopoma. Lažje jo zaznamo na posnetku, z merjenjem razdalje med palcem in kazalcem. Postopno zmanjševanje hitrosti ali amplitude gibov je značilnost upočasnitve gibov pri Parkinsonovi bolezni, redkeje pa ta fenomen vidimo pri drugih oblikah parkinsonizma.

Po izvedbi testa zdravnik z oceno od 0 do 4 oceni stopnjo motnje (Test tapkanja s prsti, MDS-UPDRS, 3.4):

(0) Pacient je test izvedel brez motenj.
(1) Nekaj od naštetega: a) pravilen ritem je prekinjen z eno ali dvema prekinitvama ali obotavljanjem; b) rahla upočasnitev; c) zmanjšanje amplitude proti koncu izvajanja.
(2) Kar koli od naslednjega: a) 3 do 5 prekinitev; b) blaga upočasnitev; c) amplituda se zmanjša na sredini izvajanja.
(3) Kar koli od naslednjega: a) več kot 5 prekinitev ali vsaj enkrat daljša zaustavitev; b) zmerna upočasnitev; c) amplituda zmanjševanje se začne po 1. dotiku.
(4) Pacient ni bil zmožen dokončati testa.

Podana ocena je subjektivna in je v veliki meri odvisna od pozornosti zdravnika pri ocenjevanju. Zdravnik nima možnosti ponovnega ogleda opravljenega testa; pacient ga lahko sicer ponovno izvede, pri čemer je treba upoštevati spremenjene pogoje - pacient se lahko utrudi in rezultati so slabši. V pomoč zdravniku smo razvili spletno aplikacijo, ki omogoča objektivno analizo in ocenjevanje testa tapkanja s prsti. Zastavljena je kot preprost ekspertni sistem za podporo odločanju pri diagnozi bolezeni. Zaledni del aplikacije skrbi za obdelavo video posnetkov in iskanje smiselnih vzorcev v podatkih. Uporabniški vmesnik je v osnovi namenjen grafičnemu prikazu analize videoposnetka. Prav tako uporabniku omogoča preprosto nalaganje video datotek, ki jih želi analizirati.

1.1 Pregled področja

V literaturi zasledimo metode za pomoč pri diagnostiki Parkinsonove bolezni z uporabo strojnega učenja [1]. Ključna problema

https://doi.org/10.51939/scores22.11

sta pomanjkanje kvalitetnih podatkov in velik problemski prostor, kar pomeni, da rešitev ni trivialna in trenutno ne poznamo najbolj primernega pristopa k reševanju. V raziskavi ~[3], ki vključuje 387 video posnetkov testa tapkanja s prsti, je bilo vključenih 13 pacientov s Parkinsonovo boleznijo; izvedena je bila klasifikacija z uporabo podpornih vektorjev, s katero je dosežena 88 % klasifikacijska točnost. V raziskavi ~[7] je opisana klasifikacija po MDS-UPDRS lestvici z uporabo orodja DeepLabCut za sledenje gibanja dlani in hitra Fourierjeva transformacija za prepoznavo porazdelitve frekvenc v časovnih vrstah posameznega premika prstov iz ene skrajne lege v drugo. Raziskava je bila narejena na 138 video posnetkih 39 pacientov. Rezultati so pokazali Pearsonovo korelacijo z MDS-UPDRS (0.69, $p < .001$).

2 RAZVOJ APLIKACIJE

Razvili smo spletno aplikacijo, ki rešuje zgoraj omenjen problem ocenjevanja testa tapkanja s prsti. Njen glavni namen je zdravniku omogočiti, da si video posnetek pacienta, ki izvaja omenjeni test, lahko večkrat (tudi upočasnjeno) predvaja in na podlagi obdelave posnetka dobi dodatne informacije, ki mu pomagajo pri postavitvi diagnoze.

2.1 Uporabniški vmesnik

Uporabniški vmesnik smo razvili z uporabo JavaScript ogrodja *Vue.js* [1]. Uporabili smo tehnologije HTML in CSS za definiranje samega izgleda aplikacije in programski jezik JavaScript za implementacijo aplikacijske logike. Uporabniški vmesnik je sestavljen iz dveh enostavnih strani oz. aktivnosti: *začetna stran* in *stran za prikaz rezultatov*.

2.1.1 Začetna stran. Na njej lahko uporabnik naloži video posnetek, ki ga želi analizirati. Na strani se nahajata 2 gumba. Prvi omogoča uporabniku, da naloži datoteko oz. video posnetek pacienta (med izvedbo testa tapkanja s prsti). Dovoljena formata posnetkov sta *mp4* in *avi*. S klikom na gumb 'Upload video' uporabnik sproži procesiranje izbranega posnetka. Posnetek se ob tem posreduje zalednemu sistemu oz. spletnemu strežniku. Strežnik je razvit v programskemu jeziku Python in vsebuje glavni program za procesiranje video posnetkov. Z uporabo ogrodja Flask [2] smo razvili REST API za komunikacijo med zalednim sistemom in spletno aplikacijo.

Po končanem procesiranju strežnik aplikaciji posreduje podatke o obdelanem video posnetku. Aplikacija obvesti uporabnika o uspešnosti. Pridobljene podatke shranimo v lokalno shrambo z namenom, da jih kasneje uporabimo na drugi strani aplikacije. Med podatki se nahajajo naslednji parametri posnetka, ki smo jih zajeli in izračunali med obdelavo:

- **Izračunana razdalja** med konico palca in konico kazalca, v vsakem trenutku posnetka,
- **Hitrost** gibanja kazalca proti palcu,
- **Pospešek** kazalca proti palcu, ter
- **Kot** in **kotna hitrost** med palcem in kazalcem dlani.

V naslednjem koraku izračunane podatke uporabniku tudi prikažemo. Tako lahko nadaljujemo z drugim delom našega vmesnika, in sicer stran za analizo in prikaz rezultatov procesiranja.

2.1.2 Stran za analizo in prikaz rezultatov. Glavni namen je uporabniku na lep način prikazati podatke predhodnega procesiranja (slika 1).

Ena izmed glavnih komponent na tej strani je obdelan posnetek, ki ga je uporabnik naložil v prejšnjem koraku. Med procesiranjem na strežniku posnetek opremimo z izrisom skeleta na palcu in kazalcu. Uporabnik si lahko posnetek pogleda in ga analizira s pomočjo izračunanih količin. Posnetek po končani obdelavi shranimo v S3 zabojnik (angl. S3 bucket [3]) in ga od tam prek HTTP zahtevkov pošljemo na našo stran. Na strani prikazan posnetek si lahko zdravnik večkrat predvaja.

Poleg posnetka na strani prikažemo tudi posamezne lastnosti gibanja dlani, izračunane med procesiranjem posnetka. Le te so predstavljene v obliki grafov. Trenutno sta na strani prikazana dva, in sicer:

- Graf dolžine med konico palca ter kazalca v odvisnosti od časa in
- Graf hitrosti gibanja palca proti kazalcu v odvisnosti od časa.

Zdravnik poleg posnetka dobi možnost podrobne analize in vpogleda v gibanje pacienta, kje se je gibanje upočasnilo, kje je prišlo do dotika, koliko časa je trajal ter kakšne so zakasnitve med gibanjem.

3 REZULTATI ANALIZE POSNETKA

V tem razdelku podrobneje predstavimo rezultate procesiranja video posnetka ter posamezne elemente strani za analizo in prikaz rezultatov.

3.1 Video posnetek s skeletom dlani

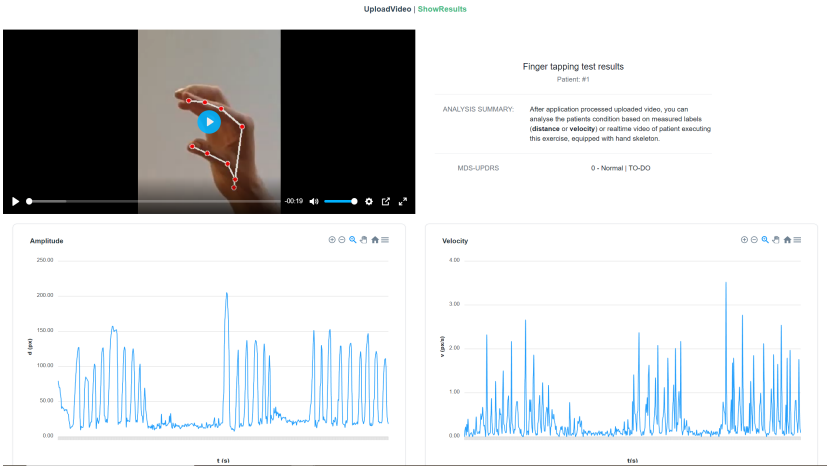
Glavni del aplikacije je video posnetek, ki ga naloži zdravnik in prikazuje pacienta med izvajanjem testa tapkanja s prsti. Po uspešnem nalaganju posnetek posredujemo zalednemu sistemu, kjer ga analiziramo in obdelamo. Iz posnetka izluščimo čim več uporabnih informacij o gibanju pacientovih prstov med izvajanjem testa.

Za obdelavo posnetka smo uporabili knjižnico OpenCV [6] in model Mediapipe Hands [4] za zaznavanje dlani na posnetku:

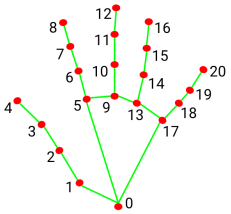
- **OpenCV** [4] je odprtokodna knjižnica, ki zajema področje računalniškega vida ter strojnega učenja. Knjižnico smo uporabili za segmentacijo posnetka na posamezne okvirje oz. slike.
- **Mediapipe Hands** [5] je model za zaznavanje dlani v realnem času. Model je implementiran v ogrodju Mediapipe [5]. Uporabili smo ga za zaznavanje dlani na posameznih slikah, pridobljenih po predhodni segmentaciji posnetka.

Nad vsako sliko dlani smo izvedli zaznavanje roke z uporabo modela Mediapipe Hands. Rezultat je množica koordinat 21 točk skeleta dlani (Slika 2). V našem programu smo za analizo posnetka uporabili točke palca in kazalca (točke od 0 do 8) in iz njihovih koordinat izračunali naslednje lastnosti gibanja: razdaljo med konico palca in kazalca, hitrost gibanja kazalca proti palcu, pospešek kazalca, kot med prstoma in kotno hitrost. Te točke smo izrisali na prvotni posnetek (Slika 3). S skeletom opremljen posnetek zaznane dlani prikažemo v oknu aplikacije. Zdravniku omogoča večkratni

[1] https://vuejs.org/
[2] https://flask.palletsprojects.com/en/2.2.x/
[3] https://aws.amazon.com/s3/
[4] https://opencv.org/
[5] https://google.github.io/mediapipe/solutions/hands.html

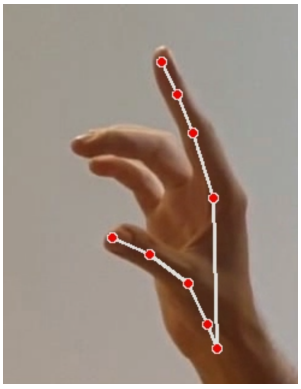


Slika 1: Stran za analizo in prikaz rezultatov.



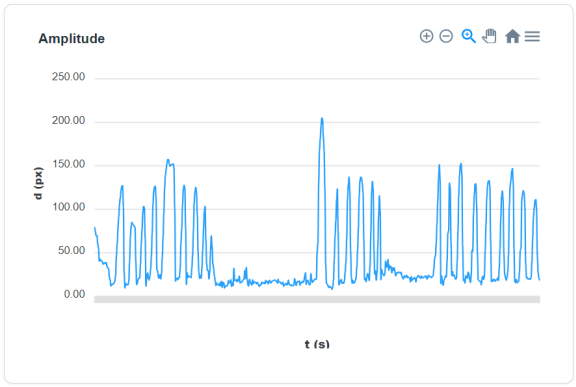
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Slika 2: Slika prikazuje 21 točk skeleta dlani, ki jih dobimo kot rezultat zaznavanja dlani z uporabo modela Mediapipe Hands.



Slika 3: Na posamezni sliki video posnetka smo, s pomočjo točk pridobljenih z zaznavanjem dlani, izrisali skelet palca in kazalca.

pogled ter počasno, pozorno analizo. Omogoča tudi preverjanje izračunanih podatkov, kar vpliva na zanesljivost končne diagnoze.



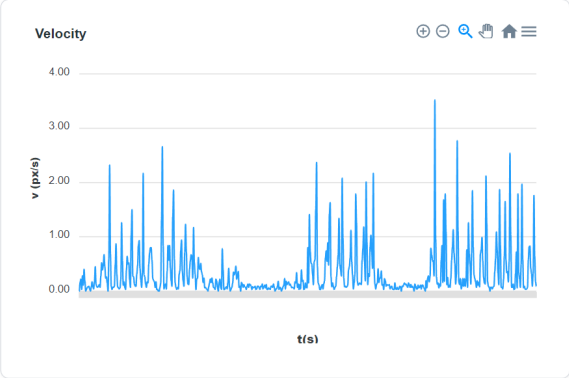
Slika 4: Graf prikazuje razdaljo med konico palca in kazalca dlani v odvisnosti od časa.

3.2 Grafi posameznih lastnosti

Na strani smo poleg video posnetka prikazali še dva grafa. Z grafi smo želeli omogočiti zdravniku podrobnejšo analizo gibanja pacienta. Iz njih lahko namreč razbere različne anomalije gibanja, ki so se pojavile med samo izvedbo vaje. Grafi prikazujejo lastnosti oz. informacije o video posnetku, ki smo jih izračunali med procesiranjem. Med številnimi, zgoraj omenjenih lastnosti smo na strani prikazali 2 grafa, in sicer:

- Graf, ki prikazuje izračunano razdaljo med konico palca in konico kazalca v odvisnosti od časa ter.
- Graf izračunane hitrosti v odvisnosti od časa, s katero se kazalec giblje proti palcu.

3.2.1 Graf razdalje med prstoma. Posnetek smo razdelili na posamezne slike in vsako posebej analizirali. Ena pomembnejših lastnosti pri oceni testa je spreminjanje razdalje med palcem in kazalcem v času. Na vsaki sliki smo izračunali razdaljo med konicama prstov; uporabniku prikažemo graf razdalje v odvisnosti od časa (Slika 4).



Slika 5: Slika prikazuje graf hitrosti gibanja kazalca proti palcu, v odvisnosti od časa.

Tako si lahko zdravnik graf pogleda in prepozna različne anomalije, ki so nastale med izvedbo testa. Zdravnik lahko opazi kje je imel pacient težave z gibanjem, spremlja amplitudo gibanja in določi ali se je pacient sčasoma utrudil in pa razpozna lahko ustavitve, prekinitve in obotavljanje pacienta med izvedbo testa. Kot primer lahko vzamemo sliko 4; graf prikazuje spreminjanje razdalje med palcem in kazalcem v odvisnosti od časa. Iz njega je lepo razvidno, da je med izvedbo testa prišlo do dveh daljših zaustavitev tapkanja. Te informacije so lahko zelo koristne za zdravnika, ko se odloča za končno oceno izvedbe testa.

3.2.2 Graf hitrosti gibanja. Graf hitrosti, s katero se kazalec giblje proti palcu, pomaga zdravniku ugotoviti ali je pri izvedbi testa prišlo do postopnega zmanjšanja hitrosti gibanja. To je namreč tudi ena izmed pomembnih značilnosti bolezni na katero mora biti zdravnik pozoren. Računali smo hitrost v konici kazalca dlani, in sicer v vsaki sliki video posnetka. Čas med slikami je približno 0.03 s, za spremembo razdalje opazovane točke kazalca pa smo uporabili kar evklidsko razdaljo med točko na predhodni sliki in pa točko na trenutni sliki. Tako smo izračunali hitrost po posamezni komponenti točke, $v_x = \frac{|d_x|}{t}$ in $v_y = \frac{|d_y|}{t}$. Po tem pa smo lahko izračunali še skupni vektor hitrosti po formuli:

$$v = \sqrt{(v_x)^2 + (v_y)^2}$$

Rezultati za vsako posamezno sliko so bili po procesiranju shranjeni in posredovani aplikaciji. Tako smo tudi v tem primeru na strani prikazali graf spremembe hitrosti v času. Tudi iz tega grafa lahko izlušči koristne informacije. Prepozna lahko ustavitve gibanja, krajše prekinitve ter ali je pri gibanju prišlo do zmanjšanja hitrosti. Slika 5 prikazuje graf spremembe hitrosti gibanja pri istem pacientu kot na sliki 4. Tudi tukaj je lepo razvidno kje je prišlo do ustavitev gibanja, ki pričakovano časovno sovpadajo na obeh slikah.

Zdravniku smo z grafi želeli podati čim več dodatnih informacij o poteku testa tapkanja s prsti, s katerimi bi si lahko pomagal pri določanju končne ocene. V prihodnosti želimo na strani predstaviti še več koristnih podatkov (npr. frekvenčna analiza, kotna hitrost, pospešek itd.) in tako še izboljšati uporabniško izkušnjo in olajšati analizo testa. Planiramo tudi testirati možnost kvantifikacije amplitude in hitrosti gibov s primerjavo parametrov, ki jih dobimo iz video posnetkov in klinično oceno amplitdue oz. hitrosti na podlagi MDS-UPDRS lestvice.

4 ZAKLJUČEK

Predstavili smo spletno aplikacijo za pomoč zdravnikom pri analiziranju testa tapkanja s prsti. Aplikacija zaenkrat prikazuje le osnovne informacije in ne posega v klinično diagnozo zdravnika. V prihodnosti si želimo iz pridobljenih podatkov ustvariti model za razpoznavanje anomalije gibanja ter na koncu tudi podati svojo oceno amplitude in hitrosti giba. Smiselna nadgradnja je prikazovanje motenj pri izvanaju testa v različnih časovnih intervalih izvajanja. Najprej moramo iz obstoječih posnetkov razbrati še nekaj dodatnih lastnosti, ki jih bo uporabljal naš model. Eden od obetavnih pristopov je frekvenčna analiza z uporabo Fourierjeve transformacije, s katero lahko razberemo vzorce v gibanju, ki s prostim očesom niso vidni. Dodatno lahko anomalije v gibanju prepoznavamo z modelom končnih avtomatov, ki je sestavljen iz stanj, v katerih se nahaja roka med izvajanjem testa. Ključno za zanesljiv sistem je testiranje na večjem številu posnetkov, zato bo v prihodnosti velik poudarek na pridobivanju ustreznih podatkov.

LITERATURA

[1] Minja Belić, Vladislava Bobić, Milica Badža, Nikola Šolaja, Milica Đurić Jovičić, and Vladimir S. Kostić. 2019. Artificial intelligence for assisting diagnostics and assessment of Parkinson's disease—A review. *Clinical Neurology and Neurosurgery* 184 (2019), 105442. https://doi.org/10.1016/j.clineuro.2019.105442

[2] Christopher G Goetz, Barbara C Tilley, Stephanie R Shaftman, Glenn T Stebbins, Stanley Fahn, Pablo Martinez-Martin, Werner Poewe, Cristina Sampaio, Matthew B Stern, Richard Dodel, et al. 2008. Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): scale presentation and clinimetric testing results. *Movement disorders: official journal of the Movement Disorder Society* 23, 15 (2008), 2129–2170.

[3] Taha Khan, Dag Nyholm, Jerker Westin, and Mark Dougherty. 2014. A computer vision framework for finger-tapping evaluation in Parkinson's disease. *Artificial Intelligence in Medicine* 60, 1 (2014), 27–40. https://doi.org/10.1016/j.artmed.2013.11.004

[4] GOOGLE LLC. 2020. *Mediapipe Hands.* https://google.github.io/mediapipe/solutions/hands.html Mediapipe Hands solution page.

[5] GOOGLE LLC. 2020. *Mediapipe Home.* https://google.github.io/mediapipe/ Mediapipe home page.

[6] OpenCV Team. 2022. *OpenCV.* https://opencv.org/ OpenCV home page.

[7] Stefan Williams, Zhibin Zhao, Awais Hafeez, David C. Wong, Samuel D. Relton, Hui Fang, and Jane E. Alty. 2020. The discerning eye of computer vision: Can it measure Parkinson's finger tap bradykinesia? *Journal of the Neurological Sciences* 416 (2020), 117003. https://doi.org/10.1016/j.jns.2020.117003

Iterated prisoner's dilemma and survival of the fittest from an ecological perspective

Martin Domajnko
martin.domajnko@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

ABSTRACT

The iterated prisoner's dilemma is a heavily studied concept in the field of game theory. It was popularized by Axelrod, and it can be used as a tool for modelling complex interactions between self-interested entities. The goal of this paper was to study the impact the environment has on the development of populations of prisoner's dilemma strategies in a simulation, where individuals interact with each other and play the iterated prisoner's dilemma game. This was done from an ecological perspective, meaning the behaviour of strategies stayed static and didn't evolve between generations, in a simulation extending the iterated prisoner's dilemma game. Additionally, the paper presents two new strategies, which are evaluated with Axelrod's original tournament and with our simulation. The implemented simulation uses Axelrod's tournament as a fitness function and fitness proportionate selection for choosing the next generation's strategies. Both of our strategies are based on the n-Pavlov strategy. They achieved average results, but none of them improved the original ones.

KEYWORDS

Prisoner's dilemma, Iterated prisoners's dilemma, n-Pavlov, Simulation, Game theory

1 INTRODUCTION

The prisoner's dilemma presents a situation in which two entities each need to choose between cooperation or defection, while being separated and unable to communicate. There are four possible outcomes: both entities cooperate, both entities defect, or one entity cooperates and one defects, and vice versa. In the first case, both entities get an equal payoff, normally called reward and marked as R. In the second case, both entities get the punishment payoff or P, and in the last case, the cooperating entity gets the sucker's payoff or S, and the defecting entity gets the temptation payoff or T. In the iterated prisoner's dilemma (IPD), two players play multiple rounds of the prisoner's dilemma game and are able to remember their opponent's previous actions [5].

In the classic example of the prisoner's dilemma game with classically rational players, defection always yields a better payoff regardless of the opponent's move. Defection is therefore a strictly dominant strategy for both players, and mutual defection presents the Nash equilibrium move [9] of the Prisoner's dilemma game. This means, that each player could only do worse by unilaterally changing their move [8]. On the other hand, in the iterated version of the game, there are two scenarios. In the first case, the game has a fixed number of rounds, which is known to both players in advance. Therefore, defection is still the dominant strategy and the proof can be constructed with the help of backward induction. In the second case, where the number of rounds is unknown or infinite, mutual defection presents still the Nash equilibrium, but there is no strictly dominant strategy. Studies on the iterated version of the game have shown that most top scoring strategies are nice. This means, that they defect only after their opponent has defected at least once.

Although the Prisoner's dilemma game is studied and applied in different fields, ranging from politics [12], economics [10] and biology [6], this paper focuses more on the implementation side of the problem. We present two new strategies for playing the iterated prisoner's dilemma game. The introduction of the paper gives a short explanation of the prisoner's dilemma game and a summary of the paper's structure. Section 2 presents two additional rules used in the iterated version of the game and gives a more detailed description of Axelrod's tournaments and experiments. This is followed up with Section 3, where we present the implemented strategies used in our experiments and explain the methods used for evaluating their performance. In Section 4, we present the interesting results and finish the paper with Section 5, with a short conclusion.

2 BASIC INFORMATION

To incentivize continuous cooperation over alternating cooperation and defection, the IPD game has two additional rules, defined by conditions $T > R > P > S$ and $2R > T + S$, where T, R, P and S present temptation, reward, punishment, and sucker's payoffs, respectively [5].

Axelrod got the idea for a computer tournament from his high school interest in artificial intelligence and also his interest in game theory starting in college. While pursuing his PhD in Political Science, the motivation for further research and understanding different ways of playing the game, were based on his desire to promote cooperation between players. Another factor was also the game's potential as a source of insight into international conflicts [3].

The IPD computer tournament consists of entrants, in this case programs, which select cooperation or defection on each move based on their decision rules and the history of the game so far. The programs for Axelrod's tournaments were provided from other experts in the field of game theory and researchers studying the Prisoner's Dilemma game [1].

https://doi.org/10.51939/scores22.12

2.1 The first Axelrod's tournament

In Axelrod's first tournament, the strategies were paired against each other in a round-robin style tournament. Additionally, each strategy played against itself and against a strategy which randomly cooperates and defects. The payoff matrix used in the tournament is presented in Table 1 and each game was played for 200 moves.

Table 1: Payoff matrix for a move in the Prisoner's Dilemma game

	Cooperate	Defect
Cooperate	3, 3	0, 5
Defect	5, 0	1, 1

In total, fourteen entries were submitted to the tournament [1]. The results of the tournament had many surprises. The most important results are:

(1) the strategy Tit for Tat won the tournamnet,
(2) none of the strategies succeeded to improve the Tit for Tat strategy, and
(3) all the top performing strategies had the property of being nice, which means, they defect only after their opponent has defected at least once.

Let us notice that the Tit for Tat strategy imitates its opponent's previous move, except in the first round when it cooperates.

2.2 The second Axelrod's tournament

The entrants providing programs for the second Axelrod's tournament were familiar with the analysis and discoveries of the first round. This is the main reason, why the results produced a better insight into the nature of effective choice in the Prisoner's Dilemma. Additionally, the size of the tournament increased from 14 to 62 entries. The rules of the tournament stayed similar to the first one, except the length of the games. The length was determined probabilistically with a 0.00346 chance of ending with each given move, instead of setting a fixed number of 200 moves to play. Again, the winner of the tournament was the Tit for Tat strategy [2]. The main finding from the paper presenting the second tournament is that the performance of the strategies is heavily dependent on the environment in which the game is played.

2.3 Axelrod's evolving new strategies

In addition to creating the previously described tournaments, Axelrod also looked at the problem from an evolutionary perspective [4]. He used the knowledge and methods from biological evolution, applying them, with the help of genetic algorithms, to a socially rich environment consisting of strategies submitted to the prisoner's dilemma tournament. The strategies, used in the simulation experiment, were represented as a string of genes on a chromosome on which genetic transformations can be applied [4]. In each step of the simulation, the strategies played against eight representatives, and based on their performance, the successful strategies were selected for a mating process in which they produced offspring. The offspring represents the next generation's strategies. Both mutation and crossover mechanisms were used for simulating evolution and discovering new strategies. The final results of his experiments have shown, that populations evolved members which were as good as the strategy Tit for Tat and also many of them reflected properties that make Tit for Tat successful [4].

3 IMPLEMENTATION OF EXPERIMENTS

The implementation of our solution was done in the Rust programming language. It heavily focuses on being extensible, adding new strategies, and customizable, changing the starting parameters of the tournament and simulation. The only condition for newly added strategy is that they need to implement the base Strategy trait we defined. For running the simulations, we also support reading the starting parameters from a file. The payoff matrix used in our experiments is the same as in Axelrod's first tournament, which is shown in Table 1.

3.1 The proposed simulation

The implemented simulation was inspired by genetic algorithms, albeit in our case, the chosen entities don't reproduce and aren't exposed to mutations but rather are cloned. This way, no new rules are introduced, and it presents a survival of the fittest simulation from a more ecological perspective rather than a strictly evolutionary one. A similar simulation was already presented in Axelrod's 1980 paper [2], which analyzes the results from his second tournament.

The proposed simulation starts with n different strategies, each having a population size of p. In every step of the simulation, each strategy plays t rounds of the prisoner's dilemma game with all other strategies. The total sum of scores achieved during those games is used as a fitness function, based on which the next generation's strategies are chosen. In the selection process, the fitness proportionate selection method, also known as roulette wheel selection, is used and through the simulation, the total number of entities stays constant. This produces a simulation where low performing strategies slowly die out, and high performing strategy's population sizes grow.

3.2 Strategies in the proposed simulation

In total, we implemented 20 strategies, 18 known ones and two of our own. All strategies are listed below with their name, a corresponding designation and a short description of their behaviour.

(1) Always Cooperate (ALLC): Always cooperates.
(2) Always Defect (ALLD): Always defects.
(3) CD: Alternates between cooperating and defecting.
(4) CCD: Alternates between cooperating twice and defecting once.
(5) DC: Same behaviour as CD, but starts with defecting.
(6) Grim: Cooperates until the opponent defects and then always defects thereafter.
(7) Prober: Plays D, C, C in the first three rounds and based on opponent's moves, chooses his further behaviour. If the opponent defected in round two and cooperated in round three, the strategy defects forever, else it applies the TFT strategy.
(8) Random (RAND): Cooperates with probability $P(C) = 0.5$.
(9) Probability p Cooperator: Cooperates with fixed probability $P(C) = p$.

(10) Reactive (with parameters $R(s, p, q)$): Cooperates with probability s in the first round and after that with probabilites p and q, based on the opponents decision in the previous round.
(11) Tit for Tat (TFT): Imitates its opponent's previous move, except in the first round when it cooperates.
(12) Tit for two Tats (TFTT or TF2T): Defects only if the opponent defected twice in a row.
(13) Two Tits for Tat (TTFT or 2TFT): Defects twice in a row if the opponent defected in the previous round.
(14) Imperfect Tit fot Tat (ImpTFT): Same behaviour as TFT, except that the probability for imitation is less than one.
(15) Suspicious Tit fot Tat (STFT): Same behaviour as TFT, except that it defects in the first round.
(16) Win-Stay-Lose-Shift (WSLS), also known as Pavlov: Cooperates if it and the opponents previous move were equal, else it defects [8, 11]
(17) n-Pavlov [7, 8]: Cooperates with probability p_n in round n. The probability in the first round is $p_1 = 1$ and the probability in round $n + 1$ is calculated based on the payoff in round n as:

$$p_{n+1} = \begin{cases} p_n + \frac{1}{n} & \text{payoff in the last round was R} \\ p_n + \frac{2}{n} & \text{payoff in the last round was T} \\ p_n - \frac{1}{n} & \text{payoff in the last round was P} \\ p_n - \frac{2}{n} & \text{payoff in the last round was S} \end{cases}$$

(18) PavlovD: Same behaviour as Pavlov but defects in the first round.
(19) My strategy simple (MSS): Cooperates the first 6 rounds, after that it either randomly cooperates every second or third move or plays as n-Pavlov.
(20) My strategy advanced (MSA): Plays as n-Pavlov. Additionally, if the probability drops below 0.3 it has a chance, equal to that probability, to forgive the opponent and cooperate the next 2 or 3 moves.

Both of our implemented strategies are based on the n-Pavlov strategy, and both exhibit the property of being nice.

4 EXPERIMENTS AND RESULTS

The implemented strategies were evaluated in a tournament similar to the Axelrod's first tournament and in a simulation, as explained in the previous section of this paper. Both evaluation methods were run multiple times, to avoid any bias introduced by strategies leveraging randomness. The impact of the tournament length on the final rankings of strategies was also tested. This section presents the results achieved.

4.1 Results of the implemented tournament

We run two tournaments with lengths 50 and 1000 moves, each 1000 times, and used the average of the scores achieved to rank the strategies. The results are collected in Table 2. In addition, we divided the scores from the long tournament by 20 for easier comparison with the short version scores.

[1] $s = 0.5, p = 0.7, q = 0.3$

Table 2: Ranking of strategies in the tournament evaluation. Rank #1 and score #1 present the results of the longer tournament with 1000 moves played, and the rank #2 and score #2 present the results from the shorter tournament with 50 moves played.

Rank #1	Rank #2	Score #1	Score #2	Name
1.	1.	2530	2532	Grim
2.	2.	2447	2459	TF2T
3.	3.	2414	2418	Pavlov
4.	4.	2407	2415	n-Pavlov
5.	5.	2405	2407	TFT
6.	7.	2317	2319	PavlovD
7.	9.	2314	2277	CCD
8.	6.	2303	2333	MSA
9.	12.	2270	2257	Random
10.	8.	2257	2280	DDC
11.	11.	2246	2258	CD
12.	10.	2230	2274	ALLD
13.	14.	2212	2222	$R(s, p, q)$[1]
14.	15.	2199	2184	$P(0.75)$
15.	13.	2194	2241	ImpTFT
16.	16.	2152	2149	ALLC
17.	17.	2150	2147	MSS
18.	18.	2055	2089	STFT
19.	19.	2022	2051	Prober
20.	20.	1947	1992	2TFT

From the results we can observe, that the winning strategy from Axelrod's tournaments landed only on place 5. The similar Tit for two Tats strategy performed better, placing second. As expected, both the Pavlov and n-Pavlov strategies placed in the top 5. The best score in both the short and long tournament, therefore placing first, was achieved by the Grim strategy. This was a small surprise, but after a more detailed look at the data, the results make sense and are also expected. Grim scores well against nice strategies, which are quite well represented in our tournament, and also takes advantage of strategies that heavily rely on randomness or forgiveness. Forgiving strategies are those that are still prepared to cooperate even when the opponent has defected against them [8]. Grim defects from the next move his opponent defected till the end, but the random and forgiving strategies will sometimes still cooperate, which brings Grim the extra score to place first. One of our strategies named My strategy advanced performed quite well, placing 8th in the short tournament and 6th in the long tournament. In comparison, My strategy simple placed only on the 17th place in both tournaments. While one of them performed rather well, none of them improved the n-Pavlov strategy on which they were based on.

4.2 Results of the proposed simulation

The first scenario we tested was the effect the length of the tournament has on the simulation's behaviour and its final outcome. The initial population size of each strategy was set to 100 members, and we tested tournament lengths of 10 and 200 moves. The performance was measured by the number of generations a specific

strategy survived, before dying out and being completely removed from the simulation. The best performing strategies, as observed from the results, were the same as seen in Table 2 and there was very little difference discovered between the tournament lengths of 10 and 200 moves.

Our experiments continued with testing different initial population sizes for the strategies and its effect on the results. This also affected the total number of strategies playing against each other and therefore the running time of the simulation. We tested initial population sizes of 10 and 100 members for each strategy. The results were almost identical to the ones produced while testing different tournament lengths.

After the initial experiments, we have also done some smaller ones, where we tested the performance of our two strategies in different environments. One of the environments was a TFT based environment, consisting only of the strategies based on the original TFT strategy (TFT, TF2T, 2TFT, ImpTFT and STFT), and another one was a Pavlov based environment, consisting only of (Pavlov, n-Pavlov and PavlovD strategies). Although the scenarios looked quite interesting, they didn't produce any significant results.

4.3 Discussion

An interesting behaviour found in a population of TFT strategies is the ability to invade a population of ALLD strategies. Additionally, a population of TFT strategies can't be invaded by a population of ALLD strategies. This was the first scenario we tested, and the results have shown, that none of our strategies exhibit that behaviour. The same holds true for the n-Pavlov strategy which they are based on.

One thing that all the experiments had in common was, that most of the low performing strategies died out quickly. If the simulations were run for enough generations, we also observed that eventually one strategy took over and no other strategy was left. This led us to a more detailed examination of the scores produced throughout the simulation. We observed that initially the minimum scores start dropping, and after the low performers die out, the minimum scores start increasing, eventually converging with the average and maximum scores. The strategies that drop out early are mostly strategies that don't fall into the nice category. Eventually all the strategies left in the simulation exhibited the property of being nice and therefore the only possible move was cooperation. From this point onwards, the behaviour of the simulation was completely based on the random generator controlling the selection process.

5 CONCLUSION

In this study, we presented two new strategies for playing the iterated prisoner's dilemma game and evaluated them on an implementation similar to Axelrod's original tournament and on our own simulation. The results of the tournament have shown, that one strategy performed quite well, while the other one performed poorly, placing only on the 17th place out of 20. Additionally, none of the two presented strategies outperformed the n-Pavlov strategy which they were based on. Lastly, we observed that most strategies dropped out of our simulation quite early and only a small number of nice strategies were left. This lead to a scenario, where the only played move was cooperation and the final outcomes were only affected by the random generator controlling the selection process.

In the future, two possible improvements could be added to our solution. One would be to increase the number of the implemented strategies, and the second one would be to reduce the imbalance between the number of strategies exhibiting the property of being nice.

REFERENCES

[1] Robert Axelrod. 1980. Effective Choice in the Prisoner's Dilemma. *The Journal of Conflict Resolution* 24, 1 (1980), 3–25. http://www.jstor.org/stable/173932
[2] Robert Axelrod. 1980. More Effective Choice in the Prisoner's Dilemma. *The Journal of Conflict Resolution* 24, 3 (1980), 379–403. http://www.jstor.org/stable/173638
[3] Robert Axelrod. 2012. Launching "The Evolution of Cooperation". *Journal of Theoretical Biology* 299 (2012), 21–24. https://doi.org/10.1016/j.jtbi.2011.04.015 Evolution of Cooperation.
[4] Robert Axelrod et al. 1987. The evolution of strategies in the iterated prisoner's dilemma. *The dynamics of norms* 1 (1987), 1–16.
[5] Robert Axelrod and William D. Hamilton. 1981. The Evolution of Cooperation. *Science* 211, 4489 (1981), 1390–1396. http://www.jstor.org/stable/1685895
[6] R Dawkins. 1976. *The Selfish Gene.* Oxford University Press, Oxford, UK.
[7] David Kraines and Vivian Kraines. 1989. Pavlov and the prisoner's dilemma. *Theory and Decision* 26, 1 (Jan. 1989), 47–79. https://doi.org/10.1007/bf00134056
[8] Steven Kuhn. 2019. Prisoner's Dilemma. In *The Stanford Encyclopedia of Philosophy* (Winter 2019 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.
[9] John Nash. 1951. Non-Cooperative Games. *Annals of Mathematics* 54, 2 (1951), 286–295. http://www.jstor.org/stable/1969529
[10] Abraham Neyman. 1985. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters* 19, 3 (1985), 227–229. https://doi.org/10.1016/0165-1765(85)90026-6
[11] Martin Nowak and Karl Sigmund. 1993. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature* 364, 6432 (jul 1993), 56–58. https://doi.org/10.1038/364056a0
[12] Glenn H. Snyder. 1971. "Prisoner's Dilemma" and "Chicken" Models in International Politics. *International Studies Quarterly* 15, 1 (03 1971), 66–103. https://doi.org/10.2307/3013593 arXiv:https://academic.oup.com/isq/article-pdf/15/1/66/5096260/15-1-66.pdf

Index of Authors