Proceedings of the 10th Student Computing Research Symposium (SCORES'24)

Maribor, Slovenia
October 3, 2024

Niko Lukač
Iztok Fister
Štefan Kohek
(Eds.)

SCORES

https://www.scores.si

University of Maribor Press

University of Maribor

Faculty of Electrical Engineering and Computer Science

Proceedings of the 10th Student Computing Research Symposium (SCORES'24)

Niko Lukač
Iztok Fister
Štefan Kohek
(Eds.)

October, 2024

Title	**Proceedings of the 10th Student Computing Research Symposium (SCORES'24)**
Editors	Niko Lukač (University of Maribor, Faculty of Electrical Engineering and Computer Science) Iztok Fister (University of Maribor, Faculty of Electrical Engineering and Computer Science) Štefan Kohek (University of Maribor, Faculty of Electrical Engineering and Computer Science)
Tehnical editors	Štefan Kohek (University of Maribor, Faculty of Electrical Engineering and Computer Science) Jan Perša (University of Maribor, University Press)
Design	University of Ljubljana, Faculty of Computer and Information Science University of Maribor, Faculty of Electrical Engineering and Computer Science University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies
Graphic material	Sources are own unless otherwise noted. The authors and Lukač, Fister, Kohek (editors), 2024
Conference	**10th Student Computing Research Symposium (SCORES'24)**
Location	University of Maribor Faculty of Electrical Engineering and Computer Science Koroška cesta 46, 2000 Maribor, Slovenia
Date	October 3, 2024
Program Committee	Niko Lukač (University of Maribor) Štefan Kohek (University of Maribor) Iztok Fister (University of Maribor) Jan Popič (University of Maribor) Klemen Berkovič (University of Maribor) Slavko Žitnik (University of Ljubljana) Timotej Knez (University of Ljubljana) Domen Šoberl (University of Primorska) Lucija Brezočnik (University of Maribor) Grega Vrbančič (University of Maribor) Jure Zabkar (University of Ljubljana) Matjaž Krnc (University of Primorska) Andrej Brodnik (University of Ljubljana) Mario Gorenjak (University of Maribor) Jani Dugonik (University of Maribor) Peter Rogelj (University of Primorska) Dušan Fister (University of Maribor) Simon Kolmanič (University of Maribor) Uroš Mlakar (University of Maribor) Damjan Strnad (University of Maribor) Marko Bizjak (University of Maribor)

Published by **University of Maribor**
University Press
Slomškov trg 15, 2000 Maribor, Slovenia
https://press.um.si, zalozba@um.si

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

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

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

Edition 1st

Publication type E-book

Published Maribor, Slovenia, October 2024

ISBN 978-961-286-914-4

DOI https://doi.org/10.18690/um.feri.6.2024

Available at https://press.um.si/index.php/ump/catalog/book/886

Organisers and sponsors:

University of Maribor

University of Maribor
Faculty of Electrical Engineering and Computer Science

Inštitut za računalništvo

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

ACM Slovenija
Slovenska podružnica društva ACM

Univerza na Primorskem
famnit

Price	Free copy
For publisher	Prof. Dr. Zdravko Kačič, Rector of University of Maribor
Attribution	Lukač, N., Fister, I., Kohek, Š. (eds.) (2024). *Proceedings of the 10th Student Computing Research Symposium (SCORES'24).* University of Maribor, University Press. doi: 10.18690/um.feri.6.2024

```
CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004(0.034.2)

STUDENT Computing Research Symposium (2024 ; Maribor)
    Proceedings of the 10th Student Computing Research Symposium [Elektronski vir] :
(SCORES'24) : Maribor, Slovenia, October 3, 2024 / Niko Lukač, Iztok Fister, Štefan Kohek
(eds.). - 1st ed. - E-knjiga. - Maribor : University of Maribor, University Press, 2024

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/886
ISBN 978-961-286-914-4 (Pdf)
doi: 10.18690/um.feri.6.2024
COBISS.SI-ID 212751875
```

Editors' Foreword

In the realm of computer science, where innovation continually reshapes our understanding of technology, the 2024 Student Computing Research Symposium (SCORES 2024) marks an important moment of progress and collaboration. This year, the Faculty of Electrical Engineering and Computer Science at the University of Maribor (UM FERI) leads the organization of SCORES, in partnership with the University of Ljubljana and the University of Primorska. These institutions have came together to provide a platform for undergraduate and graduate students, fostering their contributions to the field. This year, we are also honored to have the program committee extended with renowned international researchers. Their expertise has enriched the conference, ensuring a high standard of academic rigor and a diverse range of perspectives.

SCORES 2024 is dedicated to supporting the next generation of computer science postgraduates, offering them a stage to present their research, exchange ideas, and engage with the challenges that lie ahead. Recent advancements in artificial intelligence and data science have underscored the need for fresh perspectives and new approaches. This year's symposium features a diverse range of research, including advancements in emotion recognition technologies, computational problem-solving, and the application of video analysis in healthcare. The program also explores new methods in skill modeling, decision-making processes, and language analysis in clinical settings. Additionally, it covers innovations in device localization techniques and developments in object detection within digital environments.

As we review the ideas and research at SCORES 2024, we see the beginnings of work that will influence the future of computer science. The ideas and innovations shared here are not just academic exercises; they represent the next steps in the evolution of technology, driven by the vision and dedication of these talented students.

Finally, special thanks to the Institute of Computer Science at UM FERI as the main conference sponsor, and the UM FERI leadership for the hospitality.

Editors: Niko Lukač, Iztok Fister, Štefan Kohek

Conference Program

Plenary Speakers

Iztok Fister

University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Maribor, Slovenia

SCORES'24: History, mission and vision

In this year, the International Student Conference in Computer Science celebrate its first decade. The conference first emerged in 2014 at FERI Maribor and has continue to date. The only interruption, that the conference experienced, was during the Corona crisis in 2020.

The keynote focuses on the history, mission, and the future of the Student Conference that started with the name Student Computer Science Research Conference (StuCoSRec) in 2014, and was renamed in 2022 under the initiative of the then organizer FRI Ljubljana to Student Computing Research Symposium (SCORES).

Right from the start, the primary mission of the conference was to connect the students of the most important Computer Science Faculties in Slovenia (i.e., FERI MB, FRI LJ, and FAMNIT KP) and to foster them in publishing either the results of their seminar or individual research projects publicly. In line with this, the location of the conference was changed each year according to the current organizer. These conferences are also the place for making new acquaintances among students that could remain active throughout their whole life.

In the last three years, the conference experienced a lot of improvements as follow: introducing the keynote speakers and the best paper award, the reviewer process was escalated, while the conference organization went through a radical automation. Also the Heads of the home Faculties have started to treat it as their own property. At the FERI Faculty, the Institute of Computer Science even put itself in the role of the main sponsor of this conference.

When looking into the future, we can observe that the conference is gaining more and more importance in Computer Science, with students being aware of the importance of the flow of knowledge and experience. As a result, this conference, that is free of charge, could bring students the new views on the problems being solved and also open new ways of finding solutions. Therefore, the Steering Committee needs take care of broader internationalization.

Finally, I wish the conference a long life and as smooth a path as possible in obtaining much new and high-quality papers.

Bogdan Filipič

Jožef Stefan Institute,
Ljubljana, Slovenia

Evolutionary Computation: Overview, Trends and Perspectives

Evolutionary computation is a computational intelligence methodology dealing with theoretical studies, design and applications of search and optimization algorithms, known as evolutionary algorithms. These algorithms mimic biological evolution when iteratively searching for solutions to a given problem. They are well-suited for solving black-box optimization problems where no mathematical formulation is available and problem properties are unknown. In this presentation, we first outline different types of evolutionary algorithms and a unified approach at handling them, as well as their advantages and disadvantages. We then illustrate their capabilities with examples of successful applications to challenging real-world problems. Next, we overview current trends in evolutionary computation, including the efforts of the community in moving beyond metaphor-based algorithms, recent approaches to problem characterization aimed at better problem understanding, and machine learning of algorithm performance prediction. We conclude with future perspectives, highlighting the need for further research on understandability and explainability in evolutionary computation, and potential utilization of generative artificial intelligence techniques.

Influence of Graph Characteristics on Solutions of Feedback Arc Set Problem

Ema Leila Grošelj
eg61487@student.uni-lj.si
University of Ljubljana,
Faculty of Computer
and Information Science,
Ljubljana, Slovenia

Tomaž Poljanšek
tp51101@student.uni-lj.si
University of Ljubljana,
Faculty of Computer
and Information Science,
Ljubljana, Slovenia

ABSTRACT

In this article we present Feedback Arc Set problem and how certain graph characteristics impact the results of heuristic algorithms. We then inspect how the most promising characteristic (treewidth) helps in choosing the most appropriate heuristics for our graph.

KEYWORDS

graph, FAS, heuristics, treewidth, random forest classifier

1 INTRODUCTION

In this article, we tackled the *feedback arc set* problem, where the goal is to find the smallest set of directed edges (arcs) in a directed graph such that, when removed, the graph becomes acyclic (directed acyclic graph - DAG). We were interested in determining which characteristics of a graph suggest that a particular heuristic method might perform poorly, providing a solution not close to optimum.

Due to the trivial nature, we were not interested in the impact of graph size and aimed to normalize this effect. We collected graphs from two existing graph datasets. From these, we built a database of their strongly connected components.

The database of components was enriched with characteristics: number of nodes and arcs, graph density, radius and diameter, information on whether the graph is planar or bipartite, node connectivity, transitivity and treewidth. We also added information about distribution of some characteristics computed on individual nodes (e.g., degree, different types of centrality).

2 FEEDBACK ARC SET

First, let us state that from now on, when we say 'graph', we mean a directed and strongly connected graph with n nodes and m arcs.

Definition 2.1. Feedback arc set (FAS) of a graph $G = (V, A)$ is $A^{'} \subseteq A$ such that $G^{'} = (V, A \setminus A^{'})$ is DAG. *MFAS* (minimum FAS) is the smallest possible FAS.

In this article, we aim to approximate MFAS size using heuristics, as FAS problem is one of Karp's 21 famous NP-complete problems [10]. Unfortunately, it also does not have an approximation scheme. If a graph is already acyclic, its FAS is empty.

2.1 Upper Bound

Every arc in MFAS lies in at least one cycle. If an arc does not lie in a cycle, it does not need to be removed from the graph. This would contradict the minimality of MFAS. Thus, to break all cycles, it is sufficient to remove one arc from each cycle. However, the FAS composed of these arcs is not necessarily the MFAS, as removing one arc can break multiple cycles simultaneously, requiring fewer arcs to be removed than there are cycles. Therefore, the number of cycles is an upper bound on the size of MFAS.

However, this bound can be very loose, as shown in [3], where a graph with n nodes and m arcs can have up to 1.433^m cycles, and the number of arcs can be quadratic in the number of nodes, i.e., $m = O\left(n^2\right)$. In addition to being loose, counting the number of cycles is computationally demanding. For example, the algorithm in Python library Networkx [7] requires $O((n + m)(c + 1))$ time steps, where c is the number of cycles. This means number of cycles is potentially exponential to number of vertices so we can only count all cycles for small graphs in a reasonable time.

Another upper bound adequate also for large graphs is represented by the best result of the heuristics. This is much better since it is computed much faster. Thus in this article we take the highest heuristic result as an upper bound.

2.2 Lower Bound

For the lower bound, we can use disjoint cycles in the graph. They provide a lower bound because we need to break all these (disjoint) cycles and due to disjointness, we cannot break two cycles by removing one arc.

Note that not all (exhaustive) sets of disjoint cycles in a graph are equally strong. For instance, consider a graph with arcs $A = \{(1,2), (2,3), (3,2), (3,1), (1,3)\}$. This graph has three cycles and two sets of disjoint cycles: $\{(1,2,3)\}$ and $\{(1,3), (2,3)\}$. The idea is that by removing the cycle $(1,2,3)$ from the graph, we also break all other cycles (exhausting the disjoint cycles).

The set of disjoint cycles we get will depend on the way we search for one cycle within each iteration. If we introduce randomness in selecting the starting node and the order of nodes during the search, we obtain a random algorithm. When running the algorithm multiple times, we only consider the largest set as we aim for a tighter lower bound. In this article we ran search for disjoint cycles 10 times.

3 DATA

3.1 Data Sources

We collected graphs from two sources.

In [8] they used different generation methods to build a collection of large weighted multi-digraphs that included different topologies. Graphs there were nicely divided in groups: *De Bruijn graphs, Delaunay 3D graphs, Kautz graphs, Triangulation graphs, Small world graphs* and *Random graphs*. They derived them from ISPD98 Circuit Benchmark Suite [2].

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.1
ISBN 978-961-286-914-4

A collection of graphs presented in [5] was intended for the problem of *optimum cycle mean and ratio*. It consists of graphs from ISPD98 Circuit Benchmark Suite [2] and random graphs that they generated themselves. In these article we refer to them as *unclassified*.

We read all these graphs, broke them into strongly connected components, as breaking them into such components is usually the first step in heuristics. We wanted to ensure that the sizes of the components (e.g., many small ones and one large one) did not obscure the impact of other interesting characteristics. We then stored the components in *pickle* format for faster re-reading. If a graph contained loops (self-directed arcs), we removed them beforehand and added them to the result at the end, as not all heuristics supported graphs with loops. Sixteen graphs with either more than 5000 nodes or more than 10000 arcs were classified as 'large' graphs and omitted from the study. Thus, the main database contained 11925 graphs.

3.2 Graph Characteristics

For every graph in our database we saved the number of nodes, number of arcs, graph density, planarity, bipartiteness, diameter, radius, node connectivity (minimum number of nodes that need to be removed to disconnect the graph), transitivity (probability that the ends of two arcs that share a common node are themselves connected), and treewidth (see Subsection 3.2.1).

For nodes, we calculated degree, closeness centrality (inverse average shortest path length from the node to all other nodes), betweenness centrality (frequency with which a node is part of the shortest paths between other nodes), degree centrality (the fraction of nodes it is connected to), clustering coefficient (how many triangles a node is part of out of the possible triangles), node centrality (node influence within the network, considering both the number of neighbors and neighbors of neighbors), and PageRank (a rank obtained by the PageRank algorithm measuring the importance of a node). For each graph, we recorded the min, max, median, and interquartile range of these values.

3.2.1 Treewidth. Treewidth represents how close a graph is to being a tree, with trees having a treewidth of 1. It represents the minimum width of the largest component across all tree decompositions of the graph. The treewidth of an undirected graph was extended to directed graphs in [9].

According to [6], for a directed graph G, it holds that

$$directed\_treewidth(G) \leq treewidth(undirected(G)).$$

Equality is achieved when all arcs are bidirectional. Two heuristics, $treewidth\_min\_degree$ and $treewidth\_min\_fill\_in$ implemented in NetworkX library [7], provide an upper bound for the treewidth, which also serves as an upper bound for 'directed treewidth'. The minimum degree heuristic method repeatedly selects and removes the node with the lowest degree, while the minimum fill-in heuristic method selects the node, whose removal minimizes the number of added arcs needed to make its neighborhood a clique.

4 HEURISTICS

We have tested five different heuristics.

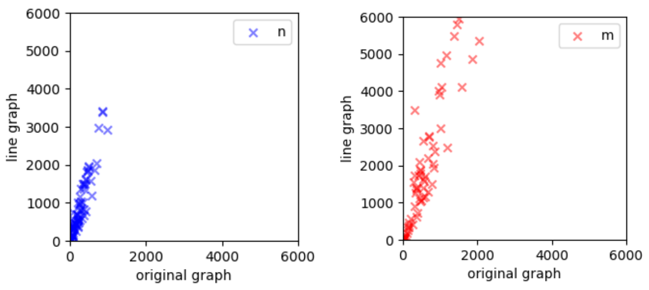


Figure 1: Increase in the number of nodes (left) and arcs (right) in the line graph.

4.1 SmartAE

We implemented the SmartAE algorithm from article [4]. First, we order the nodes, for example by indegree. Then remove outgoing arcs pointing to unvisited nodes in that order until the graph becomes acyclic. After obtaining an acyclic graph, we attempt to re-add removed arcs one by one if they do not cause cycles.

4.2 DiVerSeS - Using FVS Heuristics

The size of the FAS equals the size of the FVS (feedback vertex set, dual problem) on the line graph. The line graph is obtained by mapping arcs to nodes. In the line graph, two nodes representing arcs from original graph (u, v) and (w, x) are connected with arc (uv, wx) if $v = w$, meaning each arc in the line graph represents a directed path of length two in the original graph. This transformation is described in [12]. Line graphs are typically larger, making the problem harder. Increase in size on subsample of grafs from our database is depicted in Figure 1.

After transformation, we ran the winning FVS solver DiVerSeS from the PACE 2022 challenge [1], which dealt with FVS on directed graphs. We gave it 5 seconds to return a solution. If no solution was found, we ran it for 10 and then 40 seconds. Even then some graphs remained unsolved.

4.3 Graph hierarchy based approaches

We ran algorithms from [11]. Goal is to break cycles while still preserving logical structure (hierarchy) as much as possible. Hierarchy information identifies which edges need to be removed. Heuristics differ in a way they determine hierarchy based on different features.

We decided to test 3 approaches: greedy (FAS Greedy), PageRank rating (PageRank SCC) and Bayesian skill rating system (TrueSkill SCC), giving us another 3 heuristics. FAS Greedy was ran 5 times as it uses randomness and is also by far the fastest.

5 METHODOLOGY

We ran heuristics on the dataset and saved size of FAS and running time. [1] For each graph we determined heuristic method that found tha smallest FAS. On a tie, heuristic method with lower running time won.

For determining which graph features are most important in determining which heuristic method is the best to use, we used

[1]Implementations and dataset can be found at https://github.com/elgroselj/FAS.

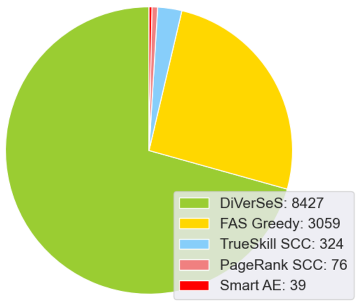


Figure 2: Best solvers.

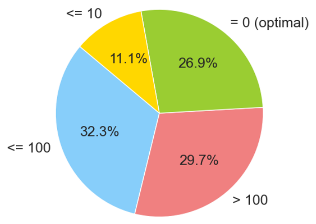


Figure 4: Gap between lower and upper bound.

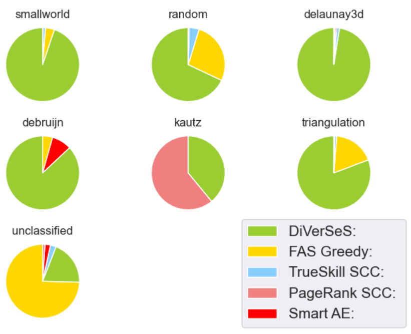


Figure 3: Best solvers by categories.

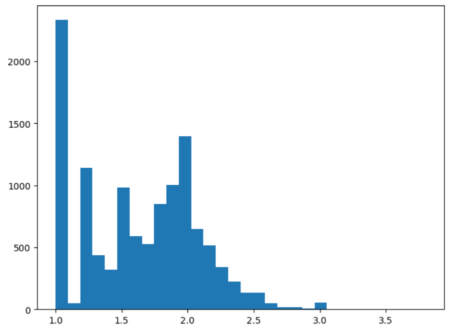


Figure 5: Histogram of the ratios between upper and lower bound.

random forest classifier: it is stable and not to difficult to explain. We trained and evaluated model using 5-fold cross-validation.

Then we evaluated feature importances. Firstly we used model's attribute $feature\_importances$, that represents accumulation of the impurity decrease within each tree. We also tested importance of features using permutation test - that is we randomly permuted values in one column at a time and observed performance degradation.

6 RESULTS

6.1 Heuristics

As shown in Figure 2, the DiVerSeS solver was the best on majority of the graphs. However, for some graph groups other heuristics gave better results as shown in Figure 3. Turns out that on $unclassified$ graphs method FAS Greedy reported the best result. On Kautz graphs we recommend to use the method PageRank SCC, while DiVerSeS method dominated on all other graph groups.

If we closely examine differences between lower and upper bounds in Figure 4 we see that in most cases solutions are well constrained - that is lower and upper bound are relatively close, giving us a narrow interval of possible FAS sizes. We prooved optimality in 26.9% of examples. In Figure 5 we see that in most cases ratios between worst and best solutions is lower than two. We have clipped the graph in Figure 5 to show the great majority of ratios, however there are some individual cases where ratio is quite high (most extreme example has ratio of 17.3). For these examples it is good to know which heuristics works best as it makes a lot of difference.

6.2 Classification and features

In classification with random forest classifier we achieved the accuracy of 0.932. This is significant improvement over the majority classifier (predicts DiVerSeS as the winner for all inputs) with an accuracy of 0.707. Feature importance provided by model's feature_importance attribute is shown in Figure 6, while permutation_importance is shown in Figure 7. Features with very little importance are left out.

We can see that treewidth is the most important characteristic according to both figures. This is not very surprising since with edges removal we create acyclic graph or a tree. Characteristics pagerank_max, number of arcs m and Katz centrality_min also have a significant importance. It is also notable that while number of nodes n has accumulated a lot of impurity decrease according to model's feature importance, it lacks at being innovative in the sense that permuting it randomly does not affect the success much, which suggests that n does not provide new information.

Figure 8 shows us that DiVerSeS generally does the best for graphs with treewidth at least 10 (this is also true for graphs with treewidth ≥ 100). For less than that FAS Greedy heuristic method gives the best results.

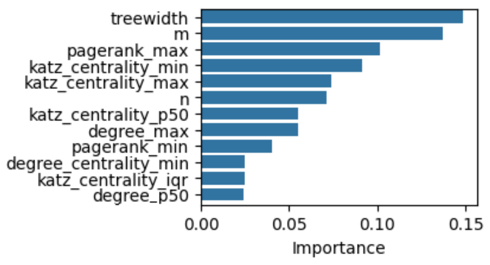


Figure 6: Feature importance.

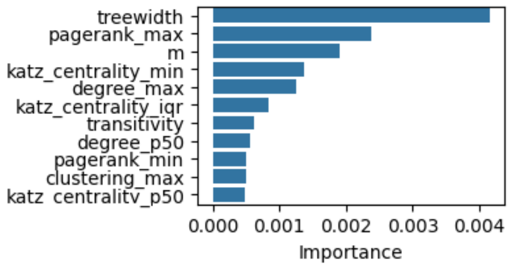


Figure 7: Permutation importance.

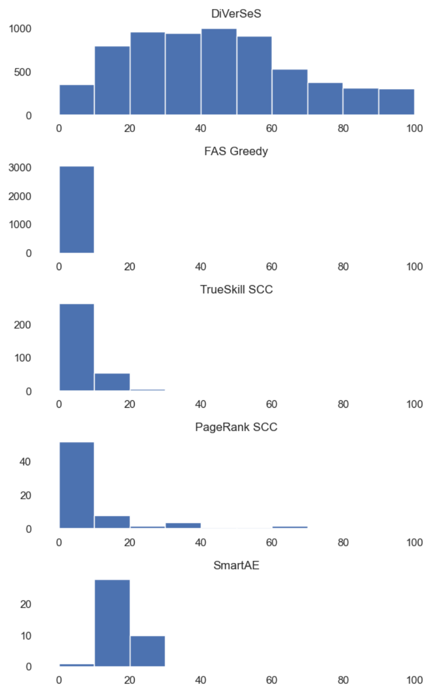


Figure 8: Histograms of treewidths by best solvers.

7 CONCLUSIONS

Treewidth is the most important graph characteristic in determining the best heuristic for graph. Number of arcs and Katz centrality also have significant impact. For graphs with higher treewidth we recommend using DiVerSeS and for lower treewidth FAS Greedy heuristic.

ACKNOWLEDGEMENTS

We sincerely thank assist. prof. dr. Uroš Čibej for his advice, guidance and for introducing us to this topic.

REFERENCES

[1] 2022. PACE2022. https://pacechallenge.org/2022/tracks/. [Accessed 26-05-2024].
[2] Charles J Alpert. 1998. The ISPD98 circuit benchmark suite. In *Proceedings of the 1998 international symposium on Physical design*. 80–85.
[3] Andrii Arman and Sergei Tsaturian. 2017. The maximum number of cycles in a graph with fixed number of edges. *arXiv preprint arXiv:1702.02662* (2017).
[4] C. Cavallaro, V. Cutello, and M. Pavone. 2023. Effective heuristics for finding small minimal feedback arc set even for large graphs. In *CEUR Workshop Proceedings*, Vol. 3606. https://ceur-ws.org/Vol-3606/paper56.pdf
[5] Ali Dasdan. 2004. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems* 9, 4 (2004), 385–418. https://doi.org/10.1145/1027084.1027085
[6] Frank Gurski, Dominique Komander, and Carolin Rehs. 2021. How to compute digraph width measures on directed co-graphs. *Theoretical Computer Science* 855 (2021), 161–185.
[7] 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 (SciPy2008)*, Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11–15.
[8] Michael Hecht, Krzysztof Gonciarz, and Szabolcs Horvát. 2021. Tight localizations of feedback sets. *Journal of Experimental Algorithmics (JEA)* 26 (2021), 1–19.
[9] Thor Johnson, Neil Robertson, Paul D Seymour, and Robin Thomas. 2001. Directed tree-width. *Journal of Combinatorial Theory, Series B* 82, 1 (2001), 138–154.
[10] Richard M Karp. 2010. *Reducibility among combinatorial problems.* Springer.
[11] Jiankai Sun, Deepak Ajwani, Patrick K. Nicholson, Alessandra Sala, Alessandra, and Srinivasan Parthasarathy. 2017. Breaking cycles in noisy hierarchies. In *Proceedings of the 2017 ACM on Web Science Conference*. 151–160.
[12] Jin-Hua Zhao and Hai-Jun Zhou. 2016. Optimal disruption of complex networks. *arXiv preprint arXiv:1605.09257* (2016).

Learning Multi-Level Skill Hierarchies with Graphwave

Simon Bele
sb95099@student.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

Jure Žabkar
jure.zabkar@fri.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

ABSTRACT

We introduce a novel framework for learning multi-level skill hierarchies in reinforcement learning environments by leveraging structural similarities in state-space graphs. To obtain structural embeddings, we use the Graphwave algorithm, which places structurally similar states in close proximity in the latent space. In the latent space, we perform hierarchical clustering of states while respecting the topology of the state-space graph. At different levels of the hierarchy we learn the options that represent the skills; a skill at each level of the hierarchy is defined using the skills from the level below. We compare our approach with the state-of-the-art method across several environments. Our results show that structural embeddings can speed up option learning significantly in certain domains.

KEYWORDS

skill hierarchy, reinforcement learning, options, structural similarity, graph embeddings

1 INTRODUCTION

In Reinforcement Learning (RL), an agent learns to make decisions by interacting with an environment; it operates on the principles of trial and error and obtains positive or negative feedback (rewards) from the environment. The overall goal of the agent is to maximize the cumulative rewards. Traditional RL approaches can struggle with scalability and efficiency as the complexity of the environment increases or the task become increasingly difficult.

A possible way to tackle this challenge is to introduce skill hierarchies in RL [1, 6]. Skill hierarchies enable the decomposition of complex tasks into simpler sub-tasks, usually improving the generalization of learned behaviors across different scenarios. This usually leads to a more efficient learning process but also produces more robust and interpretable actions.

Traditional approaches in developing these hierarchies have primarily focused on single-level structures, where skills are often defined through predefined policies or through the clustering of state transitions without considering the deeper structural relationships between these transitions. Recently, Evans et al. [3] introduced a method for learning skill hierarchies based on Louvain clustering of the state-space graph, which optimizes its modularity.

In this paper, we introduce an approach that goes beyond modularity: we use the Graphwave algorithm that identifies structural similarities within a graph. We cluster structural embeddings in latent space, thus providing a more robust foundation for skill learning. Our approach also preserves the topology of the state graph and so enables us to learn the options framework on the obtained clustering.

We evaluate our method by comparing it to the approach of Evans et al. [3]. We integrate our code into their framework and observe the learning efficiency on four domains. We show that in three out of four, our method performs significantly better while in the Four Rooms domain that features extreme modularity, the approach of Evans et al. outperforms ours.

2 RELATED WORK

A common approach in reinforcement learning involves modeling the underlying Markov Decision Process (MDP), wherein a policy $\pi : S \times A \rightarrow [0,1]$ is learned to maximize a reward function. Specifically, the action-value function $Q^{\pi}(s,a)$ for a policy π encapsulates the expected reward for states in the environment. The action-value function adheres to the Bellman equations, and the task can thus be rephrased as optimizing these equations to find the optimal policy.

The options framework in reinforcement learning is a well-established method for reasoning across multiple levels of temporal abstraction, effectively implementing Semi-Markov Decision Processes [4, 8].

An option is defined by a 3-tuple $\omega = (I_\omega, \pi_\omega, \beta_\omega)$, where $I_\omega \subseteq S$ represents the subset of the state space in which the option is executable, $\pi_\omega : S \times A \rightarrow [0,1]$ is a policy determining the probability of taking action a in state s, and $\beta_\omega : S \rightarrow [0,1]$ specifies the termination condition, indicating the probability of option termination in a given state.

To hierarchically cluster the state space, one can derive higher-level options over lower-level options, where the initiation set of the higher-level option is the union of initiation sets of lower-level options, thereby enabling options at multiple time scales.

Training options across multiple time scales necessitates generalizing the usual Bellman equations to be defined over options rather than actions, termed intra-option learning [7].

The MDP induces a state transition graph, with nodes representing states and edges denoting possible actions between states.

Several approaches leverage the state transition graph of the underlying MDP to learn skills. A notable advancement by Xu et al. [9] employs the Louvain graph clustering algorithm to partition the state transition graph into clusters, subsequently defining options as traversals across the aggregate graph of these clusters.

Previous efforts to create single-level skill hierarchies have primarily utilized different measures of centrality or various graph partitioning algorithms.

Evans et al. [3] introduce a multi-level skill hierarchy trained on the entire hierarchical clustering of Louvain. Due to the intractable problem of modularity maximization, the greedy-natured Louvain

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.2
ISBN 978-961-286-914-4

algorithm optimizes for moving nodes between partitions at each step if and only if this move results in a positive modularity gain. This can be seen as a local approach to modularity optimization. They employ macro-Q learning [5] and intra-option learning [7] to train hierarchical agents.

The above approach is novel in producing the first multi-level skill hierarchy, where it is produced automatically with no human intervention. Through it they obtain options reflecting optimizing for modularity, which they show to be useful for navigating at the top-most level of the skill hierarchy.

3 METHODOLOGY

3.1 Structural similarity embeddings

To obtain an embedding of nodes that places structurally similar nodes in close proximity within the latent space, we employ Graphwave [2], a methodology that provides strict guarantees regarding the separation of structurally equivalent nodes.

Consider an undirected graph $G = (V, E)$ with its graph Laplacian defined as $L = D - A$, where D is the degree matrix and A is the adjacency matrix. Let the eigenvector decomposition of L be given by

$$L = U\Lambda U^T, \tag{1}$$

where U is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

By applying a heat diffusion wavelet $g_s(\lambda) = e^{-\lambda s}$, define the spectral graph wavelet centered at node a as

$$\Psi_a = U\,\mathrm{Diag}(g_s(\lambda_1), \dots, g_s(\lambda_N))U^T \delta_a, \tag{2}$$

where δ_a is the Dirac delta function at node a.

To circumvent computational intractability [2], the wavelet is treated as a probability distribution over the graph:

$$\phi_a(t) = \frac{1}{N} \sum_{m=1}^{N} e^{it\Psi_{ma}}, \tag{3}$$

for time point t. The empirical characteristic function is then sampled and transformed into a vector embedding:

$$\chi_a = [\mathrm{Re}(\phi_a(t_i)), \mathrm{Im}(\phi_a(t_i))]_{t_1,\dots,t_d}, \tag{4}$$

with d being the number of samples.

This resulting $2d$-dimensional embedding ensures that structurally equivalent nodes in the graph will be at most a predefined ϵ distance apart in the ℓ_2 norm, thereby providing rigorous guarantees on the proximity of such nodes [2].

3.2 Clustering

To hierarchically cluster nodes based on their embeddings, our approach utilizes an agglomerative clustering algorithm.

This algorithm iteratively merges the nearest clusters based on the average linkage criterion. To maintain the integrity of the graph's topology, clusters are only compared if there exists a direct path between them that bypasses other clusters. The height of the hierarchy was chosen to match the height of Louvain for the sake of fair comparison between the two approaches [3], but could also be defined through any dendrogram cutting strategy.

3.3 Option learning

For the sake of comparisons with Evans et al. [3], we similarly construct the skill hierarchy as follows.

Let h represent the number of partitions produced by our algorithm when applied to the state transition graph. Each of these h partitions defines a skill layer, forming an action hierarchy with h levels of abstract actions above primitive actions. Each hierarchy level consists of skills designed to efficiently navigate between clusters of the state transition graph.

We define options for moving from cluster c_i to cluster c_j is defined by: initiation states in c_i, a policy to navigate from any state in c_i to a state in c_j, and termination upon reaching c_j.

Leveraging the hierarchical structure of the partitions, we define skills at each level of the hierarchy using the skills from the preceding level. At each hierarchy level, the policies for higher-level actions call actions (either options or primitive actions) from the level below, with primitive actions only invoked directly at the base level.

4 EVALUATION

4.1 Domains

The skill hierarchy is evaluated in four environments (Figure 1), the first three of which are different examples of the rooms environment. An empty room, two rooms connected by a bottleneck and four rooms as in [3]. The agent is given a starting position and a goal position and attempts to navigate between them as effectively as possible. The last domain we look at is the Towers of Hanoi, a classic mathematical puzzle. It involves moving a set of disks from one peg to another, following specific rules.

Each of the rooms environments feature four basic movements: north, south, east, and west. These movements steer the agent in the chosen direction unless obstructed by a wall, in which case the agent stays in place. Each action incurs a penalty of -0.001, with a bonus of +1.0 awarded upon reaching the goal state. Each run begins from a designated start state and aims for a goal state.

The Towers of Hanoi involves four disks of varying sizes positioned on three pegs. An episode commences with all disks stacked on the leftmost peg. Actions involve moving the top disk from one peg to another, ensuring no larger disk is placed on a smaller one. Each action incurs a -0.001 penalty, with an additional +1.0 reward granted upon achieving the goal state, which is when all disks are stacked on the rightmost peg.

4.2 Structural Skill Hierarchy

The hierarchy obtained through the above clustering method will cluster structurally similar nodes together.

To showcase an example, we show the hierarchical clustering of Two Rooms (Figure 2), at the lowest level the walls of the room as well as the bottleneck state are clustered together. The corners of the room are given their own individual clusters and then the center of the room is partitioned symmetrically with respect to the bottleneck state. The second level then merges most of the interiors of the individual rooms while still giving the corner states their individual clusters. The final level then merges the corner states into the walls of the room and gives three clusters which are the

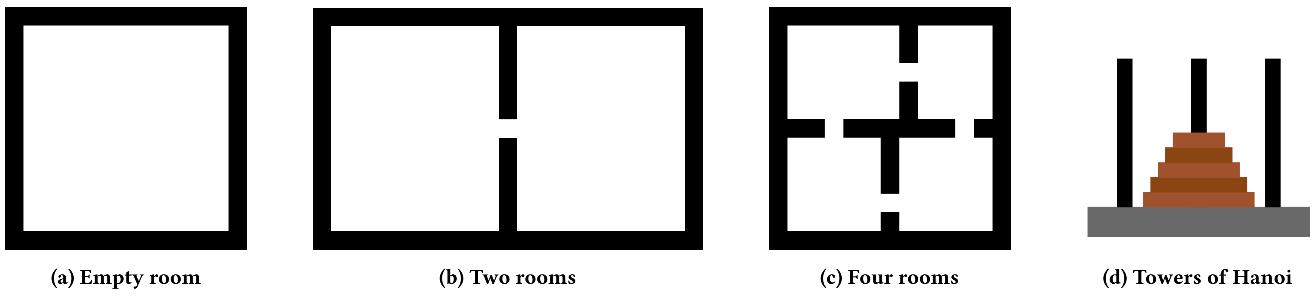


Figure 1: The environments used in the experiments. (a) Empty Room: A simple environment with no obstacles. (b) Two Rooms: An environment divided into two connected rooms. (c) Four Rooms: A more complex environment divided into four connected rooms. (d) Towers of Hanoi: A classic puzzle environment where the goal is to move disks between pegs according to specific rules.

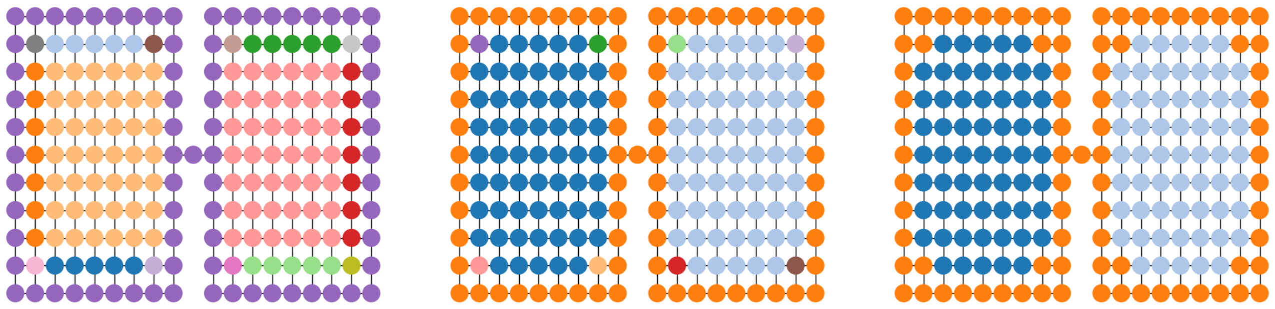


Figure 2: Hierarchical clustering of the Two Rooms environment at various levels. The lowest level (left-most image) clusters walls and bottleneck states, the second level (middle image) merges room interiors while keeping corners separate, and the final level (right-most image) combines corners into walls, resulting in three main clusters.

two interiors of the rooms and the final cluster essentially contains these two rooms.

4.3 Results

To compare with Evans et al. [3], in the analysis, we created options by building the full state transition graph and then learned their policies offline using macro-Q learning [5].

We trained all hierarchical agents with macro-Q learning and intra-option learning [7]. Shaded areas in the learning curves show the standard error, based on 40 independent trials.

The parameters set were the same as in [3], a learning rate of $\alpha = 0.4$, a discount factor of $\gamma = 1$, and initial action values of $Q_0 = 0$. An ϵ-greedy strategy with $\epsilon = 0.1$ was used for exploration. The shown learning curves represent evaluation performance. Post each training epoch, the policy was assessed (with exploration and learning disabled) in a separate environment instance.

We observe (Figure 3) that on the domain of Empty Room we quickly converge to a rewarding strategy before eventually seeing the Louvain skill hierarchy catch up. In the Two Room environment we observe much faster convergence as well as Louvain starting to catch up relatively slowly. The Four Rooms domain favours the Louvain skill hierarchy clearly, one might deduce this due to it being more important to traverse between rooms quickly which is optimally done through a clustering that relies on modularity. We outperform the Louvain skill hierarchy in the Towers of Hanoi, converging faster and having it catch up.

5 CONCLUSIONS

In this paper, we introduced a novel approach to hierarchical skill learning by leveraging Graphwave to obtain structural embeddings of the states. By clustering the states and preserving the topology of the state-space graph, we enabled efficient option learning, where options represent skills at various levels of abstraction. In our experiments, we compared the proposed approach to the state-of-the-art method by Evans et al. [3] and showed that our method can speed up the learning process significantly in some cases.

However, in some domains, optimizing for modularity obviously yields better skill hierarchies and faster option learning. It remains an open question for future research to determine which properties of a domain's state-space graph are more suited for each method. This question is related to another open challenge, namely the characterizations of a useful skill: for a given complex task, what defines a proper skill hierarchy.

Future work could also explore incrementally building the state-space graph and deriving the optimal skill hierarchy for the partially observed graph. This approach may influence how confidently the

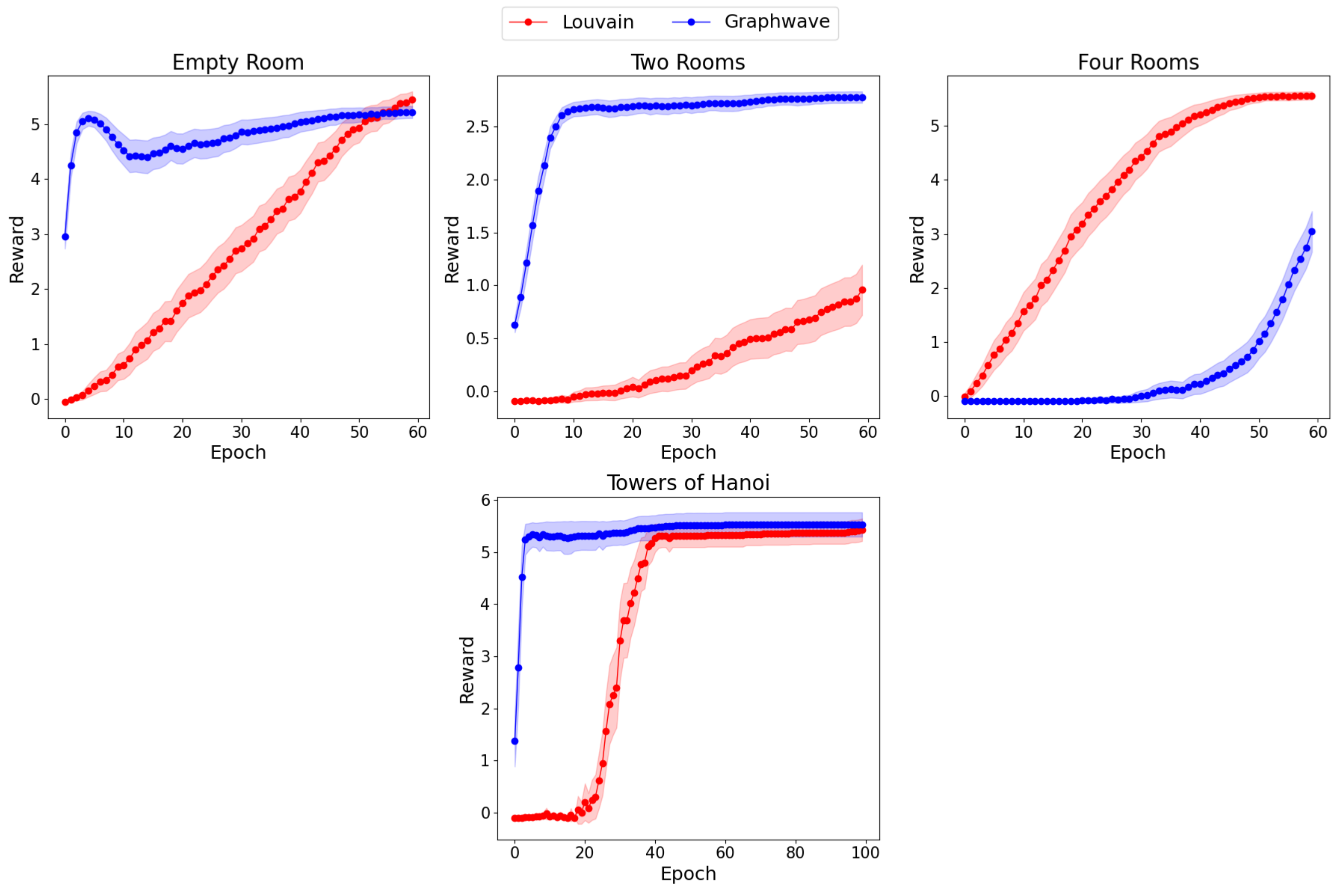


Figure 3: The following figure illustrates the performance of hierarchical agents using Louvain and Graphwave skill hierarchies in different environments: Empty Room, Two Rooms, Four Rooms, and Towers of Hanoi. We observe that in the Empty Room environment, both skill hierarchies converge quickly to a rewarding strategy, with Graphwave performing better initially but Louvain catching up over time. In the Two Rooms environment, Graphwave converges significantly faster than Louvain. The Four Rooms domain favors the Louvain skill hierarchy, likely due to the importance of quickly traversing between rooms using a clustering that relies on modularity. In the Towers of Hanoi, Graphwave outperforms Louvain, showing faster convergence and maintaining an advantage throughout. The provided plots show the reward progression over epochs for each environment, highlighting the differences in performance and convergence rates between the Louvain and Graphwave skill hierarchies.

partitioning is constructed over time, as new information becomes available and the graph evolves. One can develop algorithms that dynamically adjust the skill hierarchy based on the current state of the graph, ensuring that the hierarchy remains optimal as the environment changes. One may also pursue a similar direction in constructing skill hierarchies in problems that involve continuous state-spaces.

REFERENCES

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://doi.org/10.1609/aaai.v31i1.10916
[2] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings Via Diffusion Wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1320–1329. https://doi.org/10.1145/3219819.3220025 arXiv:1710.10321 [cs, stat]
[3] Joshua B. Evans and Özgür Şimşek. 2024. Creating Multi-Level Skill Hierarchies in Reinforcement Learning. arXiv:2306.09980 [cs]
[4] Marlos Machado, Andre Barreto, and Doina Precup. 2021. Temporal Abstraction in Reinforcement Learning with the Successor Representation.
[5] Amy Mcgovern, Richard Sutton, and Andrew Fagg. 1999. Roles of Macro-Actions in Accelerating Reinforcement Learning. (Feb. 1999).
[6] Matthew Riemer, Miao Liu, and Gerald Tesauro. 2018. Learning Abstract Options. *CoRR* abs/1810.11583 (2018). arXiv:1810.11583 http://arxiv.org/abs/1810.11583
[7] Richard S Sutton, Doina Precup, and Satinder Singh. 1998. Intra-Option Learning about Temporally Abstract Actions.. In *ICML*, Vol. 98. 556–564.
[8] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112, 1 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1
[9] Xiao Xu, Mei Yang, and Ge Li. 2018. Constructing Temporally Extended Actions through Incremental Community Detection. *Computational Intelligence and Neuroscience* 2018, 1 (2018), 2085721. https://doi.org/10.1155/2018/2085721

Integration of Named Entity Extraction Based on Deep Learning for Neo4j Graph Database

Lea Roj
l.roj@um.si
University of Maribor,
Faculty of Electrical Engineering and
Computer Science,
Maribor, Slovenia

Štefan Kohek
stefan.kohek@um.si
University of Maribor,
Faculty of Electrical Engineering and
Computer Science,
Maribor, Slovenia

Aleksander Pur
pur.aleksander@gmail.com
Ministry of the Interior,
Ljubljana, Slovenia

Niko Lukač
niko.lukac@um.si
University of Maribor,
Faculty of Electrical Engineering and
Computer Science,
Maribor, Slovenia

Abstract

The increase in unstructured textual data has created a pressing demand for effective information extraction techniques. This paper explores the integration of Named Entity Extraction (NEE) using deep learning within the Neo4j graph database. Utilizing the Rebel Large Model, we converted raw text into structured knowledge graphs. The primary objective is to evaluate the efficacy of this integration by examining performance metrics, such as processing time, graph growth, and entity representation. The findings highlight how the structure and complexity of graphs vary with different text lengths, offering insights into the potential of combining deep learning-based NEE with graph databases for improved data analysis and decision-making.

Keywords

Named entity extraction, deep learning, Neo4j, graph database, knowledge graphs

1 Introduction

The rise of digital news and social media has significantly increased the importance of NEE. As more information is generated online, extracting this information became critical for various applications, such as search engines and recommendation systems [1]. NEE, along with Relation Extraction (RE), is essential for transforming unstructured text into structured data, enabling more effective data analysis and decision-making. Building on the findings of a previous work [2] that evaluated various hyper-parameters and analyzed sensitivity performance, this paper takes a step further by exploring the integration of NEE and RE within the Neo4j graph database.

Neo4j is a graph database that provides a powerful way to store and query complex relationships between entities. This makes it well-suited for applications involving interconnected data, such as social networks, recommendation systems, and fraud detection. In Neo4j, data is stored as nodes and relationships. Nodes represent entities, while relationships represent the connections between these entities. Both can have properties (key-value pairs) to store additional information. Neo4j uses Cypher, a declarative query language specifically designed for querying graph databases [3].

Integrating Named Entity Recognition (NER) based on deep learning within Neo4j graph databases has been an area of active research and development. Ni et al. [4] addresses the challenge of translating natural language queries into graph database queries for intelligent medical consultation systems. The authors developed a Text-to-GraphQL model that utilizes a language model with a pre-trained Adapter, enhancing the semantic parsing capabilities by linking GraphQL schemas with corresponding natural language utterances.

Fan et al. [5] wrote about geological hazards, a deep learning-based NER model that was used to construct a knowledge graph from literature. This model addresses challenges such as diverse entity forms, semantic ambiguity, and contextual uncertainty. The resulting knowledge graph, stored in Neo4j, enhances the usability of geological research data.

Chaudhary et al. [6] propose a system that converts raw text into a knowledge graph using Neo4j, addressing inefficiencies in traditional tools like Spacy, NLTK, and Flair. Their method combines entity linkage and RE to convert unstructured data into a knowledge graph, leveraging graph-based NER and Linking for a contextual understanding of data. The implementation utilizes the REBEL [7] model for RE. In comparison with our approach, they use the BLINK [8] model for entity disambiguation and linking. Meanwhile we focus on efficient entity normalization by querying Wikipedia. Furthermore, while Chaudhary et al.'s system emphasizes improvements for processing large untagged datasets using graph-based NER and linking, we achieve comparable results using traditional NER through Spacy while focusing on entity filtration and knowledge enrichment. We also performed several graph analytics in Neo4j, providing deeper insights and analysis.

The objective of this paper is to demonstrate the implementation process of integrating NEE into the Neo4j graph database. It aims to evaluate the effectiveness of this integration and analyze various performance metrics. Specifically, the paper will measure the processing time required to extract named entities from text and represent them in Neo4j, analyze graph growth in relation to

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.3
ISBN 978-961-286-914-4

text length, evaluate average total neighbors score based on text length, and analyze how many entities are actually shown in graph and how many are filtered out.

The next section details the workflow from text pre-processing to graph visualization in Neo4j. The Results showcases the findings, including charts that visualize the performance metrics. Finally, the Conclusion summarizes the benefits and purpose of the integration, highlights key findings from the study, and suggests potential areas for future research and development.

2 Methodology

The workflow from text pre-processing till graph construction in Neo4j is represented in figure 1. The entire process consists of multiple crucial steps including text pre-processing, NEE, RE, entity normalization and filtration, and finally generation and visualization of the knowledge graph in Neo4j. The details of these steps have been in depth discussed in our previous paper [2].

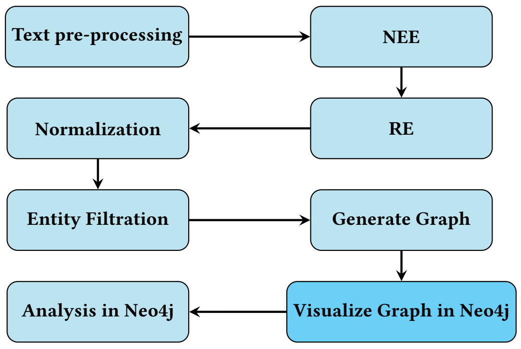


Figure 1: Workflow

Text pre-processing involves segmenting the text into manageable spans, with span length defining the number of words in each segment and overlap length ensuring coherence between consecutive spans. The length penalty manages the impact of longer sequences, while the number of beams allows simultaneous exploration of multiple sequences to find the best one. The number of returns specifies how many sequences are returned after the beam search.

NEE identifies and classifies entities by tokenizing the text, with each token corresponding to a unique word ID. The Rebel Large Model, a sequence-to-sequence model based on the T5 architecture, is employed for RE tasks [7]. This model leverages deep learning techniques to process up to 512 tokens as input and generates triplets consisting of a subject (head), object (tail), and the relationship type between them. Using a transformer-based encoder-decoder architecture, these triplets are extracted from textual spans, where each relationship is first predicted in token form and then decoded into text.

Based on the previous paper, this paper proposes improvements in normalization and entity filtration, as follows. Entity names are first standardized by converting text to lowercase and removing common prefixes, followed by verification via Wikipedia's API. Non-contributive entities, such as dates or overly generic terms, are identified and excluded using pattern recognition and categorization techniques. The system checks for duplicates or highly similar entities to prevent clutter, merging or discarding them as needed. Cosine similarity measures are used to assess and reinforce thematic links between entities, enhancing the overall coherence of the knowledge base.

2.1 Knowledge graph within Neo4j

A knowledge graph is generated from the extracted and filtered entities, and relations. This structured representation helps in visualizing the connections and relationships within the text. Finally, the knowledge graph is stored and visualized in Neo4j.

In the integration process, the data obtained using NER is saved to a graph database through the Neo4j driver. Afterwards, the method iterates over entities in the knowledge base to determine category for each entity. Using the 'MERGE' Cypher command the method either finds an existing node (based on the name) or creates a new node if none exists. Attributes such as 'url,' 'summary,' and 'category' are then added to each node.

```
MERGE (e:Entity {name: $entity})
ON CREATE SET e.url = $url, e.summary = $summary
SET e.category = coalesce(e.category, $category)
```

After adding the entities, we processed each relationship defined in the knowledge base. We ensured that both entities involved in the relationship were present in the database and then created a relationship between the entities using the 'MERGE' command if it didn't already exist.

```
MATCH  (head:Entity {name: 'EntityName1'}),
(tail:Entity {name: 'EntityName2'})
MERGE (head)-[r:RELATIONSHIP_TYPE]->(tail)
```

The knowledge graph is visualized in Neo4j to provide an intuitive and interactive representation of the extracted knowledge. Using Cypher queries, users can explore the graph, examine relationships, and derive insights from the interconnected data. To display the graph in the Neo4j application, we use the following Cypher query.

```
MATCH (n)-[r]->(m) RETURN n, r, m
```

This query retrieves all nodes (n, m) and the relationships (r) between them, displaying the graph structure in the Neo4j interface. The directed edges in the graph illustrate the relationships, providing a clear visual representation of the underlying knowledge.

2.2 Knowledge Graph Analysis in Neo4j

After constructing the knowledge graph in Neo4j, various metrics and analyses were applied to explore the structure within the graph. One such metric is the Total Neighbors score, which measures the closeness of nodes by counting their unique neighbors. It is based on the idea that a highly connected node is more likely to gain new links. The metric is calculated using the following formula:

$$TN(x,y) = |N(x) \cup N(y)|, \tag{1}$$

where N(x) and N(y) represent the sets of nodes adjacent to x and y, respectively. The Total Neighbors score measures the closeness of two nodes based on the number of unique neighbors they have. If a score is equal to 0 it indicates no closeness between the nodes, while higher scores indicate greater closeness [9].

The `gds.alpha.linkprediction.totalNeighbors` function from the Neo4j Graph Data Science (GDS) library calculates the total neighbors score between the two matched nodes (p1 and p2).

3 Results

The analysis was conducted using the text about Pablo Escobar from Wikipedia. For this paper, the original text was divided into sections of varying lengths to examine how text length influences the analysis results. Figure 2 displays a generated graph with a text length of 304 words. The graph was created using specific parameters that influence its structure and content. These parameters were heuristically determined to be span length = 30, length penalty = 0, number of beams = 5, number of returns = 2, and overlap length = 10. On the same set of parameters we measured processing time, similarity score based on total neighbors, and analyzed graph growth.

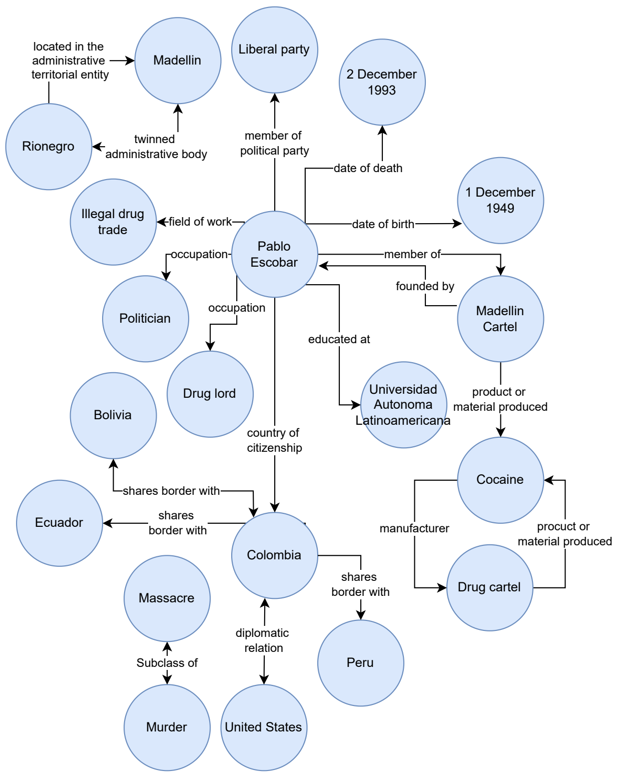


Figure 2: Generated graph with text length = 304 words

In Figure 3, the time required for graph generation is shown to increase linearly with the number of words in the text. This linearity is confirmed by a regression analysis, which yields an R^2 value of 0.9984, indicating an almost perfect fit.

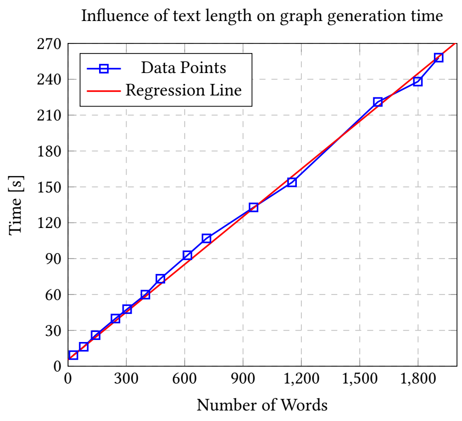


Figure 3: Influence of text length on execution time.

Figure 4 demonstrates the influence of text length on the number of recognized entities. As the number of words in the text increases, there is a corresponding increase in both the number of nodes shown and the number of entities that are recognized but not displayed. This pattern indicates that longer texts result in the recognition of more entities, although not all are displayed. The decision to display or exclude entities is determined by several processes designed to maintain the clarity and relevance of the graph. These processes include the combination and unification of similar entities, the removal of isolated entities, and the filtering out of date-related entities. These processes are essential for maintaining the graph's relevance and clarity, preventing clutter from redundant or less significant entities.

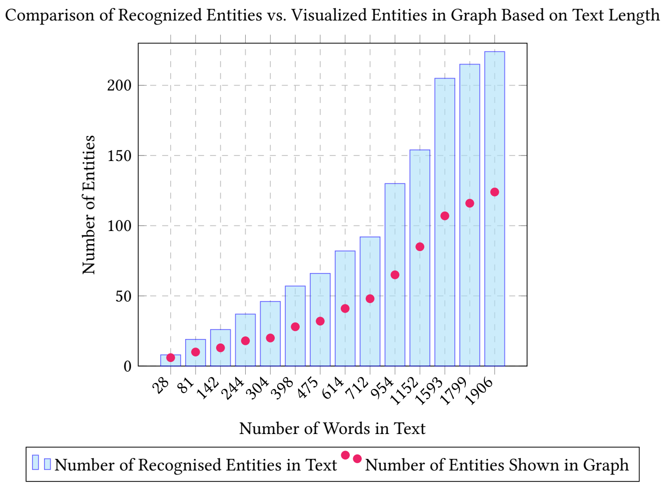


Figure 4: Comparison of recognized and visualized entities based on text length.

The following analyses represent the average similarity score for all possible pairs of entities (n1, n2) in the graph. For that is used a link prediction algorithm (totalNeighbors) to assess how connected two entities are based on their neighbors. The purpose of this is to measure how interconnected the entities are throughout the entire graph. High scores generally appear between nodes directly related through historical, contextual, or thematic associations. On the other hand dates provide low similarity scores with entities, likely indicating less direct connection or relevance to these specific dates in the dataset.

In Figure 5, the average Total Neighbors score between all nodes is represented. The values are relatively stable, mostly ranging between 1.5 and 2.0. This suggests a moderate level of similarity between entities across different text lengths, without extreme variation. This stability suggests that the entities within each text maintain a consistent level of connectivity, regardless of text length.

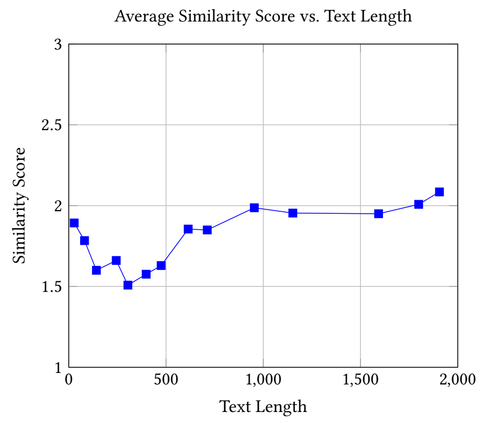


Figure 5: Average Total Neighbors Score between all nodes

Figure 6 highlights only ten strongest connections in the graph, which is particularly useful for identifying the most significant or central entities. Score increases with text length, particularly noticeable in texts longer than 900 words. This indicates that longer texts tend to have more instances of highly interconnected nodes. This is due to the increased probability of recurring entities in longer texts, which leads to more common neighbors. The text with the shortest length (28 words) has the lowest similarity score (3.2). This suggest that very short texts lack sufficient content to establish strong connections between entities. The highest scores for both average and top ten similarities occur in the longest texts (1593, 1799, 1906 words). This supports the idea that more extensive content provides more opportunities for entities to connect or relate.

4 Conclusion

The results demonstrate that text length significantly impacts the performance and outcomes of NEE within Neo4j using deep learning techniques. As text length increases, so does the processing time for graph visualization, due to the need to extract and manage a larger number of entities and relationships. Moreover, the analysis of graph structure revealed that longer texts tend to produce more nodes, both displayed and recognized but not shown. This suggests that while longer texts provide more data, they also introduce challenges in managing graph complexity, which complicates graph management and requires the consolidation of similar entities and filtering of less relevant ones to maintain clarity. Furthermore the

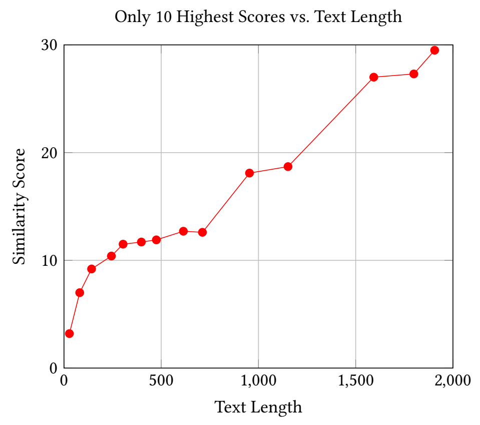


Figure 6: Average of ten highest Total Neighbors Scores

stability of the average similarity scores across various text lengths suggests a consistent level of connectivity among entities, with a noticeable increase in the strength of connections in longer texts. This supports the hypothesis that longer texts offer more opportunities for entity interconnections, which is crucial for tasks requiring comprehensive data analysis and decision-making.

In conclusion, integrating NEE with graph databases presents a promising approach for transforming unstructured text into structured knowledge. However, the complexity introduced by varying text lengths must be carefully managed to optimize both the performance and the utility of the resulting knowledge graphs. Future work could focus on exploring other NEE methods to further enhance the efficacy of this integration.

Acknowledgments

The authors acknowledge the support of the STALITA project, financed by Ministry of the Interior, Slovenia.

References

[1] Ing Michal Konkol. Named entity recognition. *Pilsen: PhD thesis, University of West Bohemia*, 2015.
[2] Lea Roj, Štefan Kohek, Aleksander Pur, and Niko Lukač. Sensitivity analysis of named entity extraction based on deep learning.
[3] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5), sep 2017.
[4] Pin Ni, Ramin Okhrati, Steven Guan, and Victor Chang. Knowledge graph and deep learning-based text-to-graphql model for intelligent medical consultation chatbot. *Information Systems Frontiers*, 26(1):137–156, 2024.
[5] Runyu Fan, Lizhe Wang, Jining Yan, Weijing Song, Yingqian Zhu, and Xiaodao Chen. Deep learning-based named entity recognition and knowledge graph construction for geological hazards. *ISPRS International Journal of Geo-Information*, 9(1), 2020.
[6] Shikha Chaudhary, Hirenkumar Vyas, Naveen Arora, and Sejal D'Mello. Graph-based named entity information retrieval from news articles using neo4j. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 320–324, 2024.
[7] Pere-Lluís Huguet Cabot and Roberto Navigli. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, 2021.
[8] Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. Zero-shot entity linking with dense entity retrieval. In *EMNLP*, 2020.
[9] Neo4j. *Total Neighbors Algorithm*, 2024. Accessed: 2024-06-25.

Proc. of the 10th Student Computing Research Symposium (SCORES'24), Maribor, Slovenia, October 3, 2024

Efficient Implementation of Spreadsheet User Application

Tjaša Repič
tjasa.repic@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Aljaž Jeromel
aljaz.jeromel@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Sašo Piskar
saso.piskar@dewesoft.com
DEWESoft d.o.o.,
Trbovlje, Slovenia

Domen Dolanc
domen.dolanc@dewesoft.com
DEWESoft d.o.o.,
Trbovlje, Slovenia

Niko Lukač
niko.lukac@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

Processing measurement data is fundamental in the field of high-tech instrumentation, where precision, collection, analysis, and visualization of data are of great importance. Extensive amounts of data ought to be displayed and processed to ensure smooth user experience. Tabular displays are therefore common, being more comprehensible for the average user. In this paper we propose a solution, envisioned by the company Dewesoft - a spreadsheet editor widget tailored for their data acquisition software DewesoftX, also compatible with separate plugins within the software. Since using commercially widespread tools to do so often results in setbacks when seeking to integrate those within existing software, we've developed an application functionally comparable to other solutions while complying with the company's existing software standards.

KEYWORDS

spreadsheet, tabular data, optimisation, user experience, data visualisation.

1 INTRODUCTION

We widely adopt spreadsheets for their familiarity and versatility, offering extensive features for data manipulation, statistical calculations, data collection, and visualization. Their accessibility makes them a preferred choice for both individuals and businesses. However, spreadsheets also have notable drawbacks. They are susceptible to various input errors, including clerical mistakes, rule violations, data-entry errors, and formula errors, which can significantly distort the data. Additionally, spreadsheets are not inherently designed for efficient data storage or seamless connectivity to relational databases, posing challenges in effective data management and retention [1, 2].

In modern spreadsheet tools, providing a clear and efficient user experience involves several essential elements. These include an intuitive interface with a clean layout, consistent design, tool tips, and help guides. User-friendly navigation is achieved through a well-organized toolbar, robust search functionality, and keyboard shortcuts. Data visualization and formatting are enhanced by features like conditional formatting, and predefined styles. Comprehensive formula support includes auto-complete, error-checking tools, and a rich library of functions. Robust data management capabilities are also crucial, including import/export options, data validation, and integration with other tools [3, 4].

Within the initial design phase of the proposed Spreadsheet plugin solution, a crucial aspect of planning involved acquiring a deeper understanding of the DewesoftX software for which the plugin development was intended [5]. Processing measurement data is a crucial aspect of the advanced test and measurement industry, where the company Dewesoft operates [6]. A key component of Dewesoft's offering is the DewesoftX software. Many fields in science, commerce and the like require precise measurement, data collection, analysis and visualization. Handling such large volumes of data can prove challenging and inefficient, reducing productivity, convenience and increasing the risk of making mistakes. The software in question has been designed specifically to solve this issue, being used across multiple industrial and commercial sectors. It supports a wide range of interfaces for data visualization, allowing for the synchronized acquisition of data from nearly any analog sensor, storage, and visualization within the same file.

When discussing tools designed to display tabular data, it is essential to consider mathability. Mathability in spreadsheet tools refers to their capacity to perform complex mathematical operations with efficiency and accuracy. This capability is crucial as it ensures precise calculations, boosts productivity, and accommodates various applications across fields such as finance, engineering, and data science [7].

Our solution enhances data handling and provides clearer visualization, prioritizing environments where precise, synchronized data acquisition is critical, such as the software itself, DewesoftX. It is meant to integrate the spreadsheet tool within the software, therefore offering more advanced data handling, synchronization and visualization capabilities tailored to the user's needs while avoiding potential issues with safety, space and integration that arise from using already existing tools. This paper presents its basic functions and the thought process behind their implementation, providing detailed explanations, as well as the results of duration and memory usage of various supported functionalities.

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.4
ISBN 978-961-286-914-4

2 METHODOLOGY

The purpose of the following subsections is to provide a breakdown, as well as a thorough explanation of the implementation process of the spreadsheet user application.

2.1 Fundamental Features

2.1.1 Spreadsheet Widget Layout. To enhance data accessibility during development and make the layout overview clearer, the widget workspace was divided into three parts. The visual widget is segmented into two primary regions. The first is the context menu, containing various button shortcuts for features that will be discussed further in this article. The software's user interface was originally developed in Delphi, using its own VCL (Visual Component Library) [8]. VCL is built on the Win32 architecture, sharing a similar structure but offering much simpler usage [9].

The second region, referred to as the spreadsheet rectangle or "TableRect", encompasses all data and its layout within the widget, including information about cells, columns, rows and the spreadsheet title. A smaller section called the data rectangle or "DataRect" additionally handles cell information. The context menu and the spreadsheet workspace can be seen on Figure 1.

We also provide users with the option to save data for future use. This is done by writing cell values, styles, merged cell information, resized columns and manually adjusted titles to a custom DewesoftX file format for saving the workspace which will further be referred to as .DXS or setup file. The data can then be read from the setup file after loading the workspace at a later time.

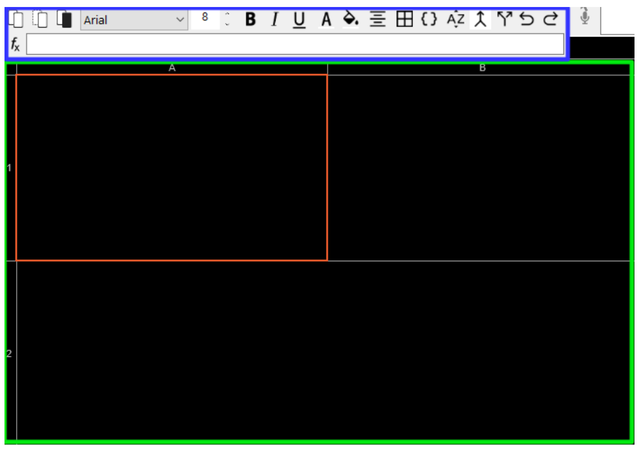


Figure 1: Separation between the context menu (blue) and area within which the spreadsheets's data is displayed (green).

2.1.2 Cell Manipulation. Initially we have made an effort of defining essential functionalities that define the main purpose and value of the application by dismantling related spreadsheet editors [10–12]. The most fundamental feature of the spreadsheet grid is the ability to insert and update data within the cells.

In order to grant each cell its own unique value we implemented a hash function, which generates a hash value from two integer inputs. These are determined by the cell's position within the grid, with the x value corresponding to the column and the y value corresponding to the row. The row index masks the lower 16 bits of the integer and shifts them 16 bits to the left. The column index is masked in the same way and remains unshifted. Using the bitwise OR operation, the two 16-bit values are combined into a single 32-bit integer. Implementation of a hash map designed to store the 32-bit value grants us a key for each cell, allowing us to insert and update data within the cell's index by combining characters received through user input into coherent values.

Where y is the row index, x is the column index, & represents the bitwise AND operation, << represents the bitwise left shift operation, | represents the bitwise OR operation, and $0xFFFF$ is a hexadecimal constant representing the lower bits, the hash key formula can be expressed as:

$$hash\_key = ((y\&0xFFFF) \ll 16) \mid (x\&0xFFFF) \qquad (1)$$

The hashmap provides $O(1)$ time complexity for retrieving data from the selected cell, which is highly desirable, as it means that the time required to perform an operation is constant and does not depend on the size of the data set.

The Spreadsheet widget also supports cell splitting and merging. Selected cells can be merged into a larger cell, which then behaves like a standard cell. To facilitate this functionality, cells include an additional parameter, "merged to", which records the cell to which they are merged. By default, for cells that are not merged, this parameter is set to -1.

A dedicated class manages cell selection within the spreadsheet, defining it by specifying the starting and ending column and row indexes. This enables users to efficiently apply operations to a range of cells, rather than being limited to individual cells.

2.2 Spreadsheet Formatting

Allowing users to stylize components in a user application is important for several reasons, including enabling customization to tailor the application's appearance to the user's individual preferences and allowing them to highlight important information and organize content according to their needs. For this purpose, we have incorporated various styling features into the spreadsheet user application.

2.2.1 Spreadsheet Stylizing. To enhance customization, we introduced a structure within the Style class containing eighteen properties applicable to all spreadsheet components, including cells, selections, rows, and columns. Sixteen of these properties handle styling aspects such as font family, size, color, cell background, bold, italic, underline, border properties, and text alignment. Only user-defined values are saved, optimizing file size and reading/writing speeds. The remaining two properties include a property mask, assigning a binary value to each style feature, and a vector of integers to prioritize styles across cells, rows, and columns, ensuring correct application when styles intersect.

2.2.2 Resizing Columns and Rows. We further enhanced spreadsheet customization by implementing the functionality for users to resize columns according to their specific needs. To resize a column, the user must trigger a mouse down event on the right edge of any column. This interaction detects the user's intention to change the column width and memorizes the index it has been detected on, then determines the mouse movement according to the following equation:

$$moveX = x - O[r - 1] - D_x, \tag{2}$$

where *moveX* represents the horizontal distance the mouse has moved during the column resizing operation, x is the current horizontal position of the mouse cursor. The offsets array O holds the horizontal positions of the left edges of each column that is currently displayed in the spreadsheet, while r represents the selected index intended to be resized. Lastly, D_x represents the x-coordinate of the data rectangle's origin. This value is subtracted to ensure the calculation is relative to the data area.

We proceed with the operation by applying the following equation:

$$resizeRatio = \max\left(\frac{moveX}{cellWidth}, 0.1\right) \tag{3}$$

The equation calculates the ratio of the mouse movement to the cell width. It then ensures that this ratio is not less than 0.1 by using the max function. The calculated ratio is then set as the new column width on the resize index. Additionally, note that the cell width is the default width of the cells, which is calculated based on the font size settings.

2.3 Undoing and Redoing Spreadsheet Actions

Within the context of the spreadsheet application, we defined the undo and redo functionality as a state machine capable of switching between the current and previous states after applying a change to the spreadsheet and calling one of said operations.

To track state changes, we have designed and implemented the "TableAction" structure. Different state changes require modifications to various types of data. The TableAction structure simplifies the process by encapsulating the type of action triggered along with parameters necessary for adjustment. We have provided detailed definitions of various Spreadsheet actions as shown in 1.

The spreadsheet actions are managed within the respective undo and redo vectors whose maximum size is set to twenty actions. Upon triggering an add action event, the initial state prior to the change is recorded in the undo vector, while the post-change state, along with its corresponding action type, is recorded in the redo vector. When the user initiates an undo or redo event, either via the context menu or keyboard shortcuts, the Algorithm 1 is executed.

To summarize, the algorithm begins by verifying the feasibility of the state change, ensuring that there is at least one recorded action in the action counter. If this condition is met, the action counter is adjusted appropriately. The algorithm then executes the necessary statements based on the type of action, ensuring that the corresponding data is modified accordingly. Finally, the visible state of the spreadsheet is updated to reflect these changes.

```
Data: Action History
Result: Undo or Redo Action
UndoOrRedo(isRedo) if no actions available then
    return;
end
get action, adjust counter;
if action is Insert/Update then
    set cell, edit value;
else if action is SetStyle then
    apply styles;
else if action is Resize then
    apply size changes;
else if action is Merge/Split then
    update merge states;
else if action is TextPaste then
    apply text;
else if action is Sort then
    apply sorting;
else if action is Paste then
    apply data and styles;
end
update visible cells;
```

Algorithm 1: Undo/Redo algorithm.

3 RESULTS

The measurements leading to the results presented in this paper were conducted on a system equipped with an AMD Ryzen 9 3900x 12-Core Processor and 64GB of RAM. It is also important to note that DewesoftX version 2024.2 was used when conducting these measurements.

For the testing, we evaluated three spreadsheet functionalities across three progressively larger cell selections. The functionalities tested included pasting values into cells (see Table 1), undoing and redoing font colour changes (see Table 2), and loading from and saving to the setup file with the font colour state being set (see Table 3 and Table 4). The cell selection ranges used for these tests were 5x5, 25x25, and 50x50.

The data used in these experiments comprised text, numeric values, and dates, with font colour applied as specified. Each evaluation was conducted ten times under identical conditions, and the results were averaged to ensure accuracy.

Table 1: Evaluation of Cell Value Pasting.

Cell Selection Range:	Memory Usage [MB]:	Duration [ms]:
5x5	4.4	0.812
25x25	16.1	13.238
50x50	44.1	50.179

Within Table 1 the data shows that memory usage increases significantly with the size of the cell selection, from 4.4 MB for 5x5 to 44.1 MB for 50x50. The duration also increases with the size of the cell selection, from 0.812 ms for 5x5 to 50.179 ms for 50x50. A larger selection is expected to be more memory-intensive than its smaller counterpart. Interestingly, a selection 100 times larger is only 10 times more memory-intensive. This can be explained by the fact that the memory required for the basic widget to display correctly is also involved within the measurement and is independent of the amount of data stored in the spreadsheet. It's also worth noting that the duration does not increase linearly.

Table 2: Evaluation of Undoing/Redoing the Font Colour Property State.

Cell Selection Range:	Memory Usage [MB]:	Duration [ms]:
5x5	0.134	0.292
25x25	0.722	1.157
50x50	1.5	5.766

Within Table 2, memory usage increases modestly with larger cell selections, from 0.134 MB for 5x5 to 1.5 MB for 50x50. The duration also increases, from 0.292 ms for 5x5 to 5.766 ms for 50x50. The memory and time required to undo/redo font colour changes grow as the cell selection range expands. Comparing the measurements of the undo/redo function with the paste-into-cells function, we can conclude that the memory usage for the former is greater than that for the latter.

Table 3: Evaluation of Loading from Setup where Font Colour Property is set.

Cell Selection Range:	Memory Usage [MB]:	Duration [ms]:
5x5	97.8	1.769
25x25	117.6	33.173
50x50	123.6	132.758

For Table 3 memory usage increases with larger cell selections, from 97.8 MB for 5x5 to 123.6 MB for 50x50. The duration, however, shows an increase from 1.769 ms for 5x5 to 132.758 ms for 50x50, indicating that loading from setup becomes more time-consuming with larger selections.

Table 4: Evaluation of Saving to Setup where Font Colour Property is set.

Cell Selection Range:	Memory Usage [MB]:	Duration [ms]:
5x5	0.234	3.752
25x25	0.293	66.618
50x50	0.356	260.720

Lastly for Table 4, memory usage shows a slight increase with larger cell selections, from 0.234 MB for 5x5 to 0.356 MB for 50x50. The duration increases significantly with the size of the cell selection, from 3.752 ms for 5x5 to 260.720 ms for 50x50. This indicates that saving to setup is considerably more time-consuming as the cell selection size grows.

As expected, the results indicate that both memory usage and duration generally increase with larger cell selection ranges across all functionalities. Pasting values and saving to setup are particularly resource-intensive, whereas undoing and redoing font colour changes show a moderate increase in resource requirements. Loading from setup shows a notable increase in duration with larger selections, highlighting the complexity of handling larger datasets.

4 CONCLUSION

In this paper, we proposed a solution for handling measurement data in high-tech instrumentation through a computationally efficient spreadsheet editor widget. This widget is tailored for integration with Dewesoft's data acquisition software, DewesoftX, aiming to offer functionality comparable to commercial spreadsheet tools while ensuring compatibility with existing software standards. The solution focuses on enhancing data accessibility and user experience by providing an intuitive interface, robust navigation, and comprehensive formatting and state manipulation features.

In future development, we aim to enhance advanced mathability features essential for an efficient spreadsheet application. Specifically, we plan to implement a formula system within the widget, enabling users to input formulas and perform calculations directly within the spreadsheet. Additionally, we intend to incorporate conditional cell formatting, which automatically changes the appearance of cells based on their content to improve data visualization and analysis. We will also continue refining the existing features, taking user feedback into consideration to ensure the highest quality user experience possible.

ACKNOWLEDGMENTS

We are deeply thankful to Dewesoft and its representatives for their collaboration on this project, without which this paper would not have been possible.

REFERENCES

[1] F. Nurdiantoro, Y. Asnar, and T. E. Widagdo. The development of data collection tool on spreadsheet format. *Proceedings of 2017 International Conference on Data and Software Engineering, ICoDSE 2017*, 2018-January:1–6, Jul. 2017.

[2] Srideep Chatterjee, Nithin Reddy Gopidi, Ravi Chandra Kyasa, and Prakash Prashanth Ravi. Evaluation of open source tools and development platforms for data analysis in engine development. *SAE Technical Papers*, pages 1–11, Jan. 2015.

[3] Sabine Hipfl. Using layout information for spreadsheet visualization. *Proceedings of EuSpRIG 2004 Conference Risk Reduction in End User Computing: Best Practice for Spreadsheet Users in the New Europe*, pages 1–13, 2004.

[4] Bernard Liengme. *A Guide to Microsoft Excel 2013 for Scientists and Engineers*. Academic Press, London, United Kingdom, 2013.

[5] Dewesoft. Introduction | Dewesoft X Manual EN. https://manual.dewesoft.com/x/introduction, 2024. Accessed: 18-08-2024.

[6] Dewesoft. DewesoftX Award-Winning Data Acquisition and Digital Signal Processing Software. https://dewesoft.com/, 2024. Accessed: 21-08-2024.

[7] P. Biro and M. Csernoch. The mathability of spreadsheet tools. *6th IEEE Conference on Cognitive Infocommunications, CogInfoCom 2015 - Proceedings*, pages 105–110, Jan. 2016.

[8] Embarcadero Technologies. VCL Overview - RAD Studio. https://docwiki.embarcadero.com/RADStudio/Sydney/en/VCL_Overview, 2024. Accessed: 18-08-2024.

[9] Thomas Lauer. *Porting to Win32™: A Guide to Making Your Applications Ready for the 32-Bit Future of Windows™*. Springer-Verlag New York Inc., New York, NY, USA, 1996.

[10] Isaac Alejo. *Google Sheets Tutorial Guide*. Google Books, Online, 2024. Accessed: 18-08-2024.

[11] LibreOffice Documentation Team. *LibreOffice 4.1 Calc Guide*. Google Books, Online, 2024. Accessed: 08-08-2024.

[12] Karl Mernagh and Kevin Mc Daid. Google sheets vs microsoft excel: A comparison of the behaviour and performance of spreadsheet users. *Proceedings of the Psychology of Programming Interest Group (PPIG) 2014 Conference*, 2014.

Improvement and Evaluation of a Heuristic Method for the Minimal Feedback Arc Set Problem

Jure Pustoslemšek
jp76466@student.uni-lj.si
University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

Ema Črne
ec51731@student.uni-lj.si
University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

Nejc Rihter
nr5256@student.uni-lj.si
University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

ABSTRACT

This paper addresses the problem of finding minimal feedback arc sets in directed graphs, a critical issue in various domains such as computational biology, scheduling and network analysis. We implement, analyse and improve a novel heuristic approach. Our improved method reuses their heuristic method for reducing solution size and uses other established techniques from both exact and approximate algorithms to speed up the algorithm. The implementation makes use of a fast network analysis library for additional speed-up.

KEYWORDS

Directed graphs, minimum feedback arc set, NP-hard problems, combinatorial optimization, heuristic methods

1 INTRODUCTION

Directed graphs are a fundamental tool in network theory. They are widely used to model systems where the direction of relationships between entities is crucial. A feedback arc set (*abbr. FAS*) is a set of edges in a directed graph such that removing those edges from the graph makes it acyclic. While the entire set of vertices V can trivially serve as a feedback arc set, the challenge lies in finding a minimal feedback arc set, i.e. a feedback arc set with the smallest number of edges.

The minimal FAS problem has various real-world applications; for instance, in social network analysis, FAS is crucial for misinformation removal and label propagation [2], it also plays an important role in computational biology and neuroscience [7], and task scheduling by breaking cyclic dependencies.

The problem of finding a minimal FAS is NP-hard. The decisional version of the problem, that is finding a FAS of a certain size, is one of the first known NP-complete problems [8]. As such, it is believed that there exists no algorithm that can solve it in polynomial time. Additionally, the problem is also very challenging to approximate. There is no known algorithm with a constant bound on the approximation ratio, making it an APX-hard [6] problem. The problem is complementary to the maximum acyclic subgraph problem and there is a natural reduction to the linear arrangement problem [3].

In this work, we focus on the implementation and improvement of a heuristic algorithm for finding a minimal FAS, as described by Cavallaro et al. [3]. We shall refer to this algorithm as the original algorithm. The algorithm begins by identifying strongly connected components (*abbr. SCC*) in the input graph G. Since any cycle is guaranteed to belong to a single SCC [4], we are able to split G into SCCs and run the rest of the algorithm on each SCC separately, collecting partial solutions into a single list to form a full solution. For the remainder of the algorithm, we assume that G is strongly connected. Two empty lists E_f and E_b are initialized to hold the forward and backward edges respectively. Vertices are then ordered according to some rule. For each vertex, we identify the forward edges and add them to E_f, thereby removing these edges from the graph. Once the graph becomes acyclic during this process, iteration stops. We then iterate in reverse order to identify the backward edges, adding them to E_b and removing them from G. Once the graph becomes acyclic, iteration stops.

To improve the solution, we apply Algorithm 1, also called *smart-AE* [3] to the smaller of the two sets, E_f or E_b. This key component aims to reduce the size of the found FAS by reintroducing the removed edges in a balanced way while avoiding creating cycles in the graph.

```
   **Input:** Graph $G$, edge list $F$
   **Output:** $AE$
1  $AE \leftarrow [\,]$;
2  **while** $F$ *is not empty* **do**
3      $PE \leftarrow [\,]$;
4      $count \leftarrow 0$;
5      $i \leftarrow 0$;
6      **while** $i + count < |V(G)|$ **do**
7          $e \leftarrow F[i + count]$;
8          Add $e$ to $PE$;
9          Add $e$ to $G$;
10         **if** $G$ *is acyclic* **then**
11             Add $e$ to $AE$;
12             $count \leftarrow count + 1$;
13         **else**
14             Remove $e$ from $G$;
15         **end**
16         $i \leftarrow i + 1$;
17     **end**
18     Remove all of $PE$ from $F$;
19 **end**
20 **return** $AE$
```

Algorithm 1: The smartAE heuristic

The *smartAE* heuristic begins with an acyclic graph G and a list of edges F, which were removed from G. An empty list AE is initialized to store edges that can be successfully reintroduced into the graph without creating a cycle. We iterate through all of the

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.5
ISBN 978-961-286-914-4

edges in F, where instead of sequentially reintroducing each edge from F, the algorithm employs a counter *count* to strategically skip over edges. This approach ensures that we reinsert edges from as many vertices as possible. During each iteration, a temporary list *PE* tracks the edges being tested. If adding an edge does not create a cycle, it is added to *AE* and *count* is incremented. Overall time complexity of the entire algorithm is $O(|E|(|V| + |E|))$.

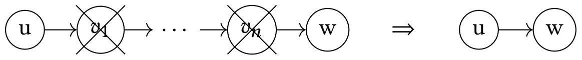
2 IMPROVEMENTS

In this section, we outline our improvements of the original algorithm. Improvements are grouped into four categories: size reduction, vertex ordering strategies, forward/backward edge removal and acyclicity checks.

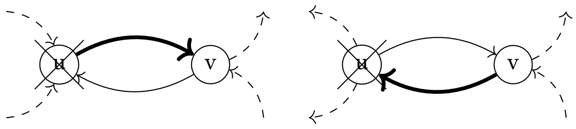
2.1 Size reduction

To reduce the size of the input graph G, we apply reduction rules based on a more general method by Baharev et al. [1]. The general method removes edges inside and on the boundary of an induced subgraph H with the following property: the size of a minimal FAS equals the upper bound of the size of the smallest edge set F whose removal breaks all cycles in G with vertices in H. In this case, we remove edges in F from G. We implemented a few straightforward rules for eliminating some simple and common patterns - we applied the following rules in rounds until a round produces no further reduction in graph size.

(1) For every self-loop e, i.e. an edge with the same entering and exiting vertex, add e to the solution.
(2) For every directed path $uv_1 \dots v_n w$ where $v_1, \dots v_n$ have in-degree 1 and out-degree 1, delete $v_1, \dots v_n$ and add the edge uw to G.



(3) For every 2-cycle uv where u has out-degree 1, delete u and add uv to the solution.
(4) For every 2-cycle uv where u has in-degree 1, delete u and add vu to the solution.



Having reduced the graph using the rules described above, we computed the strongly connected components (SCCs) as in the original algorithm. However, after computing the SCCs, we applied a technique described by Park and Akers [10], in which each SCC is further divided into its biconnected components. This efficiently breaks up the graph into even smaller strongly connected subgraphs. As biconnected components are traditionally defined for undirected graphs, we treat the SCCs as undirected to compute these components. The remainder of the algorithm is then applied to these biconnected components. Throughout this article, we will refer to these components as the SCCs.

2.2 Vertex ordering strategies

The original algorithm uses four vertex orderings in their experiments: in-degree and out-degree, both in increasing and decreasing order. We adopt all these orderings and add five more orderings. The first two are based on the difference of the in-degree ($d^-(v)$) and out-degree ($d^+(v)$) and two more are based on the difference of their degrees and their ratio. We also included a random ordering for comparison.

$$\text{degdiff}(v) = \max \left| d^+(v) - d^-(v), d^-(v) - d^+(v) \right| \tag{1}$$

$$\text{degratio}(v) = \max \left(\frac{d^-(v)}{d^+(v)}, \frac{d^+(v)}{d^-(v)} \right) \tag{2}$$

2.3 Forward/backward edge removal

The forward edge removal phase proceeds as follows. For each vertex v in G, ordered according to the chosen ordering, we remove all edges exiting v and entering vertices that follow v in the ordering, then we check if G has become acyclic. If G is acyclic, we end the edge removal phase. In the worst case, we remove forward edges of every vertex, at which point G is guaranteed to be acyclic, since the current ordering has become a topological ordering. The backward edge removal phase is almost identical, the only difference being that it removes edges that precede v in the ordering. The original algorithm chooses the smaller of E_f and E_b and only performs *smartAE* on that. We challenged this approach and performed *smartAE* on both, only then taking the smaller as the solution.

The purpose of the proposed improvement is to calculate an SCC decomposition before forward/backward edges of a vertex are removed. Then, instead of adding all edges to the solution, we skip edges that exit one SCC and enter another. Such an edge cannot be a part of a cycle. A cycle would imply that vertices from both SCCs are reachable from one another, but then all those vertices would be in the same SCC. As before, the edge removal phase terminates when G becomes acyclic. This improvement is expected to increase the running time but reduce the size of the solution.

2.4 Acyclicity checks

The most time-consuming aspect of the algorithm is restoration and removal of edges during *smartAE* process. A standard method for checking acyclicity is to find a topological ordering of the graph's vertices, i.e. a ordering of vertices such that all out-neighbors of a vertex x appear after x. If the graph contains a cycle, then it does not have a topological ordering [4]. This ordering provides an efficient way to check if inserting an edge would create a cycle. If it's a backward edge, i.e. it ends in a vertex that comes before its starting vertex, it forms cycles.

During *smartAE* we check acyclicity after restoring each edge. If the resulting graph has cycles, we immediately remove that edge, making the graph acyclic again. We avoid many calculations of a new topological ordering by exploiting this sequence of operations. Right before the use of *smartAE* at the end of forward/backward edge removal, an acyclic check is performed. During the check, we calculate a topological ordering and, if successful, store it in the variable T_c. We are then able to use this variable to make subsequent acyclicity checks trivial. When we restore an edge, we check if it is a backward edge in the T_c. If it is, the graph is no longer acyclic. We move the topological ordering from T_c into a background variable T_p and unset T_c. When we remove the last restored edge, we move the ordering in T_p back into T_c. We thus avoid a recalculation in the next acyclicity check.

3 EXPERIMENTAL RESULTS

We evaluated our implementation and improvements on the ISCAS circuit benchmark dataset [5] and on a selection of directed networks from the SNAP Large Network Dataset Collection [9]. We experimented with different configurations of the algorithm to assess the effect of each option on both speed and solution size. The ISCAS dataset primarily served as a speed benchmark. The largest instance in this dataset took about 5 to 10 minutes to complete, depending on the configuration; while the implementation of the original algorithm, which used four orderings instead of seven, took around 90 minutes. This gave us confidence to apply our algorithm on larger and more diverse networks from the SNAP dataset.

Our results from testing our implementation on the SNAP dataset are presented in Table 1. To illustrate the complexity and size of each graph, the first column lists the number of vertices and edges for each instance, while the second column provides the number of vertices and edges in the largest strongly connected component (SCC). The figures in both columns are based on reduced graph. The last column presents the best results from our different configurations, including the size of the minimal feedback arc set and the time taken to compute it.

We implemented the algorithm in Python, using a graph library written in C++. By implementing it this way, the source code is relatively easy to understand while also keeping it reasonably fast and efficient. We tested the implementation on the Arnes computing cluster with a 12-hour time limit. Each run gave us solution sizes for all vertex orderings. For runs that did not finish within the time limit, we examined the sizes of SCCs that were being computed. We searched for the largest SCC that computed at least one ordering within the time limit. This provided a rough estimate of the largest SCC size manageable in a parallel or distributed setting where each ordering is processed by two separate threads or nodes, one for forward edge removal and one for backward edge removal. In our experiments this number is between 2.8 and 2.9 million edges. The source code and raw result data is available at [11].

3.1 Impact of size reduction

The first option we tested was the use of size reductions. This reduction has two distinct effects. First, it reduces the size of the input graph. More importantly, the removal of edges and vertices can lead to a smaller SCCs, effectively shrinking the problem size and greatly reducing running time. Second, reduction rules should help decrease the size of the solution. While we add some removed edges to the solution, those edges are already guaranteed to be in the optimal solution.

For instances with the largest SCC having more than about 14,000 edges, the algorithm's running time was shorter when using reductions, with time savings increasing with input size. We find that for instances above this complexity, the benefits of reductions outweigh the cost. For a third of instances, the algorithm produced smaller solutions when reductions were not used, which is quite surprising. For these instances, *smartAE* was particularly effective, but its effect diminished after reductions. There is one instance for which the algorithm failed to complete within the time limit when not using reductions, but successfully computed a solution with configurations that used reductions.

Table 1: Solution sizes and running times for SNAP instances

Instance V-E	max-SCC V-E	Best result
7115-103689	1300-39456	7966: 14s
6301-20777	2068-9313	531: 15.3s
8114-26013	2624-10776	713: 20.9s
8717-31525	3226-13589	1186: 39s
8846-31839	3234-13453	1023: 38.8s
10876-39994	4317-18742	1721: 59.1s
22687-54705	5153-17695	1706: 1m37s
26518-65369	6352-22928	2254: 2m34s
36682-88328	8490-31706	2531: 3m44s
62586-147892	14149-50916	3361: 12m14s
75879-508837	32223-443506	141733: 37m53s
265214-418956	34203-151930	61598: 3m31s
325729-1469679	53968-304685	409862: 53m24s
281903-2312497	150532-1576314	313852: 9h43m2s
77360-828161	70355-888662	394218: 1h44m37s
82168-870161	71307-912381	403623: 1h48m43s

3.2 Vertex orderings

Different vertex orderings bring drastically different results. The orderings based on degree difference (1) and ratio (2), as well as the random ordering, give consistently worse solutions than those based on in-degree and out-degree. We should note that when comparing apparently analogous orderings, e.g. forward edges of an ascending ordering and backward edges of a descending ordering, there were minor, but non-zero differences in solution size. Such orderings were different due to the order of vertices with the same score.

The speed of the serial algorithm can be multiplied by a factor of $\frac{9}{4}$ with no effect on solution size, by only using the in-degree and out-degree orderings. Speed can be further doubled by only computing forward edge removals, but at a cost of slightly worse solution sizes. This gives us a speedup factor of $\frac{9}{2} = 4.5$. By taking into account quadratic time complexity, this would theoretically multiply the upper bound on feasible input size by 1.5, with no impact on solution size or approximately 2.12 with slightly worse solutions.

3.3 Impact of the SCC-based modification of edge removal

As described in Section 2.3, we implemented a modified version of the edge removal phase that is based on an SCC decomposition. The running time of the algorithm with this modification was more time consuming than the unmodified version. For some instances the running time doubled, while for others the impact on running time was less than 10%. The impact on solution size is more complex, though. If *smartAE* is not used, the solution is always smaller with the modification, as expected. However, when *smartAE* is applied, the solution can sometimes be larger with the modification. The difference is relatively small in those cases, ranging from 0.04% to 0.5%. This finding is somewhat unexpected as we expected this method to significantly improve solution size. This and our findings regarding reduction highlight unpredictability of *smartAE*. Similarly to our finding on vertex orderings, one can run the algorithm with both edge removal procedures to potentially achieve a slightly better solution size at a cost of running time.

3.4 Impact of smartAE

The addition of *smartAE* [3] roughly doubles the running time for all cases without the modified edge removal phase. The improvement in solution size, however, is heavily dependent on input size. Generally, larger graphs exhibit smaller improvements compared to smaller graphs. This is clearly illustrated in Figure 1, which shows the percentage of edges selected for the FAS solution, both with and without the *smartAE* procedure. The red portion of the bars represents the improvement achieved with *smartAE*. For smaller graphs, *smartAE* can reduce the solution size by half, whereas for the largest tested graphs, the improvement is as little as 1%. This observation raises questions about the usefulness of *smartAE* for very large graphs.

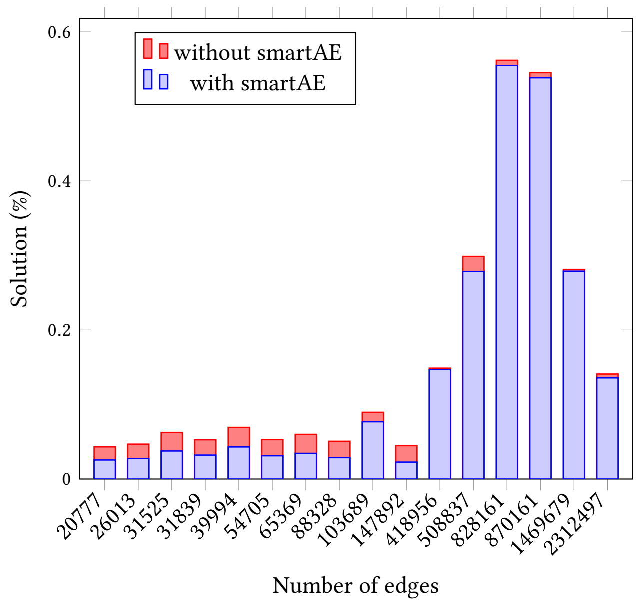


Figure 1: Percentage of solution edges compared to all edges in the graph, with and without the *smartAE* procedure.

4 CONCLUSIONS AND FUTURE WORK

The heuristic algorithm presented in this paper has proven effective in computing small feedback arc sets in large graphs. Multiple configurations have been tested with varying degrees of success.

Our implementation achieved comparable solution quality in significantly less time than the original approach on the same dataset, and it was also able to handle larger graphs. By employing size reduction techniques, we effectively decreased the complexity of input graphs, leading to faster processing times and more manageable SCCs, especially in larger graphs. We further reduced computational complexity by avoiding unnecessary recalculations during the *smartAE* process and by splitting SCCs into biconnected components. The experiments also revealed that selection of vertex ordering has a great impact on both the speed and quality of the solutions. Orderings based on in-degree and out-degree, particularly in descending and ascending orders, consistently outperformed other strategies. The modification covered in Section 3.3 did not consistently improve performance and, when it did, the benefits were usually insignificant. Additionally, it has led to less predictable results when *smartAE* was applied. The *smartAE* heuristic, while effective for reducing solution sizes in smaller graphs, showed lesser returns as graph size increased, raising questions about its efficiency and practicality in larger graphs.

However, there is still room for improvement. One is the speedup described in Section 3.3, but there are also other avenues. An obvious improvement is to implement the algorithm entirely in a compiled language like C++, eliminating the significant overhead of an interpreter. We plan on also adding more complex reduction rules based on Baharev's method [1], for example by searching for instances of tournaments or other common patterns and reducing based on those.

ACKNOWLEDGMENTS

We sincerely thank assist. prof. dr. Uroš Čibej for his guidance, advice and for introducing us to this topic.

REFERENCES

[1] Ali Baharev, Hermann Schichl, Arnold Neumaier, and Tobias Achterberg. 2021. An Exact Method for the Minimum Feedback Arc Set Problem. *ACM J. Exp. Algorithmics* 26, Article 1.4 (apr 2021), 28 pages. https://doi.org/10.1145/3446429

[2] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*. Association for Computing Machinery, 665–674. https://doi.org/10.1145/1963405.1963499

[3] Claudia Cavallaro, Vincenzo Cutello, and Mario Pavone. 2023. Effective Heuristics for Finding Small Minimal Feedback Arc Set Even for Large Graphs. In *CEUR Workshop Proceedings. vol. 3606.* https://ceur-ws.org/Vol-3606/paper56.pdf

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3 ed.). MIT Press.

[5] Ali Dasdan. 2004. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems* 9, 4 (2004), 385–418. https://doi.org/10.1145/1027084.1027085

[6] Guy Even, Joseph Naor, Baruch Schieber, and Leonid Zosin. 1997. Approximating the minimum feedback arc set in tournaments. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 464–472.

[7] I. Ispolatov and S. Maslov. 2008. Detection of the dominant direction of information flow and feedback links in densely interconnected regulatory networks. *BMC Bioinformatics* 9 (2008), 424. https://doi.org/10.1186/1471-2105-9-424

[8] Richard M. Karp. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, Raymond E. Miller and James W. Thatcher (Eds.). Springer US, Boston, MA, 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9

[9] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[10] S. Park and S.B. Akers. 1992. An efficient method for finding a minimal feedback arc set in directed graphs. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, Vol. 4. 1863–1866 vol.4. https://doi.org/10.1109/ISCAS.1992.230449

[11] Jure Pustoslemšek, Ema Črne, and Nejc Rihter. 2024. Implementation of a smartAE-based minFAS algorithm. https://github.com/jurepustos/fas-smartAE/. [Online; accessed 26-July-2024].

Counter-Strike Character Object Detection via Dataset Generation

Matija Šinko
matija11.sinko1@gmail.com
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

This paper addresses the challenge of developing robust object detection systems in the context of Valve's Counter-Strike by introducing a novel, high-quality dataset generated using a complex image generator built within the Unity game engine. This generator mimics the original game's environment and character interactions, capturing the complexity of in-game scenarios. The dataset provides a valuable resource for training models like the YOLOv9 algorithm, which we employ to develop an object detection system that achieves high precision and recall, in turn proving the usability of our dataset. Our dataset and demonstrated model could be used for object detection in future multi-modal autonomous agents, like the one we propose at the end of the paper.

KEYWORDS

Counter-Strike, object detection, AI, YOLO, autonomous agents, computer vision, data generation

1 INTRODUCTION

Artificial intelligence (AI) has opened new possibilities in gaming, with achievements like AlphaGo and AlphaStar mastering complex environments [3, 13]. These advances have sparked interest in applying AI to automate popular games. For fans of first-person shooters like Valve's Counter-Strike [12], this raises questions about AI-driven gameplay in fast-paced, strategic environments [4].

The primary problem addressed in this paper is the development of a robust object detection system for Counter-Strike, which is a crucial component for creating autonomous agents capable of human-level gameplay. Traditional approaches have relied on manually labeled datasets, which are time-consuming to create and often struggle to keep up with game updates. Additionally, previous attempts at developing AI agents for Counter-Strike have typically employed single, monolithic neural networks, leading to mixed results in terms of performance and adaptability.

In response to these challenges, we propose a novel solution: a high-quality, automatically generated dataset created using a complex image generator within the Unity game engine. This dataset is designed to train object detection models like YOLOv9 [14], which we use to identify enemy players in Counter-Strike. Furthermore, we suggest that this dataset can be a foundation for multi-model agent architectures, which could offer more robust and adaptable AI systems for gaming.

The structure of this paper is as follows: First, we provide a detailed overview of the methodology used to generate the dataset and train the object detection model. Next, we present the results of our experiments, demonstrating the effectiveness of our approach. Finally, we discuss the potential applications of the dataset in multi-model systems and conclude with suggestions for future research.

2 METHODOLOGY

2.1 Overview of Dataset Generation

This section outlines our approach to generating a dataset for training an object detection model in Valve's Counter-Strike. Using Unity, we closely replicated in-game environments and characters to ensure the training data accurately reflects real gameplay conditions.

2.2 Image Generation Pipeline

Our image generation pipeline was built for flexibility and iterative improvement, allowing quick updates to enhance model training and evaluation. This approach was key to creating a diverse and robust dataset with various in-game scenarios (see Figure 1).

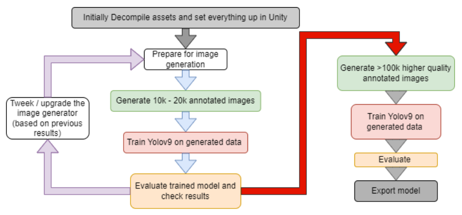

Initially Decompile assets and set everything up in Unity
Prepare for image generation
Generate 10k - 20k annotated images
Train Yolov9 on generated data
Evaluate trained model and check results
Tweek / upgrade the image generator (based on previous results)
Generate >100k higher quality annotated images
Train Yolov9 on generated data
Evaluate
Export model

Figure 1: Object detection with image generation pipeline

2.3 Detailed Environment and Character Simulation

A critical aspect of our dataset generation process was ensuring that the simulated environments and character models closely resembled those in the actual Counter-Strike game, similar to approaches using synthetic data in Unity and Unreal engines for realistic object detection [1, 10].

2.3.1 Lighting and Rendering. We used Unity's High Definition Render Pipeline (HDRP) [11] to recreate the complex lighting conditions of Counter-Strike. Realistic lighting made the dataset closely mirror the game, helping the model generalize to real gameplay.

2.3.2 Character Positioning and Inverse Kinematics (IK). To create realistic and varied character poses, we applied Inverse Kinematics (IK) algorithms[15]. IK enabled dynamic posing of characters in natural scenarios like aiming, shooting, and navigating.

2.4 Object Detection and YOLO Algorithm

Object detection is key to validating our dataset, focused on identifying enemy players in the game. We used the YOLOv9 algorithm,

DOI: https://doi.org/10.18690/um.feri.6.2024.6
ISBN 978-961-286-914-4

known for its speed and accuracy, to demonstrate the dataset's effectiveness. Training YOLOv9 on our dataset confirmed its utility. Though this study centers on YOLOv9, similar detectors could be used in more complex multi-modal agents in the future.

3 THE PROPOSED METHOD

3.1 Detailed Dataset Generation Process

As discussed earlier, Our object detectior was trained on a large Unity-generated dataset comprising varied scenarios on a detailed Dust 2 map with both Terrorist and Counter-Terrorist characters.

3.1.1 Environment and Character Creation.

(1) **Map Recreation**: The Dust 2 map was decompiled from Counter-Strike 2 game files and imported into Unity 3D. After cleaning up the geometry, we obtained a 1:1 replica of the map. (see Figure 2)

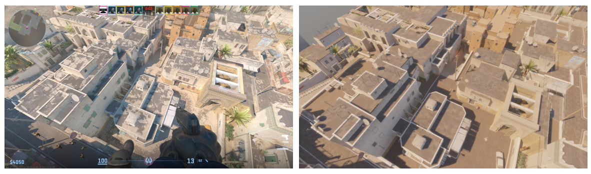


Figure 2: Comparison of Dust 2 in real game (left) and Unity Recreation (right) (Left source: Counter-Strike 2 during authors' gameplay)

(2) **Character Models**: We extracted and rigged character models for Terrorists and Counter-Terrorists using Blender, then imported them into Unity with various poses and weapons. (see Figure 6 in Appendix)
(3) **Lighting and Rendering**: Unity's High Definition Render Pipeline (HDRP) was employed to set up realistic lighting conditions, using path tracing technology to closely mimic the visuals of Counter-Strike 2. (see Figure 3)

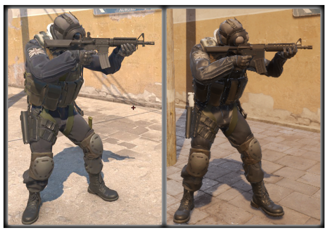


Figure 3: Comparison: Real Counter Terrorist (left) and Unity recreation (right) (Left source: Counter-Strike 2 during authors' gameplay)

3.1.2 Data Annotation and Generation.

(1) **Random Placement**: Characters were randomly placed around the map, and virtual cameras were positioned to capture various perspectives.
(2) **Dynamic Posing**: Using the Final IK plugin for inverse kinematics [7], characters were given dynamic poses aimed at different targets, adding variability to the training data.
(3) **Labeling**: Our system automatically annotated images with bounding boxes for each character, distinguishing between Terrorists, Counter-Terrorists, and their states (alive or dead).

4 RESULTS

4.1 Purpose of the Experimental Work

The goal of our experimental work was to train a YOLOv9 object detector with sufficient accuracy and recall. This would be critical for potential future use in multi-model autonomous agents, where accuracy would be needed to distinguish between friendly and enemy characters, as well as dead and alive ones. High recall would be vital for quickly and reliably spotting characters and differentiating them from the background, which would be essential for agents that rely on object detection models for shooting tasks.

4.2 Comparative Studies and Setups

We iteratively refined our image generator, creating datasets to train and evaluate the object detection model, with each iteration enhancing accuracy and recall:

(1) **Initial Model:** A dataset of 10,000 images featuring only alive characters in various poses, serving as the baseline.
(2) **Addition of Dead Characters:** Introduced separate labels for dead characters to improve recall, especially in distinguishing live enemies from the background.
(3) **First-Person Perspective and UI Overlays:** Added hands and UI elements to reduce misclassification of player arms as enemies.
(4) **Blood Splatter Effects:** Introduced blood effects to enhance precision in differentiating character states.
(5) **Name Tag-Based Identification:** Added name tags to distinguish friendly from enemy characters, as friendlies always have tags above their heads. Some tags without visible characters belong to friendlies behind walls, which we avoid classifying. This approach required training two separate models, one for each team.

These were evaluated for precision and recall improvements.

4.3 Datasets Used for Testing

To ensure the robustness and generalizability of the object detection model, we evaluated it on a variety of datasets:

(1) **Self**: This dataset was generated using the same methods as the training set, serving as a control to measure overfitting and baseline performance.
(2) **Bots CT and T Combined**: Captured from bot games in Counter-Strike, this one did not include unique player skins. This dataset is the closest to a final deployment scenario.
(3) **Batch 3**: This dataset included captures from real multiplayer games, featuring unique player skins and a variety of in-game environments.
(4) **Batch 4**: Similar to Batch 3 but sourced from different multiplayer sessions, providing additional variety in testing conditions.
(5) **Batch 1**: Captured from the Deathmatch mode, this dataset differs most from the intended deployment environment but provides insights into model generalization.
(6) **Bots, Batch 3, Batch 4 Combined**: A comprehensive dataset combining Bots CT and T, Batch 3, and Batch 4, used to test overall model performance across various conditions.
(7) **All Combined**: A super-set combining all the above datasets.

These datasets tested the model's strengths and weaknesses.

4.4 Evaluation Metrics

We measured various evaluation metrics, including Precision, Recall, F1 Score, mAP, and IoU, with detailed results available for each in the Appendix. Our focus was on Precision and Recall. High Precision is vital for accurately identifying enemy characters, avoiding misclassification of friendly units, while high Recall ensures all enemies are detected quickly and distinguished from the backround. These metrics are crucial for creating robust object detection models that could be used for potential future autonomous agents in competitive gaming.

4.5 Announcement of the Experiments Performed

We conducted experiments to evaluate object detection models trained on our dataset, testing their ability to detect and classify characters in Counter-Strike under various conditions. These experiments included assessing different character states (alive, dead) and the impact of dataset sizes and image resolutions, aiming to refine the model's precision and recall for real-world gameplay.

4.6 Detailed Descriptions of Experiments and Results

4.6.1 *Image Generator Upgrades.*

(1) **Initial Model of Only Alive Characters**: This model showed good precision but lacked recall, which is crucial for our application.
(2) **Addition of Dead Characters**: Improved the model's recall, which is crucial for distinguishing between live enemies and background elements.
(3) **First-Person Perspective and UI Overlays**: addressed the issue of confusing player arms with enemy characters. Improved recall.
(4) **Added Blood Splatter Effects**: improved both precision and recall for the terrorist dead and alive classes.
(5) **Friendly Character Identification via name tag**: Showed best results and vastly improved accuracy and recall . *Batch 1, trained in Deathmatch mode without name tags, performed worse, but this isn't a concern since the dataset is intended for Competitive mode, where name tags are always present.*

4.6.2 Cross Model Examination. The purpose of the cross-model examination is to compare the performance of different model configurations across various metrics.

As observed in Figure 4, our upgrades to the Unity Generator were successful in improving performance. For example, the recall for the counter-terrorist class improved from 0.4 to over 0.6. Graphs for the class Terrorist as well as the F1 score can be found in the Appendix A.3. While the improvements on the dead character classes seen in Figure 14 in the Appendix, were not significant, we observed notable enhancements in the performance for the terrorist and counter-terrorist classes see 12, 13. This distinction is crucial, as our primary objective is for our dataset to be suitable for potential training of agents that can accurately differentiate between alive friendly and enemy characters. The dead characters need only to be distinguished from the living, without requiring detailed differentiation among themselves. Tabled and detailed Data for these cross model examinations can be found in the Appendix A.3.

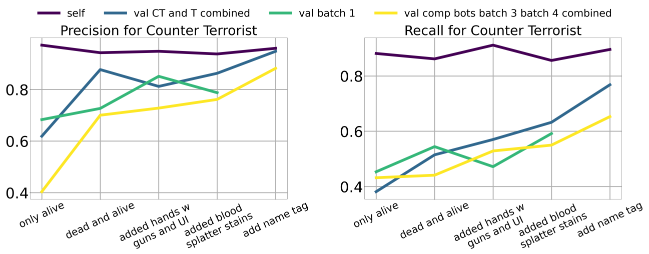


Figure 4: Counter-Terrorist class across all of our generation upgrades on 10k 640x360 images

4.6.3 Testing Different Dataset Sizes. We trained models on varying dataset sizes ranging from 1,000 to 100,000 images to examine the impact on performance. As seen in Figure 15 in the Appendix Larger training sizes improved model performance significantly.

4.6.4 Bigger Image Sizes and a Bigger Model. We explored the effects of training with larger image sizes and using a bigger YOLOv9 model and we came to the conclusion that bigger image sizes and a bigger model lead to better performance all around See Apendix A.5. That's why we trained out best model on a 1280x720 image size and with 20k images

4.6.5 Comparison with Other Counter-Strike Object Detectors. In this section, we compare our best object detection model, trained on a dataset of 20k images at a resolution of 1280x720 with name tags, to three other models developed by different authors:

- **Our Best Model**: (Terrorist team, 1280x720, 20k images).
- **Siromer's Model**: Dataset taken from [8, 9].
- **Python Lessons Model**: Dataset taken from [5, 6]. This dataset was used by Chenyang Dai [2] for object detection in their hybrid imitation training model.

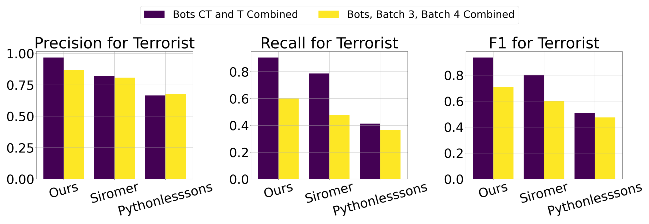


Figure 5: Comparison of object detection models

Our curated bar charts show that our object detector outperforms other methods in detecting Terrorists on the two selected datasets See table 1. For results on additional datasets and classes, see the Appendix A.6. Notably, Chenyang Dai used the 'Python Lessons' dataset to train the object detection part of their multi-model AI. This indicates that integrating our dataset into a multi-model Counter-Strike 2 system could potentially yield superior performance, as our dataset seems better than Python Lessons'. These findings support our initial hypothesis that our dataset could be used to train more complex multi-model agents.

Table 1: Detailed Metrics for Class: Terrorist

Metric	Bots CT and T Combined			Bots, Batch 3, Batch 4 Combined		
	Ours	Siromer	PyLessons	Ours	Siromer	PyLessons
Precision	0.97	0.82	0.67	0.87	0.81	0.68
Recall	0.91	0.79	0.41	0.60	0.48	0.37
F1 Score	0.94	0.80	0.51	0.71	0.60	0.47

4.6.6 Video Demonstrations of Model Performance. We provide video demonstrations of our object detection model's performance for both Counter-Terrorist and Terrorist scenarios. Available on YouTube:

- **Counter-Terrorist**: www.youtube.com/watch?v=u49CLDt8MgU
- **Terrorist**: www.youtube.com/watch?v=u49CLDt8MgU

4.7 Discussion

The improvements through our image generation pipeline have successfully enhanced model performance. The precision achieved by the model is sufficient to avoid mistakenly identifying teammates or dead characters as threats, while the recall is robust enough to reliably detect enemy players. Any remaining inaccuracies could be further refined with reinforcement-based shooting models that adapt to detection patterns when used on our generated dataset. In comparison to previous studies, such as [4], which encountered issues like agents mistakenly shooting dead characters, our approach offers a clear advantage. The dataset we generated, paired with a specific object detection model, can relieve a potential agent from the need to learn the shapes of characters, allowing it to focus more on tasks like shooting accuracy and map traversal when learning. While our model shows promise, it is currently limited to detecting objects within the Dust 2 map and is sensitive to player cosmetic skins. Additionally, the model does not differentiate between body parts, which may limit its application in more precise, action-oriented tasks.

5 CONCLUSION

5.1 Summary of Work and Key Findings

This research focused on the development and validation of a high-quality dataset generated using a Unity-based image generator for training object detection models in the context of Counter-Strike. Through iterative enhancements to our image generation pipeline, we achieved significant improvements in both precision and recall of the YOLOv9-based object detection model. This validated the effectiveness of our approach, demonstrating that synthetic data can effectively train models for complex in-game scenarios.

5.2 Best Results and Contributions

Our study made several key contributions to the field:

- **Versatile Dataset for Object Detection**: Our image generator and datasets are valuable for training object detection models in Counter Strike.
- **Effective Use of Synthetic Data Proven by Object Detection**: We showed that synthetic data can replace real-world data in training models, especially when labeled data is scarce. This was proven by achieving strong results with an object detection model trained on our generated data.
- **Future Applications**: Our work could be incorporated into future autonomous agents or used as object detection teaching exercises.

5.3 Future Work

Looking forward, there are several avenues for enhancing the capabilities of our system:

- **Expansion of Dataset and Model Generalization**: Future work will focus on expanding the dataset to include additional decompiled maps and the introduction of random cosmetic skin patterns to improve the model's generalization. Additionally, incorporating YOLOv9 pose estimation will allow for the identification of specific character body parts, thereby enhancing the model's ability to aim and shoot with greater effectivenes in a potential reinforcement learning framework.
- **Proposed Multi-Model Agent Architecture**: We propose a complex multi-model architecture that could serve as the foundation for developing autonomous agents capable of high-level gameplay in Counter-Strike see Appendix A.7.
- **Planned Dataset Publication**: We plan to publish our dataset on platforms like Kaggle, Hugging Face, and Roboflow, allowing others to use it for developing agents and practicing object detection skills.

5.4 Final Thoughts

This study underscores the potential of synthetic data and iterative model development in advancing AI for gaming. While challenges remain, particularly in bridging the gap between synthetic and real-world data, the progress made here provides a solid foundation for future innovations. The proposed multi-model architecture represents a promising direction for developing more sophisticated and capable autonomous agents, capable of performing at a high level in complex gaming environments like Counter-Strike. As AI continues to evolve, integrating reinforcement learning and advanced detection techniques will be crucial in pushing the boundaries of what these agents can achieve.

REFERENCES

[1] Per-Arne Andersen, Teodor Aune, and Daniel Hagen. 2022. Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine. *Applied Sciences* 12 (08 2022).

[2] Chenyang Dai. 2021. Counter-Strike Self-play AI Agent with Object Detection and Imitation Training. CS230: Deep Learning, Fall 2021, Stanford University, CA. (LaTeX template borrowed from NIPS 2017.).

[3] Guillaume Lample and Devendra Singh Chaplot. 2018. Playing FPS Games with Deep Reinforcement Learning. arXiv:1609.05521 [cs.AI]

[4] Tim Pearce and Jun Zhu. 2021. Counter-Strike Deathmatch with Large-Scale Behavioural Cloning. arXiv:2104.04258 [cs.AI]

[5] Python Lessons. 2020. TensorFlow 2.3.1 YOLOv4 - CSGO Aimbot. Accessed: 2024-08-30.

[6] Python Lessons. 2020. YOLOv4 TensorFlow 2.3.1 - CSGO Aimbot. Accessed: 2024-08-30.

[7] RootMotion. 2014. Final IK: The Ultimate IK Solution for Unity. Accessed: 2024-08-30.

[8] Faruk Günaydin (Siromer). 2024. Counter-Strike 2 Body and Head Classification. Accessed: 2024-08-25.

[9] Faruk Günaydin (Siromer). 2024. CS:GO/CS2 Object Detection. Accessed: 2024-08-25.

[10] et al. Steve Borkman, Adam Crespi. 2021. Unity Perception: Generate Synthetic Data for Computer Vision. arXiv:2107.04259 [cs.CV]

[11] Unity Technologies. 2024. High Definition Render Pipeline (HDRP). Accessed: 2024-08-30.

[12] Valve Corporation. 2023. Counter-Strike 2. Accessed: 2024-08-30.

[13] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. arXiv:1509.06461 [cs.LG]

[14] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. 2024. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. arXiv:2402.13616 [cs.CV]

[15] Chris Welman. 1993. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation.* Simon Fraser University.

A APPENDIX

A.1 Detailed Dataset Generation Process

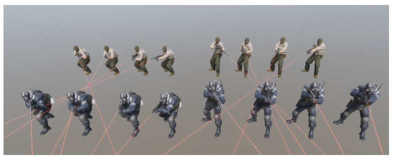


Figure 6: Various character poses for: Terrorists and Counter-Terrorists each aiming at a target (source: Our Counter-Strike recreation inside Unity 3D)

A.2 Model Upgrades

Figures related to the model upgrades and their detailed results can be found here:

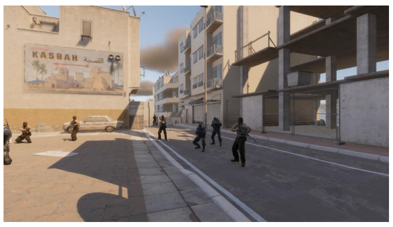


Figure 7: Alive Counter-Terrorist and Terrorist characters (source: Our Counter-Strike recreation inside Unity 3D)

Reference to the image context in the document

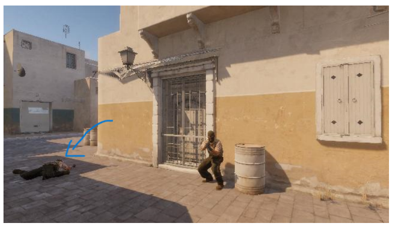


Figure 8: Dead and alive characters (source: Our Counter-Strike recreation inside Unity 3D)

Reference to the image context in the document

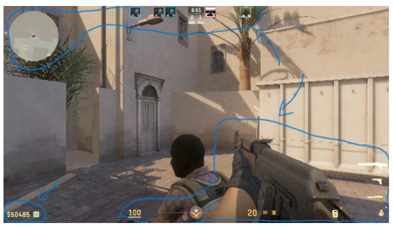


Figure 9: First-person hands and UI (source: Our Counter-Strike recreation inside Unity 3D)

Reference to the image context in the document

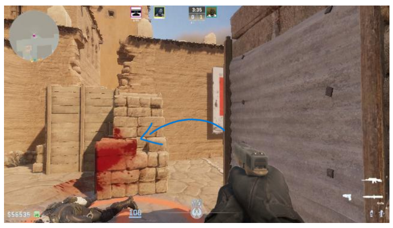


Figure 10: Blood next to a dead character (source: Our Counter-Strike recreation inside Unity 3D)

Reference to the image context in the document

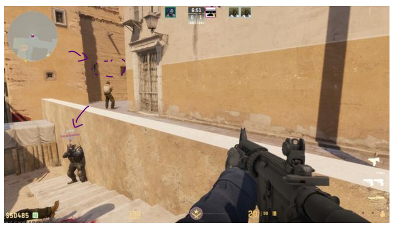


Figure 11: Image for the Coutner Terrorist model with a name tag over a Counter terrorist and a Terrorist with no name tag (source: Our Counter-Strike recreation inside Unity 3D)

A.3 Cross Model Examination data

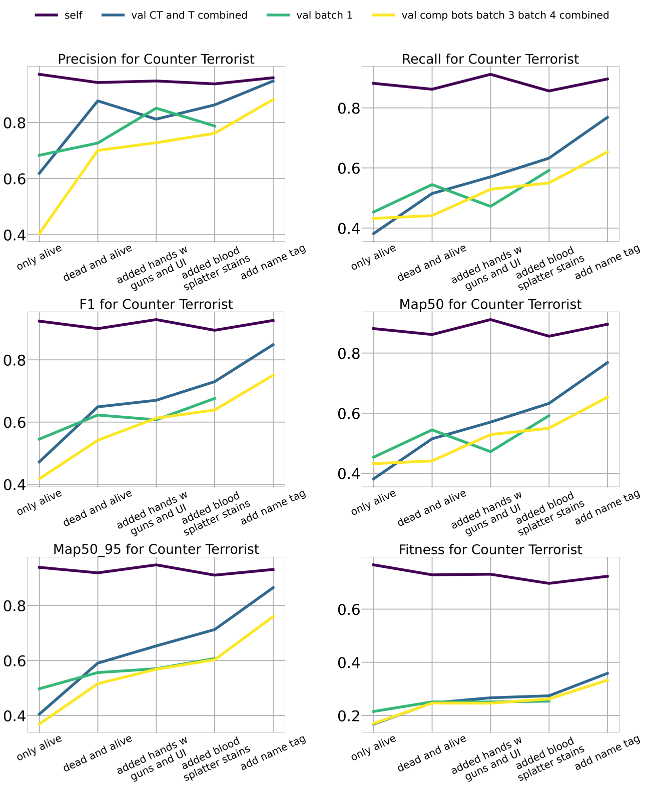


Figure 12: Counter-Terrorist class across all of our generation upgrades

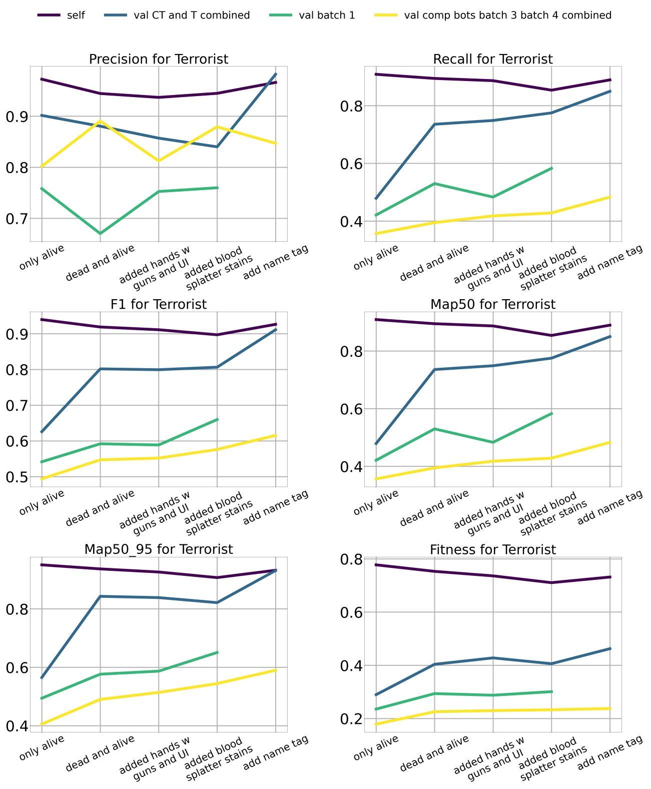


Figure 13: Terrorist class across all of our generation upgrades

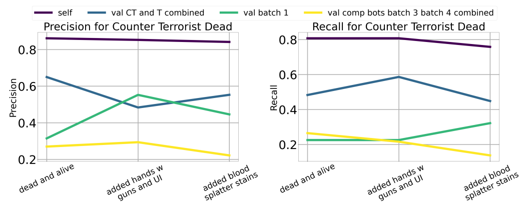


Figure 14: Counter-Terrorist dead class across all of our generation upgrades

Table 2: Precision for Class: Terrorist

Data Set	Self	CT and T Combined	Batch 1	Batch 3 & 4
Only Alive	0.972	0.901	0.758	0.802
Dead and Alive	0.944	0.880	0.670	0.890
Hands w/ Guns & UI	0.937	0.857	0.752	0.812
Added Blood Splatter Stains	0.945	0.840	0.760	0.879

Table 3: Recall for Class: Terrorist

Data Set	Self	CT and T Combined	Batch 1	Batch 3 & 4
Only Alive	0.909	0.479	0.421	0.357
Dead and Alive	0.894	0.736	0.530	0.395
Hands w/ Guns & UI	0.887	0.749	0.483	0.418
Added Blood Splatter Stains	0.854	0.775	0.583	0.428

Table 4: F1-Score for Class: Terrorist

Data Set	Self	CT and T Combined	Batch 1	Batch 3 & 4
Only Alive	0.940	0.626	0.541	0.494
Dead and Alive	0.919	0.802	0.592	0.547
Hands w/ Guns & UI	0.911	0.799	0.589	0.552
Added Blood Splatter Stains	0.897	0.806	0.660	0.576

Table 5: Combined Metrics for Class: All Metrics

Data Set	Self	CT and T Combined	Batch 1	Batch 3 & 4
Blood Dataset 1k	0.919	0.598	0.544	0.448
Blood Dataset 5k	0.920	0.608	0.519	0.416
Blood Dataset 10k	0.916	0.714	0.575	0.536
Blood Dataset 20k	0.919	0.665	0.574	0.464
Blood Dataset 100k	0.905	0.737	0.568	0.460

A.4 Testing Different Dataset Sizes

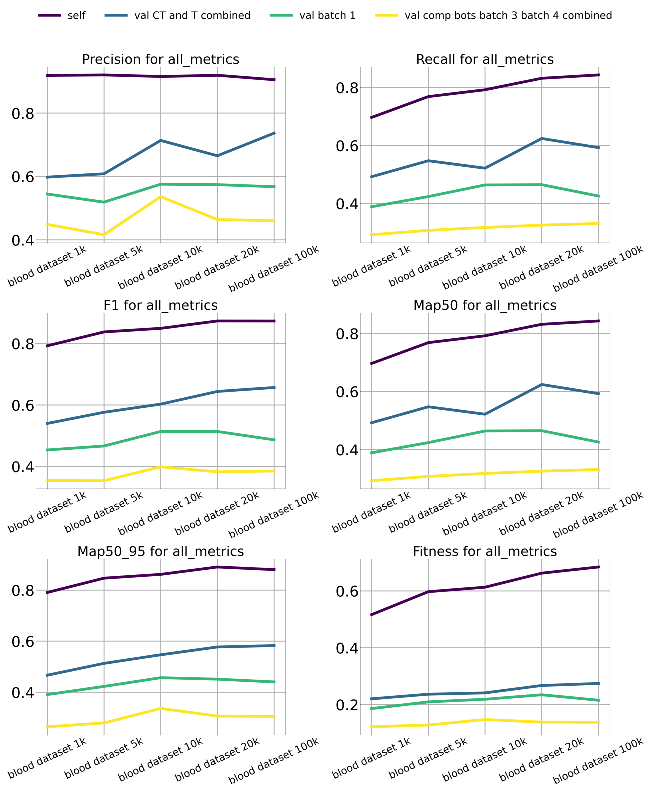
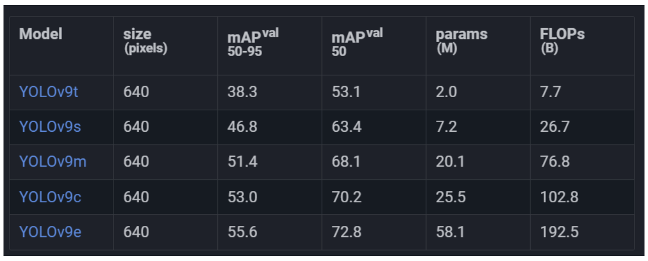


Figure 15: all metrics (classes) combined for different dataset sizes

Table 6: Detailed Metrics for Class: Counter Terrorist Dead

Data Set	Self	CT and T Combined	Batch 1	Batch 3 & 4
Only Alive	-	-	-	-
Dead and Alive	0.861	0.650	0.315	0.270
Hands w/ Guns & UI	0.853	0.483	0.552	0.294
Added Blood Splatter Stains	0.841	0.553	0.446	0.222

A.5 Bigger Image Sizes and a Bigger Model



Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	params (M)	FLOPs (B)
YOLOv9t	640	38.3	53.1	2.0	7.7
YOLOv9s	640	46.8	63.4	7.2	26.7
YOLOv9m	640	51.4	68.1	20.1	76.8
YOLOv9c	640	53.0	70.2	25.5	102.8
YOLOv9e	640	55.6	72.8	58.1	192.5

Figure 16: YOLO model sizes

YOLO offers five different model sizes 16, each with a trade-off between accuracy and training/inference speed. All of our experiments so far have been conducted using the smallest model, YOLOv9t, due to its faster inference time. The reasoning is that if our object detector is to be used as part of a real-time autonomous agent, we need the fastest possible response time (inference time). Similarly, we have used images with dimensions of 630x360 for all experiments so far, as larger images provide better accuracy but at the cost of slower processing speeds. We have also explored dataset sizes, which show a similar trend: larger datasets yield better accuracy but result in longer training times. Such adjustments to our dataset, such as changing model size, image size, or dataset size, are dynamic and easy to make when using an image generator like ours but are very rigid and time-consuming in traditional hand-labeling scenarios. In this section, we will examine how different model sizes, image dimensions, and dataset sizes affect accuracy and compare their performance. Here we compare 4 different models to see how effective different methods are (image size, dataset size, model size).

- A 10k YOLOv9 tiny model with image size (640x360)
- A 100k YOLOv9 tiny model with image size (640x360)
- A 10k YOLOv9 model with image size (640x360)
- A 10k YOLOv9 tiny model with image size (1280x720)

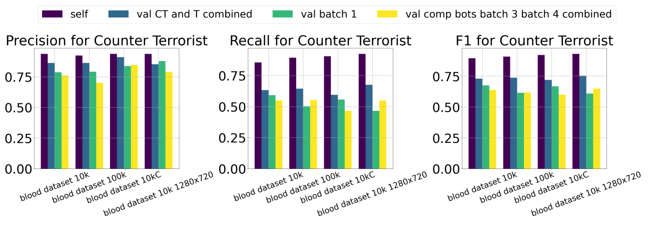


Figure 17: 1280x720 model next to previous models

Table 7: Performance Metrics for Class: Counter Terrorist

Data Set	10k T			100k T			10k C		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Self	0.937	0.856	0.895	0.924	0.894	0.909	0.937	0.905	0.921
Bots CT and T Combined	0.862	0.632	0.730	0.862	0.645	0.738	0.910	0.595	0.719
Batch 1	0.787	0.592	0.676	0.792	0.503	0.615	0.837	0.556	0.668
Bots, Batch 3, Batch 4 Combined	0.761	0.550	0.638	0.700	0.553	0.618	0.846	0.466	0.601

From the above bar charts, we can deduce that having a bigger dataset, choosing a bigger model, and using larger image sizes helps improve our model's performance. The best performance was observed in the model trained on larger images. Future work could involve training a model on 100k images of size (1280x720).

Check the Appendix to see how the (1280x720) model compares to our previous generation upgrades. A.5.

We can see that the higher image size aligns with our trend of increasing recall, which was our original goal.

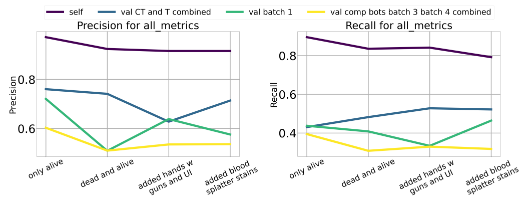


Figure 18: Counter-Terrorist dead class across all of our generation upgrades

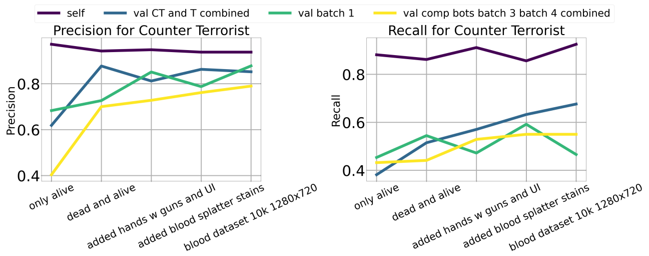


Figure 19: 1280x720 model compared to previous models

A.6 Comparison with Other Counter-Strike Object Detectors

In this section, we present a broader comparison of our model against those trained on Siromer's and PythonLessons' datasets across more metrics and datasets.

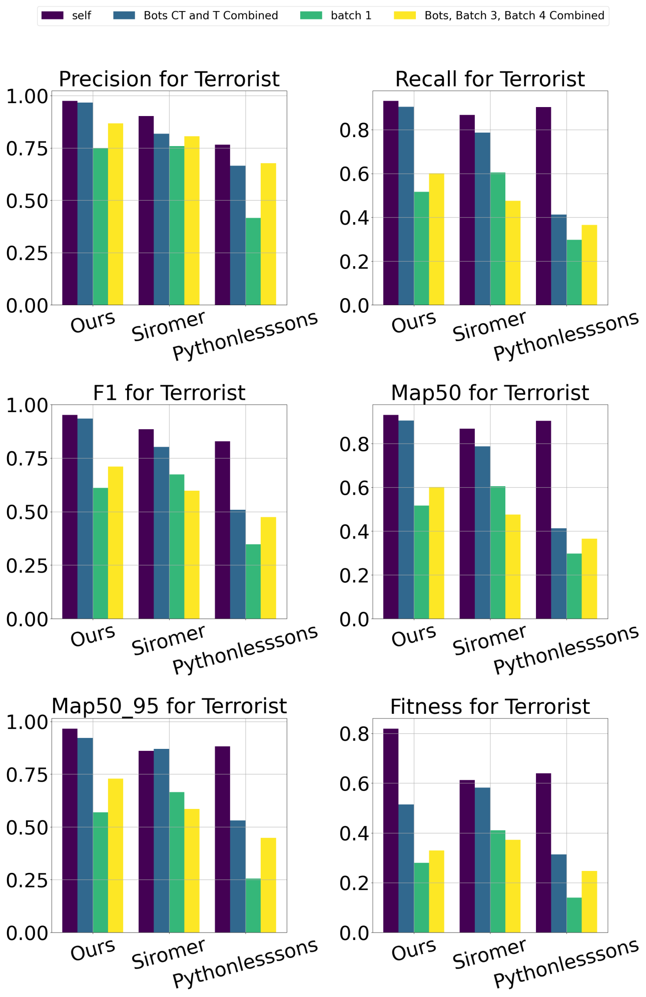


Figure 20: Comparison of object detection models expanded on the class "Terrorist"

Table 8: Expanded Metrics for Class: Terrorist

Metric	Precision				Recall			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.97	0.97	0.75	0.87	0.93	0.91	0.52	0.60
Siromer	0.90	0.82	0.76	0.81	0.87	0.79	0.61	0.48
PyLessons	0.77	0.67	0.42	0.68	0.90	0.41	0.30	0.37

Metric	F1 Score				mAP50			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.95	0.94	0.61	0.71	0.93	0.91	0.52	0.60
Siromer	0.88	0.80	0.67	0.60	0.87	0.79	0.61	0.48
PyLessons	0.83	0.51	0.35	0.47	0.90	0.41	0.30	0.37

Metric	mAP50_95				Fitness			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.97	0.92	0.57	0.73	0.82	0.51	0.28	0.33
Siromer	0.86	0.87	0.67	0.59	0.61	0.58	0.41	0.37
PyLessons	0.88	0.53	0.26	0.45	0.64	0.31	0.14	0.25

As shown in Figure 20 and seen in table 8, our model outperforms the other two models across all metrics, except when evaluated on the "Batch 1" dataset. This is not entirely surprising. As previously mentioned, our best-performing model relies heavily on the effectiveness of name tags above characters' heads to distinguish friendlies from enemies. This approach led to significant improvements but performed worse on datasets that were trained in game modes like "Deathmatch," where no name tags are displayed above friendly characters. Batch 1, as stated earlier, is trained in such a deathmatch game mode. Therefore, more traditional approaches, such as those using Siromer's and PythonLessons' datasets, generalize better in this scenario. However, this is not a major concern for us since our model is designed for the competitive game mode, where two teams of five players face off and friendly characters always have name tags above their heads.

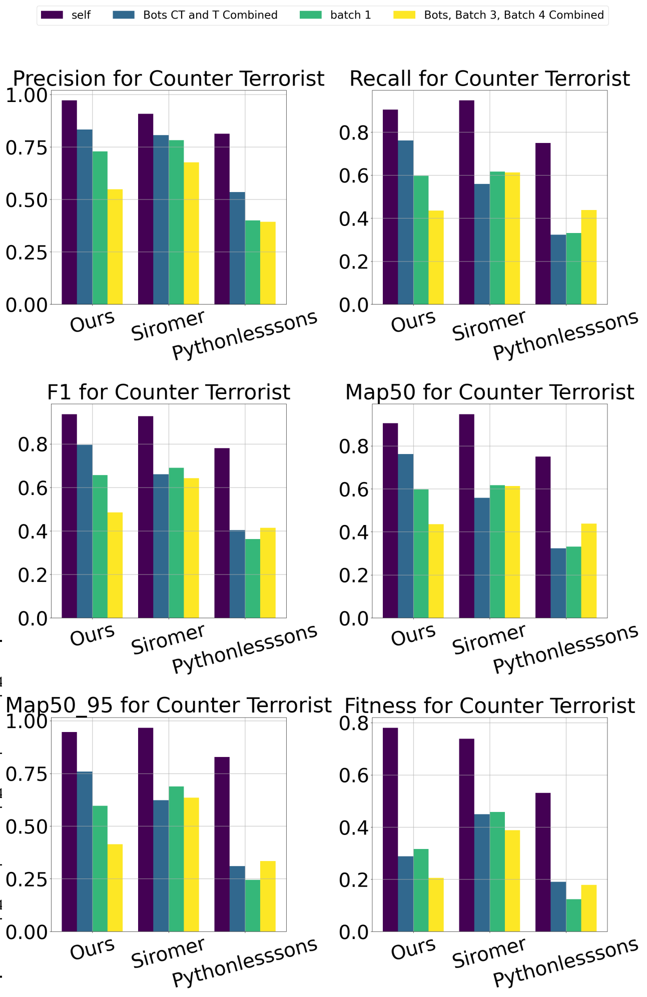


Figure 21: Comparison of object detection models expanded on the class "Counter-Terrorist"

Table 9: Expanded Metrics for Class: Counter-Terrorist

Metric	Precision				Recall			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.97	0.83	0.73	0.55	0.91	0.76	0.60	0.44
Siromer	0.91	0.81	0.78	0.68	0.95	0.56	0.62	0.61
PyLessons	0.81	0.54	0.40	0.39	0.75	0.32	0.33	0.44
Metric	F1 Score				mAP50			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.94	0.80	0.66	0.49	0.91	0.76	0.60	0.44
Siromer	0.93	0.66	0.69	0.64	0.95	0.56	0.62	0.61
PyLessons	0.78	0.40	0.36	0.41	0.75	0.32	0.33	0.44
Metric	mAP50_95				Fitness			
Data Set	Self	Bots CT & T	Batch 1	Bots, B3, B4	Self	Bots CT & T	Batch 1	Bots, B3, B4
Ours (1280x720)	0.95	0.76	0.60	0.41	0.78	0.29	0.32	0.21
Siromer	0.97	0.62	0.69	0.63	0.74	0.45	0.46	0.39
PyLessons	0.83	0.31	0.24	0.33	0.53	0.19	0.12	0.18

As seen in Figure 21 and table 9, our model performs noticeably worse on the "Counter-Terrorist" class compared to the "Terrorist" class, as shown in Figure 20. In some instances, it is even outperformed on the "Bots, Batch 3, Batch 4 Combined" dataset. This outcome is also expected because our latest model, which leverages name tags, requires training two separate models: one for when the friendly team is "Terrorists" and another for when the friendly team is "Counter-Terrorists." This approach results in some asymmetry in the detection performance between the two classes. The data shown in the previous two figures was obtained using the model trained when the friendly team was "Terrorists." Results for the model trained when the friendly team was "Counter-Terrorists" showed complementary outcomes, with the "Counter-Terrorist" team being detected more effectively. The differences, however, are minor. We chose to present the results from the "Terrorist" team model because it yielded slightly better results. In future work, it may be more beneficial to average the results across the two team-based models for each class.

Lastly, it is important to emphasize that the most relevant dataset for comparison is the "Bots CT and T Combined," as it most accurately represents competitive environments simulating real matches with bot opponents and teammates. The other datasets introduce more variability in the form of custom player cosmetics, which our generator does not currently support. However, this is unnecessary if our goal is to use our object detector exclusively for bot matches.

A.7 Future Work and Proposed Multi-Model Architecture

Proposed Multi-Model Agent Architecture: We propose a comprehensive multi-model architecture that could serve as the foundation for developing autonomous agents capable of high-level gameplay in Counter-Strike:

(1) **Object Detection for Enemy Players**: This model functions as the agent's eyes, focusing on detecting and classifying enemy players, enabling the agent to focus on shooting accuracy and map traversal.
(2) **Reinforcement Learning for Aiming and Shooting**: Acting as the agent's arms, this model uses reinforcement learning to aim and shoot at detected enemies, adjusting its behavior based on skill levels.
(3) **Object Detection on the In-Game Minimap**: This complementary model identifies player positions on the in-game minimap, providing additional spatial awareness.
(4) **Decision Making for Player Movement**: Utilizing minimap data, this model determines the agent's movement strategy, optimizing its position on the map through supervised or reinforcement learning.
(5) **3D Environment Modeling and Detection**: Enhancing environmental perception, this model employs techniques like SLAM or MiDaS to build a 3D understanding of the game world.
(6) **Dynamic Map Traversal**: Leveraging the 3D environment model, this model navigates the map dynamically, utilizing pathfinding algorithms or reinforcement learning to simulate player inputs.

Cross-Lingual False Friend Classification via LLM-based Vector Embedding Analysis

Mitko Nikov
mitko.nikov@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Žan Tomaž Šprajc
zan.sprajc@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Žan Bedrač
zan.bedrac@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

In this paper, we propose a novel approach to exploring cross-linguistic connections, with a focus on false friends, using Large Language Model embeddings and graph databases. We achieve a classification performance on the Spanish-Portuguese false friend dataset of F1 = 83.81% using BERT and a multi-layer perceptron neural network. Furthermore, using advanced translation models to match words between vocabularies, we also construct a ground truth false friends dataset between Slovenian and Macedonian - two languages with significant historical and cultural ties. Subsequently, we construct a graph-based representation using a Neo4j database, wherein nodes correspond to words, and various types of edges capture semantic relationships between them.

KEYWORDS

false friends, large language models, BERT, linguistics, natural language processing

1 INTRODUCTION

When observing individual languages, we come across homonyms, which are words that have the same spelling or pronunciation but varied meanings, such as the word "bat", which pertains to either the animal or the sports requisite. As we move from the confines of one language and observe two, we encounter chance false friends [10]. These have the same spelling but varied etymologies and meanings in different languages, such as the English word "in", which in Slovenian means "and". So, we decided to pivot our observation further and focus solely on words that have the same etymological origin and spelling whilst having different meanings in different languages, so-called semantic false friends [10, 12, 16].

A similar endeavour was undertaken by Ljubešić & Fišer [13], which attempted to identify true equivalents, partial false friends, and false friends in Slovenian and Croatian based on their spelling and semantic meaning. Our analysis will also touch on true equivalents (word pairs with the same meaning and usage [13]), partial false friends (pairs that alternate between polysemy and false friends [13]), and pure false friends.

An initial step to finding false friends could be lemmatization-based tagging [4], which is able to differentiate between parts of speech, reducing words to their root form. Which in practice means that a verb like "working" is reduced to its root of "work". Stemming is another alternative, which has already been applied to Czech together with a language-independent approach (n-gram) [9]. However, even though lemmatization proved effective for two other South Slavic languages, Croatian and Serbian [4], in our case, we expect the declension differences between Slovenian and Macedonian to be too significant for such a preprocessing step to be used.

A recently introduced method for automatic false friends detection in related languages [6] uses a linear transformation between the two vector spaces in both languages to isolate false friends. The linear transformation acts as a translation between the two languages. They [6] expect that one vector in one language should be close to its cognate partner [5] in the other language after the linear transformation, however, for false friends, this should not be the case. They use the Spanish and Portuguese Wikipedia as a corpus for the unsupervised learning of the Word2Vec models [15].

Since the linear transformation is a bijection, each vector in one of the languages is uniquely mapped to a vector in the other language. It is impossible for such a model to account for the different meanings one word can have. To solve this issue, we propose an improvement to this method by extending the vector space to use LLM embeddings [18] of meanings instead of single words.

Regarding the false friend classification between Macedonian and Slovenian, we needed to take a different approach to ground truth dataset creation. Our approach is based on finding words with the same spelling in Slovenian and Macedonian, translating them to English using a pre-trained bidirectional translator API and matching false friends accordingly. This approach also yields an unexpected amount of true friends, which are also useful to us. A prime example of a false friend would be the word "obraz", which in Slovenian means "face" and in Macedonian means "cheek". On the other hand, a true friend would be the word "jagoda", which means "strawberry" in both Slovenian and Macedonian. These and a few other examples are given in Table 1.

In the following sections, we will describe the methodology that we used to classify false friends, as well as the methodology used to create a ground truth dataset for Slovenian and Macedonian false friends. Our overview of the classification process will be based on BERT as a word to vector model, with special attention given to the embedding extraction process. Moreover, we will dive into our methodology for ground truth creation, with a special emphasis on the issues that result from the translation of words that have multiple meanings. We will also describe the graph database representation of the false friends dataset. Finally, we will present our results, comparing our methodology to an existing one. We will evaluate our results in terms of precision, recall, F1 scores, and provide a summary of our false friends ground truth dataset.

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.7
ISBN 978-961-286-914-4

Table 1: Examples of true and false friends in the Slovenian and Macedonian language.

Slovenian word	Macedonian word	Slovenian meaning	Macedonian meaning	Type of word match
obraz	образ (obraz)	face	cheek	false friend
lice	лице (lice)	cheek	face	false friend
deka	дека (deka)	blanket	that	false friend
čas	час (čas)	time	time/hour	partial false friend
jagoda	јагода (jagoda)	strawberry	strawberry	true friend
kraj	крај (kraj)	edge/end/region	edge/end/region	true friend

2 METHODOLOGY

Recently, Large Language Models (LLMs) and advanced tokenizers have revolutionized our understanding of language technologies and made significant advancements in the field. Their ability to create incredibly complex and rich context-based vector spaces opens a new area of analysis. Now, we are no longer limited by Word2Vec models but can analyze the vast variety of contextual meanings of individual words.

Thus, our first improvement of the method presented by Castro et al. [6] comes with the introduction of LLM embeddings instead of Word2Vec models. We use the pre-trained BERT Multilingual LLM [8] to extract the embeddings of tokens in our training datasets as shown in Figure 1.

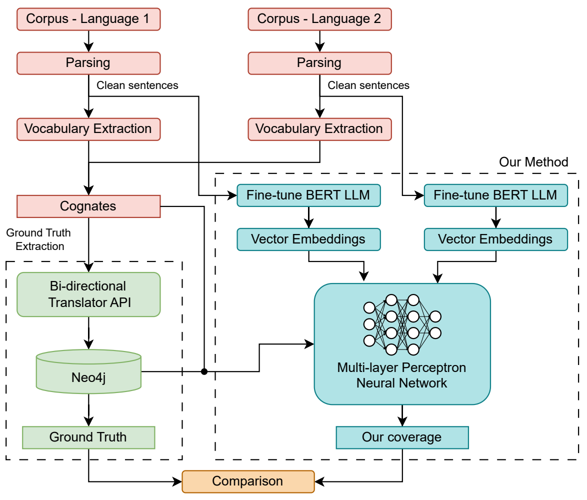


Figure 1: Our methodology

2.1 BERT as a Word to Vector Model

BERT (Bidirectional Encoder Representations from Transformers) [8] is a transformer-based model that has set new benchmarks in a variety of natural language and cross-language processing tasks [17]. Unlike previous models that processed text in a unidirectional way, BERT reads text bidirectionally, understanding the context of a word based on both its left and right surroundings. This bidirectional approach allows BERT to generate highly contextualized word embeddings.

The BERT transformer consists of multiple layers, where each layer is capable of capturing different aspects of the word's context. When we input a sentence into BERT, it tokenizes the sentence into subword units (tokens), processes these units through its multiple layers, and produces embeddings for each token at each layer. These embeddings are rich in context and can capture the nuances of word meanings in different sentences.

2.2 Embedding Extraction Process

To utilize BERT embeddings, we follow a systematic approach to extract and aggregate these embeddings:

(1) **Tokenization**: Using BERT's tokenizer, we split each word into its constituent subword tokens. This step ensures that even unfamiliar words or misspellings can be processed effectively by BERT.
(2) **Contextual Embedding Extraction**: We pass these tokens through the pre-trained BERT model to obtain embeddings. Since BERT embeddings are context-dependent, the same word can have different embeddings based on its surrounding words.
(3) **Averaging Token Embeddings**: For words split into multiple tokens, we compute the final word embedding by averaging the embeddings of all its constituent tokens. This aggregated embedding represents the word in its specific context within the sentence.

2.3 Classification of False-Friends

Instead of creating a linear transformation between vector spaces, we use the embeddings of pairs of words (correlated in both of the languages) and a training dataset of already classified pairs such as the Spanish-Portuguese dataset [7] to train a multi-layer perceptron neural network to classify pairs of unseen words as false or true friends.

The resulting embedding vector for each word is computed as the average of BERT's internal embeddings for each token comprising the word. Thus, leveraging the fixed internal embedding dimensions of BERT, where each token is represented by a vector $v \in \mathbb{R}^{768}$ space. We found that a simple dense neural network is enough for our methodology. This neural network has two hidden layers of 2000 neurons each, enough to get satisfiable results.

2.4 Creation of Ground Truth Dataset

To extend the evaluation of our method, we needed to create a new ground truth dataset, which would consist of a collection of true and false friends. The prerequisite for obtaining said collection was processing a Slovenian [1] and a Macedonian corpus [3]. The former was obtained from The Slovenian Academy of Sciences and Arts, while the latter was obtained from the University of Leipzig. The Slovenian corpus was an official list of unique Slovenian words of 354205 different headwords, while the Macedonian corpus consisted of 350921 words obtained from Wikipedia. The latter was processed using our unique word extractor, which resulted in a unique word count of 248083.

These two lists of unique words were then further processed in order to extract homographs, which are words with the same spelling, in this case pertaining to Slovenian and Macedonian words. An initial hurdle was the difference in alphabets between the two languages. Slovenian uses the Latin alphabet, while Macedonian uses the Cyrillic alphabet. To overcome this, we transliterated the Macedonian corpus into the Latin alphabet. This allowed us to compare the two lists of unique words. We based our homograph extraction on a Levenshtein distance of 0, which meant that we only extracted homographs that were identical in spelling. Our extraction of homographs thus produced 21674 homographs and 226409 non-homographs. This stage of our corpus processing thus left us with 21674 candidates for false and true friends.

Our next step was to translate each Slovenian and Macedonian homograph to English and compare their English meanings. Those homographs that produced the same meaning were categorized as true friends, while the rest were categorized as false friends. This stage of our research made apparent a flaw in our translation API. The flaw being Google's translation API [2], which only returns one translation. Moreover, the limited scope of Slovenian and Macedonian, compounded by interjections, meant that some translations were inaccurate. Said inaccuracies then resulted in false positives, which were apparent in our Neo4j database.

The Neo4j database that we filled with Macedonian words, Slovenian words, true friends, and false friends was the backbone of our visualization. The latter helped us identify potential problems with our approach, such as improper false friend connections, example given in Figure 2a, and true friend connections due to limited responses from the Google Translate API, example given in Figure 2b.

Our analysis of the Neo4j database thus yielded a lot of food for thought. An appealing approach was the classification of false friends into segregation (pairs carrying absolutely different meanings), lexical pairs (both similar and dissimilar meanings), and inclusion (one dissimilar meaning on top of all other similar meanings) as outlined in [11]. But we decided to stick with the binary classification of false and true friends.

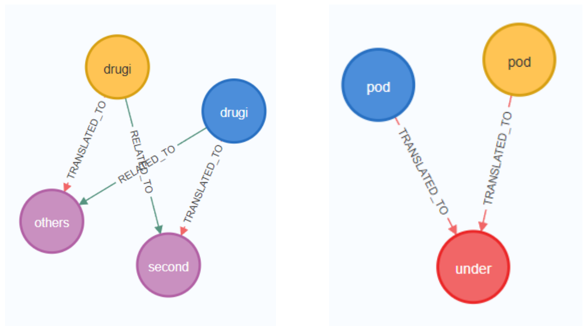
3 RESULTS

To compare and benchmark our approach, we recreated the results of the method used by Castro et al. [6]. Their method included acquiring the then-newest Wikipedia dumps (dated 20.03.2024), parsing them, and training the two Word2Vec models [14] in Spanish and Portuguese. Each model with a vector dimension of 100 took an hour and a half to train on 30 million sentences, after which we could finally derive the linear transformation necessary for translation. Their paper evaluates the method by classifying a pair of words as true or false friends given a ground truth dataset between Spanish and Portuguese [7]. Our re-testing of their method achieved the results shown in Table 2.

Our proposed method, however, showed a significant improvement on the Spanish-Portuguese dataset with an F1-score of 0.8381, shown in Table 3, without any fine-tuning on the pre-trained BERT Multilingual model. Moreover, because of the pretrained BERT model, our method including the extraction of embeddings and the training of the neural network to learn the classification of the words took under 10 minutes as opposed to the training time of the Word2Vec models of around 3 hours.

Table 2: Classification performance using the Castro et al. [6] approach on the Spanish and Portuguese dataset.

	Precision	Recall	F1-Score
False	0.7727	0.7730	0.7721
True	0.7446	0.7424	0.7427
Average	0.7586	0.7577	0.7574



(a) (b)

Figure 2: (a) The Slovenian word "drugi" could be translated as "others", which would match it with the Macedonian word "drugi" (други), or translated as "second", which results in a false friend. "Drugi" is, therefore, only 50% a false friend. (b) The Macedonian word "pod" (под) could, likewise, be alternatively translated as floor, which means that it could potentially be a false friend.

Table 3: Comparison of our and Castro et al. methodology. Note that we are not using any additional sentences for fine-tuning the Multilingual BERT Model.

F1	Castro et al.		Ours
Sentences	**30M**	**200K**	**0**
False	0.7721	0.7324	**0.8505**
True	0.7427	0.5783	**0.8258**
Average	0.7574	0.6554	**0.8381**

Table 4: Classification performance of our approach on our Macedonian and Slovenian false and true friends datasets without fine-tuning of the BERT model.

	Precision	Recall	F1-Score
False	0.8868	0.8393	0.8624
True	0.7097	0.7857	0.7458
Average	0.7982	0.8125	0.8041

Our method of extracting homographs from our Slovenian and Macedonian corpus yielded 21674 candidates for false/true friends. Further analysis using comparisons of associated English meanings resulted in 14654 true and 7020 false friends. However, some of these were false positives due to the multi-meaning nature of various words. A manual review of the 7020 false friends gave us 151 ideal false friends that are largely free of true friend overlap. We used these 151 ideal false friends as our Slovenian-Macedonian false friends dataset. The false friends manual review was followed by an extraction of 268 true friends from our initial set of 14654 true friends. These 268 true friends then comprised our Slovenian-Macedonian true friends dataset.

Using our Slovenian-Macedonian dataset of false and true friends[1], we achieved similar classification capabilities as with the Spanish-Portuguese dataset. Our results can be seen in Table 4. All experiments were run on an Intel Core i7-9700K @ 3.60GHz and GeForce RTX 2070 SUPER GPU.

4 CONCLUSIONS

In this paper, we presented a novel approach to exploring cross-linguistic connections, specifically focusing on false friends, using Large Language Model embeddings and graph databases. Our methodology leverages the advanced capabilities of BERT for generating contextualized word embeddings and a graph-based representation to capture semantic relationships. We achieved classification performance on the Spanish-Portuguese false friend dataset with an F1 = 83.81% and classification performance on our Slovenian-Macedonian dataset of F1 = 80.41% using Multilingual BERT and a multi-layer perceptron neural network. BERT was not fine-tuned using any additional sentences.

Our results indicate that LLM embeddings significantly enhance the accuracy of false friend classification compared to traditional Word2Vec models. The use of a pretrained LLM also significantly reduced the time it takes to learn the classifications from 3 hours needed to train the Word2Vec models to under 10 minutes solely for the training of the multi-layer perceptron classifier. This highlights the potential of using sophisticated language models for even more complex linguistic tasks, paving the way for more accurate and insightful cross-linguistic analysis.

A natural next step to enhancing our methodology would be incorporating larger and more diverse corpora. These would fine-tune the pre-trained BERT model on specific language pairs or domains, improving the contextual accuracy of embeddings. Moreover, larger corpora would yield additional false and true friends in our ground truth dataset. More advanced translation APIs would be capable of providing multiple translations for each word, which would result in fewer false positives when creating such a dataset.

Furthermore, extending the methodology to other cross-linguistic phenomena, such as idiomatic expressions, cognates, and loanwords, would improve our understanding of language relationships. False and true friends are, therefore, the tip of a linguistic iceberg that calls for further exploration.

[1]The datasets are available at https://github.com/mitkonikov/false-friends

REFERENCES

[1] 2013. ISJ SAZU - List of Slovenian words. http://bos.zrc-sazu.si/sbsj_en.html [Online; accessed 2. Jun. 2024].

[2] 2024. Cloud Translation API | Google Cloud. https://cloud.google.com/translate/docs/reference/rest [Online; accessed 2. Jun. 2024].

[3] 2024. Wortschatz Leipzig Macedonian Corpora. https://wortschatz.uni-leipzig.de/en/download/Macedonian [Online; accessed 2. Jun. 2024].

[4] Željko Agić, Nikola Ljubešić, and Danijela Merkler. 2013. Lemmatization and Morphosyntactic Tagging of Croatian and Serbian. *ACL Anthology* (Aug. 2013), 48–57. https://aclanthology.org/W13-2408

[5] E. Susanne Carroll. 1992. On cognates. *Sage Journals* 8 (jun 1992). Issue 2. https://journals.sagepub.com/doi/abs/10.1177/026765839200800201

[6] Santiago Castro, Jairo Bonanata, and Aiala Rosá. 2018. A High Coverage Method for Automatic False Friends Detection for Spanish and Portuguese. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, Marcos Zampieri, Preslav Nakov, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali (Eds.). Association for Computational Linguistics, Santa Fe, New Mexico, USA, 29–36. https://aclanthology.org/W18-3903

[7] María de Lourdes Otero Brabo Cruz. 2004. Diccionario de falsos amigos (espanol-portugues / portugues-espanol). https://ec.europa.eu/translation/portuguese/magazine/documents/folha47_lista_pt.pdf

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ACL Anthology* (June 2019), 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Ljiljana Dolamic and Jacques Savoy. 2009. Indexing and stemming approaches for the Czech language. *Information Processing & Management* 45, 6 (Nov. 2009), 714–720. https://doi.org/10.1016/j.ipm.2009.06.001

[10] Pedro Domínguez and Brigitte Nerlich. 2002. False friends: Their origin and semantics in some selected languages. *Journal of Pragmatics* 34 (Dec. 2002). https://doi.org/10.1016/S0378-2166(02)00024-3

[11] Ketevan Gochitashvili and Giuli Shabashvili. 2018. The issue if "false friends" in terms of learning a foreign language(Using the example of Georgian and English languages). *International Journal Of Multilingual Education* VI (July 2018), 33–41. https://doi.org/10.22333/ijme.2018.11006

[12] Diana Inkpen and Oana Frunza. 2005. Automatic Identification of Cognates and False Friends in French and English. *ResearchGate* (Jan. 2005). https://www.researchgate.net/publication/237129220_Automatic_Identification_of_Cognates_and_False_Friends_in_French_and_English

[13] Nikola Ljubešić and Darja Fišer. 2013. Identifying false friends between closely related languages. *ACL Anthology* (Aug. 2013), 69–77. https://aclanthology.org/W13-2411

[14] Long Ma and Zhang Yanqing. 2015. Using Word2Vec to Process Big Text Data. (Oct. 2015). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7364114

[15] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. 1–12. https://www.researchgate.net/publication/319770439_Efficient_Estimation_of_Word_Representations_in_Vector_Space

[16] Ruslan Mitkov, Viktor Pekar, Dimitar Blagoev, and Andrea Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine Translation* 21, 1 (March 2007), 29–53. https://doi.org/10.1007/s10590-008-9034-5

[17] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4996–5001. https://doi.org/10.18653/v1/P19-1493

[18] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving Text Embeddings with Large Language Models. *arXiv* (Dec. 2023). https://doi.org/10.48550/arXiv.2401.00368 arXiv:2401.00368

Analyzing tourist destinations in Belgrade using geotagged photos from Flickr

Vera Milosavljević
vera.milosavljevic@famnit.upr.si
University of Primorska,
Faculty of Mathematics,
Natural Sciences and Information Technologies,
Department of Mathematics,
Koper, Slovenia

Dejan Paliska
dejan.paliska@fts.upr.si
University of Primorska,
Faculty of Tourism,
Department for Sustainable
Destination Development,
Koper, Slovenia

ABSTRACT

This research aims to analyze tourist destinations in Belgrade by defining trajectories of movement of the users of platform Flickr using geotagged photos on Flickr. We defined tourist movements and used generalization techniques to identify the main tourists locations. We applied several techniques to identify frequently visited locations and predict next possible tourist spots. Our findings provide insights into popular travel patterns and suggest potential areas for tourism development.

KEYWORDS

Tourism data analysis, active user bias, geotagged photo, DBSCAN clustering, trajectory generalization

1 INTRODUCTION

Tourism is a major contributor to the global economy, offering economic benefits and fostering cultural exchange. With the growth of social media and mobile phones usage, tourists now document their journeys through geotagged photos, sharing their experiences with a wide audience. Flickr, a popular photo-sharing platform, contains a wide repository of such geotagged images and publicly aveilable API which makes it a good choice for our analysis. The users of the application are not rarely professional photographers and their focus lies in architecture, nature and country's most beautiful destinations.

Analyzing these photos provides valuable insights into tourists' behavior at destinations, and particularly into their space-time patterns. Traditional tourism data often relies on surveys and official records, which can be bias and limited in scope. In contrast, social media data offers real-time, user-generated information that reflects actual tourist activities and interests. Flickr has many active users who contribute to the platform daily.

In this paper, we focused on Belgrade as a popular tourist destination, but the insights this paper provides can be applicable to any location that has a rich dataset of photos on the platform Flickr.

2 OBJECTIVES

This research has the following objectives:

(1) Collecting geotagged photos from Flickr by using the publicly available Flick API.
(2) Data preprocessing. Defining the set of variables that are important for further analysis.
(3) Generating trajectories of movement by using geo-coordinates for each user.
(4) Trajectory analysis and visualisation for better understanding of the movements.
(5) Clustering, aggregation and generalisation of trajectories.
(6) Prediction of the next tourist movement based on several different modeling tecniques.

3 SIGNIFICANCE

Understanding the location preferences of visitors is crucial for local tourism organizations, travel agencies, and other stakeholders involved in destination development. Information about local attractions, visitor mobility, and intradestiation movement patterns can help with strategic planning, improve destination marketing, and enhance connectivity between at- tractions. By using modern technology and user-generated content, such as geotagged photos from social media, re- searchers can better understand visitor patterns and behaviors. In Belgrade, the relationships between tourist destinations and visitor mobility are under-researched, thus this study aims to address these gaps by analyzing data from Flickr platform. This will allow us to identify tourists POIs (clusters), and their mobility patterns at destinations. Additionally, while Flickr data could also be utilized to analyze tourists' emotions and destination image, these aspects are beyond the scope of this study. This research contributes to the academic field by demonstrating the application of data mining techniques in tourism analytics.

4 STRUCTURE

The rest of this paper is structured as follows: Section V details the methodology, including data collection, preprocessing, clustering, aggregation and generalisation, and the prediction techniques used. Section VI presents the results of our analysis, highlighting key findings and patterns. Section VII concludes the paper and suggests directions for future research.

5 METHODOLOGY

5.1 Tools and Technologies

All data processing and analysis were done using Python programming language, using some of the many data science libraries. The following tools were used:

- Jupyter Notebook: Computing environment that was used for the development and documentation of the code along with miniconda installer and command line.

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.8
ISBN 978-961-286-914-4

- Pandas: Used for data manipulation and analysis.
- Numpy: For numerical computations.
- Scikit-learn: Applied for clustering and other machine learning tasks.
- Geopandas: Used for geographic data processing and visualization.
- MovingPandas: Used for spatial analysis and manipulation of geospatial data.
- Mlxtend: Employed for the implementation of the Apriori algorithm and association rule mining.
- SeqMining: Applied for sequence mining to find frequent sequences.
- Matplotlib: Used for data visualization.

5.2 Data Collection

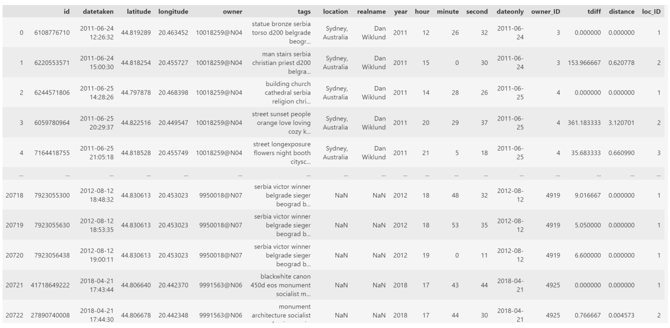
We collected 31019 geotagged photos from 1233 different users from the Flickr application. We used the Flickr public API which was granted by a unique key after becoming a user of the Flickr app. The dataset was focused on the territory of Belgrade, by using the tag "belgrade". Each photo's metadata, including the geocoordinates and timestamps, was extracted and used for further analysis.

5.3 Data Preprocessing

The preprocessing step involved cleaning and structuring the collected data to make it suitable for analysis. We defined a variable ownerID, which represents one user in one day. We defined a variable locID which represents different locations for each user. Algorithm for defining locID is shown as Algorithm 1.

The motivation for defining this variable is because we wanted to tackle the problem of an active user: a situation where one user generates many photos of the same location. We wanted to treat these photos as a single group, with the same value of locID variable. This would ensure that our analysis would be less bias.

The reduced dataset had a structure as shown in the Fig. 1 . We dropped rows which were out of boundaries of Belgrade. The rows included must have longitude variable value between 20.35 and 20.65 and latitude variable value between 44.7 and 45.1 to be considered as a photo captured on the territory of Belgrade. The final reduced dataset had 20723 rows from 550 users.



	id	datetaken	latitude	longitude	owner	tags	location	realname	year	hour	minute	second	dateonly	owner_ID	tdiff	distance	loc_ID
0	6108776710	2011-06-24 12:26:32	44.819289	20.463452	10018259@N04	statue bronze serbia torso d200 belgrade beogr...	Sydney, Australia	Dan Wiklund	2011	12	26	32	2011-06-24	3	0.000000	0.000000	1
1	6220553571	2011-06-24 15:00:30	44.818254	20.455727	10018259@N04	man stairs serbia christian priest d200 belgra...	Sydney, Australia	Dan Wiklund	2011	15	0	30	2011-06-24	3	153.966667	0.620778	2
2	6244571806	2011-06-25 14:28:26	44.797878	20.468398	10018259@N04	building church cathedral serbia religion chri...	Sydney, Australia	Dan Wiklund	2011	14	28	26	2011-06-25	4	0.000000	0.000000	1
3	6059780964	2011-06-25 20:29:37	44.822516	20.449547	10018259@N04	street sunset people orange love loving cozy k...	Sydney, Australia	Dan Wiklund	2011	20	29	37	2011-06-25	4	361.183333	3.120701	2
4	7164418755	2011-06-25 21:05:18	44.818528	20.455749	10018259@N04	street longexposure flowers night booth citysc...	Sydney, Australia	Dan Wiklund	2011	21	5	18	2011-06-25	4	35.683333	0.660990	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20718	7923055300	2012-08-12 18:48:32	44.830613	20.453023	9950018@N07	serbia victor winner belgrade sieger beograd b...	NaN	NaN	2012	18	48	32	2012-08-12	4919	9.016667	0.000000	1
20719	7923055630	2012-08-12 18:53:35	44.830613	20.453023	9950018@N07	serbia victor winner belgrade sieger beograd b...	NaN	NaN	2012	18	53	35	2012-08-12	4919	5.050000	0.000000	1
20720	7923056438	2012-08-12 19:00:11	44.830613	20.453023	9950018@N07	serbia victor winner belgrade sieger beograd b...	NaN	NaN	2012	19	0	11	2012-08-12	4919	6.600000	0.000000	1
20721	41718649222	2018-04-21 17:43:44	44.806640	20.442370	9991563@N06	blackwhite canon 450d eos monument socialist m...	NaN	NaN	2018	17	43	44	2018-04-21	4925	0.000000	0.000000	1
20722	27890740008	2018-04-21 17:44:30	44.806678	20.442348	9991563@N06	monument architecture socialist	NaN	NaN	2018	17	44	30	2018-04-21	4925	0.766667	0.004573	2

Figure 1: Reduced dataset

```
Data: database with columns 'owner_ID' and 'distance'
Result: database with 'loc_ID' column assigned to every
        row defining the location group
/* Define thresholds                                    */
1  dist_threshold = 0.0005 cumulative_dist_threshold = 0.001
/* Set initial loc_ID for each row                      */
2  final_database["loc_ID"] = 1
/* Group data by owner_ID and list distances            */
3  owner_ID_map = group database by owner_ID and
     aggregate distances;
/* Initialize row index to zero                         */
4  i = 0;
5  foreach key in owner_ID_map do
6      prev = 1 /* Initial loc_ID for current owner_ID
           */
7      cumulative_dist = 0 /* Cumulative distance for
           current loc_ID                               */
8      foreach list_value in owner_ID_map[key] do
9          cumulative_dist = cumulative_dist + list_value ;
            /* Accumulate distance */
10         if list_value > dist_threshold or cumulative_dist >
            cumulative_dist_threshold then
11             prev = prev + 1
               /* Increment loc_ID                      */
12             cumulative_dist = 0
               /* Reset cumulative distance             */
13         end
14         database.loc[i, "loc_ID"] = prev
           /* Update loc_ID for current row             */
15         i = i + 1
           /* Move to next row                          */
16     end
17 end
```

Algorithm 1: loc_ID Algorithm

5.4 Clustering

Various clustering techniques were explored to identify clusters of frequently visited locations. We experimented with DBSCAN, HDBSCAN, and OPTICS algorithms. The centroids of the clusters were identified using mean and median methods. We created visualizations to show the clustering results, trajectories, and connections between clusters.

5.5 Generalization and aggregation

Generalization techniques were applied using the Geopandas library and Douglas-Peucker Generalizer [5]. Different threshold parameters for tolerance were tested to observe the effects on generalization. A comparison was made between Douglas-Peucker Generalizer and Top-Down Time Ratio Generalizer. We defined trajectory collection using the MovingPandas [3] library which is a collection of all trajectories for each OwnerID. We used this collection as an imput parameter for DouglasPeuckerGeneralizer algorithm also defined in MovingPandas library. Fig. 2 shows the trajectories for each OwnerID. We used TrajectoryCollectionAggregator [4] from MovingPandas library to get significant points and flows between them. The results will be show in the section Results.

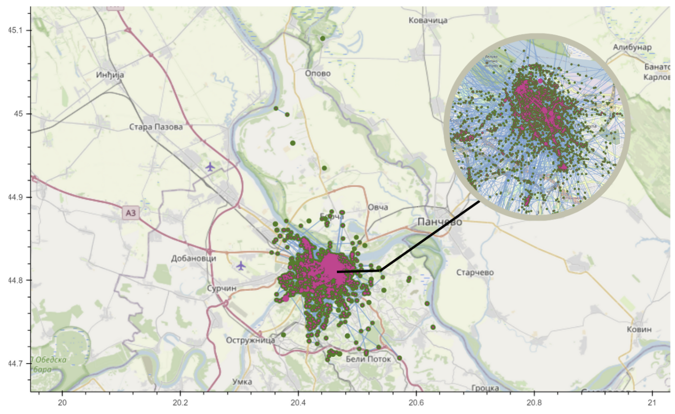


Figure 2: Trajectories for each OwnerID

5.6 Predictive Modeling

In the modeling prediction, our approach was to predict the next cluster in our spatial data analysis based on the set of visited clusters. The clustering technique that we choose was HDBSCAN because we wanted to focus on smaller clusters of tourist destinations within the city. Initially, we defined individual paths for each owner based on their clusters. We excluded outliers marked as -1 (noise) and grouped the remaining data by owner ID. Additionally, we defined a DataFrame with each owner's ID and their corresponding cluster paths. After computing the unique sequence of clusters for each owner, we generated trigrams [2] to identify recurring patterns within the paths. We defined the transition matrix and we normalized transition counts to derive probabilities of transitioning from one cluster to another. Also, we developed functions for predicting the next cluster using Markov chains, Monte Carlo simulations, and association rules derived from Apriori analysis.

6 RESULTS

6.1 Clustering

Here we will present the visualisation of the results of the clustering using HDBSCAN algorithm and median method for calculating the centroid of each cluster. Each different color in Fig. 3 represents different cluster. We calculated the number of transitions between each cluster (If one person visited cluster 1 and than after that they visited cluster 3, we would count that as 1 transition between clusters 1 and 3). Fig. 4 shows the connections between clusters. Purple points represent the centroids of the clusters, thickness of the line and the number represent how many connections are between two clusters..

6.2 Generalization

The Fig. 5 shows different thresholds for generalization by DouglasPeckuer. By analyzing these input parameters, we opted for

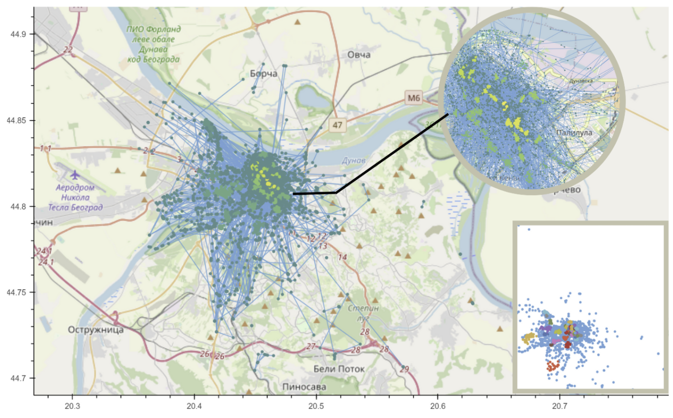


Figure 3: Clustering

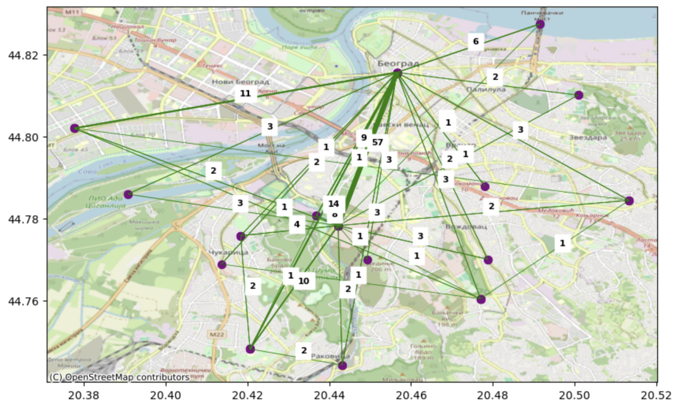


Figure 4: Connections between clusters

threshold of 0.01 as the most suitable for our research. We noticed a small difference between original and generalized trajectories of movements because our original trajectories were already filtered and reduced. We also experimented with TopDownTimeRatio Generalizer. The comparison for a singular trajectory when using DP Generalizer and TDTR Generalizer can be seen in Fig. 6.

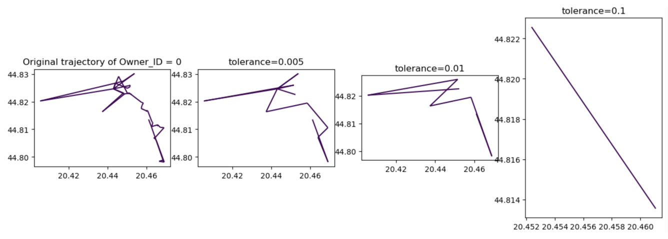


Figure 5: Different thresholds for a singular trajectory

6.3 Aggregation

The input parameter for TrajectoryCollectionAggregator was generalized trajectories with DouglasPeckuer Generalizer as instructed in

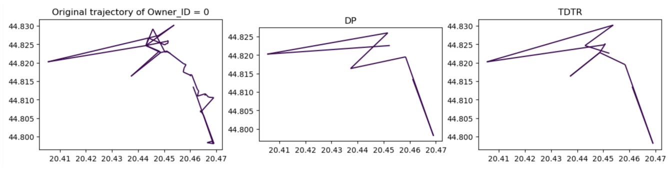


Figure 6: Comparison between TDTR and DP for a singular trajectory

the documentation of the MovingPandas library. Siginficant points and the aggregated flows between them by using this method are shown in the Fig. 7. We can observe that this method of reduction gave us similar results as clustering, identifying the centre of the city with park Kalemegdan as the most popular tourist area.

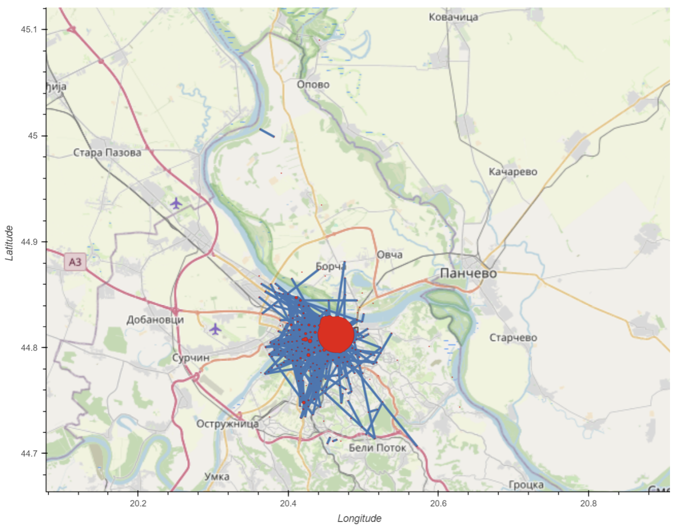
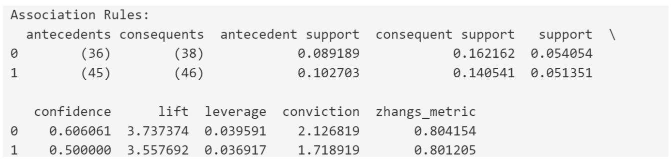


Figure 7: Aggregated flows and more popular areas

6.4 Predictive Modeling

The Markov chain analysis provided insights into the probabilities of transitions between clusters. By normalizing transition counts into probabilities, we gained insights into the likelihood of tourists moving from one cluster to another. Through Markov chain modeling, we determined the most probable next cluster after a given current cluster. For instance, after analyzing a sequence with cluster 5 which corresponds to New Belgrade, the predicted next cluster was 7, which corresponds to Zemun. This aligns geographically since these 2 locations are close. Additionally, employing Monte Carlo simulations, we had more trials to predict future clusters based on starting clusters. Lastly, by using association rules derived from Apriori [1] analysis, we predicted the next cluster by identifying the longest subsequence matching antecedents in the rules. For instance, when the current sequence contained cluster 36 (representing the Victor statue in Kalemegdan), the predicted next cluster was 38 (representing Roman well in Kalemegdan). Example output of the Apriori algorithm can be seen in Fig. 8.



```
Association Rules:
  antecedents consequents  antecedent support  consequent support   support  \
0        (36)        (38)            0.089189            0.162162  0.054054
1        (45)        (46)            0.102703            0.140541  0.051351

   confidence      lift  leverage  conviction  zhangs_metric
0    0.606061  3.737374  0.039591    2.126819       0.804154
1    0.500000  3.557692  0.036917    1.718919       0.801205
```

Figure 8: First two rows of Apriori algorithm output

7 CONCLUSION

This research introduces some improvements in analyzing tourist activities at destinations by using geotagged photos from Flickr, focusing on tourist movement patterns in Belgrade. A key innovation of this study is the development and application of the locID algorithm, which identifies unique tourist POIs and deals with the issue of overrepresentation from highly active users. This approach ensures that the analysis is more representative and less biased compared to other methods. The clustering results provided a view of frequently visited locations, offering valuable insights into the spatial distribution of tourist activity. By employing generalization techniques, the study effectively simplified tourist trajectories, making the data easier to interpret. Predictive modeling techniques, such as Markov chains, Monte Carlo simulations, and the Apriori al- gorithm, were used to predict future tourist movements. These methods demonstrated their potential for practical applications in tourism management, enabling more targeted marketing strategies and improved visitor experience planning.

ACKNOWLEDGMENT

The author would like to thank mentor professor Dejan Paliska for the support, cooperation and availability during the whole process of the research. Also, the University of Primorska and representative for this seminar prosessor Miklos for supporting this project.

REFERENCES

[1] P. Fournier-Viger, "SPMF: An Open-Source Data Mining Library," [Online]. Available: https://www.philippe-fournier-viger.com/spmf/. [Accessed: 03-Jun-2024].
[2] J. Silge and D. Robinson, "Text Mining with R: A Tidy Approach," O'Reilly Media, 2017. [Online]. Available: https://www.tidytextmining.com/ngrams. [Accessed: 03-Jun-2024].
[3] A. Graser, "MovingPandas: A Python Library for Movement Data Analysis," [Online]. Available: https://movingpandas.readthedocs.io/en/main/. [Accessed: 03-Jun-2024].
[4] A. Graser, "Trajectory Aggregator in MovingPandas," [Online]. Available: https://movingpandas.readthedocs.io/en/main/trajectoryaggregator.html. [Accessed: 03-Jun-2024].
[5] A. Graser, "Movement Analysis Tools on GitHub," [Online]. Available: https://github.com/anitagraser/movement-analysis-tools. [Accessed: 03-Jun-2024].

Volleyball Game Analysis Using Computer Vision Algorithms

Marko Plankelj
marko.plankelj@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Uroš Mlakar
uros.mlakar@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

In recent years, modern technologies have made sports more accessible to a wider audience by providing interactive data during broadcasts, reducing the risk of human error, and enhancing athletes' performance through real-time analysis and targeted training insights. This paper combines theoretical and practical approaches by developing an application based on specific convolutional neural networks for volleyball court detection and ball tracking. The results demonstrate the potential of advanced video analytics in sports, allowing users to explore the opportunities of modern technology in improving sports performance.

KEYWORDS

computer vision, convolutional neural networks, perspective transformation, volleyball, web application

1 INTRODUCTION

For people, sport has always been a form of relaxation, socializing, an opportunity to find new friendships with proven beneficial for health. The World Health Organization recommends playing sports as part of a healthy lifestyle in a program that aims to bring people closer to physical activity as an opportunity for a healthier, happier and more productive life [11]. On the other hand, we monitor the development of modern technologies and their use in a wide variety of fields, including sports, from day to day.

Sport and modern technologies such as computer vision and machine learning were completely opposite terms a few years ago, but nowadays we can hardly imagine watching or participating in sports without the inclusion of modern technologies[5]. Despite many challenges, such as poorer video quality or overlapping players, the use of systems for detecting the field, players and their actions, and tracking the ball during play is becoming more common. Whether it's a real-time analysis of the opposing team, which helps coaches find winning tactics, or simply watching a sports broadcast, during which the directors serve interactive slow-motion footage of the actions and statistical data about the players. By using them, they want to prevent controversial situations in matches, improve training and competitor analysis, predict loads in training and matches with the aim of preventing injuries, and improve the experience of spectators with analysis before, during and after the match.

2 PROBLEM DESCRIPTION

The main objective of this paper was to develop a web application that enables the users to select a video of a volleyball game and then, with the help of computer vision algorithms, automatically detect the volleyball court and analyze the trajectory of the ball, and displaying the results through a perspective projection of the court. The utilized computer vision algorithms in this paper were convolutional neural networks (CNN).

The addressed challenge can be broken down into four steps: 1) preparation of the training data, which involved collecting, preprocessing and labeling the data, which we later augmented with the aim of increasing the diversity and volume of the dataset, 2) implementation and learning of a CNN based on the prepared dataset, 3) Perspective transformation of the volleyball court, 4) Development of the online applications and integration of the trained CNN models in connection with a perspective projection to display the final results.

3 RELATED WORK

The use of modern technologies is no longer limited to pilot projects and events of lower ranks, but is gaining ground at the highest sporting events in the world. At the Olympic Games in Paris, in collaboration with Intel, technologies were introduced to improve the experience of participants and spectators. The International Olympic Committee presented a program to use artificial intelligence in sports to improve athlete performance and spectator experience [6].

At the 2022 FIFA World Cup in Qatar, semi-automatic technology was used to check prohibited positions. The technology uses twelve cameras placed under the top of the stadium to calculate the position of twenty-nine key points on each player fifty times per second. To accurately detect the impact of the ball, they use an IMU sensor, which is placed in the middle of the ball and sends data five hundred times per second [4].

In sports such as tennis and volleyball, the Hawk-eye system has been used for many years to track the path of the ball and determine its position with the help of high-speed cameras placed around the playing surface. It identifies the pixels in each frame that correspond to the ball and then compares its position using at least two image frames recorded from different camera angles to confirm or correct the position accordingly [10]. As already mentioned, even in volleyball, as in other sports, the introduction of advanced technologies is not an exception. The Balltime Platform with Artificial Intelligence Volleyball AI (VOLL-E) divides the volleyball game into segments, facilitating match analysis and player preparation. Using a CNN model that has been trained on numerous volleyball videos, it enables automatic detection of the ball and players on the court and labeling with bounding boxes. Based on the recognized positions of the ball and players, the platform recognizes game actions such as reception or defense and automatically determines the direction of the attack and displays it visually. It also calculates

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.9
ISBN 978-961-286-914-4

the ball speed and lets the users sort game elements by individual and integrate with YouTube to share clips [2].

Similar functionality to that provided by the Balltime platform described above is also provided by an Apple mobile application called Avais, which, with monitors and analyzes the volleyball game in real time. As a result, the users avoid waiting while loading a video of a volleyball match and can obtain data for analysis at the same time [1].

4 IMPLEMENTATION

The final solution was implemented in several steps using various technologies. To ensure accessibility across different devices and locations, a web application was developed, leveraging trained neural network models for detecting volleyball courts and tracking volleyball movements through image frames.

4.1 Collection, preprocessing, labeling, and augmentation of training data

We started with collecting, preprocessing, labeling, and augmenting the obtained training data. Primary data collection was performed using freely available data on the internet. Due to a lack of sufficient data, we added our own recordings from volleyball matches to the training set. We implemented a script in Python to convert video sequences into image frames and saved them in JPG (Joint Photographic Experts Group) format, one per second. Due to the different sources of training data and their dimensions, we unified their dimensions.

Once the dimensions of the image frames were standardized, we proceeded with the labeling of the training data, using two separate methods. For the first method, we manually selected six points on each image frame, and then, for each of the six selected points, recorded the x and y coordinates in JSON (JavaScript Object Notation) format for later use, as shown on Figure 1.

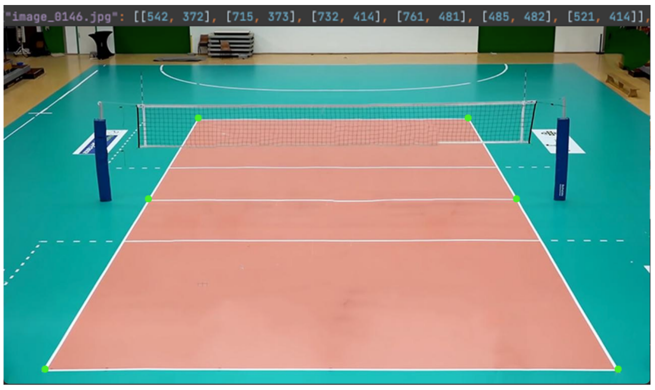

```
"image_0146.jpg": [[542, 372], [715, 373], [732, 414], [761, 481], [485, 482], [521, 414]],
```

Figure 1: Manual labeling of training data points.

As for the second technique for labeling the data used in the training set for ball detection, we utilized the functionality of the web platform Roboflow [8], which provides developers with comprehensive services for building computer vision applications, including data labeling in training sets.

Due to the smaller number of data in the training set, we decided to augment the data, as shown on Figure 2. We utilized **imgaug**, which is a dedicated library for augmenting training sets using various augmentation techniques it supports [3]. To retain the entire court on the image after augmentation, we read the coordinates of the volleyball court's corners and determined the appropriate transformations based on their distance from the image edge. Transformations were also performed in random order using the Sequential function. After the transformations, we checked whether all the marked points of the court remained within the image window; if any point was outside, we repeated the process up to five times and, in case of failure, applied horizontal flipping.

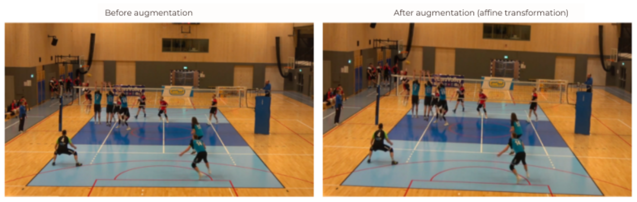


Figure 2: Original image (left) and rotated augmented image (right).

4.2 Perspective transformation

Perspective transformations have been applied in various fields, including autonomous driving, where footage from multiple cameras mounted on a vehicle was reshaped using perspective transformation to return a bird's-eye view covering the entire surroundings of the car, making it easier to assess distances between objects around the vehicle [7]. In our case, we wanted to create a bird's-eye view, that is, a top-down perspective of the volleyball court, to facilitate easier subsequent game analysis. Using a library specialized in computer vision called OpenCV, we first calculated the homography matrix, which was then used to transform the original perspective into a top-down view, as shown on Figure 3

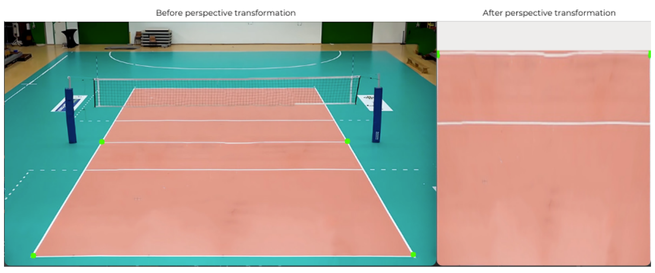

Before perspective transformation
After perspective transformation

Figure 3: Original image with court edges marked by green dots (left) and image after perspective transformation (right).

4.3 Implementation and training of CNN models

For training, we chose two separate convolutional neural network models: the segmentation neural network U-Net, which was used for detecting the volleyball court, and the YOLOv8 model, which was used for detecting the ball. In both cases, we followed an approach where separate datasets were used for training and testing, meaning none of the images used during training were later used for testing the performance of any of the neural network models. The segmentation neural network U-Net, which has a symmetrical structure provided by the encoder and decoder parts, was implemented using the open-source machine learning framework PyTorch.

For the YOLO model, we decided to use one of the newer versions, specifically version eight, developed by Ultralytics [9]. Although we could have conducted training on Ultralytic's platform, we preferred to install the Ultralytics library locally and integrate it into our framework. For training purposes, we used a pre-trained YOLOv8 model (which we further fine-tuned with our own data using previously labeled data, stored in the YAML format (a data serialization format), which was generated on the Roboflow platform during data annotation).

4.4 Web application

To ensure better accessibility, regardless of the location and device of the end user, a web application was developed. The main purpose of this application is to use trained neural network models for detecting the volleyball court and tracking the volleyball through image frames during a short segment of a volleyball match. For development we used the Flask web microframework for the backend of the application, while the interface was enhanced and improved using the open-source CSS framework Bootstrap.

5 RESULTS

The final solution is essentially divided into two main parts. In the first part, the user can choose between previously prepared clips or select their own video from the device through which they are accessing the web application. After selecting the video, the user can start the analysis by clicking a button, which is divided into four key components, as shown in Figure 4:

- Detection of the volleyball court using the trained CNN U-Net model,
- Tracking the volleyball and detecting it using the CNN model YOLOv8,
- Perspective transformation of the volleyball court,
- Attack direction, where the movement of the volleyball is shown through a sequence of image frames.

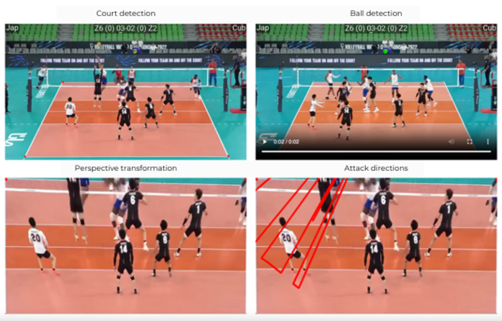


Figure 4: Web application steps: top left - Court detection with U-Net, top right - Ball tracking with YOLOv8, bottom left - Perspective transformation, bottom right - Ball trajectory across frames.

Despite the successful implementation of the desired functionalities, the application does not perform perfectly in specific situations. This becomes evident mainly in cases where the input video contains image frames different from those used to train the CNN. The most common issues we observed were:

- Different camera positions: when the volleyball match is recorded from a different camera angle than those used in the training dataset.
- Multiple lines on the court: when the video includes multiple lines that do not pertain solely to the volleyball court.
- Color scheme of the ball: in some leagues (even in top leagues, for example in Italy), they are using a ball of different color. Additionally, color schemes of the court might overlap with the ball, making it challenging to detect the ball accurately.
- Key court points or ball covered by players: when key points of the court are covered by players, as shown in the Figure 5, or by other obstacles (e.g., the net covering the line on the court farthest from the camera if the camera is positioned too low), resulting in difficulties in detecting the volleyball court or the ball.
- Real time processing limitations: based on computational resources and web application complexity, real-time streaming and analysis of video (especially with higher resolution) might introduce latency or affect performance.

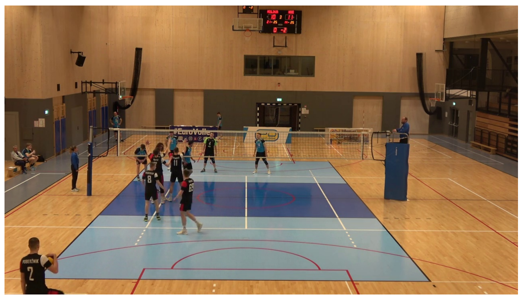


Figure 5: Key court points covered by players.

6 CONCLUSIONS

In this paper, we presented the process from the initial idea to a functional web application that provides a solution for the original concept, which was the automatic analysis and visual presentation of the results to the user. The methods used were based on collecting, preparing, labeling, and augmenting data, which were then used to train two CNN. The trained models were then used, in conjunction with the perspective transformation of the court, to analyze and visually present the results to the user. The final result was presented as a user-friendly web application, where the user can select a desired video and receive a basic analysis within seconds, including court detection and volleyball tracking.

However, despite a successful implementation, the application has some weaknesses and scenarios where the results are not as expected. Challenges include different camera positions, diverse ball color schemes or multiple balls at the same time, interference that may obscure key court points or the ball itself. Additionally, real-time processing can be affected by computational limitations and video quality, potentially impacting performance.

Nonetheless, the implemented application serves as a foundation that allows for numerous upgrades, which could be inspired by existing solutions and improve upon their shortcomings. As a final result, we could provide users with real-time statistics of a volleyball match, and the platform could be expanded to include other sports, thereby attracting a wider range of users. All these reasons encourage the realization that the integration of modern technologies into all segments of our lives, including sports, is no longer a binary question but merely a matter of time.

REFERENCES

[1] Avais. 2023. Our Features. https://www.avais.ai/features. Accessed: 2024-06-02.

[2] Balltime Academy. 2024. What is Volleyball AI. https://academy.balltime.com/getting-started/what-is-volleyball-ai. Accessed: 2024-06-02.

[3] Imgaug. 2020. Documentation. https://imgaug.readthedocs.io/en/latest/. Accessed: 2024-06-02.

[4] Inside FIFA. 2022. Semi-automated Offside Technology to be Used at FIFA World Cup 2022™. https://inside.fifa.com/technical/media-releases/semi-automated-offside-technology-to-be-used-at-fifa-world-cup-2022-tm. Accessed: 2024-06-02.

[5] B.T. Naik, M.F. Hashmi, and N.D. Bokde. 2022. A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions. *Applied Sciences* 12, 9 (2022), 4429.

[6] Olympics. 2024. IOC Takes the Lead for the Olympic Movement and Launches Olympic AI Agenda. https://olympics.com/ioc/news/ioc-takes-the-lead-for-the-olympic-movement-and-launches-olympic-ai-agenda. Accessed: 2024-06-02.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 779–788.

[8] Roboflow. 2024. Our Company. https://roboflow.com/about. Accessed: 2024-06-02.

[9] Ultralytics. 2024. YOLOv8 Models Documentation.

[10] Wikipedia. 2024. Hawk-Eye. https://en.wikipedia.org/wiki/Hawk-Eye. Accessed: 2024-06-02.

[11] World Health Organization. 2024. Sports and Health Initiative. https://www.who.int/initiatives/sports-and-health. Accessed: 2024-08-30.

A Bayesian Approach to Modeling GPS Errors for Comparing Forensic Evidence

Nika Molan
nm83087@student.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

Ema Leila Grošelj
eg61487@student.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

Klemen Vovk
kv4582@student.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

ABSTRACT

This paper introduces a Bayesian approach to modeling GPS errors for comparing forensic evidence, addressing the challenge of determining the most likely source of a single GPS localization given two proposed locations. We develop a probabilistic model that transforms GPS coordinates into polar coordinates, capturing distance and directional errors. Our method employs Markov chain Monte Carlo (MCMC) sampling to estimate the data-generating processes of GPS measurements, enabling robust comparison of potential device locations while quantifying uncertainty. We apply this approach to three datasets: one from existing literature and two newly collected datasets from Ljubljana and Novo mesto. The result is a posterior distribution of log-likelihood ratios directly comparing the two propositions, which can be transformed into likelihood ratios to comply with current standards in forensic science.

KEYWORDS

device geolocation as evidence, MCMC, digital forensics, likelihood ratio, Stan, Bayesian inference

1 INTRODUCTION

GPS as evidence has been proven problematic in court, as it has often been dismissed or not presented at all [1] due to the fear of wrong judgment or discrediting other evidence due to the uncertainty of GPS measurements. Our goal is to provide a Bayesian approach for evaluating a single GPS localization in light of two proposed locations. To give more context to why such approaches are needed, consider the following problem: a single GPS localization (evidence point *E*) was extracted from a device D found on the crime scene. Since E is a GPS measurement there is inherently some measurement error. Additionally, someone could have moved the device D during or after the crime. Investigators propose two geographical locations (P1 and P2) of where the device could have been when it measured E and we want to identify which of the two propositions is more likely. Since the conclusion is to be presented in court, we need to provide sufficiently precise verbal equivalents of the results while not misleading or misrepresenting the weight of a piece of evidence.

The contributions of this paper are summarized as follows:

- a Bayesian statistics approach utilizing Markov chain Monte Carlo sampling to estimate the data generating processes (DGPs) of GPS measurements taken from different locations to compare which DGP most likely generated a single GPS measurement obtained as forensic evidence,
- Code implementation of the proposed approach along with MCMC diagnostics, results, and visualizations made available at [2],
- two GPS measurement datasets collected in Ljubljana and Novo mesto to aid further research.

2 RELATED WORK

The increased availability of GPS logs from smartphones, activity trackers, navigation, and autonomous vehicles has increased the use of such digital evidence in court[1]. Due to the high risk of misinterpretation and wrongful judgment or discrediting other evidence, statistical methods have been proposed [3] to be able to quantify and reason under uncertainty due to GPS measurement errors.

The magnitude of GPS errors varies between devices and geographical locations. In [8] the authors report a localization error of up to 5 meters in low-cost phones, while others (see [5]) have measurement errors varying up to 100 meters.

The standard in forensic science for evidence source identification is to use likelihood ratios [7]. As different magnitudes of likelihood ratios are not easily explainable in court, forensics standards have been developed to define the verbal equivalent of likelihood ratio ranges to provide in court[4], with the current version of the standard shown in Table 1.

In [5] the authors computed a likelihood ratio to compare two proposed device locations (P1 and P2) in light of the evidence E. They also considered that errors of GPS measurements may not be equal in all directions (the horizontal error is dependent on the direction) resulting in high computational complexity due to sample dependence and brute force distribution fitting.

3 DATASETS

For all used datasets we provide data sources as well as scripts to transform data from other works to the input format for our approach. The scripts used to transform and clean the datasets that were used as inputs for modeling are also provided.

The *University of Lausanne dataset (UNIL)* was obtained from the public implementation of [5]. It consists of 699 GPS measurements that were taken from two proposal points as reference measurements on the University of Lausanne campus. The dataset is visualized in Figure 1.

Our *Ljubljana (LJ)* dataset consists of 4 predefined points (evidence *E*, and three proposal points *P*1, *P*2, and *P*3, each point is specified by latitude and longitude) and a total of 450 images captured while standing on those proposal points along with information

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.10
ISBN 978-961-286-914-4

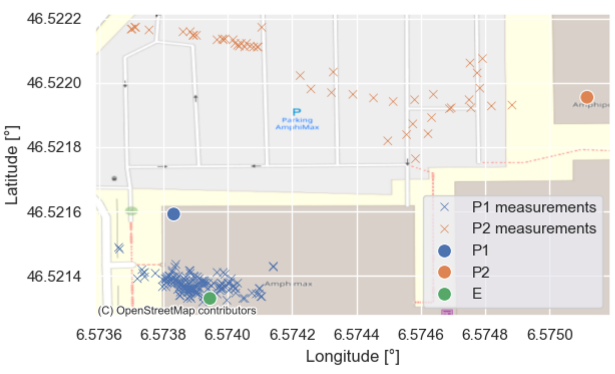


Figure 1: A geographical visualization of the UNIL dataset from [5]. E is the single localization that was recovered, while P1 and P2 are two proposed locations. Only deduplicated reference measurements for both proposals are shown. Note how reference measurements are not even on the same building as the proposal they were measured from.

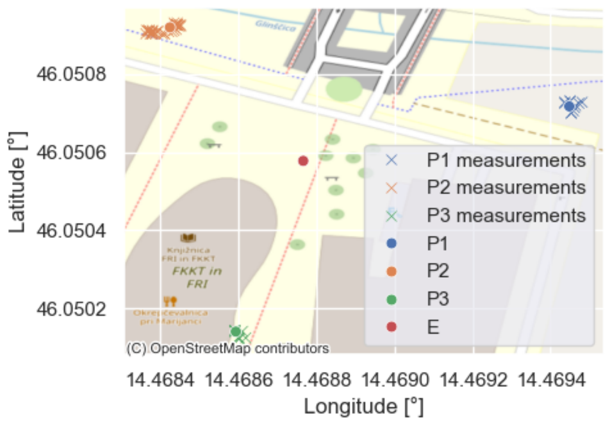


Figure 2: A geographical visualization of our LJ dataset. Only deduplicated measurements are shown.

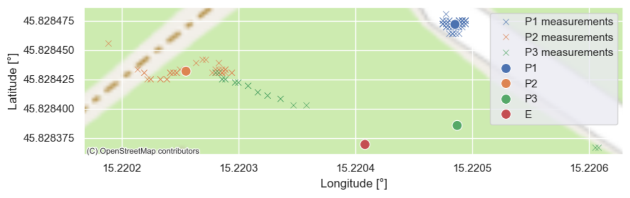


Figure 3: A geographical visualization of our NM dataset. Only deduplicated measurements are shown.

if the iPhone camera app had permission for precise location for every image. A visualization of the dataset is shown in Figure 2.

Our *Novo mesto (NM)* dataset consists of 4 predefined points (evidence E, and three proposal points $P1$, $P2$, and $P3$, each point is specified by latitude and longitude) and a total of 429 images captured using the same data collection process as the LJ dataset. A visualization of the dataset is shown in Figure 3.

Compared to the UNIL dataset, LJ and NM datasets have significantly less measurement error (on average 15 meters) and a high percentage (90%) of duplicate measurements. This is due to the measurements being done in a very short time interval (30 minutes) which resulted in a lot of cached duplicates.

4 METHODS

Unless specified, everything in this section applies to all used datasets (UNIL, LJ, and NM).

4.1 LJ and NM dataset collection

To more clearly understand what affects the accuracy of GPS evidence as well as error patterns, we collect two additional datasets with the same device. The distance between the predefined points was around 100 meters in Ljubljana and 10 meters in Novo mesto. To take images an iPhone 11 Pro was used with the default camera app. We also note granting and denying permission to precise locations to the camera app for each image. To obtain GPS measurements from images, we extract the latitude, longitude, and time of capture from the EXIF data of each image, and note the corresponding proposal point it was taken from and if precise location permission was granted.

4.2 Dataset preprocessing

Each measurement is defined by time, latitude, longitude, and the label of the proposal it was taken from. For LJ and NM data we keep only measurements that were retrieved when precise location permission was given to the iPhone camera app. We remove consecutive duplicate GPS measurements per proposal by sorting all corresponding measurements ascending by date and time, then removing all consecutive duplicates based on latitude and longitude. This is done because consecutive duplicates could be due to caching and/or rate-limiting to GPS queries. Consequently, if we try to model distance and angle errors of GPS measurements, some angles/distances will have artificially more probability mass due to duplicates, even though these duplicates are obtained from the same GPS measurement.

4.3 Transforming GPS to polar coordinates

To simplify the modeling and representation of GPS errors, each measurement was converted from (latitude, longitude) coordinates to distance (in meters) and angle (azimuth from the north, in radians) from the ground truth point (proposal) it was taken from. To illustrate the concept we show the UNIL dataset transformed to polar coordinates in Figure 4. We aim to model the distance and directionality of GPS errors taken from proposal points (and consequently their DGP) to estimate under which proposal point is the retrieved evidence point E more likely.

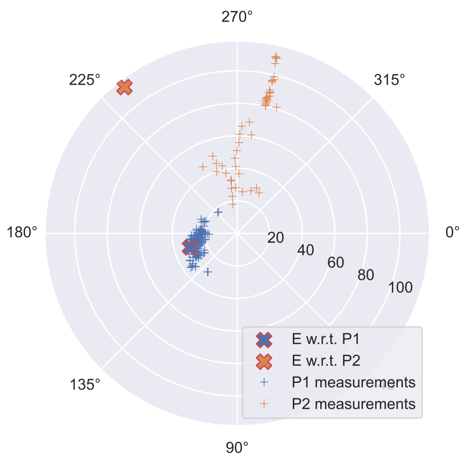


Figure 4: The UNIL dataset [5] after coordinate transformation (distance error in meters and angle - azimuth from north). Note, we only have one evidence point E, we transform it concerning (relative to, as seen from) each proposal point separately. Proposals P1 and P2 are at the center of the polar plot and measurements show the directionality (angle) and magnitude (in meters) of the GPS errors.

4.4 Probabilistic model of GPS errors

To formalize our approach to modeling GPS errors, we define a probabilistic model and estimate its parameters with MCMC sampling. The model is defined in the Stan probabilistic programming language[6] which allows users to define (log) density functions and then perform Bayesian inference with MCMC sampling.

Let $M_{ij} = (d_j, \phi_j)$ be the $i-th$ measurement measured from proposal P_j where d_j is the distance in meters from P_j to M_{ij} and ϕ_j the azimuth (in radians) of the line between P_j and M_{ij}. Analogously, let M_{ik} be the $i-th$ reference measurement measured from proposal P_k. The set of measurements for each proposal was modeled as a bivariate normal distribution:

$$
\begin{aligned}
M_{ij} &\sim \text{MultiNormal}(\mu_j, \Sigma_j)\\
M_{ik} &\sim \text{MultiNormal}(\mu_k, \Sigma_k)\\
\mu_j &= [\mu_{dist_j}, \mu_{angle_j}]\\
\mu_k &= [\mu_{dist_k}, \mu_{angle_k}]\\
\Sigma_j &\in \mathbb{R}^{2x2}\\
\Sigma_k &\in \mathbb{R}^{2x2}
\end{aligned}
$$

where μ_j is a mean vector of distance (in meters) and angle (in radians) relative to proposal P_j. Σ_j is the covariance matrix[1]. Analogously for μ_k and Σ_k relative to proposal P_k. Stan's default, non-informative priors were used for all parameters.

To compare under which proposal (P_j or P_k) is the evidence point E more likely, we compute the likelihood of E under each of the models and compute the likelihood ratio. To enhance numerical stability without loss of expressiveness the logarithms of likelihoods were used, which can later be exponentiated back. This was implemented in Stan's *generated quantities* block[2]:

$$
\begin{aligned}
\log L_j &= \log P(E_j|\mu_j, \Sigma_j)\\
\log L_k &= \log P(E_k|\mu_k, \Sigma_k)\\
\log LR &= \log L_j - \log L_k
\end{aligned}
$$

where E_j and E_k denote the evidence point E transformed relative to proposals P_j and P_k respectively.

To assess if the models capture the input data (reference measurements) well, a posterior predictive check was performed by randomly sampling points from the estimated bivariate normal models to create replicate datasets (this is also done in the *generated quantities* block in Stan). The idea is that if an estimated model fits input data well, we should be able to generate *similar*, synthetic data by randomly sampling from it. In other words, if the estimated model managed to capture the behavior of distance and angle errors in our reference measurements, it should be able to generate new, synthetic, measurements that resemble the same distance and angle errors. To visualize this, we overlay the generated synthetic data over the reference measurements (input data).

5 RESULTS

Due to the length limit of the paper we only show the full results for the UNIL dataset. However, all results, visualizations, and MCMC diagnostics are available in the provided repository.

MCMC sampling with 4 chains of 4000 samples each was performed to sample from the posterior. Standard MCMC diagnostics (trace plots, effective sample sizes, R-hat values) do not indicate any issues in convergence. Additionally, we visualize a posterior predictive check of the UNIL dataset by overlaying a random replicate dataset over the real measurements in Figure 5.

Figure 6 depicts the posterior distribution of log-likelihood ratios for the UNIL dataset along with 95% highest-density intervals. In

[1]We use the Cholesky parameterization of the Multivariate normal, which is natively implemented in Stan, so $\Sigma_j = L_j L_j'$ for efficiency and numerical stability during MCMC sampling, but omit it here for brevity

[2]Everything in the generated quantities block can be computed outside of Stan (e.g. in Python) as it is performed on the posterior draws after the MCMC sampling is done, we do it in Stan for clarity.

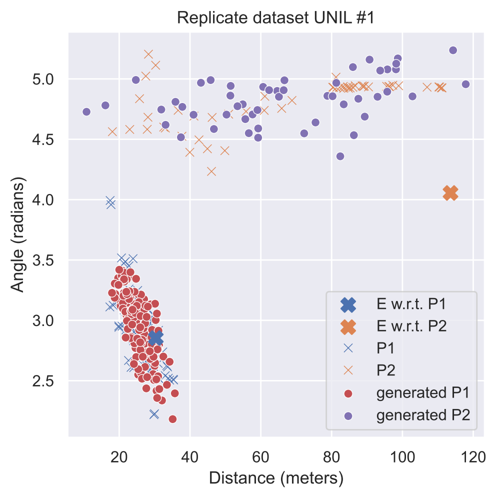


Figure 5: A posterior predictive check for the UNIL dataset in the form of a randomly selected replicate dataset for each proposal. All generated measurements for P1 and P2 are within expected regions, however, the model for P1 is better supported by the real measurements as the dataset heavily favors P1 as the source of E. This is also noted by the authors of the dataset [5].

line with the dataset, the P1 proposal is heavily favored compared to P2 to have generated the evidence point. Log-likelihood ratios can be converted back to likelihood ratios[3] which are currently the standard used in courts as per [4] and [5] to compare evidence for source-identification in forensic science.

While the model is stable and MCMC converges, even the lower-bound of the 95% highest density interval log-likelihood ratio is $\log LR = 11$, which after exponentiation is $LR = \exp 11 = 59874$, which is orders of magnitude above the highest obtainable LR range for a single piece of evidence (see Table 1 and the standard specification in [4]).

6 DISCUSSION

Our method, utilizing MCMC sampling to estimate data-generating processes of GPS measurements, offers direct uncertainty quantification, greater computational efficiency, and numerical stability due to Stan, MCMC, and working with log-likelihood ratios instead of likelihood ratios compared to the seminal method from [5]. The main limitation of our approach is the assumption that reference measurements taken (months) after the original evidence are enough to sufficiently model the DGP of GPS errors. Even if the exact device from the crime scene is used, many other variables are out of our control (GPS satellite visibility, noise and interference of GPS positioning, software updates changing the measuring process, cellular and WiFi networks that are used to improve location). Investigators should always strive to gather more actual evidence (i.e. more GPS logs from the crime scene) to directly model the errors instead of using a proxy such as reference measurements.

[3]Highest-density intervals are not equal-tailed, this means that when applying transformations, such as exponentiation to the whole distribution, the HDI will change, For such cases we recommend computing equal-tailed credible intervals that are not affected by distribution transformations.

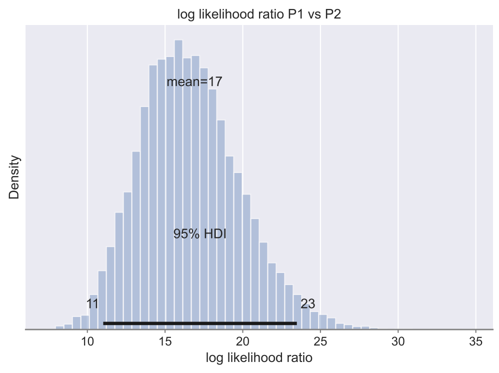


Figure 6: The posterior distribution of log-likelihood ratios for the UNIL dataset along with 95% highest-density intervals to quantify uncertainty.

Table 1: LR verbal equivalents to use in court when comparing two propositions, obtained from [4] page 39.

Range of LR	Verbal Equivalent
1-3	In my opinion the observations are no more probable if [P1] rather than[P2] were true. Therefore, the observations do not assist in addressing which of the two propositions is true.
4-10	In my opinion the observations are slightly more probable if [P1] rather than [P2] were true.
10-100	In my opinion the observations are more probable if [P1] rather than [P2] were true.
100-1000	In my opinion the observations are much more probable if [P1] rather than [P2] were true.

REFERENCES

[1] E. Casey, D.-O. Jaquet-Chiffelle, H. Spichiger, E. Ryser, and T. Souvignet. Structuring the evaluation of location-related mobile device evidence. *Forensic Science International: Digital Investigation*, 32:300928, 2020.
[2] Ema Leila Grošelj, Nika Molan and Klemen Vovk. A Bayesian Approach to Modeling GPS Errors for Comparing Forensic Evidence, 2024. https://github.com/KlemenVovk/gps_evaluation, Last accessed on 2024-08-30.
[3] C. Galbraith, P. Smyth, and H. S. Stern. Statistical methods for the forensic analysis of geolocated event data. *Forensic Science International: Digital Investigation*, 33:301009, 2020.
[4] F. S. Regulator. Development of evaluative opinions. Technical Report FSR-C-118, UK Forensic Science Regulator, Birmingham, 2021.
[5] H. Spichiger. A likelihood ratio approach for the evaluation of single point device locations. *Forensic Science International: Digital Investigation*, 2023.
[6] Stan Development Team. Stan modeling language users guide and reference manual, version 2.35, 2011–2024.
[7] M. M. Vink, M. M. Sjerps, A. A. Boztas, et al. Likelihood ratio method for the interpretation of iphone health app data in digital forensics. *Forensic Science International: Digital Investigation*, 41:301389, 2022.
[8] L. Wang, Z. Li, N. Wang, and Z. Wang. Real-time gnss precise point positioning for low-cost smart devices. *GPS Solutions*, 25:1–13, 2021.

Seven Components of Computational Thinking: Assessing the Quality of Dr. Scratch Metrics Using 230,000 Scratch Projects

Gal Bubnič
gb78843@student.uni-lj.si
University of Ljubljana,
Faculty of Natural
Sciences and Engineering,
Ljubljana, Slovenia

Tomaž Kosar
tomaz.kosar@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

Bostjan Bubnic
bostjan.bubnic@student.um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

Computational thinking has extended beyond traditional computing education recently and is becoming a broad educational movement, focused on teaching and learning critical problem-solving skills across various disciplines. Originating from computer science and programming, the most common learning method still involves educational programming languages like Scratch. Dr. Scratch is a tool designed to assess Scratch projects based on seven components of computational thinking, including abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. This study examines the quality of Dr. Scratch measurement scale. The proposed model considers computational thinking as a latent variable with seven indicators. According to the results of confirmatory factor analysis, five of the computational thinking components were measured satisfactorily, while two were below the accepted level. Based on the results, we recommend conducting an exploratory factor analysis for the potential scale refinement.

KEYWORDS

computational thinking, Dr. Scratch, block-based programming, assessment, confirmatory factor analysis

1 INTRODUCTION

Although the phrase computational thinking was introduced as a computer science concept in the early 1980s, the concept was popularized by Janette Wing in 2006 [19]. Wing described it as the ability to solve problems, design systems, and understand human behavior by leveraging fundamental computer science concepts. Recently, computational thinking has extended beyond traditional computing education into various interdisciplinary fields. It has been integrated into K-12 education, fostering problem-solving skills from an early age [10]. In addition, disciplines such as biology, physics, and social sciences are adopting computational thinking principles to tackle complex problems, analyze data, and model systems. This broadening of scope highlights the versatility and importance of computational thinking as one of the fundamental skills for the 21st century [12].

Despite its widespread adoption, there is still no consensus on the precise definition of computational thinking. Moreover, there is no consensus concerning its definitive or necessary components [5]. However, several studies have investigated the components that form its foundation. Based on the literature review: 1) Kalelioglu et al. [9] advocated that the most important components are abstraction, problem-solving, algorithmic thinking, and pattern recognition; 2) Bubnic and Kosar [5] identified abstraction and algorithms as relevant, domain independent components; 3) Lyon and J. Magana [11] argued that abstraction is the most definitional term.

Since computational thinking originates from computer science and programming, it is commonly learned and assessed today through educational programming languages like Scratch. Scratch was created by the Lifelong Kindergarten Group at the MIT Media Laboratory to provide a new environment for beginner programmers. Scratch programs are created using scripts assembled by dragging and dropping blocks, which symbolize various programming elements, such as expressions, conditions, statements, and variables. This approach helps avoiding common syntax errors, which often frustrate students. The programming environment also features interactive, 2-dimensional animations called sprites, which move on the screen according to user input or script commands. In addition, audio and video clips from webcams can be incorporated into Scratch projects.

Following the creation of Scratch, its user base grew rapidly. Due to its rapid expansion, the need for evaluation tools became more evident. In response, researchers developed several tools aimed at evaluating Scratch projects, such as Dr. Scratch [13] and Hairball [2]. Dr. Scratch is an on-line tool, which automatically assesses Scratch projects based on seven components of computational thinking, including abstraction (e.g. custom blocks), parallelism (e.g. two or more simultaneous scripts), logic (e.g. logic operations), synchronization (e.g. wait until), flow control (e.g. repeat until), user interactivity (e.g. input sound), and data representation (e.g. variables, lists).

The objective of this study was to examine Dr. Scratch's method for measuring computational thinking. The motivation for this study arises from the novelty of applying our method, particularly on such a large dataset. Utilizing a publicly available dataset containing over 230,000 Scratch projects, a latent variable model was proposed. The model considers computational thinking as a latent variable with seven indicators. Confirmatory factor analysis was used to assess the quality of the proposed model. Our results showed that five components of computational thinking were measured satisfactorily, while two were below the accepted level.

2 BACKGROUND

After the creation of Scratch, researchers and educators have started analyzing Scratch programs. However, evaluating programs in Scratch proved to be challenging due to the platform's block-based, visual format and the wide range of programming approaches used

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.11
ISBN 978-961-286-914-4

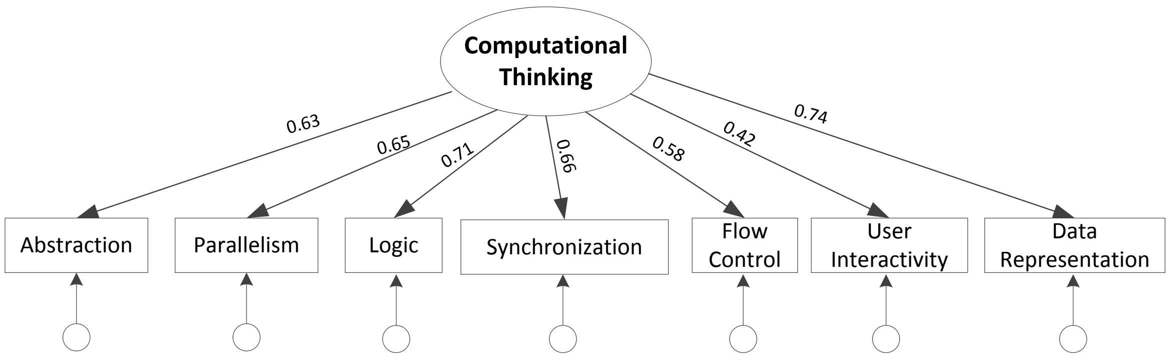


Figure 1: Latent variable model of Dr. Scratch with factor loadings derived from confirmatory factor analysis (CFA)

by beginners. These challenges led to the creation of tools for assessing Scratch programs. Hairball [2] was one of the first tools created, designed to analyze the code of Scratch projects for common programming patterns and potential coding issues. Following Hairball, Ninja Code Village [17] emerged as a platform, which offered more interactive feedback by helping users improve their coding skills through identifying areas in their projects that could be optimized or corrected. Finally, Dr. Scratch [13] was introduced to provide a more comprehensive evaluation, assessing computational thinking skills across seven indicators: abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. Each component is measured on a scale from 0 to 3, with 0 being the lowest and 3 the highest. The final score is the sum of all 7 components, thus ranging from 0 to 21 [13].

Several studies have investigated the quality of the Dr. Scratch metrics. A study by Moreno-León et al. [14] compared Dr. Scratch scores with Halstead's metrics and McCabe Cyclomatic Complexity, where vocabulary and length were the selected measures. Ninety-five Scratch projects were selected for the study, with a wide range of Dr. Scratch scores, varying from 5 to 20. According to the results, both complexity measures exhibited a strong positive correlation with the scores from Dr. Scratch. Another study by Moreno-León et al. [13] examined the ecological validity of Dr. Scratch. The sample size consisted of 109 participants, aged between 10 and 14 years, from 8 different Spanish schools. Each participant submitted a Scratch project to Dr. Scratch first. Based on the feedback, students could improve their Scratch projects using the recommendations and suggestions provided by the tool. The results showed a statistically significant score increase based on the feedback by Dr. Scratch. Convergent validity of Dr. Scratch was studied by Moreno-León et al. [16]. Fifty-three Scratch projects were evaluated by 16 specialists with a solid understanding of computer science education in the first stage of the experiment. More than 450 evaluations were conducted. The same projects were graded by Dr. Scratch in the second stage. A strong correlation was identified between scores from Dr. Scratch and evaluations by computer science education specialists. Last but not least, a discriminant validity of Dr. Scratch was demonstrated by Moreno-León et al. [15], who examined 250 Scratch projects, which were segmented into five categories, including games, art, music, stories, and animations.

3 METHOD

To assess the quality of the Dr. Scratch measurement scale, we first obtained a dataset of Scratch projects [1], which was constructed by Aivaloglou et al. [1]. The authors collected data from more than 250,000 Scratch projects, from more than 100,000 different users. After collecting data from the Scratch repository, authors also analyzed the collected projects with Dr. Scratch. As a result, the dataset comprises 231,050 Scratch projects, which were successfully evaluated using Dr. Scratch metrics [1].

After obtaining the dataset, a Grades table was extracted. A latent variable model was constructed based on the data in the Grades table. Latent variable models are statistical models that relate a set of unobservable (latent) variables and a set of observable (indicator) variables [4]. In our study, computational thinking was introduced as a latent variable with seven indicators, namely, abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. We used confirmatory factor analysis to examine the validity, reliability, and factor structure of the proposed measurement model. The model is presented in Figure 1.

4 DATA ANALYSIS AND RESULTS

Confirmatory factor analysis (CFA) was conducted using IBM AMOS 26. We used Microsoft Excel and IBM SPSS for calculating means, standard deviations, composite reliabilities (CR), and average variances extracted (AVE). The model with estimates for factor loadings is presented in Figure 1.

The χ^2 value (χ^2 (14) = 66,057) was significant, and the RMSEA was greater than the suggested threshold of 0.08 (RMSEA = 0.143). This would suggest that we had no statistical support for accepting the proposed model. However, in line with representative literature [e.g., 3], χ^2 may not be the only appropriate standard, particularly when sample sizes are large. Accordingly, we used additional fit indices to assess the goodness of fit, including GFI, RMR, NFI, IFI, and CFI. GFI was above 0.9 and RMR was lower than 0.1, which indicated a good fit of our model [8]. Furthermore, NFI, IFI, and CFI were all slightly below 0.9, but still acceptable. According to these results, we concluded that the overall model-data fit was acceptable. The measurement model fit indices are presented in Table 1.

[1]https://github.com/TUDelftScratchLab/ScratchDataset

Table 1: Model fit indices for Dr. Scratch model

χ^2	*df*	$p\,(\chi^2)$	GFI	RMR	NFI	IFI	CFI	RMSEA
66,057	14	0.001	0.919	0.058	0.870	0.870	0.870	0.143

df degrees of freedom, GFI goodness of fit index, RMR root mean square residual, NFI normed fit index, IFI incremental fit index, CFI comparative fit index, RMSEA root mean square error of approximation

Table 2: Means, Standard Deviations, Loadings, Composite Reliabilities (CR), and Average Variances Extracted (AVE) for Dr. Scratch model

Latent	Indicator	M	SD	Loadings	CR	AVE
	Abstraction	1.057	0.796	0.63		
	Parallelism	1.148	1.079	0.65		
	Logic	0.756	1.092	0.71		
CT	Synchronization	1.233	1.074	0.66	0.821	0.402
	Flow Control	1.887	0.610	0.58		
	User Interactivity	1.563	0.530	0.42		
	Data Representation	1.273	0.682	0.74		

The standardized factor loadings, composite reliability (CR), and average extracted variance (AVE) are presented in Table 2. Standardized factor loadings for abstraction, parallelism, synchronization, logic, and data representation were all higher than 0.6, varying from 0.63 to 0.74. Such results pointed toward satisfactory convergent validity of these components [7]. On the other hand, factor loadings for user interactivity and flow control were below the acceptable level of 0.6. CR was higher than the suggested threshold of 0.8 (CR = 0.82), which confirmed the reliability of the computational thinking construct [7]. On the other hand, AVE was lower than 0.5 (AVE = 0.40), which indicated that the convergent validity of the proposed measurement model might be weaker than anticipated.

5 DISCUSSION

Dr. Scratch is an assessment tool for evaluating Scratch projects based on seven components of computational thinking. This study employed confirmatory factor analysis to evaluate the quality of the Dr. Scratch measurement scale. To construct a measurement model, computational thinking was introduced as a latent variable with seven indicators, corresponding to seven components of computational thinking used by Dr. Scratch. While several studies have previously examined validity and reliability of Dr. Scratch on smaller samples, our study utilized a large sample of more than 230,000 Scratch projects.

According to the results of the confirmatory factor analysis, factor loadings of abstraction, parallelism, synchronization, logic, and data representation were above the selected threshold of 0.6. Such a threshold indicates that at least 36% of the variance in the aforementioned components is explained by computational thinking. In this context, we consider that five computational thinking components were measured satisfactorily. In addition, Hair et al. [7] suggested that, ideally, a factor loading should be at least 0.7. In this case, 49% of the variance in the observed variable is explained by the latent. According to our results, only data representation and logic surpass the 0.7 threshold. In this context, data representation tends to be the prime component on the Dr. Scratch scale.

Factor loading for flow control was slightly below the threshold (0.58), while a value for user interactivity was only 0.42. Consequently, only 18% of the variance in the user interactivity is explained by computational thinking. Accordingly, flow control and user interactivity where not measured effectively. User interactivity tends to be the weakest component on the Dr. Scratch scale.

Based on the results, CR of the proposed model (CR = 0.82) demonstrated sound reliability and high level of internal consistency. This suggests that seven components consistently measure computational thinking. However, AVE was below 0.5 (AVE = 0.4), showing that, on the average, only 40% of the variance of the computational thinking components is explained by computational thinking. Values of CR and AVE revealed a discrepancy between internal consistency and the amount of variance explained by the computational thinking. Such a discrepancy could be attributed to: low indicator quality, low factor loadings or measurement errors [6]. Regarding the low indicator quality, the high CR suggests that the indicators are reliably measuring the same construct, but the low AVE indicates that the indicators may not be capturing the construct very well. This means that while Dr. Scratch assessment items are consistent with each other, they do not explain much of the variance of the computational thinking. Concerning potential low factor loadings, loadings for user activity and flow control were lower than anticipated. Since AVE is a function of the squared factor loadings, lower loadings can result in a lower AVE even when CR is high. Regarding the potential presence of a measurement error, a lower than expected AVE could be due to high measurement error in the indicators. Namely, even if the indicators are internally consistent, significant measurement errors can reduce the proportion of variance explained by the construct.

5.1 Limitations

The results are primarily limited to the data extracted from publicly available dataset, which includes only projects submitted into the Scratch repository up until 2017. It is possible that the programming habits of Scratch users have evolved over time. Another limitation exists, because Scratch projects were not randomly selected from Scratch repository. Instead, a scraper program collected the most recent projects available at the time it was running [1]. According to Aivaloglou et al. [1], this limitation was somehow mediated by collecting a large dataset, which comprises around 1.3% of 19 million shared Scratch projects.

Another limitation of our approach stems from the fact that Dr. Scratch was designed as a formative assessment, rather than a diagnostic tool or measurement scale. To fully and comprehensively assess computational thinking, Dr. Scratch needs to be supplemented with other types of tools, such as a computational thinking test [18].

6 CONCLUSION

This study evaluated the quality of Dr. Scratch measurement scale using a publicly available dataset with more than 230,000 Scratch projects submitted to the Scratch repository up until 2017. According to the results of the confirmatory factor analysis, five computational thinking components were measured satisfactorily, whereas two were below the accepted level. In addition, lower than anticipated average variance extracted indicated potential issues with the measurement model, such as weak indicators. To address these concerns, we plan to further investigate Dr. Scratch scale in the future, using the same dataset. Exploratory factor analysis could be a valuable starting point for potential scale refinement. In addition, it would be beneficial to conduct the same analyses on Scratch projects submitted after 2017 and compare the results.

ACKNOWLEDGMENTS

The authors would like to acknowledge Marcos Román-González and Gregorio Robles, the creators of Dr. Scratch, for their valuable suggestions to improve this work.

REFERENCES

[1] Efthimia Aivaloglou, Felienne Hermans, Jesús Moreno-León, and Gregorio Robles. 2017. A dataset of scratch programs: scraped, shaped and scored. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 511–514.

[2] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. 2013. Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 215–220.

[3] Kenneth A Bollen. 1989. *Structural equations with latent variables*. John Wiley & Sons.

[4] Kenneth A Bollen. 2014. *Structural equations with latent variables*. John Wiley & Sons.

[5] Bostjan Bubnic and Tomaz Kosar. 2019. Towards a Consensus about Computational Thinking Skills: Identifying Agreed Relevant Dimensions.. In *PPIG*. 69–83.

[6] Claes Fornell and David F Larcker. 1981. Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research* 18, 1 (1981), 39–50.

[7] J Hair, B Black, B Babin, and R Anderson. 2010. *Multivariate data analysis, 7th Edition*. Pearson Prentice Hall.

[8] Litze Hu and Peter M Bentler. 1999. Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural equation modeling: a multidisciplinary journal* 6, 1 (1999), 1–55.

[9] Filiz Kalelioglu, Yasemin Gülbahar, and Volkan Kukul. 2016. A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing* 4, 3 (2016), 583.

[10] Michael Lodi and Simone Martini. 2021. Computational thinking, between Papert and Wing. *Science & education* 30, 4 (2021), 883–908.

[11] Joseph A Lyon and Alejandra J. Magana. 2020. Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education* 28, 5 (2020), 1174–1189.

[12] Ana Melro, Georgie Tarling, Taro Fujita, and Judith Kleine Staarman. 2023. What else can be learned when coding? A configurative literature review of learning opportunities through computational thinking. *Journal of Educational Computing Research* 61, 4 (2023), 901–924.

[13] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[14] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE global engineering education conference (EDUCON)*. IEEE, 1040–1045.

[15] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2017. Towards data-driven learning paths to develop computational thinking with scratch. *IEEE Transactions on Emerging Topics in Computing* 8, 1 (2017), 193–205.

[16] Jesús Moreno-León, Marcos Román-González, Casper Harteveld, and Gregorio Robles. 2017. On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2788–2795.

[17] Go Ota, Yosuke Morimoto, and Hiroshi Kato. 2016. Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational thinking concepts. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 238–239.

[18] Marcos Román-González, Jesús Moreno-León, and Gregorio Robles. 2019. Combining assessment tools for a comprehensive evaluation of computational thinking interventions. *Computational thinking education* (2019), 79–98.

[19] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

Machine Learning Approaches to Forecasting the Winner of the 2024 NBA Championship

Hana Zadravec
hana.zadravec@student.um.si
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Maribor, Slovenia

ABSTRACT

Forecasting the winner of the NBA Championship has become more important as there is a large amount of data and the league's popularity is increasing. This research investigates techniques in machine learning to predict the winner of the 2024 NBA Championship. Three methods - random forest regression, SVR, and linear regression - are used and assessed. The process includes scraping data from Basketball Reference, then analyzing and selecting features. Findings show the leading projected teams for 2024 according to each model, with random forest regression showing the best prediction. Analysis of feature importance emphasizes critical predictors like team quality rating and player performance metrics. The research highlights the capabilities of machine learning in predicting sports outcomes and indicates areas for additional research to improve predictions.

KEYWORDS

forecasting, basketball prediction, statistical analysis, NBA Championship, machine learning

1 INTRODUCTION

Forecasting the winner of the NBA championship has become increasingly accessible for sports analysts, bettors, and enthusiasts alike. This endeavor prompts the exploration and application of sophisticated analytical methodologies to enhance predictive precision. The necessity for more precise prognostications is underscored by recognizing the NBA's status as the most extensively followed professional sports league in 2022, engaging 2.49 billion individuals [4]. Comprising 30 teams in North America, the NBA stands as a premier basketball league showcasing elite players globally [5]. With an annual revenue surpassing $10 billion, the league continuously accumulates a wealth of data crucial for analysts and strategic planning within sports organizations seeking competitive advantages through data analysis. This data often informs pivotal on-field decisions regarding team formations and gameplay strategies, such as offensive or defensive approaches. Such insights can significantly impact match outcomes. Moreover, this wealth of data facilitates individual game outcome predictions in the realm of NBA contests. Ahead of each match, numerous analysts proffer their forecasts for the victor. These predictions are scrutinized by commentators on NBA platforms who provide pre-game analysis. Furthermore, a growing betting industry has arisen around prognosticating NBA matchups. This sector expands annually with a key emphasis on developing precise models adept at handling pertinent metrics in NBA games effectively. Hence, the increasing integration of machine learning models in sports represents a pivotal and adaptable strategy moving forward.

The research motivation comes from the necessity to improve sports analytics in the NBA, aiming for more precise predictions to benefit strategic decisions and operations in the betting sector. Due to the constraints of current models that often overlook important metrics, this research aims to enhance prediction accuracy by utilizing different machine-learning techniques. This study also seeks to address a deficiency in the literature, as it seldom focuses on predicting the champion of the entire championship.

This article examines forecasting NBA championship winners by utilizing three machine learning techniques: random forest, support vector regression (SVR), and linear regression. Section 2 will examine pertinent studies in the field of predicting sports performance, specifically honing in on NBA results. In Section 3, we provide a comprehensive explanation of the techniques utilized for gathering and examining data, as well as the implementation of the specified models. Next, we will discuss the results and evaluate how well each technique worked in Section 4. Section 5 explores the importance of our discoveries for future research in this area. Finally, our analysis leads to conclusions in Section 6.

2 LITERATURE REVIEW

Advancements in predictive modeling for basketball are increasingly important within sports analytics. With the growing integration of machine learning in this field, researchers continue to explore strategies for improving predictions. This section reviews studies focused on forecasting basketball outcomes using various machine-learning techniques.

Yongjun et al. [3] propose using data analysis to forecast NBA team performance by combining statistical regression methods to predict the relationship between game results and winning chances. They apply Data Envelopment Analysis (DEA) to identify optimal performance standards, evaluating their process with the Golden State Warriors, which demonstrated high predictive accuracy. The study suggests enhancing predictions by incorporating rival tactics and expanding the model for game-level forecasts for each player.

Bunker and Susnjak [1] analyze the use of machine learning methods for forecasting match outcomes in team sports. They evaluate various algorithms, including regression models, decision trees, and neural networks, assessing their predictive success with past data and player statistics. The study emphasizes advancements in machine learning that enhance accuracy and the challenges faced in

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.12
ISBN 978-961-286-914-4

practical applications. It also highlights the importance of data quality and feature selection in improving sports analytics, providing valuable insights for future research.

Yao [8] assesses how well neural networks predict outcomes compared to traditional regression models using past NBA data. The results indicate that regression models provide straightforward explanations, while neural networks are better at capturing intricate patterns, leading to increased precision. This research highlights how advanced machine learning methods can be utilized in sports analysis to improve performance prediction strategies.

The study by Huang and Lin [2] introduces an innovative method for predicting game results using regression tree models. The writers examine different elements impacting game results, like player data and team interactions, to create a targeted predictive system for the Golden State Warriors. Their research shows that regression trees can capture intricate connections in the data, leading to precise predictions of scores.

Thabtah et al. [7] explore the use of machine learning to forecast NBA game outcomes in their research. Numerous learning models, such as decision trees, artificial neural networks, and Naive Bayes, have been applied. After analyzing the data, it was discovered that important characteristics including total rebounds, defensive rebounds, three-point percentage, and the quantity of made free throws are essential for accurately predicting the outcome of games.

3 METHODOLOGY

In this section, we provide a comprehensive explanation of the approach utilized for examining NBA data in our research. All analyses were performed using a Jupyter notebook. The primary objective was to collect, process, and analyze NBA data to develop models for forecasting outcomes for the 2024 NBA season.

3.1 Data Collection

The initial step involved gathering data through web scraping methods. We sourced data from the Basketball Reference website [6], which offers extensive statistics for every NBA season up to the present day.

Web scraping was chosen for its efficiency in collecting large volumes of data without manual input. We used Python libraries such as BeautifulSoup and requests to retrieve HTML content from the web pages. The pertinent data, including player stats, team stats, and game outcomes, were extracted and organized into CSV files for ease of manipulation in subsequent analyses.

3.2 Data Preprocessing

Following data collection, we performed meticulous processing to ensure data quality and consistency. This included:

- We filled missing NaN values with the median to preserve the dataset and minimize their impact on the analysis.
- Standardizing the data as needed to maintain uniformity across the dataset.
- Normalizing features to bring them into a common scale, which is especially important for algorithms sensitive to the magnitude of input features.
- Encoding categorical variables into numerical format using label encoding, allowing for their inclusion in machine learning models.
- Removing duplicates to eliminate redundancy in the dataset and improve model performance.

3.3 Feature Selection

To enhance model performance, we addressed multicollinearity by filtering features based on their correlation. Pearson's correlation coefficient was used to assess the linear relationship between features. The coefficient r is calculated as:

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{1}$$

where $\text{cov}(X, Y)$ is the covariance between variables X and Y, and σ_X and σ_Y are their standard deviations. Pearson's r ranges from -1 to 1:

- $r = 1$ indicates a perfect positive linear relationship,
- $r = -1$ indicates a perfect negative linear relationship,
- $r = 0$ indicates no linear relationship.

We set a correlation threshold of 0.9, identifying features with an absolute value of Pearson's r above this threshold as highly correlated. Features exhibiting high correlation were removed to reduce redundancy and mitigate multicollinearity, thus enhancing model interpretability and reliability.

Additionally, we utilized Principal Component Analysis (PCA) as a dimensionality reduction technique to transform the feature space, allowing us to reduce the number of features while retaining essential information, further enhancing model performance.

3.4 Model Selection and Data Splitting

Data was divided into training and testing sets as follows:

- **Training Data:** Data from the 2005 to 2023 NBA seasons.
- **Testing Data:** Data from the 2024 NBA season.

We employed three machine learning models to forecast NBA game outcomes:

3.4.1 Support Vector Regression (SVR). **Reason for Selection:** SVR is selected for its capability to handle complex, non-linear relationships between features and the target variable (game outcome). SVR is effective in scenarios where interactions between variables are intricate.

Advantages: SVR finds optimal hyperplanes to minimize prediction errors within a specified margin, capturing subtle patterns in the data.

3.4.2 Random Forest Regression. **Reason for Selection:** Random Forest regression was selected for its ensemble method, combining decision trees to improve predictions and manage overfitting. It effectively captures complex interactions in NBA data.

Advantages: This model handles high-dimensional data, identifies key features, and is robust against outliers. It also models non-linear relationships with minimal tuning, making it versatile for both categorical and continuous variables in sports analytics.

3.4.3 Linear Regression. **Reason for Selection:** Linear Regression is chosen as a baseline model for predicting NBA champions due to its simplicity and interpretability, clarifying how features affect winning likelihood.

Advantages: It offers easy interpretation of feature impacts, with coefficients showing expected outcome changes for unit shifts in predictors. Minimal computational resources are needed for quick training and evaluation, and it serves as a reference for comparing more complex models.

3.5 Experiment

Our experiment focuses on NBA data from the 2005 season onward, due to significant changes in gameplay and statistical tracking. Prior to 2005, the game was more physical and lacked modern statistics like three-point shooting (3P%), which were introduced post-1990.

We used data from the 2005 to 2023 NBA seasons to train our models, which included 49 features related to team and player performance. Some of the features are:

- `pre_season_odds`: The odds assigned to each team before the season starts, indicating their chances of winning the championship.
- `team_rating_custom`: A custom rating for each team based on various performance metrics, reflecting their overall strength.
- `FG%`: Field Goal Percentage, representing the ratio of field goals made to field goals attempted, a key indicator of shooting efficiency.
- `3P%`: Three-Point Percentage, indicating the ratio of three-point field goals made to three-point attempts, measuring a team's effectiveness from beyond the arc.
- `max_player_rating_custom`: A custom rating for the highest-rated player on each team, capturing the impact of star players.

The target variable for prediction is `champion_share`, which represents the chances of a team winning the NBA Championship in the 2024 season. This continuous variable can take values between 0 and 1, where a higher value indicates a greater probability of a team being crowned champion. In the dataset, known winners from previous seasons are marked with a value of 1.0, signifying their championship status, while teams that did not win are represented by lower values closer to 0.

3.5.1 *Model Parameters.* Default parameters were used for all models:

- **SVR:** The Support Vector Regression (SVR) employs a Radial Basis Function (RBF) kernel, which is effective for capturing non-linear relationships. The regularization parameter $C = 1$ controls the trade-off between achieving a low training error and a low testing error, while gamma $\gamma = 0.1$ determines the influence of individual training examples on the decision boundary.
- **Random Forest:** This model consists of 100 trees with no maximum depth specified, allowing each tree to grow fully. This approach enhances model accuracy by averaging predictions from multiple trees, reducing the risk of overfitting.
- **Linear Regression:** The linear regression model uses default parameters, applying ordinary least squares to estimate coefficients without regularization. This simplicity allows it to fit the data by minimizing the residual sum of squares, serving as a benchmark for more complex models.

3.5.2 *Evaluation Metrics.* Model performance was evaluated using the following metrics:

- **Mean Absolute Error (MAE):** Measures the average magnitude of prediction errors. It is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

  where y_i represents the actual value, $\hat{y}_i$ represents the predicted value, and n is the number of observations.
- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

  where y_i represents the actual value, $\hat{y}_i$ represents the predicted value, and n is the number of observations.

4 RESULTS

The results section provides a comparison of the models based on MAE and MSE metrics. Table and figure illustrate the performance of each model and highlight predictions for the top teams in the 2024 NBA season.

4.1 Model Comparison

Figure 1 presents the comparison of MSE and MAE across the three models used in our study. The Random Forest model achieved the lowest MSE and MAE, indicating superior performance in predicting NBA outcomes compared to SVR and Logistic Regression.

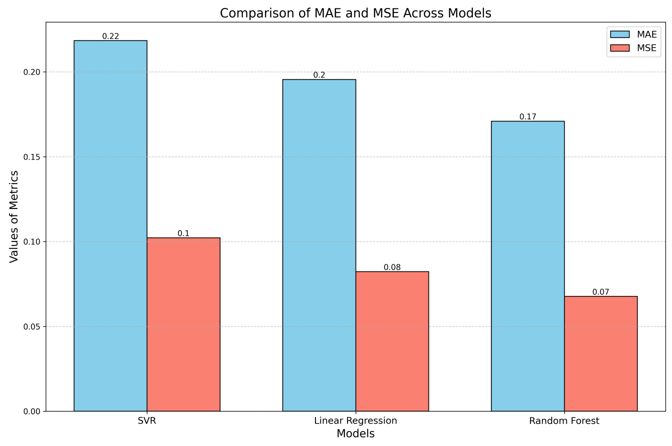


Figure 1: Comparison of MSE and MAE for each model.

4.2 Model Predictions

Table 1 presents a summary of the predicted top teams for the 2024 NBA Championship as determined by each model, along with their corresponding `champion_share` values. These values indicate the estimated probability, expressed as a decimal, of each team winning the NBA Championship in the 2024 season.

Table 1: Top 3 Predicted Teams for 2024 NBA Championship

Model	Top Predicted Teams	Predicted `champion_share`
SVR	Milwaukee Bucks	0.8691
	Boston Celtics	0.7510
	Denver Nuggets	0.6726
Random Forest	Boston Celtics	0.6485
	Milwaukee Bucks	0.5643
	Minnesota Timberwolves	0.5527
Linear Regression	Denver Nuggets	0.6706
	Milwaukee Bucks	0.6448
	Boston Celtics	0.6227

5 DISCUSSION

In this study, we assessed the performance of three predictive models—Random Forest, SVR, and Linear Regression—regarding the NBA Championship outcome for the 2024 season. The results reveal several insights and distinctions between these models.

5.1 Random Forest Model

The Random Forest model achieved the lowest MSE and MAE in predicting the NBA Championship outcome. Its ability to capture complex feature interactions enabled it to identify key determinants of the championship. Notably, it predicted the Boston Celtics as the 2024 season winners, aligning with the actual outcome and validating its predictive performance.

5.2 Support Vector Regression

Although the SVR model did not perform as well as the Random Forest and Logistic Regression models in terms of predictive metrics, it was effective in revealing intricate relationships between features. The SVR model assigned high predicted probabilities to both the Milwaukee Bucks and Boston Celtics, reflecting their strong performances throughout the season. However, the actual season outcome highlighted significant challenges for the Milwaukee Bucks, such as injuries to key players and a mid-season coaching change. These factors likely impacted their final standing, demonstrating that while SVR provided valuable insights, it may not fully account for unforeseen disruptions and their effects.

5.3 Linear Regression

Linear Regression, while not as effective as the Random Forest model in predictive metrics, still provided valuable insights. The model's predictions for the Boston Celtics and Denver Nuggets as strong contenders aligned with the final outcome of the championship. This highlights the model's utility in scenarios where more complex methods might be less interpretable. Despite its limitations, Linear Regression contributed to a broader understanding of potential championship winners.

5.4 Real-World Outcome

At the end of the 2024 season, the Boston Celtics were confirmed as the champions, validating the Random Forest model's prediction and partially supporting the SVR model's forecasts. Despite high values assigned to the Milwaukee Bucks by the SVR model, their performance was hindered by significant issues such as player injuries and a coaching change, which affected their final standing. The Minnesota Timberwolves, who were also predicted to be in contention, remained competitive until the end of the season, demonstrating that our models were accurate in predicting some outcomes.

6 CONCLUSION

This research assessed various machine learning techniques in forecasting the 2024 NBA season such as SVR, Linear Regression, and Random Forest models. The Random Forest model outperformed others, showing its capability to deal with intricate feature relationships by achieving the lowest MSE and MAE. Even though SVR and Logistic Regression were not as accurate, they still offered important information on team performance, highlighting the difficulties encountered by the Milwaukee Bucks because of injuries and coaching adjustments. This study highlights the significance of forecasting the whole season instead of single games.

One noteworthy aspect of this study is the emphasis on making predictions for the entire season, as opposed to just individual games. Our results indicate the importance of integrating current data and regularly updating it to enhance the accuracy of predictions. Future research should aim to integrate real-time data with advanced modeling techniques to more effectively adapt to the dynamic conditions and changes that occur throughout the NBA season.

In summary, incorporating various machine learning models and adjusting predictions with real-time data can improve the precision of sports predictions.

REFERENCES

[1] Rory Bunker and Teo Sušnjak. The application of machine learning techniques for predicting match results in team sport: A review. *Journal of Artificial Intelligence Research*, 75:1–22, 2022.

[2] Mei-Ling Huang and Yi-Jung Lin. Regression tree model for predicting game scores for the golden state warriors in the national basketball association. *Symmetry*, 12(5):835, 2020.

[3] Y. Li, L. Wang, and F. Li. A data-driven prediction approach for sports team performance and its application to national basketball association. *Omega*, page 102123, 2019.

[4] National Basketball Association. About the nba. https://www.nba.com/news/about, 2024. Accessed: 2024-04-30.

[5] PlayToday. Nba viewership statistics. https://playtoday.co/blog/stats/nba-viewership-statistics/, 2024. Accessed: 2024-04-30.

[6] Basketball Reference. Basketball reference. https://www.basketball-reference.com/, 2024. Accessed: 2024-04-30.

[7] Fadi Thabtah, Ling Zhang, and Nadia Abdelhamid. Nba game result prediction using feature analysis and machine learning. *Annals of Data Science*, 6(1):103–116, 2019.

[8] Alan Yao. Comparing neural and regression models to predict nba team records. *Frontiers in Artificial Intelligence and Applications*, 320:421–428, 2019.

Hit Song Prediction Through Machine Learning and Spotify Data

Andrej Natev
89221050@student.upr.si
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Koper, Slovenia

ABSTRACT

This study predicts hit songs using metadata from the Spotify API[8]. The dataset includes over 20 genres, each with 40 songs, equally divided between hits and flops, gathered using spotipy[7]. Prediction is based on the popularity feature, rated from 0-100. Models were trained on features like danceability, energy, loudness, speechiness, valence, and tempo. The dataset was split using train_test_split (10%, 20%, 33%) and kfold cross-validation with k values of 2, 5, and 10. Models were trained, evaluated, and tested, with kfold cross-validation showing the best accuracy and the least overfitting. Scikit-learn's classifiers, ensemble models, and MLPClassifier were used, with PassiveAggressiveClassifier and AdaBoost showing 60% accuracy. Ensemble methods like extra trees and random forest, along with neural networks, performed well. Gaussian Process, Naive Bayes, and ridge classifiers stood out among standard models. These results suggest that enhanced models, especially neural networks and decision tree ensembles, could improve hit prediction. Future work may explore frequency and lyric analysis.

KEYWORDS

music, genre, song, Spotify, machine learning, classification, ensemble model, support vector, neural networ, artificial intelligence

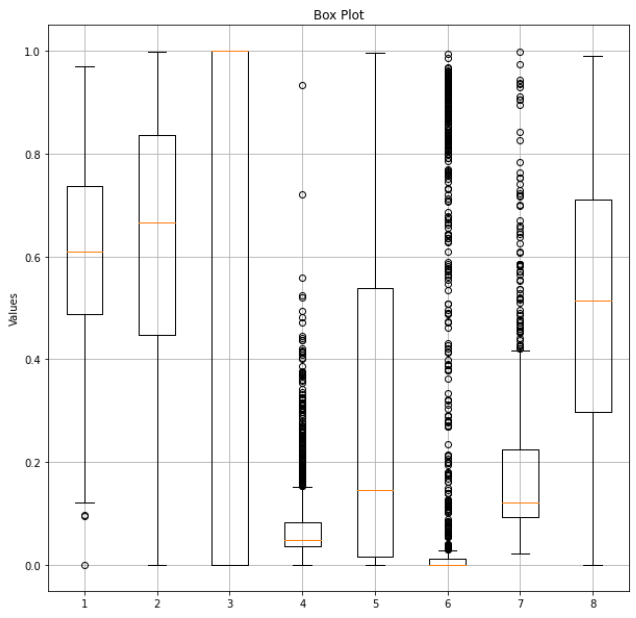


Figure 1: Box Plot for Feature Outliers

1 INTRODUCTION

This research delves into the intersection of music and data science, leveraging the Spotify Web API[8] in conjunction with the Spotipy library[7], and machine learning models. By harnessing these tools, the study[2] aims to extract and analyze track data across various genres. The primary objective is to find machine learning models capable of categorizing songs into two distinct groups: "hit" and "flop", based on a range of audio features. Popularity is a feature in Spotify's Web API[8], that represents a song's popularity the past three days from the day of extraction. It is an integer that ranges from 0-100, such that a flop is any song below 60 populartу and everything above is a hit song.

2 MATERIALS, MODELS, METHODS

2.1 Materials

In this study[2], firslt two datasets from Kaggle were utilized: "Most Streamed Songs 2023"[6] and "30000 Spotify Songs"[5]. These datasets provided a rich source of music metadata for analysis, with one of them being a training dataset and the other a evaluating dataset. Later, both of them were discarded because of the chance of overfitting. So then Spotipy[7], a Python library, was used for data extraction from the Spotify Web API[8]. Pandas and NumPy were employed for data manipulation, while the Scikit-learn[3] library facilitated feature engineering, preprocessing, data splitting, model implementation, and evaluation. And also MatPlotLib was used to be able to visually analyze the features and to present the results.

2.2 Most Accurate Models

2.2.1 MLPClassifier with ReLU and Logistic Activation Functions. The MLPClassifier is a neural network model with multiple layers of interconnected nodes. It uses the ReLU (Rectified Linear Unit) activation function, which outputs the input directly if positive or zero otherwise, helping to avoid the vanishing gradient problem. The logistic (sigmoid) activation function, which maps the input into a range between 0 and 1, is particularly useful for binary classification tasks. These activation functions enable the MLPClassifier to capture complex data patterns, improving prediction accuracy.[3]

2.2.2 ExtraTreesClassifier with Gini and Entropy Criteria. The ExtraTreesClassifier is an ensemble method that builds many decision trees using randomized splits of the training data. It uses Gini impurity or entropy as criteria to evaluate the quality of splits within the trees. Gini measures the likelihood of misclassification, while entropy measures uncertainty. By averaging predictions from multiple trees, ExtraTreesClassifier reduces overfitting and improves generalization, making it a robust choice for classification tasks.[3]

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.13
ISBN 978-961-286-914-4

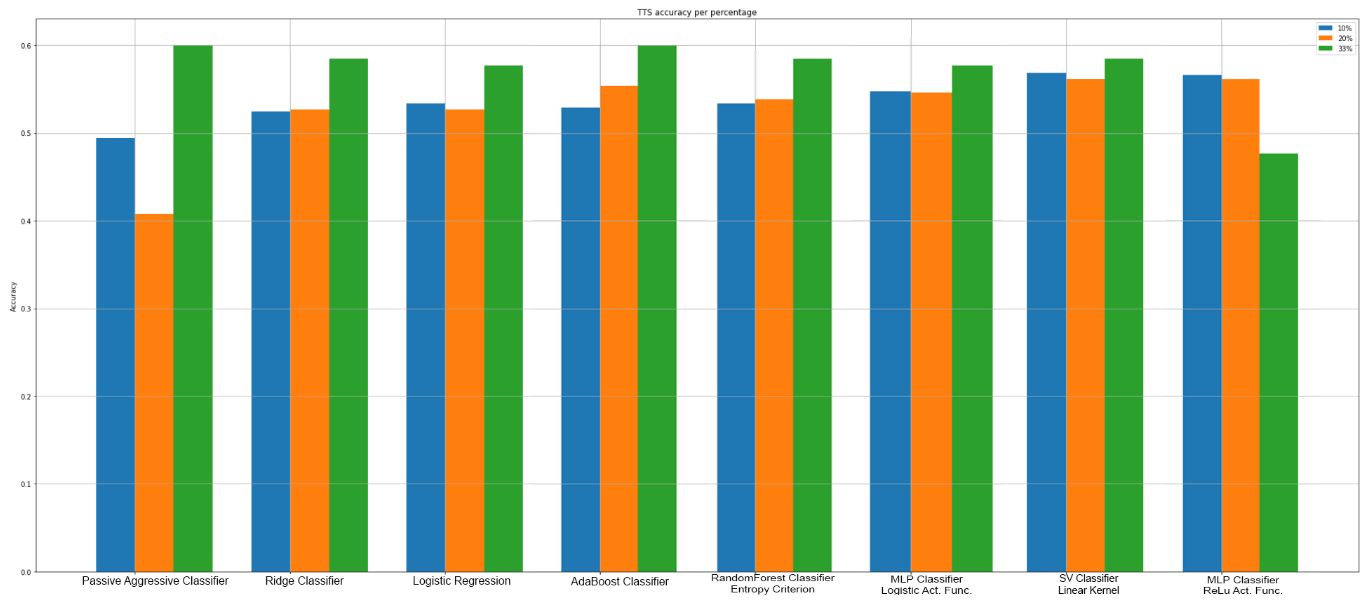


Figure 2: Bar Graph for TTS Accuracy Percentages

2.2.3 GradientBoostingClassifier. GradientBoostingClassifier incrementally builds a strong classifier by sequentially adding weak learners, typically decision trees. Each new model is trained on the residual errors of the previous models, allowing the ensemble to focus on earlier mistakes. This process iteratively reduces error, enhancing accuracy and robustness. GradientBoostingClassifier is particularly effective in complex prediction tasks requiring high precision.[3]

2.2.4 PassiveAggressiveClassifier. PassiveAggressiveClassifier is a linear model suited for large-scale learning tasks, especially in online learning. It updates parameters only when misclassification occurs (passive) and aggressively corrects errors when they do. This combination allows the model to adapt quickly, making it efficient for real-time classification tasks where speed is crucial.[3]

2.2.5 AdaBoostClassifier. AdaBoostClassifier is an ensemble method that builds a strong classifier by combining multiple weak classifiers. It adjusts the weights of misclassified instances in each iteration, focusing subsequent classifiers on difficult cases. By concentrating on previous errors, AdaBoostClassifier progressively improves accuracy, making it a powerful tool for various classification tasks.[3]

2.2.6 RidgeClassifier. RidgeClassifier is a linear model that uses L2 regularization to prevent overfitting by penalizing the magnitude of coefficients. This regularization is particularly useful in high-dimensional spaces where features outnumber observations. RidgeClassifier balances bias and variance, making it effective for classification tasks requiring generalization to unseen data.[3]

2.2.7 LogisticRegression. LogisticRegression is a linear model for binary classification tasks. It models the probability of class membership by applying the logistic function to a linear combination of input features. Trained by maximizing the likelihood of observed data, LogisticRegression effectively handles cases where the feature-target relationship is approximately linear, making it widely used for various classification scenarios.[3]

2.2.8 SVC (Linear Kernel). SVC with a linear kernel is a supervised learning model that constructs a hyperplane in a high-dimensional space to separate classes. The linear kernel computes the dot product of feature vectors, making it effective for linearly separable data. By maximizing the margin between classes, SVC with a linear kernel provides reliable and interpretable classification results.[3]

2.3 Methods

2.3.1 Data Understanding. The first step in our methodology involved understanding the datasets and the task at hand. Initially, songs were obtained from the Spotify Web API[8] based on their popularity ratings, ensuring a balanced representation of hits and flops. The popularity feature served as a crucial criterion for categorizing songs as hits or flops, with hits defined as songs with a popularity score of 60 or above, and non-hits as songs with a popularity score below 60.

2.3.2 Data Extraction. Data extraction was conducted using the Spotipy library along with the Spotipy-random add-on. The extraction process involved sourcing track data directly from the Spotify Web API[8]. In addition to leveraging Spotipy-random, genres were carefully selected to ensure diversity and fairness in the broad range of the features' values. These genres ranged from unpopular to popular and encompassed different audio features to ensure distributivity across the dataset.
During the extraction process, it was observed that songs with popularity ratings above 85 and below 60 were particularly challenging to find, even when considering genres that were both unpopular and popular. This highlights the inherent difficulty in obtaining

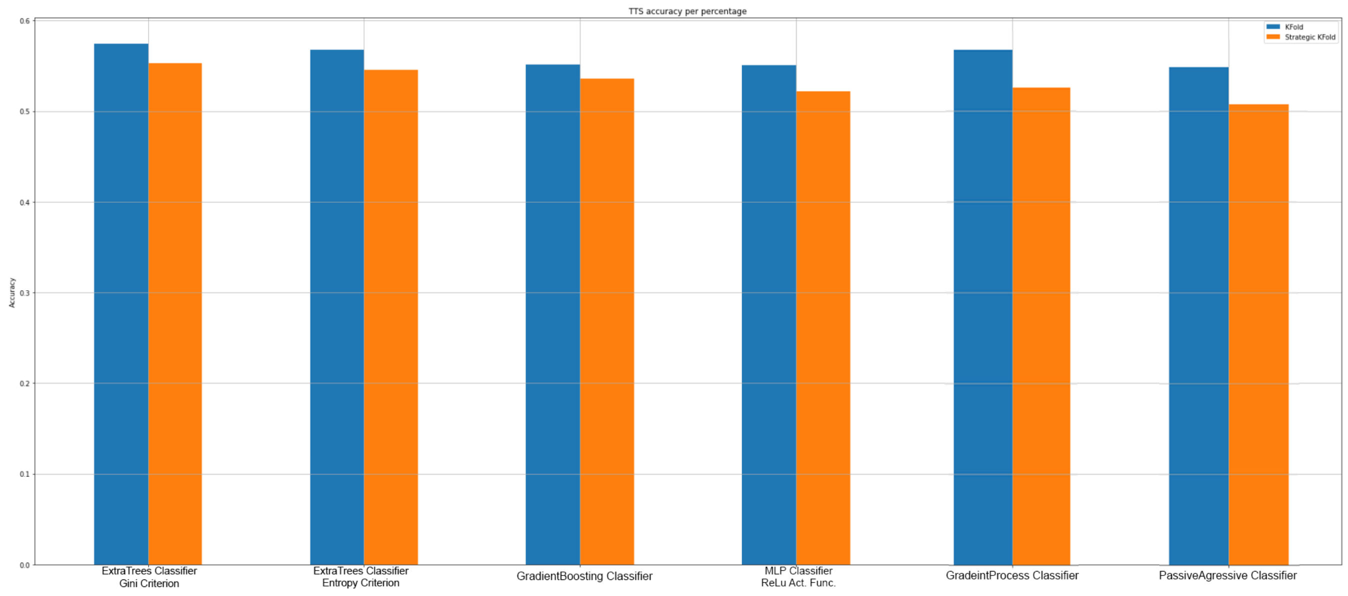


Figure 3: Bar Graph for CV Accuracy Percentages

a balanced dataset, especially when targeting specific popularity ranges.

2.3.3 Data Preparation. Following data acquisition, the dataset was prepared for analysis by incorporating audio features obtained from the Spotify API[8]. These features included danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. Additionally, data types were adjusted to ensure categorical representation for key, mode, time signature, and hit categories. After which another dataset was made that contained the same features but just scaled using the default StandardScaler from scikit-learn[3].

2.3.4 Model Training and Evaluation. During the model training and evaluation phase, it was observed that KFold and StratifiedKFold cross-validation techniques[3] encountered difficulties when handling larger splits due to the relatively small size of the dataset. With only 1299 songs available and an almost 50/50 balance between hits and non-hits, the dataset size posed challenges for these cross-validation methods to effectively cover all instances Despite these challenges, various machine learning models, including ensemble models and standard classifiers, were trained on the dataset. The performance of each model was evaluated using Train-Test-Split[3] (TTS) with varying test sizes (10%, 20%, and 33%), as well as KFold and StratifiedKFold cross-validation techniques with different fold splits (2, 3, 5, and 10).

3 RESULTS

The study[2] aimed to predict hit songs using machine learning algorithms trained on Spotify API metadata. Results revealed varying accuracies across different models and evaluation techniques. Notably, the AdaBoost Classifier and Passive Agressive Classifier achieved the highest accuracy of 60% on a test size of 33%, followed by the RandomForest (entropy criterion), and Ridge Classifiers, and Logistic Regression that demonstrated stable performance on the same test size. The MLP (logistic activation function) and the SupportVector Classifier demonstrated the highest nad the most constant through out all test sizes, 10%, 20% and 30%.
Regarding the cross-validation techniques, the stratified kfold had a constant lower accuracy throughout all models and throughout all kfolds, 2, 3, 5, and 10. ExtraTrees Classifier had the highest accuracy with both techniques and using both the gini criterion and the entropy criterion, with an an almost 60%. It was followed by the GradientProcess Classifier that had a percent higher accuracy than similarly named the GradientBoosting Classifier. Other notable mentions using the cross-validation techniques are the Passive Agressive Classifier, and the MLP Classifier with the ReLu activation function with a very similar accuracy.

4 DISCUSSION

The findings of this study[2] shed light on the possibility of machine learning algorithms to be used to predict hit songs based on Spotify metadata[8]. While some models exhibited promising accuracy rates for a really general approach to the problem, deviations from expected outcomes were observed, prompting deeper analysis. The AdaBoost Classifier achieved the highest accuracy, but only in the train-test-split. Additionally, the MLPClassifier with identity and logistic activation functions showed accuracy, suggesting the potential of neural network architectures in capturing nonlinear relationships within the data. Theoretical implications suggest the need for further investigation into ensemble models and neural networks, and hyperparameter tuning to optimize model performance.

Table 1: Model Accuracy Comparison

Model	**Accuracy (%)**	
	Train-Test-Split	**k-fold CV**
RandomForestClassifier (Entropy)	57.69 / 54.62 / 52.68	56.12
AdaBoostClassifier	60.00 / 55.38 / 52.91	53.58
ExtraTreesClassifier	48.46 / 51.54 / 55.24	56.81
ExtraTreesClassifier (Entropy)	54.62 / 55.38 / 54.31	57.81
MLPClassifier (Identity)	50.00 / 58.46 / 53.85	55.66
MLPClassifier (Logistic)	58.46 / 58.46 / 54.78	55.50

4.1 Previous Research

Compared to prior research endeavors, which often grappled with issues of data imbalance and feature scaling, this study's results represent a significant improvement. The utilization of a more balanced dataset, coupled with standardized feature scales, has led to more reliable and interpretable models. The transition from overfitted models, which yielded inflated accuracy rates, to robust and generalizable models underscores the importance of methodological rigor in data science research.[1]

5 CONCLUSION

In summary, this study[2] investigated the application of machine learning algorithms for hit song prediction using Spotify metadata[8]. Through rigorous experimentation and evaluation, we have demonstrated the potential of various classifiers and ensemble methods in categorizing songs into hits and non-hits with reasonable accuracy for a general approach. The findings contribute to the existing body of research by providing insights into the performance characteristics of different models and the impact of algorithmic parameters on predictive outcomes.

Despite achieving competitive accuracy rates, the study[2] also revealed nuances and deviations from expected results, making an even bigger need for further investigation.

Moving forward, it is imperative to address open questions surrounding the generalizability of models across diverse music genres, the robustness of predictions over time, and the incorporation of additional features such as lyrics and user-specific preferences. Moreover, future research should focus on refining model architectures, exploring ensemble models and neural networks, and optimizing hyperparameters to enhance predictive efficacy.

ACKNOWLEDGMENTS

Special recognition is also extended to the Google Developer Student Club of the University of Primorska for organizing multiple events focused on ML&AI. It was during these events that the seeds of the initial research, which had overfitting issues, were sown.[4]

REFERENCES

[1] Andrej Natev. 2023. Initial Notebook. https://www.kaggle.com/code/andrejnatev/hit-song-prediction

[2] Andrej Natev. 2024. Main Notebook. https://www.kaggle.com/code/andrejnatev/spotify-api-spotipy-hit-song-prediction.

[3] David Cournapeau. 2007. Scikit-learn Documentation. https://scikit-learn.org/stable/index.html

[4] GoogleDSC University of Primorska. 2023. ML&AI Summit. https://gdsc.community.dev/university-of-primorska-koper-slovenia/

[5] Joakim Arvidsson. 2023. 30000 Spotify Songs dataset. https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs

[6] Nidula Elgiriyrwithana. 2023. Most Streamed Songs 2023 dataset. https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023

[7] Paul Lamere. 2014. Spotipy Documentation. https://spotipy.readthedocs.io/en/2.24.0/

[8] Spotify. 2013. Spotify Web API Documentation. https://developer.spotify.com/documentation/web-api

A Data-Driven Approach for the Analysis of Ridership Fluctuations in Transit Systems

Jovan Pavlović
jovan.pavlovic@famnit.upr.si
University of Primorska,
FAMNIT,
Koper, Slovenia

Miklós Krész
miklos.kresz@innorenew.eu
InnoRenew CoE,
Izola, Slovenia

László Hajdu
laszlo.hajdu@famnit.upr.si
University of Primorska,
FAMNIT,
Koper, Slovenia

András Bóta
andras.bota@ltu.se
Luleå University of Technology,
Luleå, Sweden

ABSTRACT

This study focuses on identifying critical components within urban public transportation networks, particularly in the context of fluctuating demand and potential pandemic scenarios. By employing advanced agent-based simulations, we analyzed passenger interactions and ridership patterns across the San Francisco Bay Area's transit system. Key findings reveal specific transit stops and routes that are highly sensitive to changes in demand, often serving as bottlenecks or high-risk areas for the spread of infectious diseases.

KEYWORDS

network modeling, public transportation, agent-based simulation, community detection, demand fluctuations

1 INTRODUCTION

Efficient public transportation is vital for urban mobility, economic productivity, and public health. During the COVID-19 pandemic, transit systems worldwide were dramatically affected, resulting in a significant decline in ridership due to lockdowns, social distancing measures, and the shift to remote work [1, 6]. Physical distancing, a widely used non-pharmaceutical intervention to prevent the spread of the virus, further reduced the capacity of public transportation services, limiting their ability to meet demand [13].

Factors such as population growth, economic conditions, and environmental policies can also cause fluctuations in public transportation usage. Understanding these changes is crucial for planning resilient and efficient transit systems that can adapt to the evolving needs of cities.

Adjusting service frequency during peak and off-peak hours allows for more efficient use of resources and helps maintain service levels that meet demand without overloading the system. Additionally, rerouting or introducing new transit lines in underserved areas can improve accessibility and attract more users, or conversely, in the case of a pandemic, these changes can discourage usage to help manage public health risks. Infrastructure updates, such as upgrading stations for better crowd flow can also help transit systems adapt to changes. Therefore, it's important to identify the parts of the transit system that are most affected by changes in ridership to develop these strategies effectively.

This research uses agent-based simulations to analyze passenger interactions within transit systems. While networks traditionally depict routes and stops, improved data collection now allows tracking individual passenger interactions. Smart card data [12] and activity-based travel models [10, 11] capture detailed passenger contact patterns. However, creating accurate real-world contact networks from this data poses challenges, including computational complexity and privacy issues [4, 5].

We used activity-based travel demand models to simulate probable traveler paths in transit networks, considering demand, supply, and service details. These models are complemented by schedule-based transit assignment models, which provide accurate estimates of travel time and waiting times. We analyzed the outputs of transit assignments, considering transit route usage, congestion, and waiting times at transit stops, to identify critical components of the transit network that could be potentially affected by changes in transit demand. Additionally, we processed this data to generate contact networks. We then applied a modularity-based community detection algorithm to extract non-overlapping communities of passengers from the contact network and used these communities to further analyze critical bus routes used by different communities.

2 BACKGROUND

This work is inspired by the methodologies used in previous studies [2, 3, 7]. However, rather than explicitly modeling the spread of disease to identify high-risk transit components, it focuses on examining the components most likely to be affected by changes in ridership trends due to a pandemic or other scenarios.

The contribution of this work is to develop a framework that identifies critical components in terms of factors like changes in transit demand, vehicle capacities, and transit schedules. The insights derived from this framework can be further utilized for modeling transit operations in these scenarios.

3 METHODOLOGY

3.1 Transit simulation model

We used a schedule-based transit assignment model, FAST-TrIPs [8], to simulate passenger movement within the transit network. This model's time-dependent structure captures daily service variability

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.14
ISBN 978-961-286-914-4

and focuses on specific transit vehicle trips, which is crucial for accurately reflecting passengers' route choices based on the service schedule. FAST-TrIPs operates on a transit network composed of nodes that represent stops. Trips are connected to specific routes within this network, and transfer links connect nodes where passengers can change vehicles. This setup allows for precise modeling of both vehicle movements and passenger transfers across the transit network.

At the heart of FAST-TrIPs is the Transit Hyperpath Algorithm, which constructs a subnetwork of probable transit routes and assigns probabilities to these routes using a logit route choice model. The algorithm calculates hyperpaths by considering user-preferred arrival times and waiting time windows, allowing for the simulation of passenger journeys with a focus on real-time decision-making and path selection. Passenger movement is then modeled using a pre-estimated route choice model that incorporates factors such as in-vehicle time, waiting time, walking time (for access, egress, and transfers), and transfer penalties.

The transit assignment model generates detailed outputs, such as vehicle load profiles and passenger trajectories. The load profile provides information on the number of passengers boarding and alighting at each stop, along with timestamps, offering insight into passenger counts throughout the route. Passenger trajectories document each passenger's activities, including stop and vehicle IDs with timestamps, enabling the modeling of interactions between passengers.

3.2 Input data

FAST-TrIPs requires various input files, including transit system data stored in GTFS-PLUS format, and transit demand data that contains information about the trips individual passengers make throughout the day, including trip origins, destinations, and preferred arrival times. Additionally, path weights associated with in-vehicle time, waiting time, walking time, and transfer penalties must be specified as input.

In the current study, we used GTFS-PLUS data [1] from the San Francisco Bay Area in California from 2017, which includes 854 routes (covering bus, heavy rail, light rail, and ferry routes) and 36,058 trips serving 6,181 stops over a 24-hour weekday. On the demand side, we used data generated in the same year using the SF-CHAMP travel forecasting tool.

Since calibrated path weights were not available for the Bay Area network, we borrowed path weights from a previous study [9] corresponding to the Austin, Texas region.

3.3 Contact network

As mentioned previously, FAST-TrIPs outputs detailed passenger trajectories that can be further processed to produce a contact network. In this network, each passenger traveling within the transit system is considered a node, and edges connect any two passengers who share a vehicle trip for a positive time period. The vehicle trip refers to a specific route with a specific departure time and is unique to a single vehicle. Each edge is associated with three attributes: the contact start time, contact duration (in seconds), and the vehicle trip ID

[1]https://mtcdrive.app.box.com/s/3i3sjbzpsrbhxlwpl4v4vx9b0movferz

3.4 Community detection algorithm

We used the Clauset-Newman-Moore greedy modularity maximization algorithm [3] to find the community partition of the contact network with the highest modularity. This community detection algorithm is a hierarchical agglomeration method designed to efficiently identify community structures within large, sparse networks. Unlike traditional methods, which can be computationally expensive, this algorithm operates in a time complexity of $O(md \log n)$, where n is the number of vertices, m is the number of edges, and d is the depth of the dendrogram describing the community structure. For many real-world networks, which are sparse and hierarchical (with $m \sim n$ and $d \sim \log n$), the algorithm runs in nearly linear time, $O(n \log^2 n)$.

3.5 Limitations

The primary limitation of this study is the size of the demand data. Although the GTFS data originates from a transit network serving millions daily, computational constraints prevented us from simulating real-world demand accurately. Consequently, train routes were not filled beyond half capacity, making it impossible to realistically assess the effects of demand changes on the trains. Additionally, the dataset contains outdated transit system and demand information. However, the proposed method serves as a proof of concept and can be directly applied to more comprehensive travel datasets.

Another limitation is the lack of a detailed comparative study with state-of-the-art methodologies that aim to achieve similar objectives. This choice was due to space constraints, but future research will expand on this comparison, with findings to be published in a full-length journal paper.

4 RESULTS

4.1 Model outputs and contact network

Due to computational limits, the simulations used a reduced number of iterations to reassign passengers to alternative routes. Despite this, most passengers (41,845 out of 44,912) successfully reached their destinations, resulting in 83,280 completed trips.

Figure 1 presents a boxplot of average waiting times, aggregated by passenger, transit route, and transit stop.

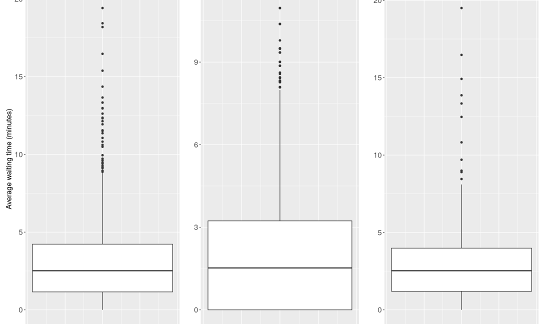


Figure 1: Boxplots of average waiting times

The derived contact network consisted of 41,845 passenger nodes and 3,530,995 contact links. The density plot of contact start times, displayed in Figure 2, peaked at 7 AM and 5 PM, reflecting typical weekday commutes. The average contact duration was 18 minutes and 43 seconds. Figure 4 shows the density plot of contact durations.

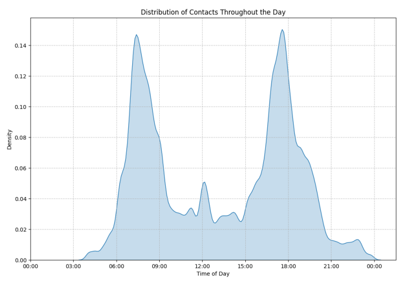


Figure 2: Density plot of contact stat times

The degree distribution of the contact network, shown in Figure 3, indicates an average of 134 contacts per person, with a maximum of 1,011, following a skewed power law distribution.

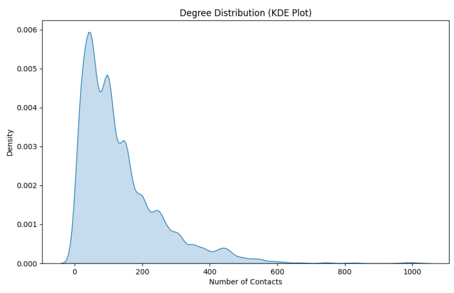


Figure 3: Degree distribution

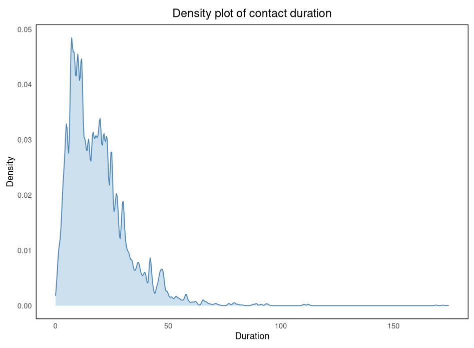


Figure 4: Density plot of contact duration

4.2 Identifying critical components

We first aimed to identify transit stops that may be sensitive to changes in demand. These stops are characterized by two key properties: they serve sufficiently large groups of people, and the average waiting times at these stops are longer than those at most other stops in the transit network. The idea is that such stops could become critical in scenarios where transit demand increases, potentially turning them into bottlenecks. Additionally, in epidemiological situations, passengers waiting at these stops might face an increased risk of infection. To focus on the most relevant stops, we filtered out those serving fewer than 100 people and sorted the remaining stops based on average waiting time. Table 1 provides information on the 10 stops with the longest average waiting times. Most of the listed stops are served by multiple bus routes and have between 100 and 200 passengers waiting at them throughout the day.

In order to identify critical transit routes we took two approaches. Firstly we identified routes whose vehicle trips are on average most congest. Due to limited transit demand here we focused only on bus routes. Table 2 shows 10 most critical bus routes identified in this way.

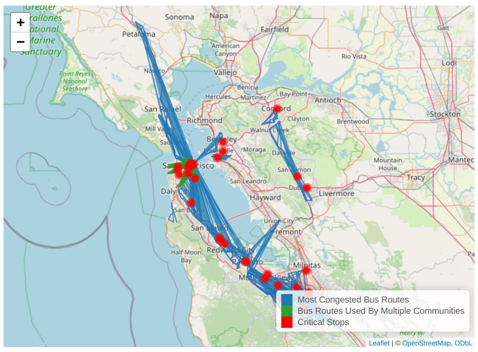
The second approach involved identifying critical trips with respect to the community structure of the contact network. Community detection algorithm divided the network into 627 communities, with the largest 10 containing 37% of all passengers in the network. We then identified transit routes used by passengers who appear in at least two of these ten communities and ranked the routes by the number of communities whose passengers travel on them. Table 3 shows the ten most critical routes identified in this way. As can be observed, all of the identified bus and trolleybus routes belong to the San Francisco Municipal Railway (SF Muni) system operating in San Francisco.

Figures 5 and 6 summarize the obtained results. Critical stops are marked in red, bus routes used by multiple communities are colored in green, and the most congested bus routes are marked in blue.

As observed, the majority of the most congested bus routes connect different cities within the Bay Area or link various cities to San Francisco. For example, several of these routes travel between Contra Costa and Alameda counties, as well as between San Mateo and Alameda counties. Additionally, some routes connect Berkeley and San Francisco, while many others link San Mateo County, Santa Clara County, Marin County, and Petaluma in Sonoma County to San Francisco. Most of the critical bus stops are concentrated in San Francisco, with several others located in the centers of various cities in the Bay Area, including Berkeley, Oakland, San Jose, and Palo Alto. As previously noted, the bus and trolleybus routes connecting different communities that commute in the Bay Area belong to the SF Muni system operating within San Francisco.

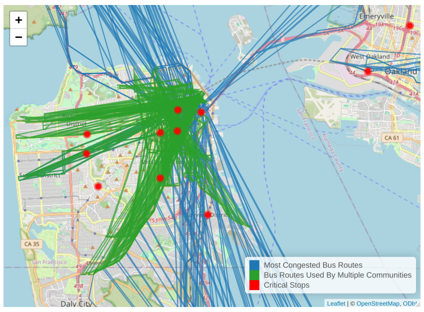
5 CONCLUSION

Using agent-based simulations and network analysis techniques, we identified transit stops and routes that are most vulnerable to changes in demand, whether due to a pandemic or other social and economic factors. Our findings show the importance of focusing on crowded routes and stops with long wait times, as these are likely to become bottlenecks when demand increases. The application of community detection to passenger contact networks further reveals how interconnected different transit routes are within major urban areas, emphasizing the importance of certain routes in keeping public transportation running smoothly.



Most Congested Bus Routes
Bus Routes Used By Multiple Communities
Critical Stops
Leaflet | © OpenStreetMap, ODbL

Figure 5: Critical components of San Francisco Bay Area transit system



Most Congested Bus Routes
Bus Routes Used By Multiple Communities
Critical Stops
Leaflet | © OpenStreetMap, ODbL

Figure 6: SF Muni transit routes

Table 1: Critical bus stops

Stop ID	Average waiting time	Number of serving routes
6420	18m 23s	5
6051	14m 21s	8
Folsom/8th	12m 8s	29
6336	11m 33s	11
103007	11m 4s	12
7186	8m 21s	16
103688	7m 51s	9
3rd/Mendell/Palou	7m 41s	6
13537	7m 13s	1
103589	6m 34s	4

Table 2: Most congested bus routes

Route ID	Capacity	Average number of pessengers
samtrans_83D	60	25
samtrans_16A	50	25
samtrans_292NA	75	20
scvta_101	63	19
scvta_182	63	18
samtrans_397	75	17
scvta_304	63	17
scvta_330	63	17
golden_gate_transit_GG76	63	14
samtrans_295	75	13

Table 3: Bus routes used by multiple communities

Route ID	Community appearance count
sf_muni_49	8
sf_muni_27	7
sf_muni_1	6
sf_muni_14	5
sf_muni_19	5
sf_muni_22	5
sf_muni_10	5
sf_muni_38_33RD	5
sf_muni_2	5
sf_muni_5	5

Additionally, during an epidemic outbreak, crowded routes and stops with long wait times can significantly contribute to the spread of infectious diseases. Areas where many passengers gather for extended periods are likely to become hotspots for infection transmission. Moreover, the interconnected nature of certain transit routes means that an outbreak starting in one part of the network could quickly spread to other areas, especially if key routes are involved. This highlights the need to closely monitor and manage these critical parts of the system to reduce the risk of widespread transmission. Implementing measures such as increasing service frequency, rerouting buses, or enhancing cleaning protocols at these critical points could be crucial in controlling the spread of diseases. These insights can also inform more effective public health strategies, such as prioritizing vaccination or testing efforts in areas served by these high-risk routes and stops.

REFERENCES

[1] Apple Inc. 2020, 2021, 2022. Apple Mobility Trends Reports.

[2] András Bóta, Lauren Gardner, and Alireza Khani. 2017. Identifying Critical Components of a Public Transit System for Outbreak Control. *Networks and Spatial Economics* 17, 4 (2017), 1137–1159.

[3] András Bóta, Lauren Gardner, and Alireza Khani. 2017. Modeling the spread of infection in public transit networks: a decision-support tool for outbreak planning and control. In *Transportation Research Board 96th Annual Meeting*.

[4] Ciro Cattuto et al. 2010. Dynamics of Person-to-Person Interactions from Distributed RFID Sensor Networks. *PLOS ONE* 5 (2010), e11596.

[5] Nicholas A. Christakis and James H. Fowler. 2010. Social Network Sensors for Early Detection of Contagious Outbreaks. *PLOS ONE* 5 (2010), e12948.

[6] Google LLC. 2020, 2021, 2022. Google COVID-19 Community Mobility Reports. Retrieved from https://www.google.com/covid19/mobility/.

[7] Laszlo Hajdu, András Bóta, Miklós Krész, et al. 2020. Discovering the Hidden Community Structure of Public Transportation Networks. *Networks and Spatial Economics* 20 (2020), 209–231.

[8] Alireza Khani. 2013. *Models and solution algorithms for transit and intermodal passenger assignment (development of fast-trips model).* PhD Dissertation. University of Arizona, Tucson, AZ, USA.

[9] Alireza Khani, Tyler J. Beduhn, Jennifer Duthie, Stephen Boyles, and Ehsan Jafari. 2014. A transit route choice model for application in dynamic transit assignment. In *Innovations in Travel Modeling*. Baltimore, MD.

[10] William H. K. Lam and Hai-Jun Huang. 2003. Combined Activity/Travel Choice Models: Time-Dependent and Dynamic Versions. *Networks and Spatial Economics* 3, 3 (2003), 323–347.

[11] Matthew J. Roorda, Juan A. Carrasco, and Eric J. Miller. 2009. An integrated model of vehicle transactions, activity scheduling and mode choice. *Transportation Research Part B: Methodological* 43, 2 (2009), 217–229.

[12] Liang Sun, Kay W. Axhausen, Dong H. Lee, and Manuel Cebrian. 2014. Efficient Detection of Contagious Outbreaks in Massive Metropolitan Encounter Networks. *Scientific Reports* 4 (2014), 5099.

[13] Alejandro Tirachini and Oded Cats. 2020. COVID-19 and Public Transportation: Current Assessment, Prospects, and Research Needs. *Journal of Public Transportation* 22, 1 (2020), 1–21. Epub 2022 Sep 13.

Automatic Assessment of Bradykinesia in Parkinson's Disease Using Tapping Videos

Matjaž Zupanič
matjaz.zupanic1@gmail.com
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

Dejan Georgiev
dejan.georgiev@kclj.si
University of Ljubljana,
Faculty of Medicine,
Ljubljana, Slovenia

Jure Žabkar
jure.zabkar@fri.uni-lj.si
University of Ljubljana,
Faculty of Computer and
Information Science,
Ljubljana, Slovenia

ABSTRACT

Parkinson's disease is a chronic neurodegenerative illness that severely affects the everyday life of a patient. The severity of Parkinson's disease is assessed using the MDS-UPDRS scale. In this study, we explore the feasibility of automatically evaluating bradykinesia, a key symptom of Parkinson's disease, from tapping videos recorded on smartphones in everyday settings. We collected a dataset of 183 tapping videos, from 91 individuals. Videos were assessed by neurologist into 5 classes of the MDS-UPDRS scale. For data extraction, we employed MediaPipe Hand, which provides a time series of hand skeleton movements. The data was preprocessed to eliminate noise and subsequently used for either feature construction or directly in neural networks. Utilizing manually created features in a multilayer perceptron classifier resulted in 61 % accuracy and an F1 score of 0.61 on our test set. Employing a fully convolutional network, we improved the accuracy to 78 % and the F1 score to 0.75. Additionally, we developed the tool for visualising tapping and displaying key data, providing detailed insights into tapping patterns.

KEYWORDS

bradykinesia evaluation, finger tapping test, Parkinson's disease, machine learning, tapping video analysis

1 INTRODUCTION

Parkinson's disease (PD) is a chronic neurodegenerative condition that profoundly impacts daily life. PD affects 1-2 % [23] of the population over the age of 65. Currently, there are more than 1.2 million cases in Europe [4] and this number is forecast to double in the near future due to the demographic problem of an aging population. Its etiology remains incompletely understood, yet researchers suggest that a combination of genetic and environmental factors contributes to its development. Factors such as exposure to polluted air, pesticides, heavy metals, and head injuries have been associated with an increased risk of Parkinson's disease. The most common symptoms include bradykinesia, which is also the main symptom, tremor, rigidity, impaired postural reflexes, and dementia. There are also numerous other symptoms that can accompany the disease, such as sleep disturbances, depression, loss of smell, and fatigue.

The standardized MDS-UPDRS [6] scale is used to assess the stage of Parkinson's disease. It consists of 4 sections that evaluate both motor and non-motor issues experienced by patients. The finger-tapping test is used to evaluate the severity of bradykinesia. This test involves asking the individual to tap their index finger and thumb as quickly as possible with a maximal span, assessing the number of pauses, time taken, decrease in amplitude, and slowing of speed, all contributing to the final score. It was estimated that up to 25 % of clinical diagnoses of PD are incorrect, due to lack of experience or attention during tests [3].

2 RELATED WORK

First automated systems for PD detection were based on wearable sensors like gyros and accelerometers [5, 17, 20, 22] or on electromyography sensors [11, 28]. The main issues with sensors are that they are commercially unavailable, require precise placement, and can interfere with tapping test. Therefore, some researchers have utilized keyboard tapping [1, 19] or tapping on a smartphone screen [7, 8] for data acquisition. Sadikov et al. [21] collected data using digital spirometry, where participants traced an Archimedes spiral on a touchscreen device. Advances in hardware and software have made computer vision combined with machine learning a viable alternative for PD recognition, allowing for home testing using a computer or smartphone. Lainscsek et al. [13] used a nonlinear delay differential equation, with the structure selected by a genetic algorithm. While other researchers used machine learning techniques, most focused on manual feature creation and utilized these features in classification models [10, 18, 29, 30] like support vector machines (SVM) and random forests (RF), or regression models [9] like support vector regression (SVR) and XGBoost. Others employed neural networks (NN) to automatically learn patterns from time series data [2, 14]. Researchers used segmentation neural networks or optical flow for hand data extraction, with MediaPipe Hand [16] becoming popular in later works. Due to limited data, most studies combined some classes, and only a few performed full-scale classification [2, 9, 14, 30].

3 METHODOLOGY

First, we collected and labeled data and preprocessed it to eliminate noise. We initially applied Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and Random Forest (RF) with manual feature extraction, then used a Fully Convolutional Network (FCN) directly on the time series.

3.1 Dataset

Since datasets of tapping videos were not publicly accessible, we assembled our own database. From each participant, we collected two videos: one of the left-hand tapping and another of the right-hand, as they are independent from each other and have their own score. Videos were recorded at a resolution of 3840 x 2160 or 1920

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.15
ISBN 978-961-286-914-4

x 1080 at 60 or 30 fps. All PD patients were recorded in a clinical setting, while some participants of the healthy group were recorded in various other environments.

Table 1: Number of videos in each class.

MDS-UPDRS score	0	1	2	3	4	Sum
Number of videos	49	51	53	23	7	183
Percentage %	26	28	29	13	4	100

We excluded videos with significant tilts, incomplete hand visibility, and participants scoring above 0 on MDS-UPDRS without confirmed PD. In total, we compiled 183 videos from 91 different individuals. The distribution of data between classes can be observed in Table 1. As this study involves clinical data, ethical approval was obtained as part of a larger research project approved by the Neurological Clinic.

3.2 Preprocessing

Since datasets were collected without any professional equipment we were dealing with different illuminations, angles, camera tilts, distances between camera and hand, noise, and motion blur. Videos were cut, so only hand is visible, due to privacy and faster processing. We used Mediapipe Hand [16] to extract the thumb and index finger positions and computed the Euclidean distances between their endpoints to generate time series data. Occasionally, there were single-frame misses where Mediapipe detected previous and next frames but missed the current one. We used linear interpolation to fill these gaps, but avoided interpolating longer gaps to preserve tapping integrity. To reduce noise caused by inaccurate detections from MediaPipe, we implemented a combination of low-pass and moving average filter. Filtering also helped to eliminate tremor which is not part of MDS-UPDRS, but masked tapping. We balanced filtering and signal preservation using weaker filters. The Low-pass filter addressed high-frequency noise from tremors and MediaPipe Hand's misalignment, which caused slight shifts between frames even when the hand was still. We implemented the Butterworth low-pass filter due to its flat response. The cutoff frequency set uniquely for each input based on a specified influence percentage k, as defined by the equation: $f_{cutoff} = f_{max} + \frac{\text{bandwidth}*k}{2}$. Additionally, a moving average with a window size of 5 was used to further smooth the data and reduce erroneous detections, such as spikes. We restricted tapping sequences to a maximum of 15 taps, as the MDS-UPDRS scale does not require longer sequences and participants may tire during extended sessions. We also applied min-max normalization to account for varying distances between the camera and the hand. The finalised graphs of the processed data are depicted in Figure 1.

3.3 Feature Engineering

In the **1st method** we created a larger number of features following the MDS-UPDRS scale to comprehensively describe finger tapping. In addition to Euclidean distances between the endpoints of the thumb and index finger, we included distances between the last joints and absolute wrist movement as additional time series data. We hypothesized that these metrics could provide supplementary but limited insights into finger tapping, although movements in these areas are typically less pronounced. The time series of distances represents the amplitude spectrum, from which we derived 4 additional spectra: velocity, acceleration, frequency, and spectrum of amplitude peaks. Additionally, we included the spectrum of absolute wrist movement. From these 6 spectra, we extracted 193 statistical features. Our goal was to capture dependencies at global and local levels, describing hesitations, slowdowns, amplitude decreases, data distributions, tapping energy, and other characteristics.

In **2nd method** we designed features closely aligned with the MDS-UPDRS scale, categorizing them into 3 parts reflecting its criteria. The first part assesses hesitations and freezes, the second part measures reduced speed and the third part evaluates decreased amplitudes. In our final analysis, we utilized 145 features, with 90 being coefficients from linear regressions to assess reductions in amplitudes and velocities of finger tapping. These coefficients were derived from local maxima of amplitudes and velocities. The remaining 55 features were derived from amplitudes, velocities, their extremes, and autocorrelated velocities and amplitudes.

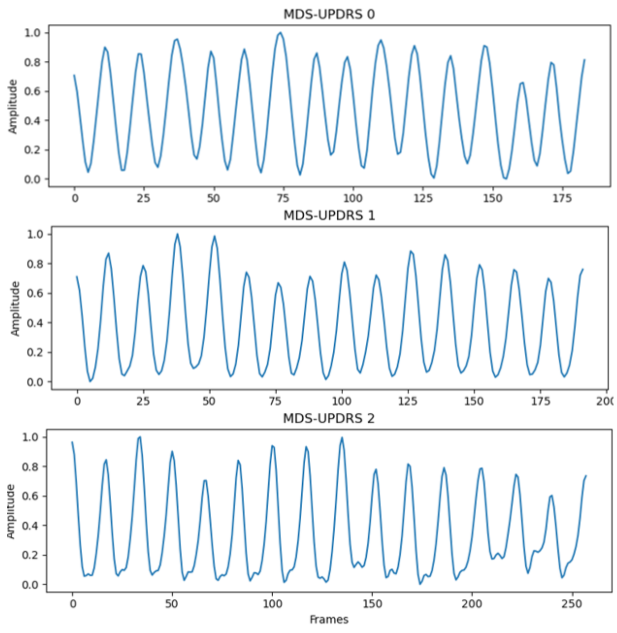


Figure 1: Amplitude graphs of finger tapping.

3.4 Neural network

We opted for FCN due to its simplicity and efficiency, leaving the exploration of more complex models for future work. We tested the FCN presented by Li et al. [14], using preprocessed time series data directly as input. Since the selected FCN is limited to processing equally long inputs, we padded our time series data with 0 at the end. We later modified the FCN by adding convolutional layers, dropout layers, early stopping, and adjusted input layers to handle 2D inputs consisting of amplitudes and velocities. The architecture of our extended FCN is shown in Figure 2.

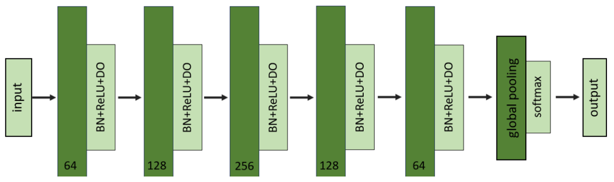


Figure 2: FCN for classification of bradykinesia.

4 EVALUATION

We created a visualization tool for analyzing finger-tapping dynamics, featuring a built-in video player with a MediaPipe Hand skeleton overlay. Velocity and amplitude graphs on the right side indicate the current frame with a vertical red line (Figure 3). Taps are denoted by red dots and the tool includes a freeze labelling option. This interactive analysis of tapping dynamics could be useful for neurologists in diagnosing and monitoring motor disorders.

All tests were conducted using 10-fold cross-validation due to the relatively small dataset. The F1 score was calculated as the Macro F1= $\frac{1}{C}\sum_{i=1}^{C} \mathrm{F1}_i$, where C is the number of classes and $\mathrm{F1}_i$ is the F1 score for class i. In Methods 1 and 2, we used SelectKBest [26] for feature selection. As score functions we tried ANOVA F-test [24] and mutual information for a discrete target [25], with the latter performing better overall. By experimenting with various feature counts, we identified the optimal number that maximized the model's F1 score. Results are detailed in Table 2. Overall, SVM performed the worst in Methods 1 and 2, while RF and MLP achieved the best results, likely due to their superior ability to model complex patterns. Method 1 outperformed Method 2 across all metrics. Approximately 83 % and 73 % of misclassifications from Methods 1 and 2, respectively, differed from the reference tapping scores by exactly 1 class. The FCN from Figure 2 significantly outperformed Methods 1 and 2, achieving a 77 % accuracy. FCN excels at capturing both local and global dependencies in signals by using filters of varying sizes.

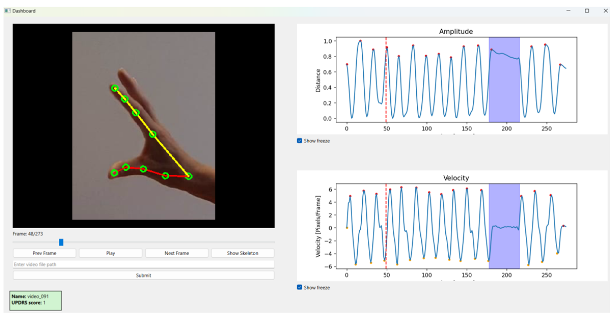


Figure 3: Tool for detailed visualization of finger tapping.

5 DISCUSSION

Our data set was diverse, assembled by tapping videos of different people among all MDS-UPDRS classes. Since the dataset was collected without any professional equipment we were dealing with different illuminations, angles, camera tilts, distances between hand and camera, noise, and motion blur. That required a robust approach, with filtering being an important part. To balance noise reduction and signal preservation, we opted for milder filtering.

Table 2: Results were obtained via 10-fold cross-validation. FCN refers to Li et al.'s neural network [14], while FCN+ denotes our modified version (Figure 2). MLP was used for Method 1, while RF was used for Method 2.

Model	Accuracy %	F1 %	Precision %	Recall %
FCN	72	62	84	63
FCN+	**77**	**75**	**88**	**75**
Method 1	61	62	67	58
Method 2	60	55	67	58

Due to the low capture speed and fast movement of fingers, motion blur was present in the videos. To address this, we tested two NNs for motion blur removal: Ghost-DeblurGAN [15] and PDV_net [27]. However, both methods introduced artefacts in the frames, prompting us to discontinue their use. We also experimented with upscaling the resolution to 200 % of the original size using Video2x [12]. This aimed to enhance image clarity, potentially improving MediaPipe Hand's skeleton detection precision and reducing noise. Testing on a smaller upscaled subset showed minimal differences in classification performance but significantly increased processing time, prompting us to abandon this approach due to time constraints. Similarities were observed across different classes of tapping, as shown in Figure 1, where the graphs appear similar. Various factors contribute to the overlapping of classes. Small differences in tapping styles between adjacent classes mean even minor decreases in velocity or amplitude can significantly impact the final classification. Normalization contributes to reduced differentiation between classes by masking tapping instances with low amplitudes. However, it is necessary to account for variations in recording distances and resolutions. Additionally, recording angles can distort actual finger distances, leading to misleading data representation. Another factor for class overlap is the possibility of human errors in video assessments. However, within the same class, tapping behaviours can vary significantly. For example, in class 4, participants often showed varying abilities: some managed to perform a few taps despite severe difficulties, while others were unable to tap at all.

Direct comparisons with related research may be challenging due to the use of different datasets. When comparing our Methods 1 and 2 with the method by Yu et al. [30], who derived features based on MDS-UPDRS, we achieved lower scores. They reported 80 % accuracy and 79 % recall on a test set of only 15 videos recorded as close to a 90-degree angle as possible. Frame interpolation they used might distort tapping details with artificial data, risking the reliability of their classification outcomes. Islam et al. [9] investigated SVR, LightGBM, and XGBoost regressor, achieving up to 55 % accuracy. This is lower than the 61 % accuracy we achieved with Method 1, possibly due to their larger database of 489 videos, less effective preprocessing and a feature set of 65 features that may not fully capture tapping dynamics. The FCN presented by Li and colleagues [14] achieved 72 % accuracy on our dataset, compared to the 80 % reported by the authors. We attribute the slightly lower classification performance on our data to its complexity and heterogeneity, over 37 % smaller size, and the use of 10-fold cross-validation compared to their 5-fold approach. However, by enhancing the FCN (Figure 2) we improved prediction accuracy to 77 %. Alam et al. [2] reported 81 % accuracy and F1 score of 0.81 on their test set using a graph neural networks (GNN).

6 CONCLUSION

In conclusion, Method 1 with MLP provided the best performance between the manual methods, with better overall metrics and 10 % fewer misclassifications differing by one class from the reference value. The modified FCN+ (Figure 2) further improved accuracy to 77 %. Results are expected, since with manual time-invariant feature extraction it is challenging to capture all unique patterns at various scales in a tapping sequence of around 400 frames. In the future, we plan to expand the dataset, as our class with an MDS-UPDRS score of 4 had a limited population. Increasing the dataset will help achieve more precise results and provide more data to create an unseen test set. Due to only one labeler, we plan to involve another neurologist to cross-validate our dataset and eliminate potential human errors. We also plan to explore regression models on extracted features to predict continuous severity scores, offering a more detailed evaluation of bradykinesia.Given neural networks' superior performance, we aim to explore graph neural networks (GNN) for handling all data points extracted by MediaPipe.

REFERENCES

[1] Warwick R Adams. 2017. High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing. *PLOS ONE* 12, 11 (11 2017), 1–20. https://doi.org/10.1371/journal.pone.0188226

[2] Zarif U Alam, Saiful Islam, Ehsan Hoque, and Saifur Rahman. 2023. PULSAR: Graph based Positive Unlabeled Learning with Multi Stream Adaptive Convolutions for Parkinson's Disease Recognition. https://doi.org/10.48550/ARXIV.2312.05780

[3] Nin P S Bajaj, Vamsi Gontu, James Birchall, James Patterson, Donald G Grosset, and Andrew J Lees. 2010. Accuracy of clinical diagnosis in tremulous parkinsonian patients: a blinded video study. *Journal of Neurology, Neurosurgery & Psychiatry* 81, 11 (2010), 1223–1228. https://doi.org/10.1136/jnnp.2009.193391

[4] Parkinson's Europe. 2024. Parkinson's Statistics. https://parkinsonseurope.org/facts-and-figures/statistics/ Accessed: 7-3-2024.

[5] Joseph P Giuffrida, David E Riley, Brian N Maddux, , and Dustin A Heldman. 2009. Clinically deployable Kinesi technology for automated tremor assessment. *Movement Disorders* 24, 5 (2009), 723–730. https://doi.org/10.1002/mds.22445

[6] 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, Bruno Dubois, Robert Holloway, Joseph Jankovic, Jaime Kulisevsky, Anthony E Lang, Andrew Lees, Sue Leurgans, Peter A LeWitt, David Nyenhuis, Warren C Olanow, Olivier Rascol, Anette Schrag, Jeanne A Teresi, Jacobus J van Hilten, and Nancy LaPelle. 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.

[7] Dimitrios Iakovakis, Stelios Hadjidimitrio, Vasileios Charisis, Sevasti Bostantjopoulou, Zoe Katsarou, Lisa Klingelhoefer, Heinz Reichmann, Sofia B Dias, José A Diniz, Dhaval Trivedi, Ray K Chaudhuri, and Leontios J Hadjileontiadis. 2018. Motor impairment estimates via touchscreen typing dynamics toward Parkinson's disease detection from data harvested in-the-wild. *Frontiers ICT* 5 (2018).

[8] Dimitrios Iakovakis, Stelios Hadjidimitriou, Vasileios Charisis, Sevasti Bostantzopoulou, Zoe Katsarou, and Leontios J Hadjileontiadis. 2018. Touchscreen typing-pattern analysis for detecting fine motor skills decline in early-stage Parkinson's disease. *Scientific Reports* 8, 1 (2018).

[9] Saiful Islam, Wasifur Rahman, Abdelrahman Abdelkader, Sangwu Lee, Phillip T Yang, Jennifer L Purks, Jamie L Adams, Ruth B Schneider, Earl R Dorsey, and Ehsan Hoque. 2023. Using AI to measure Parkinson's disease severity at home. *npj Digital Medicine* 6, 156 (2023). https://doi.org/10.1038/s41746-023-00905-9

[10] Jacek Jakubowski, Anna P Chromik, Jolanta Chmielinska, Monika Nojszewska, and Anna K Pruszczyk. 2023. Application of imaging techniques to objectify the Finger Tapping test used in the diagnosis of Parkinson's disease. *Bulletin of the Polish Academy of Sciences. Technical Sciences* 71 (2023), art. no. e144886. https://doi.org/10.24425/bpasts.2023.144886

[11] Hyoseon Jeon, Woongwoo Lee, Hyeyoung Park, Hong J Lee, Sang K Kim, Han B Kim, Beomseok Jeon, and Kwang S Park. 2017. Automatic Classification of Tremor Severity in Parkinson's Disease Using a Wearable Device. *Sensors (Basel)* 17, 9 (Sept. 2017).

[12] k4yt3x. 2024. Video2x. https://github.com/k4yt3x/video2x Accessed: 16-02-2024.

[13] Claudia Lainscsek, Peter Rowat, Luis Schettino, Dongpyo Lee, D Song, Cristophe Letellier, and Howard Poizner. 2012. Finger tapping movements of Parkinson's disease patients automatically rated using nonlinear delay differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22, 1 (2012). https://doi.org/10.1063/1.3683444

[14] Zhu Li, Lu Kang, Miao Cai, Xiaoli Liu, Yanwen Wang, and Jiayu Yang. 2022. An Automatic Evaluation Method for Parkinson's Dyskinesia Using Finger Tapping Video for Small Samples. *Journal of Medical and Biological Engineering* 42, 3 (1 2022), 351–363. https://doi.org/10.1007/s40846-022-00701-y

[15] Yibo Liu, Amaldev Haridevan, Hunter Schofield, and Jinjun Shan. 2022. Application of Ghost-DeblurGAN to Fiducial Marker Detection. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 6827–6832. https://doi.org/10.1109/IROS47612.2022.9981701

[16] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo L Chang, Ming G Yong, Juhyun Lee, Wan T Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. 2019. MediaPipe: A Framework for Building Perception Pipelines. *ArXiv* abs/1906.08172 (2019). https://doi.org/10.48550/arXiv.1906.08172Focustolearnmore

[17] ALexander Meigal, Saara Rissanen, Mika P Tarvainen, Stefanos Georgiadis, Pasi A Karjalainen, Olavi Airaksinen, and Markku Kankaanpää. 2012. Linear and nonlinear tremor acceleration characteristics in patients with Parkinson's disease. *Physiological measurement* 33, 3 (2012), 395.

[18] Adonay S Nunes, Nataliia Kozhemiako, Christopher D Stephen, Jeremy D Schmahmann, Sheraz Khan, and Anoopum S Gupta. 2022. Automatic Classification and Severity Estimation of Ataxia From Finger Tapping Videos. *Frontiers in Neurology* 12 (2022).

[19] Atemangoh B Peachap, Daniel Tchiotsop, Valérie Louis-Dorr, and Didier Wolf. 2020. Detection of early Parkinson's disease with wavelet features using finger typing movements on a keyboard. *SN Applied Sciences* 2, 10 (2020).

[20] Cameron N Riviere, Stephen G Reich, and Nitish V Thakor. 1997. Adaptive Fourier modeling for quantification of tremor. *Journal of Neuroscience Methods* 74, 1 (1997), 77–87. https://doi.org/10.1016/S0165-0270(97)02263-2

[21] Aleksander Sadikov, Jure Žabkar, Martin Možina, Vida Groznik, Dag Nyholm, and Mevludin Memedi. 2015. Feasibility of Spirography Features for Objective Assessment of Motor Symptoms in Parkinson's Disease. In *Artificial Intelligence in Medicine*, Lucia Sacchi John H Holmes, Riccardo Bellazzi and Niels Peek (Eds.). Springer International Publishing, Cham, 267–276.

[22] Arash Salarian, Heike Russmann, Christian Wider, Pierre R Burkhard, Françios J G Vingerhoets, and Kamiar Aminian. 2007. Quantification of Tremor and Bradykinesia in Parkinson's Disease Using a Novel Ambulatory Monitoring System. *IEEE Transactions on Biomedical Engineering* 54, 2 (2007), 313–322. https://doi.org/10.1109/TBME.2006.886670

[23] Claudia Schulte and Thomas Gasser. 2011. Genetic basis of Parkinson's disease: inheritance, penetrance, and expression. *The Application of Clinical Genetics* 4 (2011), 67–80. https://doi.org/10.2147/TACG.S11639

[24] scikit learn. 2024. fclassif. https://parkinsonseurope.org/facts-and-figures/statistics/ Accessed: 7-3-2024.

[25] scikit learn. 2024. mutual info classif. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif Accessed: 7-3-2024.

[26] scikit learn. 2024. Select K Best. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html Accessed: 28-02-2024.

[27] Hyeongseok Son, Junyong Lee, Jonghyeop Lee, Sunghyun Cho, and Seungyong Lee. 2021. Recurrent Video Deblurring with Blur-Invariant Motion Estimation and Pixel Volumes. *ACM Trans. Graph.* 40, 5, Article 185 (2021), 18 pages.

[28] Molly M Sturman, David E Vaillancourt, , and Daniel M Corcos. 2005. Effects of aging on the regularity of physiological tremor. *Journal of neurophysiology* 93, 6 (2005), 3064–3074.

[29] Stefan Williams, Samuel D Relton, Hui Fang, Jane Alty, Rami Qahwaji, Christopher D Graham, and David C Wong. 2020. Supervised classification of bradykinesia in Parkinson's disease from smartphone videos. *Artificial Intelligence in Medicine* 110 (2020), 101966. https://doi.org/10.1016/j.artmed.2020.101966

[30] Tianze Yu, Kye W Park, Martin J McKeown, and Jane Z Wang. 2023. Clinically Informed Automated Assessment of Finger Tapping Videos in Parkinsonrsquo;s Disease. *Sensors* 23, 22 (2023). https://doi.org/10.3390/s23229149

Exploring Mathematical Decision-Making Through EEG Analysis

Riste Micev
riste.micev1@gmail.com
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Koper, Slovenia

Peter Rogelj
peter.rogelj@upr.si
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Koper, Slovenia

ABSTRACT

In this study, mathematical decision-making tasks were used to provide further details on the flow of information across a number of brain regions, with the objective of finding out whether connectivity patterns are informative in predicting decisional outcomes. The experiment consisted of showing 50 mathematical expressions to each participant, and they decided on their correctness by pressing buttons. Neural activity and button presses were recorded by means of the g.tec Nautilus EEG device, equipped with 64 electrodes. A detailed epochs analysis was conducted with regard to participant responses. Advanced techniques of signal analysis were applied, including Granger causality, Phase Locking Value, and Complex Pearson Correlation Coefficient. This research aims to determine how the following tools could distinguish events from states, get aware of their limitations, and develop novel analysis techniques for better discrimination of brain processes. This research is specifically focused on using mathematical reasoning as a model to study decision-making processes. Our objective is to test existing and develop novel methods for gaining deeper understanding of the brain dynamics involved in discrete cognitive activities.

KEYWORDS

EEG, mathematical decision-making, neural connectivity, connectivity analysis, neural signal classification, EEGNet

1 INTRODUCTION

1.1 Background

Electroencephalography (EEG) devices are core tools in neuroscience for the monitoring of brain activity through the detection of electrical potentials in different places on the scalp [4]. They find applications in a wide variety of both clinical and research settings during the investigation of brain activity and connectivity. The ability to understand the brain processes is crucial for advancements in neuroscience, medical diagnostics and brain-computer interfaces. Identification and improvement of methods that are capable of classifying events and explaining the underlying decision process is also of a great importance.

In this study we aim to set the whole pipeline for conducting such research, which consists of data acquisition and analysis. For our brain process of interest we selected mathematical reasoning, which exemplifies decision-making processes. It is selected because of its complexity that enables layered analysis of sub processes, as mathematical thinking involves complex cognitive processes that engage multiple brain regions. Mathematical decision-making tasks require the integration of numerical processing, working memory, and logical reasoning, making them an ideal for studying brain connectivity.

1.2 Objectives

Our primary objective with this research is to assess existing techniques for connectivity analysis and to develop a comprehensive pipeline for the analysis of brain processes (during mathematical decision-making tasks), with the focus on creation and refinement of methodologies that would be able to classify and explain these processes. Through the comprehensive analysis we also aim to validate two specific hypotheses.

- **H1** - Mathematical thinking causes unique connectivity patterns, differentiable from resting state brain activity.
- **H2** - True and false answers can be distinguished by their EEG signals.

The motivation behind this study is to contribute to the understanding of cognitive processes by providing insights into the neural dynamics of decision-making.

2 LITERATURE REVIEW

EEG has established itself as an invaluable tool in cognitive neuroscience, particularly for exploring brain activity in real time. Its application in understanding the neural mechanisms of mathematical decision-making has drawn considerable interest due to the dynamic and complex nature of the task. Previous studies have demonstrated that specific brain regions, particularly within the frontal and parietal lobes, are significantly active during mathematical cognition, reflecting the intricate process of problem-solving and numerical reasoning [1, 2].

Basic methods for EEG analysis rely on statistical analysis of independent electrode signals, and do not enable reliable differentiation of complex brain activities. This can be achieved by additionally considering the mutual signal interdependence as a reflection of utilization of brain networks, known as brain connectivity analysis. There are several accepted brain connectivity analysis methods, which include Phase Locking Value (PLV), Weighted Phase Locking Index (wPLI), Complex Pearson Correlation Coefficient (CPCC) [7] and Granger Causality (GC). Most methods including PLV, wPLI and CPCC are not directional and rely on analyzing phase differences between the electrode signals. GC is a directional method, developed by Clive Granger in the 1960s, and determines whether one time series can help predicting another one. Applied on EEG data it can reveal directional influences between different neural regions covered by corresponding EEG electrodes.

Granger causality has been widely used in neuroscience to explore the temporal dynamics of brain activity. For example, it has

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.16
ISBN 978-961-286-914-4

been applied to EEG signals to investigate the functional connectivity between different brain areas during various cognitive states. Recent research by Seth, Barrett, and Barnett (2015) [6] has further demonstrated the effectiveness of GC in identifying directed functional interactions in neuroscience and neuroimaging time-series data. Their findings indicate that GC can reveal insights into the functional circuits involved in perception, cognition, and behavior. This research also emphasizes both the theoretical foundations and practical applications of GC while discussing its limitations and computational strategies, thus solidifying its role as a crucial tool in neuroscience.

With the advances in artificial intelligence the use of artificial neural networks also affects EEG analysis. One of the most promising artificial neural network architectures for classification of EEG data is EEGNet [3], a compact convolutional neural network designed for EEG-based brain-computer interfaces. Recently, it has been shown that neural networks can contribute to understanding of underlying processes, by computing saliency maps [5]. As such, artificial neural networks could also be extended and utilized to reveal connectivity patterns.

3 METHODOLOGY

3.1 Participants

For the purpose of the study, we recruited 15 participants from the university. Each participant provided written informed consent before participating in this experiment. This work was approved by the university's ethical committee to ensure the study conformed to ethical standards for studies involving human participants.

3.2 Equipment

- EEG Headset: g.tec Nautilus EEG device with 64 channel electrodes.
- Base Station: Connected to the EEG headset for data transmission.
- Trigger Box: Connected to the base station, equipped with two response buttons.
- Optical Sensor: Connected to the trigger box to detect changes in the visual stimuli.
- Recording Software: g.recorder for capturing and storing EEG data.

3.3 Procedure

The experiment was set in the following way:

A participant was seated comfortably in a noise-free, dimly lit room to help eliminate other external factors that might cause discomfort. An EEG headset was fitted on the head of the participant, making sure that the contact of all the electrodes on the scalp is good to ensure high-quality signals. The headset was then connected to the base station and trigger box.

The experiment consisted of 50 mathematical equations that were shown for 10 seconds each on a computer screen, where the research participants had to determine whether the equation was correct or incorrect. Responses were marked by pressing one of the two corresponding buttons connected to the trigger box. At the end of each equation, there was a resting phase of 3 seconds where the subjects could rest before the next equation appears.

An optical sensor was used to exactly capture the display time of each equation, thus ensuring correct synchronization with the visual presentation and EEG data. This setup allowed for exact timing of participant responses relative to the presentation of the equations.

In this experiment we recorded EEG data using the g.recorder software, that captured the continuous EEG signals on all activities of 64 channels at a sampling rate of 250 Hz. The software also recorded the presentation timings of the stimuli and participant responses according to the detection of the optical sensor and trigger box. This setup ensured that all relevant events will be accurately time-stamped and synchronised with the EEG data.

Each of the participants completed the experiment in an individual session, which in total lasted approximately 15 minutes. The data was stored safely for later preprocessing and analysis.

3.4 Data Collection

Figure 1 shows the raw EEG data recorded from one of the subjects while performing the mathematical decision-making task. EEG signals, captured from 64 channels, are shown here along with their respective labels on the y-axis, which represent electrode positions at different locations on the scalp, and the x-axis represents time in seconds.

In this visualization, specific triggers are marked to indicate important events during the experiment. The green line represents a trigger at the moment the equation on screen changes, thus signaling the beginning of a new arithmetic problem. This trigger is important in synchronizing EEG data with the exact moment each equation is presented to the participant, thus providing the ability to analyze the neural response to the stimulus very precisely.

The red line marks the trigger corresponding to the event of a user pressing the "incorrect" button, thus signaling his/her decision that the equation presented is wrong.

Although not shown in the current image, a blue line is used as a trigger to mark the event when a participant presses the "correct" button, indicating their decision that the equation is correct.

3.5 Data Preprocessing

One of the most important steps in satisfying the quality and reliability of the recorded signal before detailed analysis is the preprocessing of EEG data. MATLAB with the EEGLAB toolbox was used for this purpose, where advanced functionalities were applied to deal and handle with the intricate nature of EEG data. The preprocessing pipeline started by filtering all frequencies of the raw EEG data outside the frequency range of interest. That was easily accomplished with the help of a bandpass filter with the limiting frequencies of 0.1 Hz and 45 Hz. This filtering step was quite important to avoid noise or other effects due to muscle activity, electrical interference, etc.

After filtering, the EEG data was re-referenced to the common average reference. This involved averaging the signal from all electrodes and subtracting this average from each individual electrode's signal to give clarity to the signal and remove common noise. Common average referencing is conducted as a standard operation in preprocessing when carrying out EEG. This operation helped to normalize data across different channels.

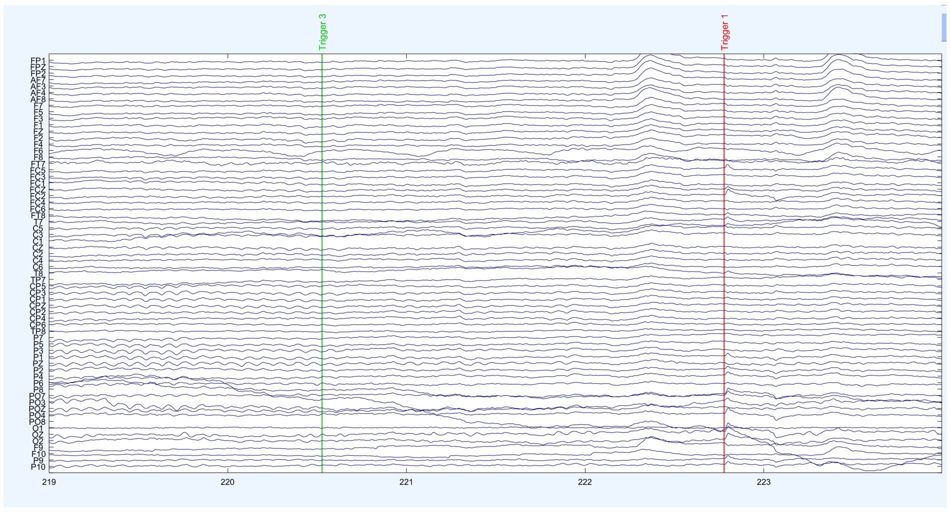


Figure 1: Visualisation of the raw signals data.

Artifacts were removed by Independent Component Analysis (ICA), where the EEG signal was decomposed into independent components. With the help of ICA and the ICLabel add-on in MATLAB, components related to common artifacts due to eye movements, blinks, and muscle activity were identified and isolated. Removing these artifact components from the data ensured that the remaining signals are more representative of the true neural activity.

3.6 Hypothesis Testing

The basic idea of our hypothesis testing approach revolves around developing and training classification methods on epochs that we define specifically around key events (equation change, incorrect/correct marking). For example, for testing the first hypothesis **H1**, two primary states can be defined:

- **Rest** - Epochs taken from a 3-second window just before a new equation appears - shown by this green line in our recording setup. This is the period not active for making any judgment, which in turn gives our baseline or rest state.
- **Active** - In contrast active state epochs taken from a 3-second window just prior to the participants' responses since these active states are thought to carry neural signatures related to the cognitive processes of judgment and decision making on the mathematical expressions.

In this way we can directly compare neural activity in both "decision-making" and "resting" conditions, with the specific objective of the identification of distinct patterns that could validate our hypotheses about the differential brain connectivity in different cognitive states.

For the second hypothesis, **H2**, testing adapts methodologies developed for H1 but focus on epochs particularly related to the correctness of the participant's response. It is hypothesized that EEG signals could differentiate between true and false answers of participants in mathematical decision-making tasks. The epochs were extracted in a similar way with 3 second intervals before the blue and red triggers in the dataset.

3.7 Connectivity Matrices

Connectivity matrices serve as a fundamental tool in neuroscience for visualizing and quantifying the intricate patterns of neural interactions within the brain. These matrices can be derived with the use of the different connectivity analysis techniques mentioned above.

In Figure 2 we can see the Granger causality matrices obtained from one of the subjects' EEG recordings in resting and active cognitive states respectively. Each matrix describes the directional influences between pairs of EEG electrodes over the scalp. The x-axis labels denote the influencing electrodes, and the y-axis labels indicate the influenced electrodes. Each cell in this matrix thus corresponds to a pair of electrodes; the color of each cell reflects the strength of causal influence from the electrode on the x-axis to the electrode on the y-axis.

The color scale, ranging from 0 to 0.18, is provided at the right side of the matrices. The colors change from cool colors like blue, indicating very weak causal influence, to warm colors like yellow, representing very strong influences. This scale will help the eye in assessing the strength and distribution of connectivity across the brain.

4 RESULTS

After segmenting the epochs and preparing the EEG data from all participants, we used the EEGNet neural network to classify resting and active states, in order to validate **H1**. The network was trained with 80% of the data and tested with the remaining 20%.

This resulted in a classification accuracy of about 84%, showing that distinct neural connectivity patterns are present during mathematical decision-making tasks compared to resting states.

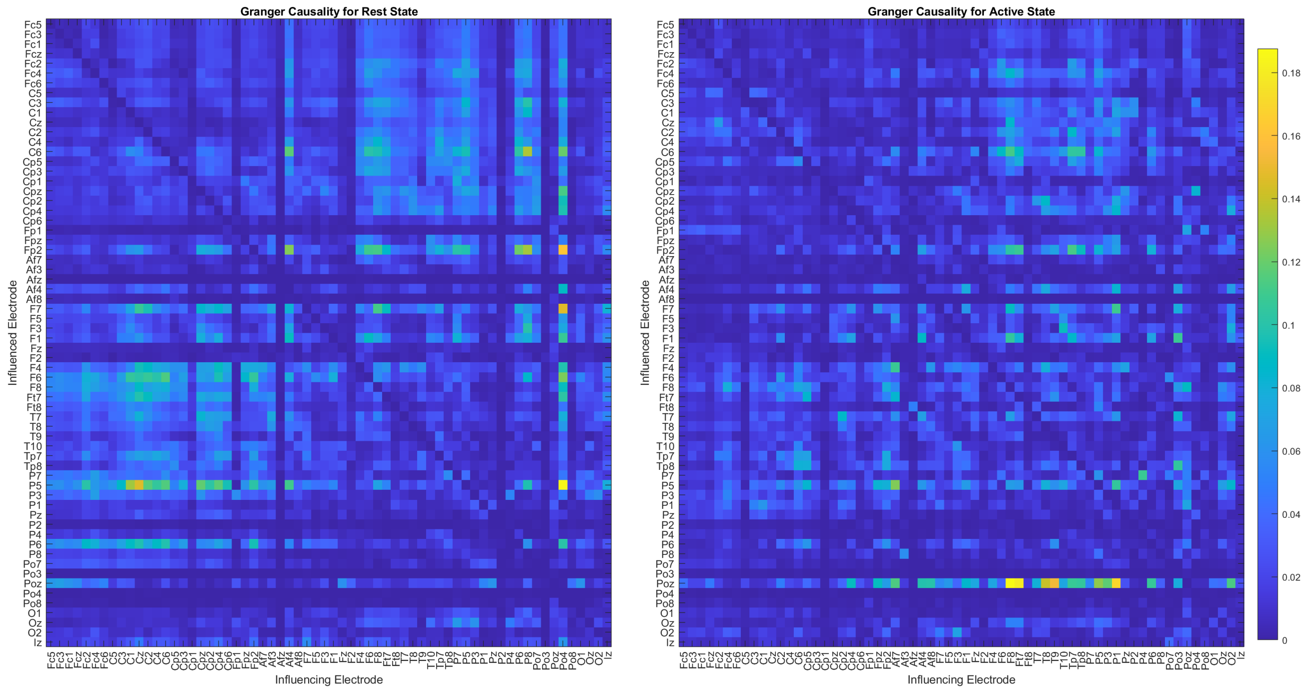


Figure 2: Visualisation of the connectivity matrices of rest and active states.

The findings support our hypothesis that, mathematical thinking causes unique connectivity patterns, differentiable from resting state brain activity. This suggests the promising capability of the EEGNet to discriminate between rest and active states based on the neural data collected around the event-defined epochs.

We also did some testing on the second hypothesis **H2**. Initial tests using the EEGNet neural network for epochs related to correct and incorrect responses resulted in classification accuracies of about 50%, which is clearly insufficient. These results suggest two possible explanations: either the EEG signals do not contain enough distinguishing information, or the applied methods, are not yet optimized to detect subtle differences in brain activity.

Given these results, we will continue to refine our analytical methods and to explore alternative models for a better representation of neural dynamics. Hypothesis **H1** has proven to be a more accessible goal, while hypothesis **H2** presents a greater challenge. Should we confirm **H2**, it could revolutionize how we estimate knowledge and decision-making processes based on neural data.

5 CONCLUSION

The main objective of this research is the development and enhancement of methodologies to analyse EEG signals during cognitive tasks, with a special emphasis on mathematical decision-making. The strategy taken in this research provides a model for future works on more complex cognitive phenomena. It indicates the need for precise acquisition of data, sophisticated preprocessing strategies, and new analytical techniques in an attempt to capture and interpret correctly the activity in the brain.

In summary, this research adds a great deal into the field of development of methodologies that further improve our understanding of cognitive processes and pushes the boundaries of how we can interact with technology using Brain Computer Interfaces (BCI) and analyse neurological conditions. Further evolution of these methods is likely to close the gap between human cognitive functions and machine interpretation, setting the stage for possible future advances that may change neurological healthcare and technology interfacing.

REFERENCES

[1] Mike X Cohen. 2014. *Analyzing Neural Time Series Data: Theory and Practice.* The MIT Press.

[2] S. Dehaene, N. Molko, L. Cohen, and A. J. Wilson. 2004. Arithmetic and the brain. *Current Opinion in Neurobiology* 14, 2 (2004), 218–224.

[3] V. J. Lawhern et al. 2018. EEGNet: a compact convolutional network for EEG-based brain-computer interfaces. *Journal of Neural Engineering* 15, 5 (2018), 056013.

[4] G. A. Light, L. E. Williams, F. Minow, J. Sprock, A. Rissling, R. Sharp, N. R. Swerdlow, and D. L. Braff. 2010. Electroencephalography (EEG) and Event-Related Potentials (ERPs) with Human Participants. *Current Protocols in Neuroscience* 52 (2010), 6.25.1–6.25.24.

[5] S. Mortier et al. 2023. Classification of Targets and Distractors in an Audiovisual Attention Task Based on Electroencephalography. *Sensors* 23, 23 (2023), 9588.

[6] A. K. Seth, A. B. Barrett, and L. Barnett. 2015. Granger Causality Analysis in Neuroscience and Neuroimaging. *Journal of Neuroscience* 35, 8 (2015), 3293–3297.

[7] Z. Šverko, M. Vrankić, S. Vlahinić, and P. Rogelj. 2022. Complex Pearson Correlation Coefficient for EEG Connectivity Analysis. *Sensors (Basel)* 22, 4 (2022), 1477.

Analysis of Verbal Fluency in Slovenian Language in Patients With Schizophrenia

Mila Marinković
mm9136@student.uni-lj.si
University of Ljubljana,
Faculty of Computer and Information Science,
Ljubljana, Slovenia

Polona Rus Prelog
polona.rus@psih-klinika.si
University Psychiatric Hospital Ljubljana,
Ljubljana, Slovenia

Martina Zakšek
martina.zaksek@gmail.com
Splošna bolnišnica Celje,
Celje, Slovenia

Jure Žabkar
jure.zabkar@fri.uni-lj.si
University of Ljubljana,
Faculty of Computer and Information Science,
Ljubljana, Slovenia

ABSTRACT

This study investigates verbal fluency in the Slovenian language among individuals diagnosed with schizophrenia compared to healthy controls. Participants completed a verbal fluency task, which involved producing as many words as possible starting with a specific letter in Slovenian within a set time limit. The analysis included statistical testing and semantic similarity measures using FastText embeddings. Significant differences were found between the groups in terms of the number of correct and total words produced. While semantic similarity showed minimal differences, global optimality divergence revealed notable disparities. These findings highlight the utility of comprehensive analytical approaches in understanding verbal fluency deficits in schizophrenia, emphasizing the need for nuanced methods to capture the complexity of cognitive impairments in this population.

KEYWORDS

verbal fluency, schizophrenia, Slovenian language, semantic analysis, statistical analysis

1 INTRODUCTION

Verbal fluency tests are widely used to assess cognitive function and linguistic abilities in various clinical populations, including individuals with schizophrenia. These tests, which require participants to say words based on specific criteria, provide valuable insights into semantic memory, executive function, and language processing capabilities.

Schizophrenia is a chronic mental disorder characterized by symptoms such as cognitive disorganization, impaired semantic processing, and executive dysfunction. These symptoms often manifest as deficits in verbal fluency, where affected individuals typically produce fewer words and commit more errors, such as repetitions, intrusions, and neologisms. Understanding these verbal fluency deficits is crucial for developing targeted cognitive and linguistic interventions.

Previous studies [1–8] have documented that individuals with schizophrenia exhibit notable impairments in verbal fluency tasks, producing fewer words and making more errors compared to healthy controls. To address this gap, we employed a comprehensive analytical approach combining traditional statistical tests with advanced semantic similarity measures using FastText embeddings.

We hypothesized that while local semantic relationships might not differ significantly between groups, broader semantic coherence and structural organization of speech would be markedly impaired in schizophrenia. By leveraging advanced techniques such as FastText embeddings, we aimed to uncover deeper insights into the semantic characteristics of verbal fluency in schizophrenia.

In this paper, we present a detailed analysis of verbal fluency performance in individuals with schizophrenia compared to healthy controls. We discuss the implications of our findings for understanding the cognitive and linguistic disruptions associated with schizophrenia and propose directions for future research to further explore these impairments.

2 RELATED WORK

The analysis of verbal fluency in individuals with schizophrenia has been extensively researched to understand the cognitive and neural mechanisms underlying the disorder. This study draws upon several key pieces of related work that have influenced our methodology and analytical approaches.

Nour et al. [8] investigated the semantic trajectories in schizophrenia by analyzing verbal fluency tasks using a computational model of word embeddings. Their study highlighted the reduced semantically guided word selection in people with schizophrenia and its correlation with hippocampal disruptions. This approach underscored the importance of semantic distance in understanding cognitive disorganization in schizophrenia and inspired our use of FastText to compute word embeddings and analyze semantic distances between words generated during verbal fluency tasks.

Galaverna et al.[1] conducted a detailed analysis of errors in verbal fluency tasks among individuals with chronic schizophrenia. Their research emphasized the prevalence of perseverative and intrusion errors in verbal fluency tasks, highlighting significant moderators such as the severity of negative symptoms, formal

University of Maribor Press
DOI: https://doi.org/10.18690/um.feri.6.2024.17
ISBN 978-961-286-914-4

thought disorder, and pharmacological variables. This study provided crucial insights into the patterns of errors (intrusions, repetitions, neologisms) in verbal fluency tasks, which are essential for understanding the cognitive deficits associated with schizophrenia.

Ojeda et al. [5] explored the relationship between verbal fluency and other cognitive domains in patients with schizophrenia and healthy controls. Their findings indicated that while healthy controls' verbal fluency was primarily predicted by processing speed, in patients with schizophrenia, it was more closely related to working memory. This study highlights the differing cognitive mechanisms underlying verbal fluency performance in schizophrenia and informed our consideration of different cognitive variables in our analysis.

Grimes et al. [2] examined the stability of verbal fluency abilities in outpatients with schizophrenia over a one-year period. They found that verbal fluency abilities remained stable over time, providing evidence against significant longitudinal decline in these cognitive domains among individuals with chronic schizophrenia. This study's findings on the stability of verbal fluency informed our understanding of the temporal consistency of cognitive impairments in schizophrenia.

Lehtinen et al.[4] presented a systematic administration and analysis approach for verbal fluency tasks, highlighting the importance of detailed scoring guidelines and exploring various underlying cognitive processes. Their method provided strong inter-rater reliability and demonstrated significant effects of education and gender on verbal fluency performance, reinforcing the need for comprehensive analysis beyond total scores. This study's emphasis on clustering, switching, and error analysis informed our analytical approach to understanding the cognitive processes involved in verbal fluency tasks in schizophrenia.

Kosmidis et al. [3] studied verbal fluency in institutionalized patients with schizophrenia, focusing on age-related performance decline. They found that elderly patients exhibited a disproportionate decline in phonemic fluency compared to younger patients, while semantic fluency remained relatively stable. This research highlighted the impact of aging on cognitive strategies like clustering and switching, which are critical for verbal fluency tasks. The findings underscore the importance of considering age and institutionalization duration when analyzing verbal fluency in schizophrenia

Nogueira et al. [7] provided normative data on semantic and phonemic verbal fluency tasks for a European Portuguese population, considering the effects of age, gender, and education. Their study demonstrated that age and education significantly affect verbal fluency performance, while gender has a more variable impact. This research supports the need for demographic adjustments in verbal fluency assessments and informed our methodology in adjusting for these variables in our analysis.

These studies collectively emphasize the multifaceted nature of verbal fluency impairments in schizophrenia, necessitating the use of advanced analytical techniques to capture the underlying cognitive and linguistic disruptions. Our approach, combining traditional statistical tests with semantic similarity measures using FastText embeddings, aims to build on this foundation to provide deeper insights into the verbal fluency deficits in schizophrenia.

3 METHODOLOGY

This section outlines the methodology used in our study to investigate verbal fluency deficits in individuals with schizophrenia compared to healthy controls. We describe the participants, procedures, data collection, and data analysis methods employed to gather and analyze the data.

3.1 Participants

The study involved a total of 126 participants, divided into two groups: 58 individuals diagnosed with schizophrenia and 68 healthy controls. The participants were matched for age and gender to ensure comparability between the groups. All participants were 18 years or older. Exclusion criteria included an inability to speak Slovenian, a history of intellectual disability, organic brain conditions, or substance abuse. For healthy controls, additional criteria included no history of psychiatric disorders or substance abuse. The study was approved by the Medical Ethics Committee of the Republic of Slovenia, and all participants provided written informed consent.

3.2 Procedure

Participants were asked to perform a verbal fluency task in which they had to say as many words as possible that start with the letter "L" in Slovenian within one minute. This task was administered individually in a quiet room to minimize distractions. All responses were recorded for subsequent analysis. Demographic data, including age, gender, education level, marital status, employment status, and hospitalization history, were collected prior to the test to provide a comprehensive overview of the sample population.

3.3 Data Collection

All data were recorded and stored in a secure database. The verbal fluency responses were transcribed and annotated for analysis. Each word was evaluated for its accuracy and categorized as correct, intrusion (an incorrect word not fitting the criteria), repetition (same word used more than once), or neologism (made-up word). The timestamps for each word were also recorded to facilitate temporal analysis. Additionally, FastText embeddings were computed for each word to enable semantic similarity analysis.

Table 1: Demographic Characteristics of the Participants.

Measure	Schizophrenia Patients	Healthy Controls
Total Participants	58	68
Average Age (years)	46.05	46.71
Prevalent Education Level	Primary school	High school
Avg. Elementary School grade-point	3.57	4.72
Avg. Secondary School grade-point	3.46	4.39
Male Distribution	29	35
Female Distribution	29	33

Demographic data, summarized in Table 1, were analyzed to ensure that the groups were comparable in terms of age and gender, so that any differences observed in verbal fluency performance are less likely to be confounded by these factors. Although there were differences in the average education level between the schizophrenia and healthy control groups, we verified that within each education level, there were no significant differences between the groups, ensuring that education level did not confound the verbal fluency comparisons.

3.4 Data Analysis

The collected data underwent various analyses to explore the differences in verbal fluency between individuals with schizophrenia and healthy participants. The following analytical techniques were employed:

(1) Statistical Analysis: A t-test was conducted to compare the total number of words and the number of correct words produced by the two groups, where correct words are those that are neither intrusions, repetitions, nor neologisms. Before using the t-test, we checked that the data was normally distributed. This statistical test provided evidence of differences in verbal fluency performance between individuals with schizophrenia and healthy controls.

(2) Semantic Similarity Analysis: For this study, we used FastText embeddings to capture semantic relationships between words produced during the verbal fluency tasks. FastText is a word embedding technique designed to capture the semantic meaning of words. It breaks words into character-level n-grams, which allows it to capture more contextual information and better handle rare or morphologically complex words. This makes FastText particularly effective for languages like Slovenian, as it can better represent linguistic nuances and provide more meaningful embeddings for semantic similarity analysis. Using these embeddings, we calculated cosine similarity, mean semantic distance, local optimal divergence, and global optimality divergence to capture the semantic relationships and coherence of word sequences.

By combining these analytical approaches, our study aims to provide a comprehensive understanding of the verbal fluency impairments associated with schizophrenia, contributing valuable insights to cognitive functioning.

4 RESULTS

The results of the verbal fluency tests conducted on both individuals with schizophrenia and healthy individuals are summarized in this chapter. Figure 1 and 2 display the most frequently spoken words by each group. Figure 1 presents the top five words spoken by healthy individuals, along with the occurrences of these words among people with schizophrenia. Similarly, Figure 2 illustrates the top five words spoken by individuals with schizophrenia, along with the occurrences of these words in the healthy group.

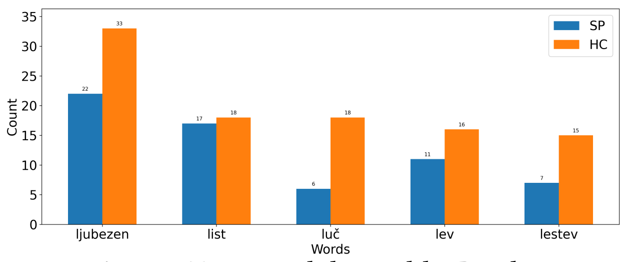


Figure 1: Top 5 Words by Healthy People.

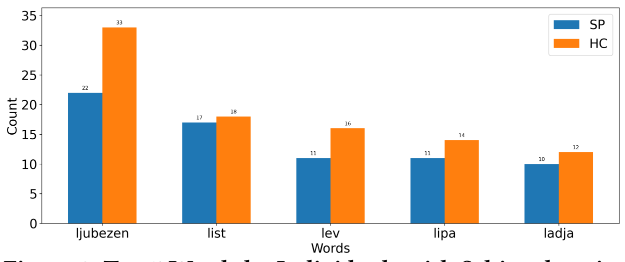


Figure 2: Top 5 Words by Individuals with Schizophrenia.

In addition to the graphical representation of the top words, Table 2 provides a summary of various key metrics from the verbal fluency tests. This includes the total number of different words, the number of unique words, and the counts of intrusion, repetition, and neologism words for both groups.

Table 2: Comparison of Verbal Fluency Test performance between individuals with schizophrenia (SP) and healthy controls (HC). The bottom rows summarize the total number of different words across all users and overlapping words between the groups.

Total number of	SP	HC
Different words	176	247
Unique words	60	131
Intrusion words	44	8
Repetition words	21	8
Neologism words	20	0
Different words across all users	307	
Overlapping words between observed groups	116	

The analysis indicates significant differences in verbal fluency between healthy individuals and those with schizophrenia. The following sections will provide a detailed examination of the data, including statistical analyses and further discussion on the implications of these findings.

4.1 Statistical Analysis

The statistical analysis compared the total number of words and the number of correct words (no intrusion, no repetition, no neologism) produced by participants in both groups.

Table 3: Mean and Standard Deviation (SD) of Words Produced by Healthy Controls (HC) and Individuals with Schizophrenia (SP).

Measure	HC (Mean ± SD)	SP (Mean ± SD)
Correct Words	10.32 + 4.24	6.86 + 4.19
Total Words	10.69 + 4.32	8.52 + 5.12

The results of the t-test, after confirming that the data is normally distributed, are summarized in the table below:

These results indicate significant differences between the groups, with healthy participants producing more correct and total words than participants with schizophrenia.

Table 4: T-test Results for Correct and Total Words by Group.

Measure	t-statistic	p-value
Correct Words	4.77	< 0.001
Total Words	3.04	0.002

4.2 Semantic Similarity Analysis

Using FastText embeddings, we calculated various semantic similarity measures. The average cosine similarity for the group with schizophrenia was 0.2, while for the healthy group, it was 0.19.

In addition to the average cosine similarity, we analyzed other measures which are summarized in Table 5.

Table 5: Semantic Similarity Measures.

Measure	t-statistic	p-value
Mean Semantic Distance	-0.59	0.554
Global Optimality Divergence	2.75	0.007
Local Optimality Divergence	0.29	0.769
Repetitions	-2.26	0.026
Intrusions	-1.83	0.070
Neologisms	-4.47	<0.001

The differences in global optimality divergence and occurrences of intrusions, repetitions, and neologisms were significant, highlighting disruptions in the overall semantic coherence and increased errors in individuals with schizophrenia.

5 DISCUSSION

The results of this study highlight significant differences in verbal fluency performance between individuals with schizophrenia and healthy controls. Healthy participants produced a higher number of correct words and exhibited more coherent and interconnected semantic structures compared to individuals with schizophrenia. These findings are consistent with established research on the cognitive impairments linked to schizophrenia, particularly in the domain of verbal fluency.

In addition to the differences in correct word production, individuals with schizophrenia also showed a significantly higher frequency of errors, including repetitions, neologisms, and intrusions. These errors are characteristic of the cognitive disorganization associated with schizophrenia and reflect the impaired executive function and semantic processing commonly observed in the disorder. The increased number of neologisms and intrusions further highlights the semantic and linguistic disruptions that differentiate individuals with schizophrenia from healthy controls.

Furthermore, although the use of FastText embeddings was a key aspect of this study, the method was not sensitive enough to capture local semantic disruptions. Measures such as cosine similarity and mean semantic distance, focusing on local semantic relationships, failed to highlight significant differences between the groups. This outcome suggests that while local word relationships may remain relatively preserved in individuals with schizophrenia, or that FastText embeddings may not effectively capture subtle local disruptions, the broader semantic coherence was impacted. This was evidenced by the significant differences in global optimality divergence, demonstrating a marked reduction in overall word sequence coherence among individuals with schizophrenia.

5.1 Conclusions

In conclusion, our study highlights the significant cognitive and linguistic impairments in individuals with schizophrenia, particularly in verbal fluency performance. We matched participants on age and gender to ensure comparability between the groups. While education is known to influence cognitive abilities, our analysis confirmed that within each educational level, there were no significant differences between the two groups. However, a larger sample size is needed to increase the power of our statistical analyses and allow for better generalizability and more in-depth exploration of all demographic variables.

Additionally, while FastText embeddings provided useful insights into semantic coherence, they were not sensitive enough to capture more subtle cognitive impairments. Future studies should explore alternative methods to provide a more comprehensive understanding of verbal fluency deficits in schizophrenia, contributing to improved diagnostic and therapeutic strategies.

REFERENCES

[1] Flavia Galaverna, AdriÃ¡n M. Bueno, Carlos A. Morra, MarÃa Roca, and Teresa Torralva. 2016. Analysis of errors in verbal fluency tasks in patients with chronic schizophrenia. *The European Journal of Psychiatry* 30 (12 2016), 305 – 320.

[2] Kyrsten Grimes, George Foussias, Gary Remington, Kathryn Kalahani-Bargis, and Konstantine Zakzanis. 2020. Stability of Verbal Fluency in Outpatients with Schizophrenia. *Psychiatry Research* 295 (10 2020), 113528. https://doi.org/10.1016/j.psychres.2020.113528

[3] Mary Kosmidis, Vassilis Bozikas, Christina Vlahou, Grigoris Kiosseoglou, George Giaglis, and Athanasios Karavatos. 2005. Verbal fluency in institutionalized patients with schizophrenia: Age-related performance decline. *Psychiatry research* 134 (05 2005), 233–40. https://doi.org/10.1016/j.psychres.2005.02.003

[4] Nana Lehtinen, Ida Luotonen, and Anna Kautto. 2021. Systematic administration and analysis of verbal fluency tasks: Preliminary evidence for reliable exploration of processes underlying task performance. *Applied Neuropsychology: Adult* 30 (09 2021), 1–13. https://doi.org/10.1080/23279095.2021.1973471

[5] Ojeda Natalia, Pedro Sanchez, Javier Peña, Edorta Elizagárate, Ana Yoller, Juan Larumbe, Miguel Gutiérrez-Fraile, Leonardo Casais, and Jesús Ezcurra. 2010. Verbal Fluency in Schizophrenia Does Cognitive Performance Reflect the Same Underlying Mechanisms in Patients and Healthy Controls? *The Journal of nervous and mental disease* 198 (04 2010), 286–91. https://doi.org/10.1097/NMD.0b013e3181d61748

[6] Angel Nevado, David Del Río, María Teresa Martín-Aragoneses, José M. Prados, and Ramón López-Higes. 2021. Preserved semantic categorical organization in mild cognitive impairment: A network analysis of verbal fluency. *Neuropsychologia* 157 (2021), 107875. https://doi.org/10.1016/j.neuropsychologia.2021.107875

[7] Dália Nogueira, Elizabeth Reis, and Ana Vieira. 2016. Verbal Fluency Tasks: Effects of Age, Gender, and Education. *Folia Phoniatrica et Logopaedica* 68 (12 2016), 124–133. https://doi.org/10.1159/000450640

[8] Matthew M. Nour, Daniel C. McNamee, Yunzhe Liu, and Raymond J. Dolan. 2023. Trajectories through semantic spaces in schizophrenia and the relationship to ripple bursts. *Proceedings of the National Academy of Sciences* 120, 42 (2023), e2305290120. https://doi.org/10.1073/pnas.2305290120

Index of Authors

Proceedings of the 10th Student Computing Research Symposium (SCORES'24)

Niko Lukač (ed.)
niko.lukac@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science
Maribor, Slovenia

Iztok Fister (ed.)
iztok.fister@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science
Maribor, Slovenia

Štefan Kohek (ed.)
stefan.kohek@um.si
University of Maribor,
Faculty of Electrical
Engineering and Computer Science
Maribor, Slovenia

ABSTRACT

The 2024 Student Computing Research Symposium (SCORES 2024), organized by the Faculty of Electrical Engineering and Computer Science at the University of Maribor (UM FERI) in collaboration with the University of Ljubljana and the University of Primorska, showcases innovative student research in computer science. This year's symposium highlights advancements in fields such as artificial intelligence, data science, machine learning algorithms, computational problem-solving, and healthcare data analysis. The primary goal of SCORES 2024 is to provide a platform for students to present their research, fostering early engagement in academic inquiry. Beyond research presentations, the symposium seeks to create an environment where students from different institutions can meet, exchange ideas, and build lasting connections. It aims to cultivate friendships and future research collaborations among emerging scholars. Additionally, the conference offers an opportunity for students to interact with senior researchers from institutions beyond their own, promoting mentorship and broader academic networking.

KEYWORDS

student conference, computer and information science, artificial intelligence, data science, data mining

University of Maribor Press

ISBN 978-961-286-914-4
DOI: https://doi.org/10.18690/um.feri.6.2024

University of Maribor Press