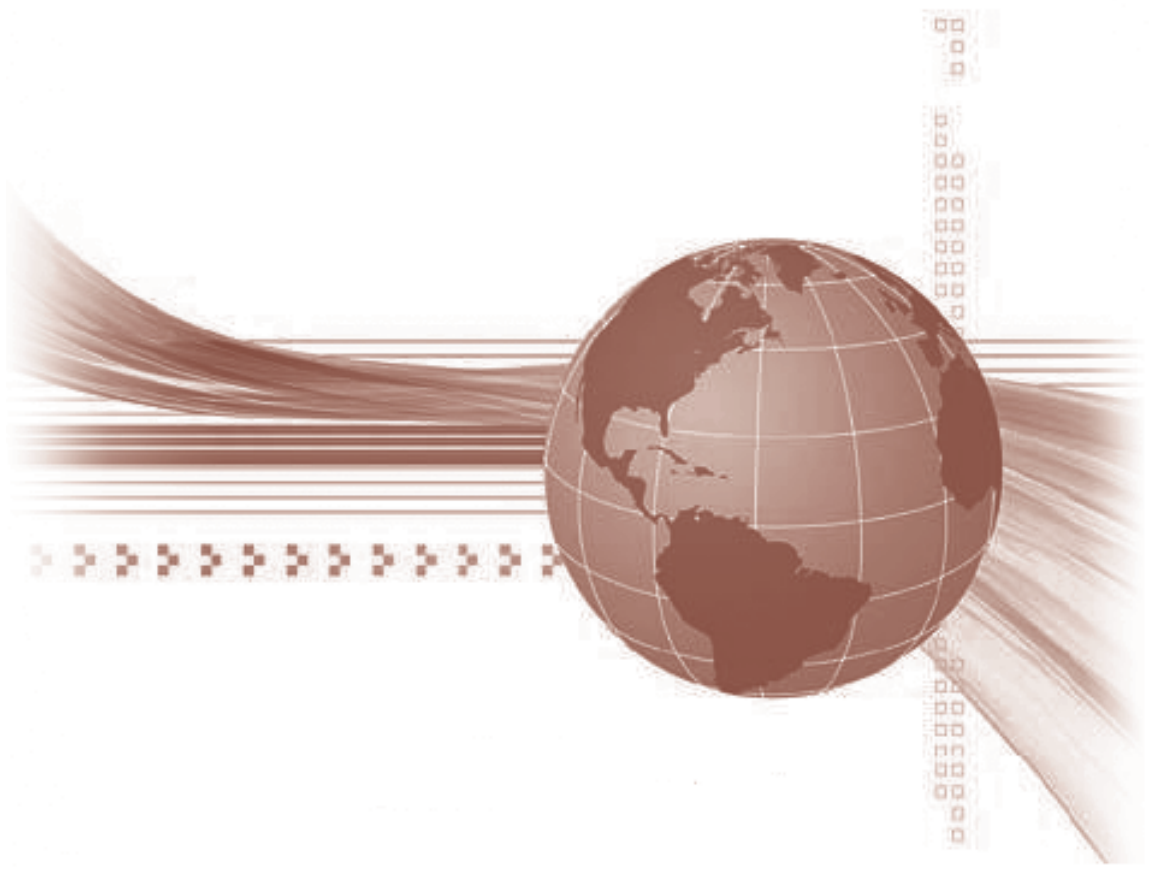
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI

UNIVERSITATEA BABEŞ-BOLYAI
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
BABEŞ-BOLYAI UNIVERSITÄT
BABEŞ-BOLYAI UNIVERSITY
TRADITIO ET EXCELLENTIA



INFORMATICA

1/2024

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 1/2024

January - June

ISSN (online): 2065-9601; ISSN-L: 2065-9601

Published by Babeș-Bolyai University

EDITORIAL BOARD

YEAR **Volume 69 (LXIX) 2024**
MONTH **JUNE**
ISSUE **1**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

1

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.01

MATCHING APICTORIAL PUZZLE PIECES USING DEEP LEARNING

RALUCA-DIANA CHIŞ

Abstract. Finding matches between puzzle pieces is a difficult problem relevant to applications that involve restoring broken objects. The main difficulty comes from the similarity of the puzzle pieces and the very small difference between a pair of pieces that almost match and one that does. The proposed solution is based on deep learning models and has two steps: firstly, the pieces are segmented from images with a U-Net model; then, matches are found with a Siamese Neural Network. To reach our goal, we created our own dataset, containing 462 images and just as many masks. With these masks, we built 3318 pairs of images, half of them representing pieces that fit together and half that do not. Our most relevant result is estimating correctly for 290 out of 332 pairs whether they match.

1. Introduction

The aim of this paper is to present the design and implementation of a solution for the complicated problem of matching *apictorial* puzzle pieces. The term *apictorial* is used in this work with the meaning that the pairs of puzzle pieces are found only by looking at the shape of the pieces and neglecting the model on top of them, as we aim for a robust solution, that could suit pieces with any type of patterns. This problem is a complicated one, due to the fact that the puzzle pieces are very resembling and it is hard to differentiate between them. This requires a very well-defined puzzle piece edge extraction, that does not compromise their shapes at all, according to [5]. Moreover, the search space is considerably large, considering that for each side of a piece, the program has to compute as many comparisons as images in the dataset. Normally, the number of comparisons should have been multiplied by 4 (the

Received by the editors: 23 October 2023.

2010 *Mathematics Subject Classification.* 68T45, 68U10.

1998 *CR Categories and Descriptors.* I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding – *Shape*; I.4.6 [**Image Processing and Computer Vision**]: Segmentation – *Edge and feature detection*.

Key words and phrases. U-Net, Siamese architecture, Edge-matching, Puzzle Pieces.

number of possible rotations of a piece), but for this research it was considered that all the pieces are oriented as in their final position from the puzzle.

This research topic revolves around finding pairs of puzzle pieces, that complement each other perfectly. This has great applications in the reassembly process of fragmented objects, which is a very time-consuming task for humans and of high interest in archaeology, according to [17] and [15]. The main difference between solving a puzzle and restoring an object is that the former can be treated as a two-dimensional problem. Willis et al. [20] associate the reconstruction of ancient artefacts with solving a *real-world geometric puzzle*. In the same paper, the authors present puzzle piece matching as a more regular version of the problem of artefact piece matching, since puzzles have a more predictable shape and were not exposed to erosion prior to the reconstruction. Besides this, it is mentioned that the edges of jigsaw pieces are more simple to separate, because the corners of each side are easily identifiable.

The main contribution of the current study refers to the creation of an automated pipeline designed for pairwise matching of puzzle pieces. The process is done in two steps. Initially, images of puzzle pieces are segmented to extract the shape of each piece. For training this model, an original dataset was created, consisting of images of real puzzle pieces and corresponding masks. The second step of the pipeline involves the creation of a deep learning model for matching pairs of puzzle pieces according to their shape, which to our knowledge has never been tried before. A Siamese neural network performs edge matching for all segmented pieces and the matching score is computed for a series of generated pairs of puzzle pieces. Our most relevant result gives an accuracy of 87.34% for the edge-matching task on the test set, which means that for the majority of the puzzle pairs, the model managed to correctly predict if they match or not.

The current paper is structured in five sections as follows. The second section details the existing literature in the field. Section 3 presents the development of the segmentation and the edge-matching model. The fourth section describes the environment in which the experiments were carried out, what results we achieved and how they compare to the outcomes of other papers. Finally, conclusions are drawn and possibilities for improvement in the future are mentioned.

2. Related work

In this section we will review the most recent research papers on techniques for finding matches between different pieces, using images of them. Since we have established that our aim is not to solve a puzzle, we will only briefly address the parts of the papers related to generating puzzle solutions. The

methods presented in what follows expose both two and three-dimensional approaches and are focused on image processing techniques. Some of them are not completely automatic and require a specialist for constant feedback. One can easily notice a common pattern for all papers exposed here. They all follow a similar structure: photos or scans of the pieces (puzzle pieces or other types) are taken, followed by instance segmentation or contour curve extraction, and using the resulting features, edge matching between pieces is performed. The results of the papers presented below will be presented in Section 4.5 and they will be compared to the ones of the current research.

As far as the data used in the most relevant papers on edge-matching are concerned, they consist of images of torn documents [2], patches of papyrus or ostraca fragments images [18] and [14], images of unconventional puzzle pieces [6] and [7], 3D scans of broken objects [1], and synthetically generated three-dimensional objects [5]. Regarding the segmentation task, the authors propose methods that differ slightly from each other. Grim et al. [5] use Bezier curves with selected control points, while in [6] images were segmented using active contours [9] and smoothed using a naïve spline algorithm. In [7] the edge curvature is computed with integral invariants and Alagrami et al. [1] proposed a graph-based method for extracting the breaking curves of an object, with which the sides of the object were identified.

For the task of matching the edges, the approach proposed in [8] and in [1] uses the Iterative Closest Points (ICP) algorithm. Authors of [6] and [5] divided each boundary curve into a set of bivertex axes with the extended Euclidean signature method. All sets of axes were compared to each other and a confidence score on their matching was computed. In [2] the features lists for each corner detected were joined with the Minkowski Sum.

In a recent survey addressing the solving of Jigsaw puzzles [13], emphasis is placed on the utilization of deep learning techniques exclusively for methodologies employing square or rectangular patches extracted from images. This particular perspective proves valuable for discerning the relationship between two fragments belonging to the same object [18] or predicting the permutation of image patches to compose a comprehensive image [11]. However, our objective diverges from this approach. Our model seeks puzzle pieces that complement each other in shape rather than similarity, as the synergy of two matching puzzle pieces lies in their complementarity. To our knowledge, there are no previous experiments applying a Siamese network for comparing the shapes of two puzzle pieces.

We focus specifically on the shape of the pieces in the images. In the context of patches, continuity in terms of the patterns in the image is sought. This represents the main difference between our edge-matching model and the

Siamese neural network from [14]. Moreover, in the architecture proposed by Ostertag et al., the two inputted images are passed through four convolutional blocks and then 4 patches with a width of 10 pixels are extracted from the feature map of each branch. Each patch is subtracted from its opposite belonging to the second branch, the results are concatenated and the output consists of a 5 probabilities linear vector, each of them representing the similarity of each side. The disadvantage of this method that we overcome with our proposed architecture is that by creating those 10-pixel wide patches the overall context is lost and important chunks might be missing for some puzzle pieces.

3. Methodology

The manner in which we have approached the problem of matching puzzle pieces from images consists of two phases: the development of a model for segmenting puzzle pieces from images and the development of a deep-learning model for finding matching pairs of pieces.

3.1. **Segmentation.** The model for image segmentation has a very large influence on the success of the attempted approach, as the predictions generated by it will be the network input for edge-matching. The higher the quality of the segmentations, the better the piece-matching results will be. In particular, the edge of the pieces is preferable to be as well defined as possible, but the difficulty arises when the shadow of a piece appears in the image, which misleads the model about its edge.

We have decided to use for this task the model called U-NET, due to its effectiveness in capturing fine-grained details. The authors of this model [19] prove the usefulness and the very good performance of their network architecture, by applying it to three segmentation tasks: segmentation of neuronal structures in electron microscopic recordings, segmentation of cells on a polyacrylamide substrate recorded by phase contrast microscopy, and HeLa cells on a flat glass recorded by differential interference contrast microscopy. The results of the U-NET outperform the state-of-the-art (up until 2015) on these tasks.

This U-NET architecture is based on using fewer images than usual for training neural networks while exploiting the given images more efficiently. The authors use data augmentation techniques to achieve this performance, by elastically deforming images. In this way, the model can learn invariance to different types of deformations, which in their case of biomedical segmentation was very useful, because deformation in tissue is very common. For our case, data augmentation is also useful because images of puzzle pieces can vary greatly in terms of object position, brightness, colour, and so on.

The architecture of the network comprises a downsampling of the feature map, followed by an upsampling of these features. The model encodes an image, extracts features using multiple convolutional layers. Then, the decoder upsamples the features using the transpose convolution and concatenates them using a process, called skip connection. Through this process, the model skips some of the network layers and feeds the output of one layer as the input to the next layers. The output of the network is a segmentation mask. The pixels of this mask have values ranging from 0 to 1 and for increasing the chances of the edge-matching model, a threshold of value 0.4 was applied, meaning that all pixels under this value were converted to 0 and 1, respectively. The value of the threshold was set experimentally.

3.2. **Edge matching.** What comes after segmenting the puzzle pieces is finding pairs that match, depending on the shape of their sides. For this matter, we chose a deep-learning approach, following a Siamese architecture. The idea of this architecture dates back to 1993 [3], when a group of researchers proposed the use of two identical neural networks merged at the output. Their goal was to check if two signatures written on a touch-sensitive pad are identical. They obtained impressive results on test data, the model being able to detect a real signature in 95.5% of the cases and forgeries in 80%.

Siamese networks are used for measuring the similarity or the dissimilarity between two images, by comparing the feature vectors outputted by the twin networks. Our architecture is presented in Fig. 1. The network comprises two identical branches with shared weights, meaning that the parameters (weights and biases) of these branches are the same. Each branch is made out of three convolutional blocks. Each of them is composed of 2 convolutional layers with 64, 128, and 256 filters and kernel sizes of 11x11, 5x5, and 3x3 respectively, one batch normalization layer, one ReLu activation layer, and lastly, a max-pooling layer, which reduces the image size by 2.

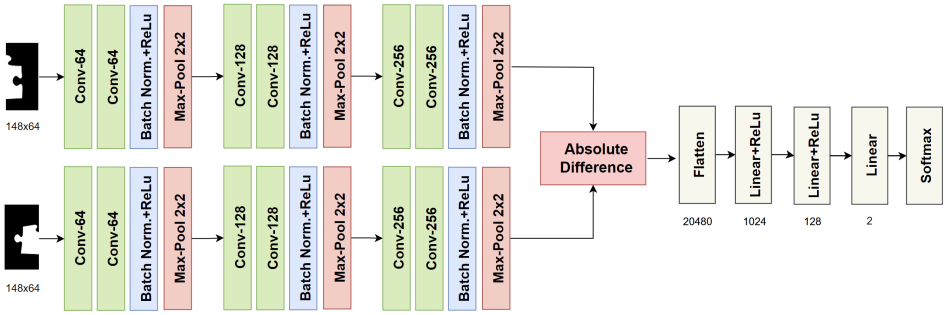


FIGURE 1. Edge-matching network architecture

At this point in the network, most of the Siamese architectures propose computing a loss between the two feature vectors, usually a contrastive loss, as in [4]. We tried out this method too, but the results were disappointing. The best model we obtained with this approach was able to make correct estimations only in half of the cases. Moreover, the Euclidian distance between one image and 50 others from the test set, was the same for 25 of them and equal to 0.26116937. This value for the distance would mean that the images are compatible, but when we reviewed them, we discovered both suitable images and completely not suitable ones. These facts led us to the conclusion that this path cannot be successful, since the model was focusing on finding if the two input images are similar or not and this was not what we specifically needed.

Inspired by [18] and [14], we decided to subtract the two feature vectors element by element. The result was flattened and we obtained a one-dimensional feature vector of size 20408. As visible in Fig. 1, the vector is afterwards passed through a couple of linear transformations. After the first linear transformation, we tried adding a dropout layer, with a probability of 0.5, to prevent the model from overfitting. In some situations, this layer helped, but not always, as we will discuss in the next section. Lastly, the Softmax layer computes the probability of the two images forming a match or not. Further on, we used the Cross-Entropy loss from PyTorch [16], to compute the distance between the probabilities outputted by the model and the true ones.

4. Experimental results

Based on the models presented above, experiments were conducted for both tasks. In the following, the most relevant experiments will be presented, discussed and compared, together with the dataset and the methodology used in their evaluation. The code used for experiments is available here[1].

4.1. **Data.** For this study, a dataset was created taking into account the use of various puzzle piece sizes, the need for having a minimum of several hundred images for training, testing and validation, and a reduced difficulty for the pre-processing and mask generation step. The dataset is available here[2].

The first step in creating the dataset was to photograph seven puzzles with different numbers of pieces and different piece sizes, ranging from 1.6x1.8 cm to 4.9x4.1 cm. All the pieces had the classic shape of puzzle pieces, namely four sides, each with an indent or an outdent. The exception was the edge pieces, which had one or two straight sides. The pieces were placed face down, on a black background, to have a bigger contrast between the piece

[1]https://github.com/RalucaChis/PiecePerfect

[2]https://www.kaggle.com/datasets/ralucachiss/pieceperfect-dataset/

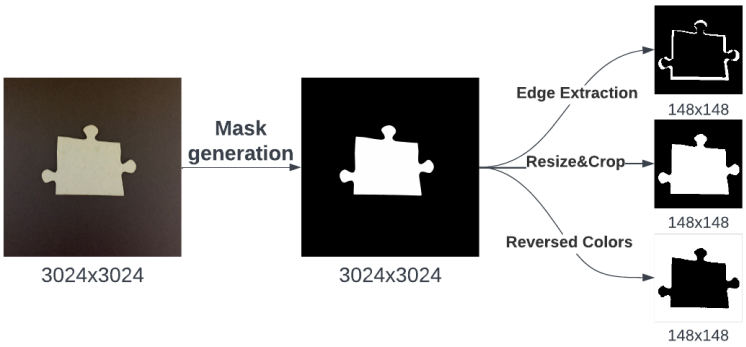


FIGURE 2. Puzzle piece images preprocessing pipeline: mask generation, mask resize and crop, mask with reversed colours and an only-edge mask

and the background, which helped enormously the binarization process. The photographs were taken with an iPhone, version XS, with a focal length of 26 mm, aperture ratio of f/1.8, and image size of 3024x3024. In total, we took 462 pictures, belonging to 7 different puzzles. The quality of the images was good enough for our task, but only very late in the research a possible problem in the data collection occurred to us. Since we did not use a fixed setup for taking the pictures (a camera setup [8] or a photocopier [7]), the distances between the camera and the pieces may be different by a few centimetres, thus affecting the ratio between the size of the piece and the image.

To generate the masks, images were first opened in grayscale mode and then converted into binary images with thresholding, separating foreground (objects of interest) and background. Experiments revealed that a threshold value of 140 (with pixels ranging from 0 to 255) was optimal.

The generated masks have white pixels for puzzle pieces (value 1) and black pixels (value 0) for the background. Since the masks had some black gaps inside the pieces, we made use of the "floodFill" method from OpenCV to fill the gaps. FloodFill is a technique for changing the colour or value of a connected region of pixels in an image, starting from a seeding point. The filling process began from the left top corner of the image, by converting to white the whole background. The colours of the retrieved image were inverted and then we applied a logical OR operation on the resulting image and the original one. The obtained masks have very good quality and correspond completely to the shape of the pieces (Fig. 2).

The obtained dataset consists of 462 pictures of puzzle pieces, suitable for training a segmentation model. However, the model was limited to identifying

puzzle pieces without an image and a simple background. Therefore, we generated five additional variants for each image, randomly selecting backgrounds from a set of 13 and foregrounds from a set of 19. This augmentation resulted in a dataset of 2772 images (original 462 + 5*462 augmented), significantly improving segmentation performance on test images.

The initial images have a size of 3024x3024 pixels. Since working with very large images is difficult, we chose to resize the images to 256x256 pixels. After performing several experiments with unsatisfactory numeric results on the edge-matching model, we found that reducing the background area by cropping the images improved model performance. For this, we cropped the images, keeping 5 pixels on each side between the piece and the edge of the image. To maintain uniform mask sizes without losing quality through resizing, we added black padding to each side, resulting in images standardized to the largest size (148x148 pixels). For future reference, we will call this set *DS1*.

Furthermore, we thought it might help if, when analyzing a pair of masks, the second image had the colours reversed. In this way, the side of the first piece would be white, and the outside of the pair side of the second image would still be white, and thus the model would more easily identify the compatibility of the two pieces. So we generated a set of images with the colours reversed (*DS2*). Also, in order to simplify the work of the model, we generated a set of images containing only the edges of the pieces (*DS3*). To obtain them, we resized the masks to 135x135 pixels, added black padding to the images on all sides until we obtained pictures of 148x148 pixels with the piece in the centre, and then applied a logical XOR operation between the initial mask and the transformed one. Fig. 2 contains an example mask from each of the three resulting sets.

For efficient image management, we scanned each puzzle solved and numbered each piece. The numbers on each puzzle were manually translated into a matrix, which we used to create the pairs of pieces. Thus, for each piece we considered the neighbouring pieces in the 4 directions (top, bottom, left, right) as compatible, and the neighbours on the diagonal (top-left, top-right, bottom-left, bottom-right) as incompatible.

We started from the premise that all pieces are oriented in the final position in the puzzle, so between any two puzzle pieces there are 4 possible matches, plus the case where they do not match at all. An example of the 4 matching cases can be seen in Fig. 3.

The compatibility of two pieces is determined by how well two of their sides complement each other. Thus, to encourage the model to pay more attention to the edges of the pieces, we chose to train the model with the images cut in half for all experiments. Following the pre-processing procedures applied to

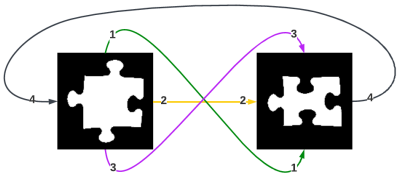


FIGURE 3. Matching Possibilities: 1&3 are top-down matches (the top of a piece matches the bottom of the other and vice-versa) and 2&4 are left-right matches.

the masks, in each of them the piece is in the centre of the image, so there is no risk that a half of the image does not contain any part of the piece. Prior to cropping, the images were rotated 90°, 180° or 270°, according to their match-case, so that the right side of the first image (first from left to right) matches the left side of the second image. Since we rotated and cropped the images, the task of differentiating the four types of match is almost impossible, so we chose to classify the pairs as binary, match or non-match. With this strategy, we obtained a total of 3318 entries, of which 1618 represent matches and 1700 non-matches.

4.2. **Evaluation methods.** In order to evaluate the segmentation model, we split the image dataset into three sets: for training (70%), validation (20%) and testing (10%). One of the evaluation metrics used is L1 loss, which is also known as the absolute error loss. It is defined as the absolute difference between the predicted and actual values, as follows: $L1 = |y_{actual} - y_{predicted}|$. The mean absolute error (MAE), which is computed as the mean of L1 values, will be used for evaluating the mean value of pixels, that have not been correctly predicted (with values from 0 to 1, where 1 states that all pixels have been wrongly classified and 0 the opposite).

Moreover, the Dice coefficient was used to estimate how well the mask overlaps the input image. The Dice coefficient is computed as twice the intersection of the predicted and ground truth masks, divided by the sum of their areas: Dice coefficient = (2 * Intersection) / (Sum of areas). It takes values in the range of 0 and 1, where 1 indicates a perfect overlap, while 0 indicates no overlap.

For the edge-matching task, the main criterion applied for evaluating the model is the accuracy of the predictions. The pairs discussed in Section 4.1 were divided into three sets: training (70% of them), validation (20%) and testing (10%). The accuracy of this task is simply computed by dividing the number of correctly estimated pairs from the test set, by the total number of pairs. For an improved perspective, we also computed the F1 score. This evaluation method is computed as a harmonic mean of precision and recall,

with the following formula: $F1 = 2 * (precision * recall)/(precision + recall)$. The F1 score ranges from 0 to 1, where 1 represents the best possible value and 0 is the worst.

In addition to these classical evaluation methods, we considered it important to analyze the performance of the model in the context of finding the ideal pair of a part. We chose from the test set the pieces that had a pair and analysed the fit between one piece and all the others. If the true match is among the top 10 pieces with a very high probability of being a match, we will consider it a success. For future reference, we will call this evaluation *top_10*.

4.3. **Segmentation results.** The segmentation part of the problem did not impose as many difficulties as the edge-matching task, so a small number of experiments was adequate. After a few tries, we noticed that the model behaved best when using the Adam optimizer and a batch size of 16. The difference between using a learning rate of 0.003 and 0.0003 was minimal. After applying the second value, the mean absolute error (MAE) obtained was 0.003490, and the Dice coefficient value on average was 0.9751, confirming that the model learns the proposed task very well.

The results of both experiments are visible in Table 1. A mean value as high as 0.97 out of 1 for the Dice coefficient suggests that the masks generated by the segmentation model overlap almost perfectly with the ground truth. This result is further supported by the very low value achieved for the MAE score, 0.0034. The progress of the L1 loss during training supports the good performance of the model (Fig. 4). The values range from 0.0896 to 0.0092 for training and from 0.5906 to 0.0123 for validation. The trainings have been conducted for 30 epochs, as we noticed that the loss stabilizes and fluctuates very little after the first 10 epochs.

4.4. **Edge matching results.** The edge-matching task was difficult, so several experiments were needed to come up with at least a satisfactory solution. Table 2 exposes the numerical results and we are going to detail each of the experiments in what follows. All experiments were done with a batch size of value 64, a learning rate of 0.001 and with the Adam optimizer [10]. These specifications have been established experimentally. The batches are computed before training and images are randomized. For all experiments, the images were halved. As we noticed that accuracy and loss stabilize after 30 epochs, we did all our experiments with this number of epochs. All the experiments described in the next chapter are done on Google Colaboratory, using GPU Nvidia T4, with 16 GB.

Inspired by the architecture shown in [18], where the authors present a model with an accuracy of 79% on matching papyrus fragments, we tried

TABLE 1. Experimental results for image segmentation with U-Net.

Dataset	**Batch Size**	**LR**	**Optimizer**	**MAE**	**Dice coeff.**
DS1	16	0.003	Adam	0.004968	0.9649
DS1	16	0.0003	Adam	**0.003490**	**0.9751**

TABLE 2. Experimental results for edge-matching (columns *Img1* and *Img2* state the dataset used as input for the first and second branches respectively; *Drp* stands for *Dropout Layer*)

Id	**Img1**	**Img2**	**Drp.**	**LR**	**Opt.**	**Acc. %**	**F1**	***top_10* %**
1	*DS1*	*DS1*	NO	0.001	Adam	51.2048	0	0
2	*DS1*	*DS1*	YES	0.001	Adam	83.4337	82.9721	10.17
3	*DS1*	*DS1*	NO	0.001	Adam	84.6385	84.9557	12.34
4	*DS1*	*DS2*	YES	0.001	Adam	80.4216	79.4952	**17.36**
5	*DS3*	*DS3*	YES	0.001	Adam	**87.3493**	**85.6164**	8.982

flattening the feature vectors outputted by the two identical branches and only afterwards perform the absolute difference between the two (experiment with id *1*). The model performed poorly, predicting that almost all pairs are not matching, which lead to very low results for all three performance measures. The behaviour of the model is unexpected and worth investigating in future.

The second experiment (id *2*) is based on the architecture described in Section 3.2, having as input the preprocessed puzzle pieces images, rotated and cut in half, as described in Section 4.1. Although this approach obtained very good results on all evaluation methods, it was outperformed with more than one percent on each metric by the experiment in which we removed the dropout layer from the neural network (experiment with id *3*). The reason for this lies in the fact that the model used had a rather simple design, which was not prone to overfitting, so the dropout layer could be omitted. This was not the case for the last two experiments. The fourth experiment performed better when we used the dropout layer and the same for the fifth. This happened thanks to the ability of the dropout layer to reduce the sensitivity of a model, which helped the model focus on the most salient features.

The experiment with id *4* was very similar to the one with id *2*, with the exception that for experiment *4* the second inputted image had its colours inverted (white became black and vice-versa). While the accuracy and the F1 score value were considerably lower than in the approach with id *3*, the model managed to find the perfect match of a piece among the top 10 most suitable piece images in 29 out of 167 cases. On the other side, for the experiment *5*

we used as input the images containing only the edge of the puzzle pieces. In this case, the model obtained the best accuracy, 87.3493%, and the highest F1 score, 85.6164. However, it performed the worst on the *top_10* evaluation, with only 8.9820% of matches identified. This discrepancy can be explained by the fact that the model did a little overfitting, and when evaluating *top_10*, it obtained unrealistically high probabilities for many parts (in one of the cases the probabilities were 1.4511e-07 non-match and 1.0000e+00 match) and it was difficult to find the true pair.

Regarding the evolution of the cross-entropy loss at each epoch for the training and validation sets, the results stabilize somewhere after 30 epochs (Fig. 4). They fluctuate between 0.693 and 0.4101 for training and between 0.6933 and 0.4367 for the validation process. As expected after discussing the outcomes of the experiment *5*, the loss in this case was the lowest, for both training and validation, while the approach with id *1* had the highest loss values.

4.5. **Discussion.** The results presented in the previous section can be summed up to achieving an MAE score as low as 0.0034 for the segmentation task and accuracy as high as 87.34% for the edge-matching model. Even though these are very good results from our perspective, some issues remain, which will be discussed at length below.

Regarding the segmentation task, as mentioned, good results were achieved on the test set. For situations where only one piece appears in an image, and the contrast between the piece and the background is high enough, the model manages to do the segmentation correctly. In a more complicated situation, however, it fails to estimate the mask accurately enough so that the mask can be used for further edge-matching. There is certainly room for improvement in the model, especially on the part of the diversity of the data used as input, but from the perspective of the purpose of this paper, the current capability of the mask estimation model is satisfactory.

As for the results obtained for the evaluation *top_10*, an example of what they look like can be seen in Fig. 5. Despite the actual pair not being within the top 10, all featured images depict pieces remarkably similar to the sought pair, each possessing a framing-compatible hole. Distinguishing the true pair is challenging, even for a human, due to minimal differences between pieces. Notably, with a smaller image set, finding the actual match becomes more probable. In our experiments, the task was challenging, with a test set comprising 332 pairs. Another attribute that influences the model results for edge-matching is the ratio between the piece and the image sizes. Some images were taken at a shorter distance between the camera and the piece, and others at a longer distance. Thus, the model may learn to classify as non-match two

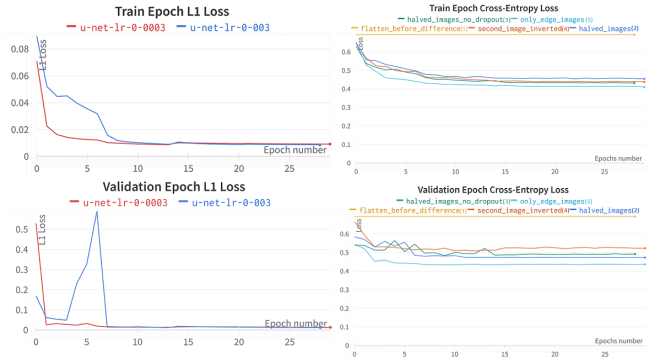


FIGURE 4. Epoch loss for training and validation on segmentation (left column) and on edge-matching (right column)

pieces that in reality match, but the images do not overlap perfectly because of the different size ratios. To investigate the extent of this issue, we constructed a confusion matrix (Fig. 6) for the model trained using the *DS3* dataset (experiment id *5*). False negatives account for 11.14% of the total predictions, but only 11.9% of the incorrect predictions are false positives, revealing that the model is more prone to generating false negatives than false positives.

For the specific situation of a puzzle, where the solution is guaranteed to be unique, the process can be continued with the assembly of the puzzle. The compatibility of a piece with the top 10 possible pairs is very high, with small differences between them, which is why, for the purpose of automatic puzzle assembly, we consider it appropriate to find tuples of 4 pieces that match two by two, and then match them all together. For what we set out to do, this step is not necessary.

Comparing the results presented in this paper with those obtained by other researchers is difficult. One reason is that in some papers the authors used

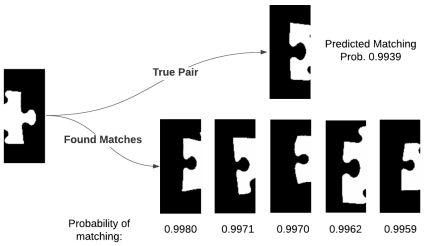


FIGURE 5. Example of very probable matches found for a piece, yet none of them is the true pair.

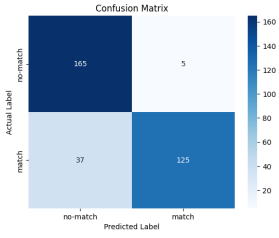


FIGURE 6. Confusion matrix for the edge-matching model trained with *DS3*

image patches ([18], [14]), as discussed in Section 2. In [18], the highest accuracy obtained for predicting if two patches belong to the same image is 79%, while in a similar situation, we obtained 87.3493%. Paper [14] exposes no numerical results done on a test set, but presents a 96% accuracy achieved on the validation set. Our best accuracy on the validation is only 89.15%, but our dataset had just 2322 entries for training and 664 for validation, while in the mentioned paper a set of 6000 entries was used for training and 1000 for validation. An overfit on the training data cannot be excluded in their case, since the authors of [14] mentioned poor results in reconstructing an image from patches, despite the high accuracy.

Another difficulty comes from the fact that some works exhibit working with three-dimensional models of artefacts [8] or 3D puzzles [5], and our solving is strictly based on 2D images. While [8] lacks numerical results, in [5] the model manages to solve a 3D 18-piece puzzle successfully. However, when tested on a broken ostrich eggshell, only 11 out of 15 shell pieces are placed correctly.

As for the papers [6], [7], [12] and [2], they come closest to our work in terms of the proposed problem, because they use 2D puzzle pieces, i.e. pieces of documents, as input data. What distinguishes our work is the use of deep learning techniques, whose help is difficult to estimate in comparison with image processing techniques. This happens especially because in the previously mentioned works the methodology for evaluating the experiments is to measure the time taken to reassemble the puzzle, i.e. the document, while this was not a main focus of our work. Also, in the case of [2], the reassembled documents are cut into only 4 parts, and the puzzle pieces used in [6] and [12] have very different shapes from each other, much more irregular than the pieces of a normal jigsaw puzzle. These aspects favour the success of the model. In terms of results, Makridis et al [12] present the assembly of a puzzle, without mentioning the time needed for the whole process, and in [2] the best image preprocessing time was 0.394s, and the best joining time was 0.336s. The results provided in [6] refer to the solving of a 48-piece puzzle in 58 minutes and a 67-piece one in 31 minutes. The same 48-piece puzzle was solved in 3 minutes with the approach from [7], which is a great improvement. However, the approach from [7] is not as robust, since the authors mentioned a higher assembly time in comparison with [6], for the same 67-piece puzzle.

5. Conclusions and future work

The aim of this paper is to develop a solution to the problem of matching puzzle pieces from images, using deep learning methods. Solving this problem is the first step in the digital approach to the difficult task of restoring artefacts (e.g. putting together ostraca pieces). The solution proposed consists of two

phases: segmenting the puzzle pieces, followed by finding matching pairs for each side of a puzzle piece.

Regarding segmentation, the method applied is the U-Net model, trained with 462 images with puzzle pieces and related masks. The main result obtained in this respect is a score for the dice coefficient of 0.9751 for the test set, which represents a very good result given that the highest possible value is 1 for perfect overlap.

The edge-matching task has been addressed with a Siamese architecture model. Two masks are processed at the same time by two identical networks, and then the absolute difference between the two feature vectors is computed. The result is flattened, linearized, and at the end, the output consists of two probabilities: match or non-match. The main result obtained is an accuracy as high as 87.3493% and an F1 score of 85.6164, for a number of 332 pairs of masks used in the test. This result was achieved when the masks used to feed the network were processed so that they contained only the contours of the pieces. Reversing the colours of the second mask also proved effective, so the model managed to find for 17.36% of cases among the best 10 predictions (the most compatible images), the ideal match of the piece in the initial image.

In the future, we plan to improve the image set with more puzzle pieces in different contexts, to help the segmentation model identify the contours of the pieces in any situation. Also, for the edge-matching model, the results of the *top_10* evaluation need improvement, which could be done by developing a more complex model and diversifying the dataset. The differences between the puzzle pieces are small, at least for the puzzles we used, which made learning the matches difficult.

The current study contributes to the exploration of artefact reconstruction, obtaining good results on both segmentation and edge-matching of images with puzzle pieces.

References

[1] Alagrami, A., Palmieri, L., Aslan, S., Pelillo, M., and Vascon, S. Reassembling broken objects using breaking curves, 06 2023.

[2] Biswas, A., Bhowmick, P., and Bhattacharya, B. Reconstruction of torn documents using contour maps. In *IEEE International Conference on Image Processing 2005* (2005), vol. 3, pp. III–517.

[3] Bromley, J., Bentz, J., Bottou, L., Guyon, I., Lecun, Y., Moore, C., Sackinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence 7* (08 1993), 25.

[4] Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Llados, J., and Pal, U. Signet: Convolutional siamese network for writer independent offline signature verification, 2017.

[5] Grim, A., O'Connor, T., Olver, P., Shakiban, C., Slechta, R., and Thompson, R. Automatic reassembly of three-dimensional jigsaw puzzles. *International Journal of Image and Graphics 16* (04 2016), 1650009.

[6] Hoff, D., and Olver, P. Automatic solution of jigsaw puzzles. *Journal of Mathematical Imaging and Vision 49* (05 2014).

[7] Illig, P., Thompson, R., and Yu, Q. Application of integral invariants to apictorial jigsaw puzzle assembly. *Journal of Mathematical Imaging and Vision 65* (2021), 340–353.

[8] Jampy, F., Hostein, A., Eric, F., Laligant, O., and Truchetet, F. 3d puzzle reconstruction for archeological fragments. *Proceedings of SPIE - The International Society for Optical Engineering 9393* (03 2015).

[9] Kass Michael, Witkin Andrew, T. D. Snakes: Active contour models. *International Journal of Computer Vision 1* (1988), 321–331.

[10] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014).

[11] Li, R., Liu, S., Wang, G., Liu, G., and Zeng, B. Jigsawgan: Auxiliary learning for solving jigsaw puzzles with generative adversarial networks. *IEEE Transactions on Image Processing 31* (2022), 513–524.

[12] Makridis, M., and Papamarkos, N. A new technique for solving a jigsaw puzzle. In *2006 International Conference on Image Processing* (2006), pp. 2001–2004.

[13] Markaki, S., and Panagiotakis, C. Jigsaw puzzle solving techniques and applications: a survey. *The Visual Computer 39*, 10 (2023), 4405–4421.

[14] Ostertag, C., and Marie, B.-A. Matching ostraca fragments using a siamese neural network. *Pattern Recognition Letters 131* (2020), 336–340.

[15] Papaioannou, G., Schreck, T., Andreadis, A., Mavridis, P., Gregor, R., Sipiran, I., and Vardis, K. From reassembly to object completion: A complete systems pipeline. *ACM Journal on Computing and Cultural Heritage 10* (2017), 8:1–8:22.

[16] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: Crossentropyloss. `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`, 2021.

[17] Pintus, R., Pal, K., Yang, Y., Weyrich, T., Gobbetti, E., and Rushmeier, H. A survey of geometric analysis in cultural heritage. *Computer Graphics Forum 35*, 1 (2016), 4–31.

[18] Pirrone, A., Aimar, M. B., and Journet, N. Papy-s-net: A siamese network to match papyrus fragments. In *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing* (New York, NY, USA, 2019), HIP '19, Association for Computing Machinery, pp. 78–83.

[19] Ronneberger, O., P.Fischer, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), vol. 9351 of *LNCS*, Springer, pp. 234–241.

[20] Willis, A. R., and Cooper, D. B. Computational reconstruction of ancient artifacts. *IEEE Signal Processing Magazine 25*, 4 (2008), 65–83.

Department of Computer-Science, Faculty of Mathematics and Computer Science, Babeş,-Bolyai University, Cluj-Napoca, Romania

Email address: `raluca.chis@ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.02

3D DEFORMABLE OBJECT MATCHING USING GRAPH NEURAL NETWORKS

MIHAI-ADRIAN LOGHIN

Abstract. Considering the current advancements in computer vision it can be observed that most of it is focused on two dimensional imagery. This includes problems such as classification, regression, and the lesser known object matching problem. While object matching ca be viewed as a solved problem in a two dimensional space, for a three dimensional space there is a long way to go, especially for non-rigid objects. The problem is focused on matching a given object to a target object. We propose a solution based on Graph Neural Networks that tries to generalize over multiple objects at once, based on self-attention and cross-attention blocks for the network. To test our solution, we utilised five convolutional operators for the layers of the model. The convolutional operators we compared included GCNConv, ChebConv, SAGEConv, TAGConv, and FeaStConv. This paper aims to find the best operators for our architecture and the task. Our approach obtained favourable results for predicting the barycentric weights for the model, while struggling on predicting the triangle indexes. The best results were obtained for the models using GCNConv, for the triangles index prediction and FeaStConv for the barycentric coordinates prediction.

1. Introduction

In the most recent years we have observed multiple new technologies that require or are improved by usage of 3D models. This ranges from applications in medicine [13] to research oriented papers about simulations of 3D environments. Still, very few papers talk about how we can relate 3D objects to real-world objects. By creating a solution for relating 3D objects to real-world

Received by the editors: 29 November 2023.

2010 *Mathematics Subject Classification.* 68T45, 68U05.

1998 *CR Categories and Descriptors.* I.2.10 [**ARTIFICIAL INTELLIGENCE**]: Vision and Scene Understanding – *3D/stereo scene analysis*; I.3.5 [**COMPUTER GRAPHICS**]: Computational Geometry and Object Modeling – *Geometric algorithms, languages, and systems*.

Key words and phrases. graph neural networks, object matching, 3D objects, deformable objects.

objects, software such as NextMed [13] could be improved. On a similar aspect, the validity of such an application is given by frameworks like the one presented in [26]. By starting this research we could end up with applications that would easily allow to overlap digital reconstructions over real-world objects automatically. This could be especially useful in medicine, where we could have a 3D recreation of an organ that could be moulded automatically over the real organ.

In this paper, we will explore one of the first steps in relating real-world objects to those from a virtual world. The process consists of first seeing if it is even possible to relate 3D objects to one another by focusing on deformable objects. While 2D image matching or object matching can be considered an optimised problem [16, 19, 30], the same cannot be said for cases where the objects can suffer various deformations such as cuts, holes, or surface changes. Furthermore, very few, if any, experiments have focused on creating a reference-object agnostic solution for the problem.

Throughout this paper, we will present the overall field of study, how we constructed a dataset for the problem and how the model for different kinds of solutions have been made. Our focus will be on exploring solutions that work based on multiple inputs, data from multiple datasets, and multiple shape classes for the objects. We aim to obtain a good comparison between multiple model architectures for deformable object matching, mainly focusing on comparing convolutional operators that work on graphs, representing 3D objects. The solution that we provide will be based on using self and cross-attention blocks in a graph neural network to take into account both the deformed shape and the target shape.

Our paper is structured into eight sections starting with the introduction. In Section 2 we present the current state-of-the-art and other approaches in the field of study, to properly define it. Section 3 will be used to discuss the problem of object matching, and which solutions and datasets for the problem are currently available. After defining the problem, Section 4 will be used to define what graph neural networks are and what the different convolution operators for those networks are. We will present our proposed model design and its evaluation in Section 5. The results and experimental setup will be presented in Section 6 and then they will be further discussed in Section 7. We will go over a few conclusions and further research possibilities in Section 8.

2. Related work

To start off, we need to focus on what has already been accomplished in this field. As such, in this section we will present the already solved problem or existing models related to the problem at hand.

At the very beginning of research related to object matching, most of the solutions were focused on works related to two dimensional images. While articles such as [17, 19, 32] have already solved the problem, they also indicate possible approaches for solutions in higher dimensions. Article [16] in particular shows the possibility of using graph neural networks (GNN) for the problem with very promising results.

Deformable objects, especially if related to matching between three dimensional objects, have been especially of interest in challenges for the field. A clear example of this are the articles [4, 8, 9, 22] related to the SHREC event in various years. The results from the challenge show that the current state-of-the-art is represented by Partial Functional Maps, followed closely behind by an approach using Random Forests. This indicates the possibility of using a machine learning approach for the problem and creating a model that can learn to associate one model to multiple objects. With partial functional maps, the authors of [22] managed to obtain a matching percentage of around 80% for deformations involving both cuts and holes. This method creates a form of mapping between the points of one object to another, treating them like functions.

On the side of usefulness for the field, we do not need to look further than the field of robotics. Matching a predefined 3D object to a real-world counterpart has been essential in multiple papers [20,31,34], including even solutions outside of robotics [26] based on rigid objects. Still, the problem in robotics is that it relies heavily on using an almost perfect environment and object representation to the 3D counterpart.

Other methods for deformable object matching are based on point cloud data. One of those solutions can be found in [23], focusing on human-shaped 3D objects. The authors present a method that uses learning based on functional maps, focusing only on heaving self-attention blocks that embed the objects. They claim to achieve state-of-the-art results obtaining an error of $5.4e-2$. A similar solution, closer in structure to what we propose, for learning point cloud matching is presented in [12]. This solution provides a network based on self and cross-attention blocks using the Chamfer distance as its loss function. Compared to the previous solution, the authors validate their approach on multiple shape classes for the 3D objects. On SHREC 2019, they obtained an accuracy of 15.3% and an error of 5.6, beating the other methods they compared with. Given that both approaches obtained state-of-the-art

results, it is easy to say that in regards to using point clouds the solutions are well optimised. Still, the discrepancy between the error and accuracy for the last method shows the need for further investigation of the architecture and of the results obtained by other methods.

Considering everything that has been mentioned so far, it should be clear that the problem itself still requires a lot of improvements. This is especially true when it comes to constructing a machine learning approach for the problem, as we will be doing in the following sections.

3. Deformable object matching

In this section, we will discuss the problem of 3D deformable object matching, some already existing solutions for the problem and how a dataset for the problem looks like. To do all of this, we will separate our findings into two subsections. The first subsection will be for the larger problem, while the second will be dedicated to defining our dataset.

3.1. **Problem definition.** Deformable object matching, compared to matching rigid bodies, is a lot more complicated than it might seem. It requires finding a way to transform the given object into something that more closely resembles the reference object. The main problem here is finding the correct type of transformation needed, as it goes beyond basic transformations such as translation, rotation and scaling [5, 26, 29]. Regarding this, we can use barycentric weights to solve the problem [4, 8, 22].

$$p = \sum_{i=1}^{3} \lambda_i * v_i, where : \sum_{i=1}^{3} \lambda_i = 1 \text{ [14]} \tag{1}$$

Considering Equation 1, we can transform any point from a deformed object to a position that more closely matches that of the target object. In the equation, λ_i with $i \in \{1, 2, 3\}$ represents the barycentric weights for a given triangle from target object. The weights are chosen in such a way that by multiplying each weight to the vertices of the triangles and then summing them we get a approximation of a node from the deformed object to the target object [4,8,14,22]. The transformation could lead to an association like that from Fig. 3, given the rightmost result.

3.2. **Dataset.** An important part of our project was gathering relevant and consistent data from across multiple datasets. In the end, we have found two datasets with very similar formats that still offered their own unique elements for the problem. The datasets that were used are SHREC 2016 [4,22] and SHREC 2019 [8]. The first dataset offers deformations such as body

movements, holes and cuts. On the other hand, the second dataset offers deformations such as surface changes, wrinkles, and joint movements.

In total, the datasets offer 276 three dimensional objects spread across 11 classes. The classes are for the most part evenly distributed, with the main exceptions being the human class having the highest number of objects and the glove class having the lowest number of objects [4, 8, 22]. For our testing environment, we have constructed a dataset with a 70-30 train-test split. We have split the dataset evenly for all the classes.

4. Graph Neural Networks (GNN)

As stated in Section 2, the current state-of-the-art is represented by the use of Functional Maps, closely followed behind by the use of Random Forests [4, 8, 9, 22]. Both of the mentioned approaches fail to properly generalize to different types of damage. While there is another approach using Graph Neural Networks [21], even that one still bases its final layer on Functional Maps. We are researching the possibility of using GNNs without any other attachments. For that, we will look into how different types of convolutions can improve the results.

Defining graph neural networks is essential for our research. As such, we will be looking into the general definition for this machine learning architecture and how it can be used to suit our needs. We defined multiple subsections for this case, to understand not just the general definition, but also the subsequent definitions needed for our model architecture.

4.1. **Definition.** Graph neural networks are a variant of neural networks such that they can work on graphs. In our case, a graph is defined as $G = (V, E)$, where V represents the set of nodes in the graph and E is the set of edges that connect the nodes. We can also associate a matrix $A \in \mathbb{R}^{N \times N}$ to the graph, representing the adjacency matrix. Using this, and the power of neural networks we can create predictions at node, edge, and graph level [18, 35, 37].

Defining a mathematical expression for GNNs requires taking into consideration information about graph structures and neural networks at once. This consists of using information regarding nodes, edges, and learnable weights, which are represented in the following equation:

$$h_i^{t+1} = f(h_i^t W_i + \sum_{j \in N_i} \frac{1}{c_{ij}} h_j^t W_j) \text{ [35, 37]} \tag{2}$$

To understand GNNs even better, we will analyse Equation 2. In this equation h_i^t represents the vector representation of node i at time t, W_i represents the learnable weights for the given node i (which are not always present), c_{ij}

represents a non-weighted connection between nodes i and j, N_i is considered the node neighbourhood of node i and W_j represents the learnable weights for the given node j. In the equation, f represents a propagation function and if needed, can be interpreted as an activation function. This equation stands at the base of how most, if not all, GNNs are constructed [35, 37]. While it might suffer changes from implementation to implementation, the main idea remains the same.

4.2. **GCNConv.** This graph convolution was created for the task of semi-supervised classification. Still, this does not mean it can not be used for scenarios such as ours. Considering this, we will look at what the main aspects of this convolutional operator are and what kind of results were obtained using it in the original presentation of the method [18,37]. If we were to start from equation 2, we can form a new equation, with a similar structure.

$$H^{t+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^tW^t) \text{ [18]} \tag{3}$$

Equation 3 represents the propagation rule of the network. Similarly, we work just like before with the node level information, like in Equation 2. Here H^t represents the activation results of layer l and for $t = 0$ it represents the results of the initial graph, similarly to how h_i^t was used in the previous equation. W^t represents, just as before, the learnable weights for layer l. σ denotes the activation function for the layer, and it can take multiple forms. In the original paper, the authors have used both softmax and ReLU as possible approaches. $\widetilde{D}_{ii} = \sum_{j=0} \widetilde{A}_{ij}$ is considered a diagonal degree matrix, for which $\widetilde{A} = A + I$ is the adjacency matrix with self inserted loops [18].

In the original evaluation, the authors of the method have shown through rigorous testing that their method could beat other approaches. In their experiments, they have noted improvements over multiple runs. They have also shown how their method can get over 80% accuracy where other methods would only get at most 75% [18].

This convolutional operator has also been proven to be time-efficient. It was shown that for the most part it performs the same on GPU and CPU, only slowing in performance when there are more edges in the graph. For a number of 1k edges the authors have noted a performance of around 10^{-3} seconds/epoch and only at 10M nodes did they reach 10 seconds/epoch on GPU [18].

4.3. **ChebConv.** The ChebConv operator was created with the idea of creating GNNs with fast localized spectral filtering. This means that the authors

implemented ideas similar to pooling in their approach [6]. Those can be better portrayed by the following equation:

$$H^{t+1} = \sum_{k=0}^{K-1} \Theta_k T_k(\widetilde{\Lambda}) \text{ [6]} \tag{4}$$

This convolutional operator is described in full in Equation 4. Here, Θ is a vector of polynomial coefficients, more precisely Chebyshev coefficients. $T_k(\widetilde{\Lambda})$ is a Chebyshev polynomial of order k and it is determined recurrently. To start the recurrent process we consider $\widetilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I$, where I is the identity matrix and Λ is the diagonal matrix formed by the frequencies of the graph, $T_0(\widetilde{\Lambda}) = H$, and $T_1(\widetilde{\Lambda}) = \widetilde{\Lambda}H$. We can then consider the recurrent function $T_k(\widetilde{\Lambda}) = 2\widetilde{\Lambda}T_{k-1}(\widetilde{\Lambda}) - T_{k-2}(\widetilde{\Lambda})$. The authors have noted that the entire operation would have a computation cost of $O(K|\epsilon|)$ [6].

For the experimental phase of the research, the authors have revisited various datasets for which solutions exist using classical CNNs and other machine learning methods. Those datasets include MNIST and 20NEWS. For the MNIST dataset, they have shown an almost exact performance with CNNs. Still, the proposed architecture outperformed other methods in terms of time efficiency. As for the 20NEWS dataset, the model did not manage to outperform the Multinomial Naive Bayes approach. Although it did not perform better in this case, the authors noted that their proposed architecture still outperforms other fully connected neural networks [6].

In their research, they have also tackled the influence of graph quality on the results. They have noted that the way the graph is constructed is the most important part for their operator to work. For this, they considered a comparative use of image and text graphs. What their study has shown is that the method works best on image graphs, but due to the limitation of text graphs, it cannot outperform the current state-of-the-art [6].

4.4. **SAGEConv.** This method was created on the assumption that not all nodes in the graph need to be used for determining the embedding of one node. SAGEConv came as a way to essentially improve the already existing approach of using low-dimensional node embeddings for large graphs. As such, they proposed a framework called GraphSAGE [15]. The framework also stands as one of the more popular approaches for graph convolutions [37]. In our research we are only interested in the convolutional operator of this framework and how it can help us. We will detail our findings and explain what we will be using from this framework.

The SAGEConv operator can be described in the form of two equations, of which the latter is optional. This idea comes from showing the operator in two ways, a standard equation operator and an optimised operator that can be used for further improving the model [15].

$$h_i^{t+1} = W_1 h_i^t + W_2 * mean_{j \in N_i}(h_j^t) \text{ [15]} \tag{5}$$

$$h_j^{t+1} = \sigma(W_3 h_j^t + b) \text{ [15]} \tag{6}$$

We can formulate the equation for the convolutional operator based on the information from Equations 5 and 6. In those equations W_i for $i \in \{1, 2, 3\}$ represent the weight matrices that will change during the training process. As previously used, σ can represent any activation function. In the original article, the authors have not presented any functions that would be more favourable to be used. h_i represents the feature vector for a node v_i [15].

In the implementation of the convolutional operator, the most interesting part is the relation between Equation 5 and Equation 6. By that, we are referring to the fact that the relations can work independently of one another, at least according to the PyTorch Geometric implementation. We can then consider that the combined use of the two equations is an improved version of the convolutional operator over the graph [10, 15]. In our work, we only considered the base version, without the additional improvement to the method. This is due to it requiring a more complicated implementation of our model.

For the qualitative evaluation of the operator, the authors have noted a comparison against four other methods. Those methods include the DeepWalk algorithm, a random classifier, logistic regression, and a hybrid between raw features and DeepWalk embeddings. They have also noted extended versions of their algorithm that use the operator from Subsection 4.2, an LSTM, a mean operator, and a pooling operator [15, 18].

The testing took place on three datasets. Those datasets were based on citations, Reddit posts, and the PPI dataset. On all three of those, the proposed algorithm has outperformed all other methods, with the most notable results being from the GCNConv and LSTM variants. The algorithms were tested on both supervised and unsupervised environments. The authors have noted that their method does generalize across graphs [15].

4.5. **TAGConv.** In the previous three subsections, we have considered the use of the more popular convolutional operators. Now it is time to get into more problem-specific operators, mainly those that were created to work directly with 3D objects. Graph convolutions can defined on the spectral or vertex domains, of which the authors of TAGConv have chosen the latter.

The name of the model stands for Topology Adaptive Graph Convolution and it is mainly based on the idea that the network/operator will adapt to the topology of the graph [7].

$$H^{t+1} = \sum_{k=0}^{K} (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})^k H^t W_k \text{ [7, 10]} \tag{7}$$

Since we are interested in the convolutional operator from the entire network, we will only look at that part of its equation. The operator is represented by Equation 7 and it is adjusted to fit its PyTorch Geometric implementation. In this case, we consider K to have the base value of 3, representing the number of hops. A represents the adjacency matrix of the graph and $D_{ii} = \sum_{j=0} A_{ij}$ the diagonal degree matrix. Considering that, $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalization of the diagonal matrix. As always, W represents the learnable weights for the convolutional operator [7, 10].

This operator, like the previous ones, has been tested mainly on standard benchmark datasets. Those datasets include Pubmed, Citeseer, and Cora. While the model was tested against other standard methods such as DeepWalk and deep convolutional neural networks, it also tested against other graph neural networks such as ChebNet and GCN, presented in the first two subsections. According to the authors, TAGConv outperforms the other methods on all three datasets, with an accuracy of over 80% on two of the datasets [7].

While topology is an important part of defining geometric forms, it is not enough. For the next subsection, we will be looking at our final operator, created to work perfectly with 3D objects. Just like now, we will be looking at it from a theoretical and applied perspective.

4.6. **FeaStConv.** For our final operator, as previously stated, we will focus on direct applications to 3D objects. This operator was constructed from the need to create something like convolutional neural networks, but for 3D shapes. Its name stands for Feature-Steered Graph Convolutions and its original evaluation was done directly on 3D meshes for a variety of problems regarding shape analysis [33].

FeaStConv can be better described in the following equation:

$$h_i^{t+1} = \frac{1}{|N_i|} \sum_{j \in N_i} \sum_{k=1}^{K} q_k(h_i^t, h_j^t) W_k h_j^t \text{ [10, 33]} \tag{8}$$

The equation, like in the previous subsection, is defined according to its implementation in the original article and the PyTorch Geometric implementation. Equation 8 describes the convolutional operator in full. The most unique elements of the equation are represented by K and q_k representing the number of attention heads and an activation function, respectively. In this equation $q_k(h_i, h_j) = softmax_j(u_k^T(h_j - h_i) + c_k)$, where W_k, u_k and c_k are trainable parameters [10, 33].

As mentioned, above, the operator was tested and experimented with using problems related to 3D shape analysis. Those problems included 3D shape correspondence and part labelling. For 3D shape correspondence, the model was compared to other methods such as PointNet, ACNN, GCNN and MoNet. The model proposed by the authors outperformed all other models by a lot, having an accuracy of 98% at most and 88% at least. For the part labeling problem, they used a dataset based on ShapeNet [2] and compared it with four other methods, some of which were mentioned before. While it did not outperform any of the other methods, except one where the difference was of 0.1%, the model still got overall similar results [33].

5. Model and evaluation

We need to properly define the model architecture and evaluation on our data. We will explore this in two subsections, dedicated to each subject. Following that, we will focus on the experiments defined by us.

All of the implementation effort was done using PyTorch and PyTorch Geometric for the loss function and for the implementation of the convolutional operators, respectively. We have chosen those frameworks, based on the number of operations that they had implemented and their usage in other papers [10, 25].

5.1. **Model.**

Our model architecture is based on the idea of using self and cross-attention blocks. Those blocks are a necessity for the problem, as we have to work with multiple 3D objects at once for one result. As a reference for constructing our model we have used SuperGlue [30] and later validated it based on articles that came out during our research that tried to use similar architectures for the use of self-attention and cross-attention blocks [21]. To further emphasise the validity of our approach, we also considered looking into approaches that try to solve the problem in other contexts, such as point cloud data [12]. We only used other models as architectural references, rather than for specific layer parameters.

Fig. 1 is a visualization of our model and how it works. The desired output, as referenced in Section 3 is an [n,4] vector. In this case, n represents the number of nodes in the graph and 4 is the size of the output for each node,

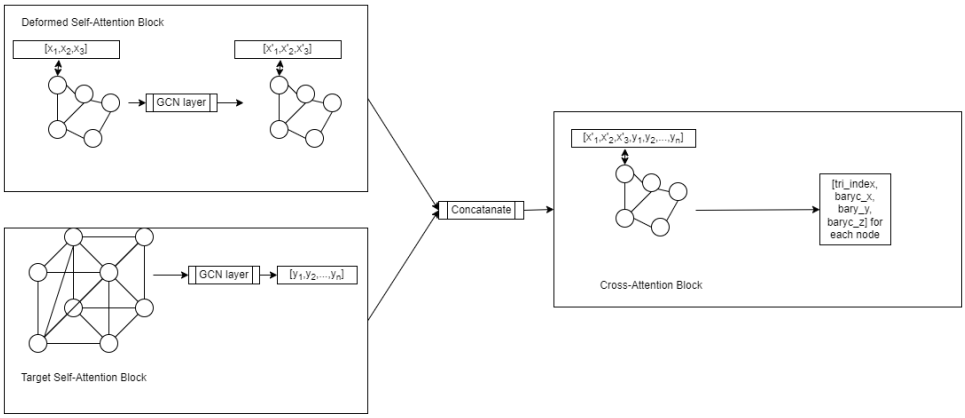


FIGURE 1. A diagram describing the Graph Convolution Network (GCN) created to address the problem of object matching. The model works using a dual input: the deformed object and the reference object. In the third layer of the model, the two outputs are concatenated and the final result will be a tensor of size four representing: the triangle of correspondence, and the barycentric weights for the given triangle.

representing the triangle index and barycentric weights. The self-attention blocks are meant for reinterpreting the object graphs based on node neighbourhoods and the cross-attention block is meant to learn how the two graphs can interact.

For the cross-attention block and deformed self-attention block we have used the approach of reducing the number of hidden channels after each individual convolution. First, it is divided by 2 and then by 4. In the case of the reference/target self-attention block, we used the opposite approach as we wanted to obtain a larger embedding of the graph at the end. As such, we first multiply the number of hidden channels by 2 and then by 4. Between the convolutions, we have used a ReLU activation function and for the output we have used the ReLU for the triangle output and SoftMax for the barycentric part of the output.

5.2. **Training and evaluation.** We have decided to evaluate our model using three loss functions. The loss functions were determined based on the required outputs of the model in regard to the dataset. As such, we have determined that there needs to be a loss for the triangle index $L1_{Triangle}$, one

for the barycentric weights $L1_{Baryc}$ and one for the combined loss of the two $L1_{Combined}$.

$$L1_{combined} = L1_{triangle} + L1_{baryc} \tag{9}$$

For our experiments, we have used the standard definition for the loss function, which is used to determine $L1_{Triangle}$ and $L1_{Baryc}$. In Equation 9 we can observe the combined loss function which represents the sum of the other loss functions and which was used to train the model [3, 11, 24]. To complete the training requirements, we also need to mention the use of the Adam optimizer with a learning rate of $3e-4$, which was chosen based on its proven improvements in performance [1, 36].

We also consider the mean geodesic error as a possible loss function, as it was used on the SHREC 2016 dataset [22]. In the end, we opted against this. After further research, the function did not seem standard enough to be selected.

6. Experiments

6.1. **Setup.** To cover all possible architectural and data combinations, we have decided to explore the problem in four main ways. Each of the three experiment types will have a role in the evaluation of the model in a new environment to see how it reacts under new conditions. The experiments are: [4, 8, 22]

- Experiments on the combined dataset from SHREC 2016 and 2019 [4, 8, 22]
- Experiments on the SHREC 2016 dataset [4, 22]. To evaluate if the standalone dataset offers a better training environment.
- Experiments on the SHREC 2019 dataset [8]. Just like the previous point, it will be used to evaluate the consistency of the results on a standalone dataset. Additionally, it will also help in defining which dataset is better as a training set,
- Experiments on some of the single classes from the SHREC 2016 dataset [4, 22]. An experiment type that will help determine if the problems that we found are related to the use of multiple classes or due to the model that is being used.

Additionally, the experiments will be done for all the convolutional layers defined in Section 4. Besides the change in convolutional layers, we also applied changes in terms of hidden channels, for a number of 20 epochs each and a batch size of 20. An exception for the batch size was used in the case of the single class experiments, where we used a batch size of one.

For the experiments done on the entire dataset we have only used the best combinations obtained from the other experiments. We also reduced the number of epochs by five and expended the batch size by ten, due to time constraints and computational availability in the used environment.

Conv. type	Ch.	Embed	$L1_{Triangle}$		$L1_{Baryc}$		$L1_{Combined}$	
			train	test	train	test	train	test
GCNConv	16	100	7.4E+03	2.2E+04	4.2E-01	1.3E+00	7.4E+03	2.2E+04
		500	5.7E+03	1.7E+04	4.3E-01	1.4E+00	5.7E+03	1.7E+04
	32	100	7.3E+03	2.3E+04	4.1E-01	1.3E+00	7.3E+03	2.3E+04
		500	5.3E+03	1.6E+04	4.6E-01	1.4E+00	5.3E+03	1.6E+04
ChebConv	16	100	1.1E+07	2.7E+07	4.4E-01	1.4E+00	1.1E+07	2.7E+07
		500	7.0E+06	1.6E+07	4.4E-01	1.4E+00	7.0E+06	1.6E+07
	32	100	2.9E+07	6.7E+07	4.6E-01	1.4E+00	2.9E+07	6.7E+07
		500	8.6E+06	1.9E+07	4.5E-01	1.4E+00	8.6E+06	1.9E+07
SAGEConv	16	100	7.4E+03	2.3E+04	4.5E-01	1.4E+00	7.4E+03	2.3E+04
		500	6.8E+03	2.0E+04	4.4E-01	1.4E+00	6.8E+03	2.0E+04
	32	100	5.9E+03	1.8E+04	**4.0E-01**	**1.2E+00**	5.9E+03	1.8E+04
		500	**5.7E+03**	**1.7E+04**	4.2E-01	1.3E+00	**5.7E+03**	**1.7E+04**
TAGConv	16	100	7.8E+03	2.3E+04	4.2E-01	1.2E+00	7.8E+03	2.3E+04
		500	7.1E+03	2.3E+04	4.1E-01	1.3E+00	7.1E+03	2.3E+04
	32	100	7.2E+03	2.3E+04	4.3E-01	1.3E+00	7.2E+03	2.3E+04
		500	6.0E+03	1.9E+04	4.2E-01	1.3E+00	6.0E+03	1.9E+04
FeaStConv	16	100	9.9E+03	3.1E+04	3.5E-01	1.1E+00	9.9E+03	3.1E+04
		500	9.4E+03	2.9E+04	3.6E-01	1.1E+00	9.4E+03	2.9E+04
	32	100	9.5E+03	3.0E+04	3.6E-01	1.1E+00	9.5E+03	3.0E+04
		500	8.0E+03	2.5E+04	3.8E-01	1.2E+00	8.0E+03	2.5E+04

TABLE 1. The results for a single class in the SHREC 2016 dataset [22]. The class was chosen so that it would have a single reference object.

6.2. **Results.** Table 4 is the table that contains the evaluation of the model in the context of the full dataset that was originally presented in Section 3. As for Tables 2 and 3, they represent the results for individual datasets that formed the full dataset. Finally, Table 1 presents the model aggregated results when evaluated on a single class.

The scope of the results is to determine the best possible combination. Our research has so far only focused on determining the best architectural combination for a graph neural network for the task. We only considered longer amounts of training for the best combination, due to our limited computational power. Furthermore, the experiments were designed in such a way as to allow us to find the weakest links in our dataset and approach. We will touch up more on our decisions in the next section.

Conv. type	Ch.	Embed	$L1_{Triangle}$		$L1_{Baryc}$		$L1_{Combined}$	
			train	test	train	test	train	test
GCNConv	16	100	6.3E+03	1.9E+04	4.2E-01	1.3E+00	6.3E+03	1.9E+04
		500	5.5E+03	1.7E+04	4.3E-01	1.3E+00	5.5E+03	1.7E+04
	32	100	5.9E+03	1.8E+04	4.3E-01	1.4E+00	5.9E+03	1.8E+04
		500	5.4E+03	1.7E+04	4.6E-01	1.4E+00	5.4E+03	1.7E+04
SAGEConv	16	100	6.3E+03	1.9E+04	4.6E-01	1.4E+00	6.3E+03	1.9E+04
		500	6.2E+03	1.9E+04	4.4E-01	1.4E+00	6.2E+03	1.9E+04
	32	100	5.8E+03	1.7E+04	4.1E-01	1.3E+00	5.8E+03	1.7E+04
		500	**5.5E+03**	**1.6E+04**	4.3E-01	1.3E+00	**5.5E+03**	**1.6E+04**
TAGConv	16	100	6.7E+03	2.1E+04	4.2E-01	1.3E+00	6.7E+03	2.1E+04
		500	6.3E+03	2.0E+04	4.2E-01	1.3E+00	6.3E+03	2.0E+04
	32	100	6.2E+03	1.9E+04	4.3E-01	1.3E+00	6.2E+03	1.9E+04
		500	6.2E+03	2.0E+04	4.6E-01	1.4E+00	6.2E+03	2.0E+04
FeaStConv	16	100	8.3E+03	2.6E+04	**3.8E-01**	**1.2E+00**	8.3E+03	2.6E+04
		500	-	-	-	-	-	-
	32	100	-	-	-	-	-	-
		500	-	-	-	-	-	-

TABLE 2. Our model's results on the SHREC 2016 dataset [22].

Conv. type	Ch.	Embed	$L1_{Triangle}$		$L1_{Baryc}$		$L1_{Combined}$	
			train	test	train	test	train	test
GCNConv	16	100	6.3E+03	2.0E+04	4.7E-01	1.5E+00	6.3E+03	2.0E+04
		500	5.7E+03	1.8E+04	4.5E-01	1.4E+00	5.7E+03	1.8E+04
	32	100	5.5E+03	1.7E+04	4.3E-01	1.4E+00	5.5E+03	1.7E+04
		500	5.5E+03	1.8E+04	**4.1E-01**	**1.3E+00**	5.5E+03	1.8E+04
SAGEConv	16	100	6.6E+03	2.1E+04	4.2E-01	1.3E+00	6.6E+03	2.1E+04
		500	6.1E+03	1.9E+04	4.3E-01	1.4E+00	6.1E+03	1.9E+04
	32	100	5.8E+03	1.8E+04	4.6E-01	1.5E+00	5.8E+03	1.8E+04
		500	**5.5E+03**	**1.7E+04**	4.4E-01	1.4E+00	**5.5E+03**	**1.7E+04**
TAGConv	16	100	6.5E+03	2.2E+04	4.5E-01	1.5E+00	6.5E+03	2.2E+04
		500	6.1E+03	2.1E+04	4.6E-01	1.5E+00	6.1E+03	2.1E+04
	32	100	6.2E+03	2.0E+04	4.5E-01	1.4E+00	6.2E+03	2.0E+04
		500	6.1E+03	4.0E-01	4.5E-01	1.5E+00	6.1E+03	2.1E+04
FeaStConv	16	100	7.9E+03	2.6E+04	4.5E-01	1.4E+00	8.0E+03	2.6E+04
		500	-	-	-	-	-	-
	32	100	7.6E+03	2.5E+04	4.4E-01	1.4E+00	7.6E+03	2.5E+04
		500	-	-	-	-	-	-

TABLE 3. Our model's results on the SHREC 2019 dataset [8].

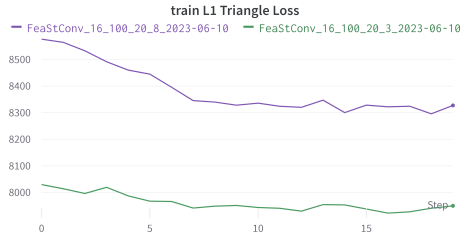
Conv. type	Ch.	Embed	$L1_{Triangle}$		$L1_{Baryc}$		$L1_{Combined}$	
			train	test	train	test	train	test
GCNConv	32	500	**5.4E+03**	**1.7E+04**	4.5E-01	1.4E+00	**5.4E+03**	**1.7E+04**
SAGEConv			5.6E+03	1.7E+04	4.3E-01	1.3E+00	5.6E+03	1.7E+04
TAGConv			6.2E+03	2.0E+04	4.5E-01	1.4E+00	6.2E+03	2.0E+04
FeaStConv	16	100	8.3E+03	2.6E+04	**3.8E-01**	**1.2E+00**	8.3E+03	2.6E+04

TABLE 4. Our model's results on the dataset formed by combining the SHREC 2016 and 2019 datasets [8, 22].

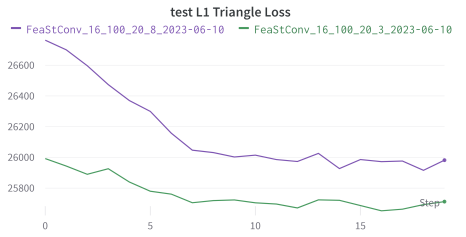
7. Discussion

When confronted with the problem of predicting large numbers for the triangle indexes, we have determined a shortcut for the training procedure. The outputs here are in the tens of thousands and are rather hard to learn for a neural network. To simplify the process we have tested multiplying the output with multiples of ten. The best results were obtained when multiplying the output by 100.

In our approach, we have encountered several benefits and some downfalls. To start off, we have observed a huge under-performance while using ChebConv as observed in Table 1. Since we started with single class experiments to get an initial idea of how to continue the rest of the experiments, we have removed all experiments using ChebConv architectures from the other experiment types. We motivate this choice by arguing that if an architecture under-performance on a single class, it has no way of performing better when put against multiple classes.



(A) Train results.



(B) Test results.

Figure 2. A visual representation of the triangle index loss results using FeaStConv for the individual datasets. The purple line represents the results for the SHREC 2016 [4, 22] dataset, while the green line represents the results for the SHREC 2019 [8] dataset.

To further remove some of the experimental difficulties for our complete dataset, we have experimented with the individual datasets too. The experiments can be seen in Tables 2 and 3. We have come to understand that there is not a huge performance difference between the models on the two datasets, some of which are further established by the results from Fig 2. Related to the graphical results are the results for the FeaStConv architecture. We have observed that this convolutional operator performs poorly for the triangle index prediction, but outperforms all other models on the barycentric predictions. Furthermore, it is the only architecture that works better with a smaller size

embedding and smaller size channels. Some problems can observed, as there are empty cells in the tables. The model seemed to have caused a memory overflow for the GPU in our environment. Considering this, we had to give up on running some of our experiments.

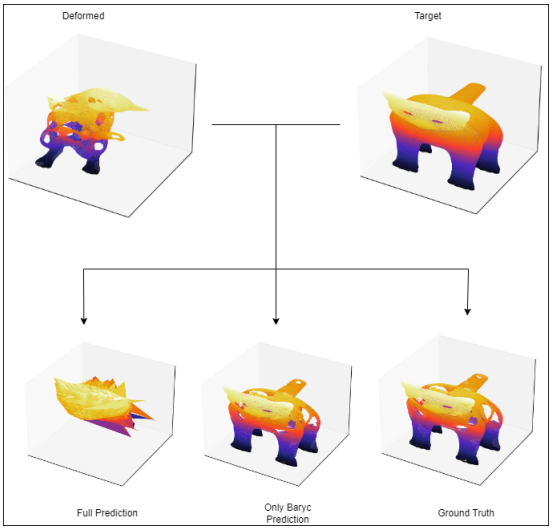


FIGURE 3. An illustration of the predictions for the seventh image with holes type damage from SHREC 2016 [4, 22]. This diagram contains, from left to right, the full prediction of the model after 20 epochs, only the barycentric prediction with correct triangles and the ground truth for this case.

For our final few experiments, one with each adequate model, we have chosen the bigger size architectures. The only exception to this rule is the previously presented model. In Table 4, one can observe the results for what we considered to be the best model. The observations made so far, on the smaller datasets, have remained true. As such, we can consider that the best model architecture is the one using GCNConv, followed by the one using FeaStConv.

To get a better representation of our model's performance, we will now be referring to Fig. 3. Here the reader is free to observe how the model performs when only using the barycentric predictions together with the correct triangle indexes. When using the full output, the shape ends up being clustered to something alike a centre of gravity. This does not mean that all the predictions are wrong, but the wrong prediction can have a huge effect on the model's performance.

Considering only the barycentric part of the output, the model's performance becomes almost indistinguishable from the ground truth. This further indicates that the next steps of our research should be focused on making the model perform better on the triangle index part of the output. Several methods could be used here, such as using an output that gives the indexes of three nodes and only later validating the correctness of this output.

8. Conclusions and further research

In our experiments, we have shown the potential of using various model architectures for deformable object matching. The best overall results were obtained using the GCNConv model with 32 starting hidden channels and an embedding size of 500, for the prediction of the triangle index. In the case of predicting the barycentric weights, the best results were obtained using the FeaStConv model, with of 16 starting hidden channels and an embedding size of 100.

We consider that the results give the right direction to continue developing the GNN model. Seeing that two distinct architectures have given the best results on the two targets of the model, we might want to look more into using them. A combined environment for the two convolutions is not out of the question, nor verifying their hyper-parameters.

Considering the conclusions and every experiment done so far, we have taken into consideration a few possible approaches for further research. Those approaches consist of changes and additions to the model and the dataset.

A possible improvement to the dataset is to consider several basic geometric forms such as cubes, spheres, and cones and then apply random deformations over them. This could show the potential of using synthetic data to train a more robust model.

As mentioned in the very first section, the final scope of this research would be to see if it can match a complex 3D object to a partial reconstruction of the object from a 2D image. This could be done by considering the use of a state-of-the-art depth estimation model such as MiDaS [27,28], which would help us evade the need to use cameras that already have the technology implemented.

Considering the possible new environment, starting from a picture, there will also be the possibility of using a new dataset constructed from that. The dataset could consist of the partial 3D objects given by the depth model and an associated complete 3D object. The only problem with this idea is the need to have more annotated data.

References

[1] Adedigba, A. P., Adeshina, S. A., Aina, O. E., and Aibinu, A. M. Optimal hyperparameter selection of deep learning models for COVID-19 chest x-ray classification. *Intell Based Med 5* (Apr. 2021), 100034.

[2] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[3] Christoffersen, P., and Jacobs, K. The importance of the loss function in option valuation. *Journal of Financial Economics 72*, 2 (2004), 291–318.

[4] Cosmo, L., Rodol, E., Bronstein, M. M., Torsello, A., Cremers, D., and Sahillioglu, Y. Partial Matching of Deformable Shapes. In *Eurographics Workshop on 3D Object Retrieval* (2016), A. Ferreira, A. Giachetti, and D. Giorgi, Eds., The Eurographics Association.

[5] Cosmo, L., Rodol, E., Masci, J., Torsello, A., and Bronstein, M. M. Matching deformable objects in clutter. In *2016 Fourth International Conference on 3D Vision (3DV)* (2016), pp. 1–10.

[6] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2016), NIPS'16, Curran Associates Inc., p. 38443852.

[7] Du, J., Zhang, S., Wu, G., Moura, J. M. F., and Kar, S. Topology adaptive graph convolutional networks. *CoRR abs/1710.10370* (2017).

[8] Dyke, R. M., Stride, C., Lai, Y.-K., Rosin, P. L., Aubry, M., Boyarski, A., Bronstein, A. M., Bronstein, M. M., Cremers, D., Fisher, M., Groueix, T., Guo, D., Kim, V. G., Kimmel, R., Lhner, Z., Li, K., Litany, O., Remez, T., Rodol, E., Russell, B. C., Sahilliolu, Y., Slossberg, R., Tam, G. K. L., Vestner, M., Wu, Z., and Yang, J. Shape correspondence with isometric and non-isometric deformations. In *Eurographics Workshop on 3D Object Retrieval* (2019), S. Biasotti, G. Lavou, and R. Veltkamp, Eds., The Eurographics Association.

[9] Dyke, R. M., Zhou, F., Lai, Y.-K., Rosin, P. L., Guo, D., Li, K., Marin, R., and Yang, J. SHREC 2020 Track: Non-rigid shape correspondence of physically-based deformations. In *Eurographics Workshop on 3D Object Retrieval* (2020), T. Schreck, T. Theoharis, I. Pratikakis, M. Spagnuolo, and R. C. Veltkamp, Eds., The Eurographics Association.

[10] Fey, M., and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).

[11] Gao, X., and Fang, Y. A note on the generalized degrees of freedom under the l1 loss function. *Journal of Statistical Planning and Inference 141*, 2 (2011), 677–686.

[12] Ginzburg, D., and Raviv, D. Dual geometric graph network (dg2n) iterative network for deformable shape alignment. In *2021 International Conference on 3D Vision (3DV)* (Dec. 2021), IEEE.

[13] Gonzlez Izard, S., Snchez Torres, R., Alonso Plaza, ., Juanes Mndez, J. A., and Garca-Pealvo, F. J. Nextmed: Automatic imaging segmentation, 3d reconstruction, and 3d model visualization platform using augmented and virtual reality. *Sensors 20*, 10 (2020).

[14] Grozdev, S., and Dekov, D. Barycentric coordinates: Formula sheet. *International Journal of Computer Discovered Mathematics 1* (03 2016), 72–82.

[15] Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 10251035.

[16] Joo, H., Jeong, Y., Duchenne, O., and Kweon, I. S. Graph-based shape matching for deformable objects. In *2011 18th IEEE International Conference on Image Processing* (2011), pp. 2901–2904.

[17] Kim, J., and Shontz, S. M. An improved shape matching algorithm for deformable objects using a global image feature. In *Advances in Visual Computing* (Berlin, Heidelberg, 2010), G. Bebis, R. Boyle, B. Parvin, D. Koracin, R. Chung, R. Hammound, M. Hussain, T. Kar-Han, R. Crawfis, D. Thalmann, D. Kao, and L. Avila, Eds., Springer Berlin Heidelberg, pp. 119–128.

[18] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *CoRR abs/1609.02907* (2016).

[19] Lazarou, M., Li, B., and Stathaki, T. A novel shape matching descriptor for real-time static hand gesture recognition. *Computer Vision and Image Understanding 210* (2021), 103241.

[20] Li, Y., Wang, Y., Case, M., Chang, S.-F., and Allen, P. K. Real-time pose estimation of deformable objects using a volumetric approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), pp. 1046–1052.

[21] Lian, Y., and Chen, M. Ca-cgnet: Component-aware capsule graph neural network for non-rigid shape correspondence. *Applied Sciences 13*, 5 (2023).

[22] Lhner, Z., Rodol, E., Bronstein, M. M., Cremers, D., Burghard, O., Cosmo, L., Dieckmann, A., Klein, R., and Sahillioglu, Y. Matching of Deformable Shapes with Topological Noise. In *Eurographics Workshop on 3D Object Retrieval* (2016), A. Ferreira, A. Giachetti, and D. Giorgi, Eds., The Eurographics Association.

[23] Marin, R., Rakotosaona, M.-J., Melzi, S., and Ovsjanikov, M. Correspondence learning via linearly-invariant embedding, 2020.

[24] Muthukumar, V., Narang, A., Subramanian, V., Belkin, M., Hsu, D., and Sahai, A. Classification vs regression in overparameterized regimes: Does the loss function matter? *J. Mach. Learn. Res. 22*, 1 (jan 2021).

[25] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[26] Prisacariu, V. A., and Reid, I. D. Pwp3d: Real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision 98*, 3 (Jul 2012), 335–354.

[27] Ranftl, R., Bochkovskiy, A., and Koltun, V. Vision transformers for dense prediction. *ArXiv preprint* (2021).

[28] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

[29] Rodol, E., Bronstein, A. M., Albarelli, A., Bergamasco, F., and Torsello, A. A game-theoretic approach to deformable shape matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 182–189.

[30] Sarlin, P., DeTone, D., Malisiewicz, T., and Rabinovich, A. Superglue: Learning feature matching with graph neural networks. *CoRR abs/1911.11763* (2019).

[31] Schulman, J., Lee, A., Ho, J., and Abbeel, P. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 1130–1137.

[32] Veltkamp, R. Shape matching: similarity measures and algorithms. In *Proceedings International Conference on Shape Modeling and Applications* (2001), pp. 188–197.

[33] Verma, N., Boyer, E., and Verbeek, J. Dynamic filters in graph convolutional networks. *CoRR abs/1706.05206* (2017).

[34] Wu, Y., Yan, W., Kurutach, T., Pinto, L., and Abbeel, P. Learning to manipulate deformable objects without demonstrations. In *Robotics Science and Systems* (07 2020), pp. 65–76.

[35] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems 32*, 1 (2021), 4–24.

[36] Yang, L., and Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing 415* (2020), 295–316.

[37] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open 1* (2020), 57–81.

Department of Computer Science, Babes-Bolyai University, 1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

Email address: mihai.loghin@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.03

A COMPREHENSIVE EVALUATION OF ROUGH SETS CLUSTERING IN UNCERTAINTY DRIVEN CONTEXTS

ARNOLD SZEDERJESI-DRAGOMIR

Abstract. This paper presents a comprehensive evaluation of the Agent BAsed Rough sets Clustering (ABARC) algorithm, an approach using rough sets theory for clustering in environments characterized by uncertainty. Several experiments utilizing standard datasets are performed in order to compare ABARC against a range of supervised and unsupervised learning algorithms. This comparison considers various internal and external performance measures to evaluate the quality of clustering. The results highlight the ABARC algorithm's capability to effectively manage vague data and outliers, showcasing its advantage in handling uncertainty in data. Furthermore, they also emphasize the importance of choosing appropriate performance metrics, especially when evaluating clustering algorithms in scenarios with unclear or inconsistent data.

1. Introduction

Clustering algorithms play an important role in uncovering patterns and structures from unlabelled data across several scientific and engineering domains [5, 12, 16, 14, 7]. The added value of these algorithms lies in their ability to group data points based on underlying similarities, thereby facilitating a deeper understanding of dataset characteristics without prior knowledge of the group identities. In real-world contexts where uncertainty and ambiguity often pervade, the ability to discern coherent groups within a dataset becomes indispensable. However, traditional clustering techniques are often inadequate in environments driven by uncertainty, including the presence of hybrid data (vague data or outliers). This limitation shows the necessity for innovative approaches that can robustly handle the complexities induced by the uncertainty and ambiguity of such landscapes.

Received by the editors: 5 March 2024.

2010 *Mathematics Subject Classification.* 68T37, 68T99.

1998 *CR Categories and Descriptors.* I.5.3 [**Pattern Recognition**]: Clustering – *Algorithms*; I.5.3 [**Pattern Recognition**]: Clustering – *Similarity measures* .

Key words and phrases. rough sets, clustering, metrics.

The introduction of rough sets theory by Pawlak [28] has facilitated the development of clustering algorithms capable of handling uncertainty more effectively. Rough sets have mostly been applied to feature extraction [2, 17, 22, 27, 35, 19, 37, 33, 4], their use in direct cluster modeling being significantly less common. The approaches from [21, 25, 24, 20] investigate rough sets clustering, but they are all partitioning methods. ABARC [11], on the other hand, is a hierarchical clustering algorithm that distinguishes itself by its adeptness at detecting hybrid data by using rough sets, as well as isolating outliers, thereby promising enhanced clustering performance in scenarios driven by uncertainty in data.

The evaluation of clustering algorithms in the context of uncertainty-driven environments needs to take into account the particularities detected in data. Internal and external performance metrics serve as critical tools in this process, providing insights into an algorithm's ability to generate cohesive and well-separated clusters while aligning with external validity measures when ground truth is available.

This paper aims to perform a comprehensive comparison of the ABARC algorithm against several supervised and unsupervised learning algorithms, employing a suite of performance metrics to assess each algorithm's efficacy across standard datasets. Through this comparative analysis, our aim is to show the strengths and limitations of the ABARC algorithm and its counterparts, thereby contributing to the ongoing research on optimal clustering approaches in the context of data uncertainty.

The paper is structured as follows: Section 2 presents an overview of the clustering algorithm based on rough sets, Section 3 illustrates the comprehensive experiments we made, including evaluation based on external, internal and rough metrics and Section 4 draws the conclusions of this paper and presents potential future work.

2. Rough sets clustering

Rough sets [28] represent an effective methodology for addressing data uncertainty and vagueness, without the need of membership functions (which could be hard to build) like in fuzzy set theory. Employing an equivalence relation R within a dataset U, rough set theory proposes a mechanism to approximate uncertain subsets $X \subseteq U$ via two distinct and precise sets: the lower and upper approximations. The lower approximation is comprised of elements that are surely in X and it is defined as $R^{\downarrow}(X) = \{x \in U : [x]_R \subseteq X\}$, where $[x]_R$ represents the equivalence class of x under R. Conversely, the upper approximation includes elements that possibly belong to X, defined as $R^{\uparrow}(X) = \{x \in U : [x]_R \cap X \neq \emptyset\}$. The boundary region, delineated as

$Bnd_R(X) = R^{\uparrow}(X) - R^{\downarrow}(X)$, consists of objects that cannot be definitively classified as belonging or not belonging to the subset X. Accordingly, the rough set of X relative to R is denoted as $RS(X) = \{R^{\downarrow}(X), R^{\uparrow}(X)\}$.

Rough sets clustering [11] uses rough sets theory to effectively group a dataset into clusters while acknowledging the existing uncertainties and ambiguities in data.

Definition 1. *Given a dataset U (universe of discourse) and an equivalence relation R on U, the goal of **rough sets clustering** is to partition U into a set of clusters $\{C_1, C_2, \ldots, C_k\}$ such that:*

- $U = \bigcup_{i=1}^{k} C_i$ *and* $C_i \cap C_j = \emptyset$ *for* $i \neq j$.
- *Each cluster C_i is represented by its lower and upper approximations ($R^{\downarrow}(C_i)$, $R^{\uparrow}(C_i)$) with respect to R.*
- *The boundary region for each cluster C_i is given by $Bnd_R(C_i) = R^{\uparrow}(C_i) - R^{\downarrow}(C_i)$.*

In the context of clustering: (1) objects in the lower approximation of a cluster *definitively* belong to that cluster; (2) objects in the upper approximation *might* belong to the cluster (3) objects in the boundary region of a cluster may belong to the boundary regions of other clusters.

In Algorithm 1 we show an overview of the ABARC rough sets clustering algorithm from [11].

Algorithm 1 Rough Sets Clustering

Require: X (dataset), $imax$ (number of trials), λ (similarity limit)
Initialize AG (set of agents) with one agent for each instance in X.
For each agent in AG, assign it to a unique cluster.
for $i = 1$ to $imax$ **do**
  for each $agent_k$ in AG **do**
    Find a similar agent (sa_k) using a similarity threshold λ.
    if sa_k is found **then**
      Move $agent_k$ to the cluster of sa_k.
    end if
  end for
end for
for each cluster representative R_k **do**
  Find similar clusters based on a rough similarity limit.
  Update and unify clusters based on similarity.
end for
Handle outliers by assigning them to the closest cluster.

The algorithm receives a dataset X, along with a maximum trial limit $imax$ and a similarity threshold λ, as inputs. In this appproach, each element of X is represented by an agent and the set of all agents is denoted with AG. Initially, every agent is allocated to a distinct cluster, leading to a total of n clusters corresponding to n agents.

Iteratively, up to $imax$ rounds, the algorithm refines the clustering structure by allowing each agent $agent_k$ to seek peers within the similarity boundary set by λ. Upon identifying a similar agent sa_k, $agent_k$ relocates to sa_k's cluster, thus similar agents are grouped together.

Based on the representative of each cluster, the algorithm checks the similarity among clusters through these representatives. If any clusters are similar, they are merged into a unified cluster. This step ensures that clusters that are close to each other or have significant overlap can be combined to form a more cohesive and meaningful cluster. It is possible for a representative to be similar to more than one other cluster representative in which case the corresponding data (rough instances) will be treated as it would belong to several clusters.

An optional phase addresses outliers - agents that do not seem to fit into any other cluster - by assigning them to the nearest cluster, thus ensuring that all data points are included in a cluster.

3. Experimental evaluation

In this section we calculate various external (Section 3.1), internal (Section 3.2), as well as rough (Section 3.3) metrics as we compare the ABARC approach with several other algorithms in the literature. The ABARC algorithm is compared with other approaches in three scenarios: including all instances, eliminating only outliers and eliminating both outliers and rough instances (i.e. eliminating all hybrid data).

The experiments are performed on the following datasets: Iris [8], Seeds [18], and Wine [9]. These datasets have been chosen primarily for benchmarking purposes and, secondly, because they present challenges such as the presence of outliers and instances that are not linearly separable, which makes them suitable for applying the ABARC algorithm.

3.1. **External evaluation metrics.**

3.1.1. *Metrics.*

- Accuracy - in a clustering context, it represents the percentage of instances that were correctly predicted out of all instances

- Precision - focuses on how many predicted instances were classified correctly for a given class:

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

- Recall - focuses on how many actual instances were predicted correctly for a given class:

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

- F1-Score - incorporates both precision and recall using harmonic mean (thus punishing extreme values) with even weights, this metric is also for a given class:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Macro F1-Score - combines the F1 Score from each class using arithmetic mean (this can be applied to precision and recall metrics too):

$$Macro\ F_1 = \frac{1}{|C|} * \sum_{c \in C} F_{1_c}$$

  where C is the set of classes, and c is a class
- Weighted Average F1-Score - same as Macro F1-Score but with class sizes used as weights (again this can be used for precision and recall too):

$$Weighted\ Average\ F_1 = \frac{1}{|C|} * \sum_{c \in C} |c| * F_{1_c}$$

- Micro F1-Score (Accuracy) - to calculate this we take all the samples together and compute precision and recall, and then the F1 Score. In cases where the number of predicted instances is equal to the actual instances, we will have the following equation hold:

  $Micro\ Precision = Micro\ Recall = Micro\ F_1 = Accuracy$

- Kappa Score - when talking about Kappa Score we need to introduce new terms: *Agree* which is the proportion of correctly predicted instance over all instances (similar to Accuracy) and *Chance Agree* which is computed from the probabilities of predicting a class or being in a class, formally:

$$Agree = \frac{1}{N} * \sum_{c \in C} |predicted_c \cap actual_c|$$

TABLE 1. Overall supervised metrics for the Iris dataset.

Case Study	Class	Instances	Prec	Recall	F1	Kappa
Clusters with hybrids	Macro	163	91.19	91.19	91.15	-
	Weighted	163	90.8	90.88	90.8	-
	Micro	163	90.798	90.798	90.798	-
	Kappa	163	-	-	-	86.2
Clusters without outliers	Macro	152	90.64	90.74	90.64	-
	Weighted	152	90.13	90.21	90.12	-
	Micro	152	90.132	90.132	90.132	-
	Kappa	152	-	-	-	85.1
Clusters without hybrids	Macro	126	98.35	98.35	98.35	-
	Weighted	126	98.41	98.41	98.41	-
	Micro	126	98.413	98.413	98.413	-
	Kappa	126	-	-	-	97.6

where N is the total number of instances, $predicted_c$ is the set of all instances predicted in class c, and $actual_c$ is the set of all instances actually being in class c.

$$Chance\ Agree = \sum_{c \in C} \frac{|predicted_c|}{N} * \frac{|actual_c|}{N}$$

$$Kappa\ Score = \frac{Agree - Chance\ Agree}{1 - Chance\ Agree}$$

3.1.2. *Results and discussion.*

Iris dataset. Analyzing the results from Table 1 we can see that outliers make no real difference, but when we eliminate rough instances we get much better results on all metrics.

We have also compared our metrics to some related work. We have used as a comparison the following results from [36]: KMEA, WKME, EWKM, ESSC, AFKM, SC, SSC-MP, ERKM; and from [29]: Bayes Network Classifier, J48, Random Forest, OneR. In Table 2 we can see that the F1-Score for the two ABARC cases is better than all of the others, but the Kappa Score is better only after removing hybrids.

We have also compared to algorithms from Scikit learn [38] like Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Decision Tree (DT), Gaussian Naive Bayes (GNB), Support Vector Machines (SVM). From Table 3 we can again conclude that removing hybrids is essential to have the best values for all the metrics.

TABLE 2. Comparison with related work on Iris dataset.

Algorithm	Precision	Recall	F1-Score	Kappa Score
KMEA [23]			81.2	
WKME [13]			79.8	
EWKM [15]			82.6	
ESSC [6]			84.8	
AFKM [1]			81.6	
SC [31]			47.2	
SSC-MP [32]			76.7	
ERKM [36]			90.2	
Bayes Network Classifier				89
J48				94
Random Forest				93
OneR				91
ABARC /w hybrids	91.2	91.2	91.2	86.2
ABARC /wo hybrids	**98.4**	**98.4**	**98.4**	**97.6**

TABLE 3. Comparison with Scikit learn on Iris dataset.

Algorithm	Precision	Recall	F1-Score	Kappa Score
LR	95.4	95.2	95.2	92.9
LDA	98.2	97.9	97.9	96.9
KNN	97.8	97.8	97.7	97.0
DT	95.3	95.1	95.1	92.9
GNB	95.7	95.5	95.5	92.8
SVM	97.8	97.7	97.7	96.9
ABARC /w hybrids	91.2	91.2	91.2	86.2
ABARC /wo hybrids	**98.4**	**98.4**	**98.4**	**97.6**

Seeds dataset. When taking a look in Table 4, the overall metrics show small difference when outliers are removed, and eliminating rough instances does not seem to make any difference. A potential reason why hybrid data might have such a small impact is the reduced number of outliers and rough instances.

We have used the same algorithm from Scikit learn. When taking a look on Table 5 we can observe that our approach is nowhere near being the best. This can once more happen because the algorithm's performance is not affected by rough instances and outliers.

Wine dataset. Wine is one of the datasets where hybrids make difference. The idea can be observed when we take a look at the overall metrics in Table 6: outliers improve metrics, but rough instances are the ones that make the real difference bumping all metrics to above 99%.

TABLE 4. Overall supervised metrics for the Seeds dataset.

Case Study	Class	Instances	Prec	Recall	F1	Kappa
Clusters with hybrids	Macro	213	91.52	91.61	91.5	-
	Weighted	213	91.55	91.84	91.63	-
	Micro	213	91.549	91.549	91.549	-
	Kappa	213	-	-	-	87.3
Clusters without outliers	Macro	192	91.13	91.59	91.28	-
	Weighted	192	91.15	91.42	91.21	-
	Micro	192	91.146	91.146	91.146	-
	Kappa	192	-	-	-	86.7
Clusters without hybrids	Macro	178	91.46	92.23	91.77	-
	Weighted	178	91.57	91.79	91.62	-
	Micro	178	91.573	91.573	91.573	-
	Kappa	178	-	-	-	87.2

TABLE 5. Comparison with Scikit learn on Seeds dataset.

Algorithm	Precision	Recall	F1-Score	Kappa Score
LR	90.0	90.2	89.5	84.9
LDA	**95.9**	**96.2**	**96.0**	**94.2**
KNN	91.7	92.0	91.4	87.7
DT	88.3	88.5	88.1	82.7
GNB	89.7	90.1	89.5	84.9
SVM	92.6	92.7	92.5	89.2
ABARC /w hybrids	91.5	91.6	91.5	87.3
ABARC /wo hybrids	91.5	92.2	91.8	87.2

TABLE 6. Overall supervised metrics for the Wine dataset.

Case Study	Class	Instances	Prec	Recall	F1	Kappa
Clusters with hybrids	Macro	186	93.2	94.79	93.59	-
	Weighted	186	93.55	94.3	93.51	-
	Micro	186	93.548	93.548	93.548	-
	Kappa	186	-	-	-	90.2
Clusters without outliers	Macro	165	94.23	95.31	94.32	-
	Weighted	165	94.55	95.31	94.49	-
	Micro	165	94.545	94.545	94.545	-
	Kappa	165	-	-	-	91.8
Clusters without hybrids	Macro	148	99.24	99.29	99.26	-
	Weighted	148	99.32	99.34	99.32	-
	Micro	148	99.324	99.324	99.324	-
	Kappa	148	-	-	-	99.0

We have done the one of the comparisons from Iris on the Wine dataset too, illustrated in Table 7. As there is only one metric interpreting the results

TABLE 7. Comparison with related work on Wine dataset.

Algorithm	F1-Score
KMEA [23]	94.6
WKME [13]	94.8
EWKM [15]	90.4
ESSC [6]	95.0
AFKM [1]	94.3
SC [31]	86.9
SSC-MP [32]	58.4
ERKM [36]	89.9
ABARC /w hybrids	93.6
ABARC /wo hybrids	**99.3**

TABLE 8. Comparison with Scikit learn on Wine dataset.

Algorithm	Precision	Recall	F1-Score	Kappa Score
LR	95.0	96.1	95.3	93.1
LDA	97.6	97.8	97.6	96.5
KNN	66.4	68.2	65.9	51.5
DT	89.6	89.2	89.2	84.3
GNB	97.5	97.8	97.6	96.5
SVM	60.7	39.5	30.4	9.5
ABARC /w hybrids	93.2	94.8	93.6	90.2
ABARC /wo hybrids	**99.2**	**99.3**	**99.3**	**99.0**

is trivial and we can see the same tendency: ABARC with hybrids has good performance but not good enough to be better than all of the other related work, but when we remove hybrids the value becomes almost perfect, thus being the best of all.

We have the comparison with Scikit learn algorithms on the Wine dataset too. Table 8 shows that the performance of ABARC is comparable to that of the related work, albeit slightly lower. Nevertheless, the advantage of our approach is that it can also detect and isolate hybrid data.

3.2. **Internal evaluation metrics.**

3.2.1. *Metrics.*

- Purity - a measure of the extent to which clusters contain a single class:

$$Purity = \frac{1}{N} \sum_{k \in K} max_{c \in C}\ a_{ck}$$

where N is the number of instances, K is the set of clusters, and C is the set of classes, and $a_{ck} = |c \cap k|$

- Entropy - a measure of uncertainty

$$Entropy = \sum_{k \in K} \frac{|k|}{N} * (-\sum_{c \in C} P(a_{ck}) * \log_2 P(a_{ck}))$$

where

$$P(a_{ck}) = \frac{a_{ck}}{|c|} = \frac{|c \cap k|}{|c|}$$

- V-Measure - is again based on entropy but considers homogeneity and completeness with different importance. Homogeneity means that a clustering must assign only those datapoints that are members of a single class to a single cluster, completeness is symmetrical to homogeneity: a clustering must assign all of those datapoints that are members of a single class to a single cluster. Formally we calculate homogeneity, completeness and V-Measure as:

$$H(C|K) = -\sum_{k \in K} \sum_{c \in C} \frac{a_{ck}}{N} * \log \frac{a_{ck}}{\sum_{c \in C} a_{ck}}$$

$$H(C,K) = -\sum_{c \in C} \frac{\sum_{k \in K} a_{ck}}{|C|} \log \frac{\sum_{k \in K} a_{ck}}{|C|}$$

$$h = \begin{cases} 1 & H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C,K)} & \text{otherwise} \end{cases}$$

$$H(K|C) = -\sum_{c \in C} \sum_{k \in K} \frac{a_{ck}}{N} * \log \frac{a_{ck}}{\sum_{k \in K} a_{ck}}$$

$$H(K,C) = -\sum_{k \in K} \frac{\sum_{c \in C} a_{ck}}{|C|} \log \frac{\sum_{c \in C} a_{ck}}{|C|}$$

$$c = \begin{cases} 1 & H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K,C)} & \text{otherwise} \end{cases}$$

$$V_\beta = \frac{(1+\beta) * h * c}{\beta * h + c}$$

3.2.2. *Results and discussion.* Considering the results from Table 9 we can observe that on Iris the accuracy and purity drops a bit as we eliminate outliers and rough instances, but the entropy and the V-Measure, after dropping both of them, are significantly better, which makes us assume that outliers and rough instances do not really affect homogeneity but they affect completeness. On the Seeds dataset we cannot see any real difference when eliminating them, thus they do not affect our performance. Finally, on Wine we can see all metrics improve, entropy and V-Measure improve significantly, so on this

TABLE 9. Unsupervised performance measurements for the Iris, Seeds and Wine datasets.

	Case Study	Inst	Acc	Entropy	Purity	V
Iris	Clusters with hybrids	150	98.66%	0.0803	0.987	0.733
	Clusters without outliers	139	98.56%	0.0847	0.986	0.717
	Clusters without outliers and rough	126	98.41%	0.0204	0.984	0.932
Seeds	Clusters with hybrids	210	92.857%	0.0839	0.929	0.721
	Clusters without outliers	190	92.105%	0.0863	0.921	0.711
	Clusters without outliers and rough	178	91.573%	0.0829	0.916	0.719
Wine	Clusters with hybrids	178	97.753%	0.0569	0.978	0.8
	Clusters without outliers	157	99.363%	0.0418	0.994	0.854
	Clusters without outliers and rough	148	99.324%	0.0088	0.993	0.97

dataset eliminating them makes our results almost perfect regardless of the metric used.

To compare with some related work we used the following results:

- KMEA, WKME, EWKM, ESSC, AFKM, SC, SSC-MP, ERKM [36]
- Bayes Network Classifier, J48, Random Forest, OneR [29]
- KM, EWKM, AFKM, FCM, SCAD, Entropy-based Variable Feature Weighted Fuzzy k-Means (EVFWFKM) [30]
- UFT-k-means, k-prototypes, Improved k-prototypes, KL-FCM-GM [34]

These are used in Table 10, and there can be multiple entries for a single algorithm (ex. EWKM) as they are taken from different results probably run using different configuration. The first comparison from Table 10 again suggests that ABARC has the performance, this time on Seeds too. Although both the compared metrics are the best in our approach, on the dataset Iris and Seeds the accuracy is actually better with hybrids than without them (this can happen when hybrid instances are accidentally put in the cluster specified by the official documentation), but the entropy values are always better without hybrids. Other algorithms does not seem to be even close to the values reported by ABARC in any of the cases.

When we compare to the Scikit learn algorithms we have the same results as for the supervised metrics. They are much better on Seeds dataset, but our approach especially without hybrids has much better performance on Iris and Wine dataset from both metrics' point of view.

3.3. **Rough evaluation metrics.**

3.3.1. *Metrics.* We have also evaluated the ABARC algorithm against the following rough indices from [26]:

TABLE 10. Unsupervised comparison with related work for the Iris, Seeds and Wine datasets.

Dataset	Algorithm	Accuracy	Entropy
Iris	KMEA [23]	80.54%	
	WKME [13]	78.47%	
	EWKM [15]	82.09%	
	ESSC [6]	84.66%	
	AFKM [1]	81.27%	
	SC [31]	80.66%	
	SSC-MP [32]	71.20%	
	ERKM [36]	90.36%	
	Bayes Network Classifier	92.66%	
	J48	96%	
	Random Forest	95.33%	
	OneR	94%	
	KM [23]	88.67%	
	EWKM [15]	89.78%	
	AFKM [1]	90.67%	0.299
	FCM [3]	82.67%	0.395
	SCAD [10]	88.67%	0.395
	EVFWFKM [30]	92.67%	0.294
	ABARC /w hybrids	**98.66%**	0.08
	ABARC /wo hybrids	98.41%	**0.02**
Seeds	UFT-k-means	89.05%	
	k-prototypes	86.67%	
	Improved k-prototypes	84.76%	
	KL-FCM-GM	57.62%	
	ABARC /w hybrids	**92.857%**	0.084
	ABARC /wo hybrids	91.573%	**0.083**
Wine	KMEA [23]	94.43%	
	WKME [13]	94.71%	
	EWKM [15]	90.24%	
	ESSC [6]	95.06%	
	AFKM [1]	93.99%	
	SC [31]	87.07%	
	SSC-MP [32]	58.65%	
	ERKM [36]	90.16%	
	ABARC /w hybrids	97.753%	0.057
	ABARC /wo hybrids	**99.324%**	**0.009**

(1) Average Accuracy, α index - it is the average of the ratio of the number of objects in lower approximation to that in upper approximation of each cluster, it captures the average degree of completeness of knowledge about all clusters: $\alpha = \frac{1}{|K|} \sum_{k \in K} \frac{\omega * A_k}{\omega * A_k + (1-\omega) * B_k}$ where A_k is the size of the lower approximation of cluster k, B_k is the size

TABLE 11. Unsupervised comparison with Scikit learn algorithms for the Iris, Seeds and Wine datasets.

Dataset	Algorithm	Accuracy	V-Measure
Iris	LR	94.67	86.5
	LDA	97.33	91.7
	KNN	96.67	89.9
	DT	95.33	86.7
	GNB	95.33	87.2
	SVM	96.00	88.6
	ABARC /w hybrids	**98.66**	73.3
	ABARC /wo hybrids	98.41	**93.2**
Seeds	LR	90.00	73.6
	LDA	**96.19**	**88.1**
	KNN	91.90	77.3
	DT	88.10	67.7
	GNB	90.00	73.4
	SVM	92.86	78.5
	ABARC /w hybrids	92.857	72.1
	ABARC /wo hybrids	91.573	71.9
Wine	LR	95.51	85.1
	LDA	98.32	94.2
	KNN	69.70	39.8
	DT	90.46	75.9
	GNB	96.59	90.4
	SVM	38.19	9.7
	ABARC /w hybrids	97.753	80.0
	ABARC /wo hybrids	**99.324**	**97.0**

of the boundary region of cluster k and ω is the weight of lower approximation (we used 0.6)

(2) Average Roughness, ρ index - represents the average degree of incompleteness of knowledge about all clusters: $\rho = 1 - \alpha$

(3) Accuracy of Approximation, α^* index - it captures the exactness of approximate clustering: $\alpha^* = \frac{\sum_{k \in K} \omega * A_k}{\sum_{k \in K} \omega * A_k + (1-\omega) * B_k}$

(4) Quality of Approximation, γ index - it is the ratio of the total number of objects in lower approximations of all clusters to the cardinality of the universe of discourse: $\gamma = \frac{1}{N} \sum_{k \in K} A_k$

3.3.2. *Results and discussion.* We have compared our rough indices results with the ones reported in the book in Table 12 (only on Iris and Wine datasets). From the comparison we can say that our approach matches the algorithms discussed in the related work on the Iris dataset. The first three indices are slightly lower but the last one is significantly better, probably meaning that

TABLE 12. Rough indices for the Iris, Seeds and Wine dataset.

Dataset	Iris				Wine			
Algorithm	α Index	ρ Index	α^* Index	γ Index	α Index	ρ Index	α^* Index	γ Index
$RFCM^{MBP}$	0.999971	0.000029	0.999963	0.625000	0.8387	0.1613	0.9251	0.5000
RFCM	0.999986	0.000014	0.999988	0.800000	0.8918	0.1082	0.9259	0.8275
RPCM	0.999983	0.000017	0.999985	0.553333	0.8433	0.1567	0.9306	0.6255
RFPCM	**0.999987**	**0.000013**	**0.999989**	0.766667	0.9012	0.0988	0.9258	0.7234
ABARC	0.999980	0.000020	0.999981	**0.913333**	**0.9989**	**0.0011**	**0.9989**	**0.9438**

overall our rough score is better but it is slightly worse on one of the clusters. On the Wine dataset all indices are the best in the ABARC algorithm's case, meaning that we have better rough clustering from all points of view.

4. Conclusions and future work

In this paper, we have carried out a comprehensive evaluation of the Agent BAsed Rough sets Clustering (ABARC) algorithm, which is a new approach for clustering in uncertain environments. Experiments were done using standard datasets and against multiple supervised and unsupervised methods too. Besides this evaluation, we also analyze the impact of several internal and external metrics, especially in the context of unpredictability.

The results suggest that removing hybrids increases the performance of the ABARC algorithm. Compared to other approaches on Iris and Wine datasets the algorithm outperforms all the related approaches with respect to almost any of the considered metrics. This outcome emphasises the importance of hybrid data detection and hence the need of applying algorithms that are tailored to uncertainty driven environments.

As a future work we plan to analyze rough instances and outliers even more in order to potentially gain extra relevant information about the given datasets as well as work on applying ABARC in other domains, like software engineering, biology, chemistry or even medicine.

References

[1] Bachem, O., Lucic, M., Hassani, H., and Krause, A. Fast and provably good seedings for k-means. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 55–63.

[2] Bera, S., Giri, P. K., Jana, D. K., Basu, K., and Maiti, M. Multi-item 4d-tps under budget constraint using rough interval. *Applied Soft Computing 71* (2018), 364 – 385.

[3] Bezdek, J. C., Ehrlich, R., and Full, W. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences 10*, 2-3 (Jan. 1984), 191–203.

[4] Bharadwaj, A., and Ramanna, S. Categorizing relational facts from the web with fuzzy rough sets. *Knowledge and Information Systems 61*, 3 (Dec 2019), 1695–1713.

[5] Coy, S., Czumaj, A., and Mishra, G. On parallel k-center clustering. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures* (New York, NY, USA, 2023), SPAA '23, Association for Computing Machinery, p. 65–75.

[6] Deng, Z., Choi, K.-S., Chung, F.-L., and Wang, S. Enhanced soft subspace clustering integrating within-cluster and between-cluster information. *Pattern Recognition 43*, 3 (Mar. 2010), 767–781.

[7] Fielding-Singh, P., and Fan, J. X. Dietary patterns among us children: A cluster analysis. *Journal of the Academy of Nutrition and Dietetics* (2023).

[8] Fisher, R. A. *UCI Machine Learning Repository: Iris Data Set.* http://archive.ics.uci.edu/ml/datasets/Iris, 1936.

[9] Forina, M. *UCI Machine Learning Repository: Wine Data Set.* https://archive.ics.uci.edu/ml/datasets/wine, 1991.

[10] Frigui, H., and Nasraoui, O. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition 37*, 3 (Mar. 2004), 567–581.

[11] Găceanu, R. D., Szederjesi-Dragomir, A., Pop, H. F., and Sârbu, C. Abarc: An agent-based rough sets clustering algorithm. *Intelligent Systems with Applications 16* (2022), 200117.

[12] Hong, J., and Kim, S.-W. C-affinity: A novel similarity measure for effective data clustering. In *Companion Proceedings of the ACM Web Conference 2023* (New York, NY, USA, 2023), WWW '23 Companion, Association for Computing Machinery, p. 41–44.

[13] Huang, J., Ng, M., Rong, H., and Li, Z. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 5 (May 2005), 657–668.

[14] Janowski, A. M., Ravellette, K. S., Insel, M., Garcia, J. G., Rischard, F. P., and Vanderpool, R. R. Advanced hemodynamic and cluster analysis for identifying novel rv function subphenotypes in patients with pulmonary hypertension. *The Journal of Heart and Lung Transplantation* (2023).

[15] Jing, L., Ng, M. K., and Huang, J. Z. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge and Data Engineering 19*, 8 (Aug. 2007), 1026–1041.

[16] Karim, S. M., Habbal, A., Hamouda, H., and Alaidaros, H. A secure multifactor-based clustering scheme for internet of vehicles. *Journal of King Saud University - Computer and Information Sciences 35*, 10 (2023), 101867.

[17] Kato, Y., Saeki, T., and Mizuno, S. Considerations on the principle of rule induction by strim and its relationship to the conventional rough sets methods. *Applied Soft Computing 73* (2018), 933 – 942.

[18] Kulczycki, P. *UCI Machine Learning Repository: Seeds Data Set.* https://archive.ics.uci.edu/ml/datasets/seeds, 2012.

[19] Lei, L. Wavelet neural network prediction method of stock price trend based on rough set attribute reduction. *Applied Soft Computing 62* (2018), 923 – 932.

[20] Li, Y., Fan, J.-c., Pan, J.-S., Mao, G.-h., and Wu, G.-k. A novel rough fuzzy clustering algorithm with a new similarity measurement. *Journal of Internet Technology 20*, 4 (2019), 1145–1156.

[21] Lingras, P., and West, C. Interval set clustering of web users with rough k-means. *J. Intell. Inf. Syst. 23*, 1 (2004), 5–16.

[22] Liu, Y., Qin, K., and Martinez, L. Improving decision making approaches based on fuzzy soft sets and rough soft sets. *Applied Soft Computing 65* (2018), 320 – 332.
[23] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (Berkeley, Calif., 1967), University of California Press, pp. 281–297.
[24] Maji, P., and Pal, S. *Rough-fuzzy pattern recognition: applications in bioinformatics and medical imaging*, vol. 3. John Wiley & Sons, 2012.
[25] Maji, P., and Pal, S. K. Rough set based generalized fuzzy -means algorithm and quantitative indices. *Trans. Sys. Man Cyber. Part B 37*, 6 (2007), 1529–1540.
[26] MAJI, P., and PAL, S. K. *ROUGH-FUZZY PATTERN RECOGNITION*. Wiley, 2012.
[27] Pamucar, D., Stevic, Z., and Zavadskas, E. K. Integration of interval rough ahp and interval rough mabac methods for evaluating university web pages. *Applied Soft Computing 67* (2018), 141 – 163.
[28] Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.
[29] Rajhasthan, Sharma, K., and College, S. Classification of iris dataset using wekas, Dec 2019.
[30] Singh, V., and Verma, N. K. An entropy-based variable feature weighted fuzzy k-means algorithm for high dimensional data. *arXiv preprint arXiv:1912.11209* (2019).
[31] Tarn, C., Zhang, Y., and Feng, Y. Sampling clustering. *CoRR abs/1806.08245* (2018).
[32] Tschannen, M., and Bolcskei, H. Noisy subspace clustering via matching pursuits. *IEEE Transactions on Information Theory 64*, 6 (June 2018), 4081–4104.
[33] Wang, P.-C., Su, C.-T., Chen, K.-H., and Chen, N.-H. The application of rough set and mahalanobis distance to enhance the quality of osa diagnosis. *Expert Systems with Applications 38*, 6 (2011), 7828 – 7836.
[34] Wei, M., Chow, T. W., and Chan, R. H. Clustering heterogeneous data with k-means by mutual information-based unsupervised feature transformation. *entropy 17*, 3 (2015), 1535–1548.
[35] Xie, X., Qin, X., Yu, C., and Xu, X. Test-cost-sensitive rough set based approach for minimum weight vertex cover problem. *Applied Soft Computing 64* (2018), 423 – 435.
[36] Xiong, L., Wang, C., Huang, X., and Zeng, H. An entropy regularization k-means algorithm with a new measure of between-cluster distance in subspace clustering. *Entropy 21*, 7 (July 2019), 683.
[37] Yang, H.-H., and Wu, C.-L. Rough sets to help medical diagnosis - evidence from a taiwan's clinic. *Expert Systems with Applications 36*, 5 (2009), 9293 – 9298.
[38] https://scikit-learn.org.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Computer Science Department, Cluj-Napoca, România

Email address: arnold.szederjesi@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.04

COMPETITIVE INFLUENCE MAXIMIZATION IN TRUST-BASED SOCIAL NETWORKS WITH DEEP Q-LEARNING

ANIKÓ KOPACZ

Abstract. Social network analysis is a rapidly evolving research area having several real-life application areas, e.g. digital marketing, epidemiology, spread of misinformation. Influence maximization aims to select a subset of nodes in such manner that the information propagated over the network is maximized. Competitive influence maximization, which describes the phenomena of multiple actors competing for resources within the same infrastructure, can be solved with a greedy approach selecting the seed nodes utilizing the influence strength between nodes. Recently, deep reinforcement learning methods were applied for estimating the influence strength. We train a controller with reinforcement learning for selecting a node list of given length as the initial seed set for the information spread. Our experiments show that deep Q-learning methods are suitable to analyze the competitive influence maximization on trust and distrust based social networks.

1. Introduction

Monitoring the information spread in social networks is beneficial for public opinion analysis, evaluating and marketing strategies. The influence maximization [3] problem aims to maximize the information coverage, while minimizing the cost associated with the degree of information spread. Competitive influence maximization [1] refers to the optimization problem, when multiple entities operate on the same social network and each of them attempts to maximize the information spread in parallel. In viral marketing, multiple companies often target the same audience with similar products. The goal of each individual company is to maximize their own revenue and persuade the

Received by the editors: 1 March 2023.

2010 *Mathematics Subject Classification.* 68T05.

1998 *CR Categories and Descriptors.* G.2.2 [**DISCRETE MATHEMATICS**]: Graph Theory – *Network problems*; G.3. [**PROBABILITY AND STATISTICS**]: Markov Processes; I.2.6 [**ARTIFICIAL INTELLIGENCE**]: Learning – *Connectionism and neural nets*.

Key words and phrases. influence maximization, reinforcement learning, Q-learning.

most possible individuals within a social network to choose their product over a competitor's.

Reinforcement learning is a computational approach to learn a specific task based on an agent-environment interaction. The agent's learning process is guided by a reward received: successful steps towards the completion of a predefined task are associated with positive feedback. Reinforcement learning has been applied to several real-world inspired optimization problems, such as robotics, epidemiology, scheduling and routing problems. Moreover, the reinforcement learning setting can be extended to the influence maximization problem [2, 6, 11].

The present work argues that deep reinforcement learning is suitable for constructing initial seed sets for competitive influence maximization on social networks displaying trust-distrust relationships. Two distinct mechanisms are analyzed to construct initial seed sets for two competing actors: joint- and iterative seed selection. The effectiveness of possible seed sets is compared based on the number of activated nodes after simulating polarity related independent cascade on trust-distrust networks.

This article is organized as follows. In Section 2, the polarity related competitive influence maximization problem is described. Section 3 presents the reinforcement learning setting and a deep reinforcement learning method, namely Deep Q-network. The conducted experiments and results are shown in Section 5. Finally, our conclusions and opportunities for improvement are summarized in Section 6.

2. Influence maximization

Influence maximization [3] targets the optimization of information spread in social networks starting from a set of source nodes. Let K denote the maximum number of nodes in the seed set. In [3] the authors also proposed a greedy baseline algorithm under the two main existing diffusion models, namely Independent Cascade and Linear Thresholds. The influence maximization problem is NP-hard [3], therefore, in addition to approaches proposed specifically optimize influence maximization, various soft computing methods can be applied to alleviate the computational requirements e.g., reinforcement learning.

Carnes et al. extended the independent cascade model to the competitive scenario, where actors with opposing interests are present, introducing two influence spread mechanisms: the distance based and the wave propagation model. Both models are suitable for constructing a seed set greedily to address the problem of competitive influence maximization (CIM).

Conventional influence maximization methods are biased simulating influence spread groups with different attributes [7]. Thus, the balanced influence

maximization was proposed to examine influence spread in attributed social networks [7]. The baseline algorithm for the balanced influence maximization problem is Attribute-based Reverse Influence Sampling algorithm, that achieves the efficiency of conventional Influence Maximization methods and manages to conserve the initial attribute distribution of the sampled social network [7].

In [2] the authors proposed a reinforcement learning framework regarding influence maximization problem in random graphs. For the selection of initial source points for the information cascade a Markov Decision Process is proposed. Markov Decision Processes may be solved by applying single agent reinforcement learning [10]. The autonomous agent selects the source nodes to broadcast an initial message, policy improvement is applied to approximate the action-value function. The reward received for selecting certain source nodes shows the degree of information dissemination in the network after simulating information cascade with a finite time-horizon.

Recently, hierarchical generative embedding was implemented with the goal to map the network nodes to a lower-dimensional embedding space [11]. The learned node representation is utilized for estimating the influence strength between two nodes and the most influential nodes are selected greedily in regard to the learned representation. The method is evaluated on various social networks based on real-world data, such as citation networks [8].

In [6] deep reinforcement learning was studied to construct an estimator to determine the expected influence of nodes. Network embedding is applied to construct a vector representation of nodes, the obtained vectors are utilized as an input for a deep Q-network [9] that approximates the expected influence. The seed set optimizing the influence spread is constructed by selecting nodes with the objective of maximizing the expected influence. The node selection is performed in one iteration, all embeddings are computed and the top k candidates are appointed as the seed set.

2.1. **Polarity related influence maximization.** The influence maximization problem formalized in [3] features social networks having a single type of relation between individuals. Polarity related influence maximization [5] operates taking into account two opposing type of relationships. Methods addressing influence maximization can be extended to solve the polarity related influence maximization by applying polarity related independent cascade [5].

In the competitive influence maximization setting, where two distinct actors attempt to influence vertices in of the same network, multiple node activation statuses occur. A vertex is considered inactive if none of the actors managed to influence them yet. In the case of activated nodes, two additional states are distinguished representing the polarity of activated nodes. Furthermore,

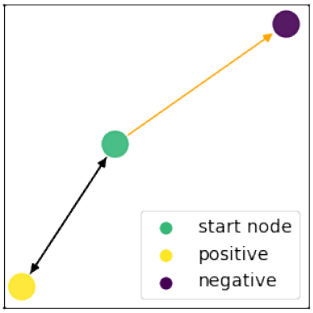


FIGURE 1. Example for polarized activation of nodes. The green node is marked as positive activating its neighbors following outgoing edges, the neighbors that trust the starting node (black edges) turn positive, while nodes that distrusted the starting node (orange edges) become negative

the polarity of nodes also marks the actor which influenced the current node. The vertices activated by the reinforcement learning agent are called positive vertices, whereas the vertices activated by the adversary are called negative ones.

We construct a seed set by selecting a k number of nodes to be activated in the beginning of the simulation. Then, a polarized node activation is simulated (see Figure 1) in accordance with [5]: if a given vertex would activate one of its' neighbors, the neighbor will choose for itself: (i) the same sign (positive or negative) if the edge between the source node and the activated neighbor is positive, or (ii) opposite sign if the edge between the two nodes is negative. If an n node is activated in a simulation step t, n tries to activate its' inactive neighbors in the next time step $t+1$. In following time steps starting from $t+2$, n no longer broadcast information toward its' inactive neighbors and no longer activates individuals in the social network. In this model, each node will be activated only once and will preserve its positive or negative status over the simulation.

3. Reinforcement learning

Reinforcement learning (RL) trains an action policy to optimize the behavior of an agents in an observable or partially observable environment [10]. Previous experiences of agent-environment interaction characterized by a reward signal are utilized to optimize solving a predefined task. The optimization

problem of training an RL agent can be formalized as a Markov Decision Process (MDP) [10]. The $\{\mathcal{S}, \mathcal{A}, p, \mathcal{R}, \gamma\}$ tuple is an MDP if the Markov property holds true, which states that the immediate reward r and the agent's next state s_{i+1} is defined by the the previous state s_t and the a_t action taken:

$$p(s', r \mid s, a) = Pr\{s_{t+1} = s', R_{t+1} = r \mid s_t = s, a_t = a\}$$

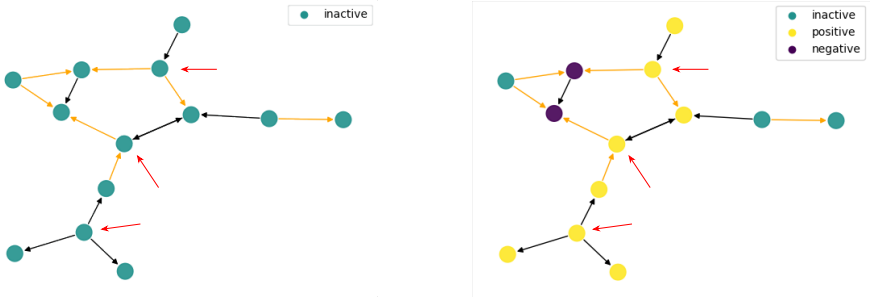
Optimal action selection policies may utilize action-value estimators to assess the potential benefits of selecting an action a in a given state S. We denote with $Q(s, a)$ the pay-off for the agent for taking action a in the s state.

Deep Q-Networks (DQN) were introduced in [9] utilizing neural networks to approximate the associated gain for possible state-action pairs. The action-value function for a state-action pair – denoted by $Q(s, a)$ – measures the goodness of choosing the a action over any other available action in state s. The optimal policy receiving a state as input is constructed as a greedy action selector regarding the estimated Q-values. The experience replay mechanism is implemented to generate training batches for the Q-network. The agent's interaction with the environment is saved into a buffer. In each training step, Q-values are calculated for $s_t, a_t, s_{t+1}, r_{t+1}$ state transitions drawn from the experience buffer. The weights of the neural network are updated with the objective to minimize the temporal difference [10] calculated for the current batch of state transitions. The temporal difference error is computed using a target network, a periodic backup of the trained Q-network. The target network is robust in regard to abrupt changes of the Q-values, hence, a more stable training approach is obtained.

4. Proposed methods

Several approaches exist for interpreting the influence maximization as a reinforcement learning problem. With the scope of formalizing the influence maximization as a Markov Decision Process, which may be solved by applying deep Q-learning methods, we describe two models.

4.0.1. *Joint seed selection.* The activation states for the vertices of the social network(s) are encoded with integers, establishing the state representation of the reinforcement learning problem. Inactivated nodes are labelled as 0, activated and positive nodes get 1, activated and negative nodes get -1 labels, respectively. The seed set for the first agent producing positively activated nodes is selected as one action of the RL agent. The possible seed sets are obtained based on the social network infrastructure before the deep Q-learning takes place. The episode consists of 1 iteration: both the agent and its' adversary select their seed set, given the independent cascade model, the degree



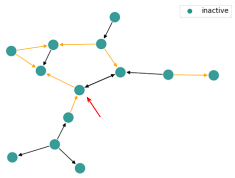
(A) Selecting 3 seed nodes (red arrows) (B) Activation status of the nodes

FIGURE 2. Select 3 seed nodes with joint seed selection. Red arrows point to seed nodes that activate their neighbors based on outgoing edges: positive edges preserve the sign of activation, while activation is switched when following negative edges

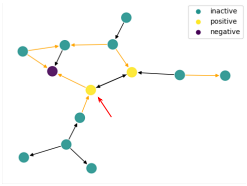
of information dissemination is obtained, and the reward is the number of activated and positive nodes.

Figure 2 illustrates the joint seed selection in a directed graph that has positive and negative edges. The budget allocated for the seed set is 3 and the seed nodes will be marked as positive. The RL agent receives as input the activation statuses of the network nodes and selects a 3 length list of seed nodes generated from all nodes present in the network (Fig. 2a). Neighbors are activated simultaneously according to the polarized independent cascade model described in Section 2.1. Following the outgoing edges of the seed nodes, inactive nodes are activated positive edges leading to positive neighbors, while negative edges result in negative neighbors (Fig. 2b). Then, the activated neighbors may activate the remaining inactive neighbors. The neighbors of the seed nodes do not have outgoing edges that point to inactive nodes (see Fig. 2b), thus, the simulation of polarized independent cascade is finished. The RL agent receives the total number of activated and positive nodes, which is 8 in this case.

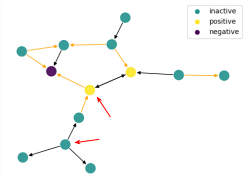
4.0.2. *Iterative seed selection.* The state representation and the encoding of the nodes is identical as described in Section 4.0.1. However, the initial seed set is assembled in an iterative manner. The reinforcement learning episode consists of a maximum k number of iterations, in each iteration the actors select an inactivated node to be added to their respective seed sets. The immediate reward received by the agents in each iteration is going to be the change in the number of activated nodes.



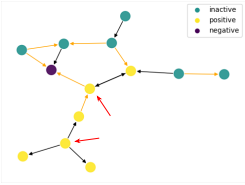
(A) Select the first seed node



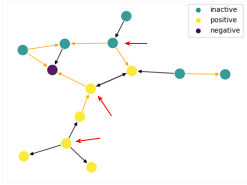
(B) Activate the neighbors of the first seed node



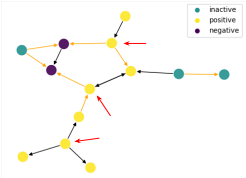
(C) Add the second seed node



(D) Inactive neighbors are activated



(E) Select the last seed node



(F) Activate the neighbors of the last seed nodes

FIGURE 3. Selecting 3 nodes for a seed set in an iterative manner in a directed graph with positive and negative edges. Red arrows point to seed nodes that activate their neighbors based on outgoing edges: positive edges preserve the sign of activation, while activation is switched when following negative edges

Figure 3 illustrates the iterative seed selection process in a directed graph that has positive and negative labels associated with each edge with a budget of $k = 3$. The seed set is initialized as empty at the beginning. Each node is selected independently, after activating the new seed node and marking the seed node as positive, a polarized independent cascade step is performed. The first seed node shown on Figure 3a activates one of its neighbors as positive, while the other neighbor becomes negative (Figure 3b). After updating the activation statuses, the RL agent receives as a reward the number of positive nodes including the seed node; the reward for the first seed node is 2. The second seed node is chosen from the inactive nodes (Figure 3c) and attempts to activate its neighbors. Previously activated nodes remain with the original activation status indifferent to their neighbors becoming activated, as shown on Figure 3d, the first seed node remains positive although it is connected to another positive node. The reward associated to selecting the second seed node for the given activation state is 4. Finally, the third seed node is selected (Fig. 3e), the 2 neighbors are newly activated (Fig. 3f) yielding a reward of 2.

5. Experiments

5.1. **Data.** We conducted our experiments on trust and distrust based social networks constructed from user relations on the `epinions.com` consumer review site [4]. Users from the website are considered nodes and nodes are connected (by edges) if the corresponding users trust or distrust one another in the context of the review site. Weights assigned to edges detail the kinship between the respective users: 1 encodes trust between users, while -1 refers to distrust. For modelling trust relations within users, directed graphs are preferred because trust (or distrust) between users is not necessarily symmetrical, i.e. *user A* trusts *user B*, however *user B* does not trust *user A*.

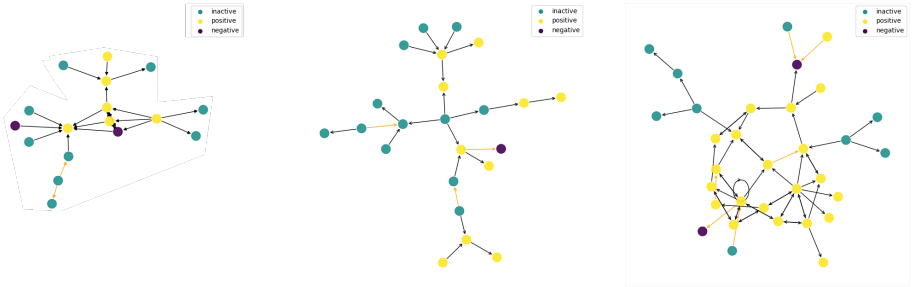
The social network constructed from `epinions.com` contains over 100000 nodes and 800000 edges.[4]. Analysis of the constructed social network showed that the probability of edges being positive is higher for nodes with a larger number of neighbors, while negative edges tend to act as bridges between positive clusters [4]. The original network was sub-sampled to experiment with deep Q-learning methods alleviating computational demands needed for processing large networks and assessing scalability of reinforcement learning approaches.

Weakly connected graphs are generated by selecting *nodes* from the original network using the following method. First, an initial node is chosen from the original network generated by a uniform distribution. A neighbor-pool that will contain the nodes connected to the sample graph nodes but not in the sample graph is initialized as an empty set. Whenever a new node is added to the sub-graph, the neighbor-pool is extended with all the nodes connected to the selected node not already in the pool, both incoming and outgoing edges are considered. In the following steps, nodes are selected from the neighbor-pool by a uniform random distribution and added to the sub-sample graph. The sampling terminates when the neighbor-pool does not have any more candidates, or the size of the sub-sample network reaches a certain threshold. Small-scale sub-networks were obtained by selecting an upper limit of 17, 23 and 32 number of nodes within a sample graph. Medium scale graphs were generated with 93 and 340 nodes, respectively.

5.2. **Experimental setup.** Two distinct reinforcement learning models are described to address the competitive influence maximization problem. Given the competitive nature of the optimization problem, two actors are distinguished to operate on the social network. We study optimal (policy) configurations for the actors separately, the currently analyzed actor is going to be referred to as the *agent* and RL methods are applied for generating possible initial seed sets for the selected actor. The two competing actors determine

Graph	Number of nodes	Number of edges	$K_{positive}$	Number of positive nodes	Number of negative nodes
G-17	17	24	3	7	2
G-23	23	22	4	9	1
G-32	32	55	4	13	2

TABLE 1. Selecting the starting nodes with joint seed selection by a controller trained with DQN on small trust-distrust graphs



(A) G-17 (B) G-23 (C) G-32

FIGURE 4. Activation status of nodes in small trust-distrust networks determined by the DQN approach

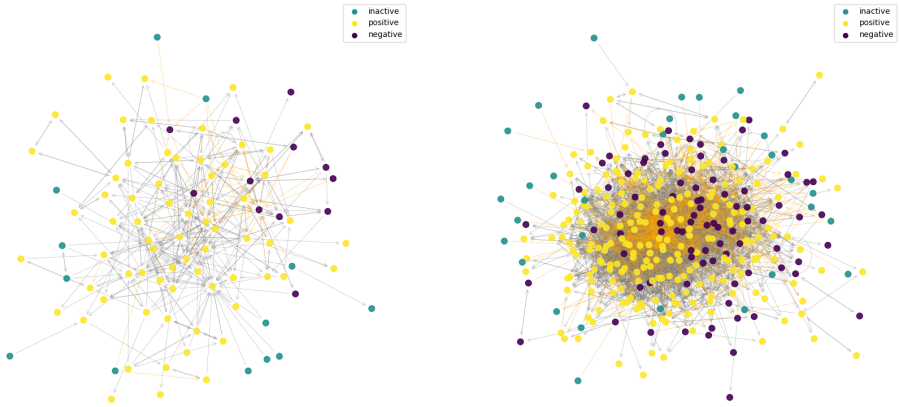
their initial seed sets simultaneously, then the influence spread is simulated under the competitive independent cascade [1] diffusion model. The actors take turns to activate nodes within the network: the vertices activated by the first actor are marked as positive, while vertices activated by the second actor are labeled negative.

For the sake of simplicity, the social network is assumed to be known during our experiments and only the policy generating the initial seed set is optimized. For different graph instances, new policies are trained using deep Q-learning. Training with a fixed social network architecture aims to reduce the magnitude of the optimization problem.

5.3. **Results.** In this paper, we evaluated deep Q-networks (DQN, [9]) approach for selecting initial seed sets for the competitive influence maximization problem [1] in signed trust based social networks. Two distinct reinforcement

Graph	Number of nodes	Number of edges	$K_{positive}$	Number of positive nodes	Number of negative nodes
G-93	93	444	1	68	12
G-340	340	4958	1	222	86

TABLE 2. Selecting the starting nodes with iterative seed selection by a controller trained with DQN on trust-distrust graphs



(A) G-93 (B) G-340

FIGURE 5. Activation status of nodes in medium size trust-distrust networks determined by the DQN approach

learning models were applied to formalize the influence maximization problem and construct the solution space.

Joint seed selection (see Section 4.0.1) is suitable to operate on trust-distrust based social networks with limited number of nodes. In Table 1, 3 distinct social networks are presented alongside with the seed sets for the influence maximization problem determined by the DQN method. The inactive, positively, and negatively activated nodes are determined by following polarity related independent cascade [5]. The structure of the evaluated small networks and the activation of nodes given the seed sets determined by the controller trained with DQN are shown on Figure 4.

When applying joint seed selection, the dimensions of the action space increase exponentially with the number of vertices in the social network. The

memory consumption and required execution time can be reduced by determining the elements of the seed node set in an iterative manner (see Section 4.0.2). Table 2 summarizes the quantitative characteristics of training with DQN for the iterative selection of seed nodes in trust-distrust social networks in the context of competitive influence maximization. In case of the larger sample networks, the distribution of inactive, positively and negatively activated nodes for the evaluated social networks is shown on Figure 5.

5.4. **Discussion.** Experimental results show that deep reinforcement learning is suitable for proposing seed sets for the competitive influence maximization problem on polarized networks. In this paper, two actors operate on social networks of various sizes that describe trust-based relationships. At the evaluation step, the same policy is used to select seed sets for both actors. The nodes activated by the first actor are reported as positive nodes, whereas the nodes activated by the second actor are reported as negative nodes. The actors select nodes for the seed set simultaneously; however, because nodes cannot change their activation in the conducted experiments, the final activation status of a node selected by both actors is positive. In the evaluated trust-distrust graphs, the first actor has an advantage over the second actor, and more nodes are positive than negative after both actors selected a seed set.

The joint seed selection described in Section 4.0.1 is feasible to determine seed sets for the competitive influence maximization problem. The joint seed selection method does not scale well due to the fact that the action space increases rapidly with the budget for the seed set. Experiments show that iterative seed selection (Section 4.0.2) can be utilized with the DQN approach to operate on medium-scale networks. In case of the G-93 and G-340 sub-networks, a $K = 1$ length seed set activates as positives a large proportion of the network nodes (see Table 2). The original social network is observed to contain positive clusters [4]. The occurrence of a larger number of positive connections of nodes facilitates the information spread in the medium-scale sub-networks.

6. Conclusions and future work

Reinforcement learning proceeds to extract meaningful information from past agent-environment interactions. Reinforcement learning is suitable for addressing NP-hard optimization problems, such as influence maximization. Deep Q-Networks, a well-known reinforcement learning method, were trained to optimize the competitive influence maximization problem on polarized networks. The solution space of the influence maximization problem increases rapidly with the number of nodes and the size of the seed set. To alleviate the impact on memory consumption and the time necessary for training of the

models several techniques can be applied, e.g. introducing a filtering step to exclude infeasible actions, constructing the seed node set by selecting nodes one by one. Future work includes analyzing the application areas of influence maximization and applying network embedding methods to project the node representation into a latent space.

References

[1] Carnes, T., Nagarajan, C., Wild, S. M., and van Zuylen, A. (2007). Maximizing influence in a competitive social network: A follower's perspective. In *Proceedings of the Ninth International Conference on Electronic Commerce*, ICEC '07, page 351–360, New York, NY, USA. Association for Computing Machinery.

[2] Chen, M., Zheng, Q., Boginski, V., and Pasiliao, E. (2021). Influence maximization in social media networks concerning dynamic user behaviors via reinforcement learning. *Computational Social Networks*, 8.

[3] Kempe, D., Kleinberg, J., and Tardos, É. (2003). Maximizing the spread of influence through a social network. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137-146.

[4] Leskovec, J., Huttenlocher, D. P., and Kleinberg, J. M. (2010). Signed networks in social media. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.*

[5] Li, D., Xu, Z., Chakraborty, N., Gupta, A., Sycara, K. P., and Li, S. (2014). Polarity related influence maximization in signed social networks. *PLoS ONE*, 9.

[6] Li, H., Xu, M., Bhowmick, S. S., Rayhan, J. S., Sun, C., and Cui, J. (2022). Piano: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, pages 1–13.

[7] Lin, M., Li, W., and Lu, S. (2020). *Balanced Influence Maximization in Attributed Social Network Based on Sampling*, page 375–383. Association for Computing Machinery, New York, NY, USA.

[8] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

[9] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602.*

[10] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* The MIT Press, second edition.

[11] Xie, L., Huang, H., and Du, Q. (2022). A hierarchical generative embedding model for influence maximization in attributed social networks. *Applied Sciences*, 12(3):1321.

Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Mihail Kogalniceanu str, Cluj-Napoca, Romania
Email address: aniko.kopacz@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.05

AUTOMATIC DETECTION OF VERBAL DECEPTION IN ROMANIAN WITH ARTIFICIAL INTELLIGENCE METHODS

MĂLINA CRUDU

Abstract. Automatic deception detection is an important task with several applications in both direct physical human communication, as well as in computer-mediated one. The objective of this paper is to study the nature of deceptive language. The primary goal of this study is to investigate deception in Romanian written communication. We created a number of artificial intelligence models (based on Support Vector Machine, Random Forest, and Artificial Neural Network) to detect dishonesty in a topic-specific corpus. To assess the efficiency of the Linguistic Inquiry and Word Count (LIWC) categories in Romanian, we conducted a comparison between multiple text representations based on LIWC, TF-IDF, and LSA. The results show that in the case of datasets with a common subject such as the one we used regarding friendship, text categorization is more successful using general text representations such as TF-IDF or LSA. The proposed approach achieves an accuracy of the classification of 91.3%, outperforming the similar approaches presented in the literature. These findings have implications in fields like linguistics and opinion mining, where research on this subject in languages other than English is necessary.

1. Introduction

Automated deception detection merges fields of research such as sociology, interpersonal psychology, communication studies, philosophy, and computational models of deception detection. The recognition of a misleading way of behaving is a task that has acquired expanding interest because of the quick development of deception in written sources, particularly the ones from cyberspace. Moreover, its applications in identifying potential harm for people and society make this challenge a relevant one and necessary to be resolved. In general, the benefits of solving this problem are reflected in domains such as

Received by the editors: 29 April 2024.

2010 *Mathematics Subject Classification.* 68T50.

1998 *CR Categories and Descriptors.* I.2.7 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing –*Text analysis*.

Key words and phrases. Deception Detection, Text Classification, Natural Language Processing, Machine Learning.

business, jurisprudence, law enforcement, and national security. Text-based information of any structure, such as news articles, client surveys, political discourses, social media contents, witnesses' reports, and so on are at present used in deception research as they portray the ideal context of lying in genuine circumstances.

Considering the problem of automatic detection of deceptive language in Romanian written texts, relatively modest efforts, if any have been made, the spotlight being put on languages that are widely spoken such as English, Spanish, or Italian. Most of the previous work has focused on the psychological or social aspects of lying. They concentrated on deceiving and its relation to the cultural dimension of individualism/collectivism and not so much on the specificity of linguistic aspects of falsehood in Romanian.

Taking into account the introduced issue, there are various difficulties that can be noticed. One of these is the fact that the information provided does not present any additional data apart from the written language itself.

Researchers often take deception language as a whole, overlooking individual highlights that may distinguish one speaker from the others, and assuming that everyone lies in much the same way. Rather than comparing each individual sample of misleading language to its equivalent control text, the complete collection of "false" testimonies is compared against the set containing "true" claims. It is worth noting that the fundamental disadvantage of a corpus of "genuine" language is the difficulty in getting a sample of instances of language in which a speaker tells the truth for the purpose of comparison. Taking these factors into account, this study aims to analyze deception indications in written Romanian, which is a unique area of study, that has not been explored yet.

Because the implicit assumption about the homogeneity of language indications of deception contradicts earlier work from psychological and sociological disciplines and raises fundamental problems about the application of current deception tools on Romanian texts, our key research goals are:

RG1. Explore which language markers and indications are more successful in distinguishing deception given a piece of Romanian text on the topic of friendship.

RG2. Related to the previous research goal, we create and evaluate the effectiveness of a wide range of binary classifiers for predicting the truthfulness and deceptiveness of texts.

RG3. Determine whether or not the Latent Semantic Analysis (LSA) method is better suited for this task compared to other different data representations such as the ones based on Linguistic Inquiry and Word Count (LIWC) or term frequency-inverse document frequency (TF-IDF).

The paper is organized as follows. Work related to our problem is discussed in Section 2. The methodology described in Section 3 incorporates the preprocessing stage, text representations, and data analysis. Section 4 is concerned with the development of the classification models. Section 5 presents the experimental results and discussions. Conclusions and directions for further research are presented in Section 6.

2. Related work

There are verbal signals to deceptive behavior that are part of the existing verbal lie detection methods utilized by professional lie catchers and scholars [16]. Automatic linguistic methods have been utilized to analyze the linguistic elements of the constitution of deceptive language in English generally.

Typically, specialists have used the word classes specified in the Linguistic Inquiry and Word Count, or LIWC [12], which is a linguistic examination tool that creates a taxonomy of words based on psychologically meaningful categories. It has been utilized to investigate matters such as personality, psychological acclimation to different changes, social judgments, tutoring dynamics, and mental health.

LIWC was for the first time used by Pennebaker's research group for several studies on the language of deception [9]. Through five different experiments, they collected a corpus of real and fake texts as part of their research. The factors that were considered to be relevant predictors in at least two of the experiments were: self-reference terms, references to others, exclusive words, negative emotion elements, and motion words. The justification behind the underperformance in a number of studies might be the fact that the verbal signals of deception in oral contact do not transpose in written communication and the other way around.

LIWC has been used for the examination of deception in written language. Research in this field has been addressed by computational linguists and a relevant example is represented by [8], who applied LIWC for post hoc analysis, evaluating many linguistic characteristics on a corpus of 100 fake and true statements on three contentious themes - the survey being similar to [9]. As an initial experiment, they used two ML classifiers: Naïve Bayes and Support Vector Machine. Both algorithms have been trained using word frequencies, like a Bag-of-Words model. They achieved an average classification accuracy of 70%, which is altogether higher than the 50% baseline. Based on this information, they computed a dominance score linked with a certain word class within the set of misleading texts as a measure of salience. The word coverage, or the linguistic item's weight in the corpus, was then calculated. Therefore, they determined some particular characteristics of deceptive texts.

In this strand of research, [10] used the same two ML classification algorithms. For their training, apart from drawing a comparison between lexically-based deception classifiers and a random guess baseline, the authors additionally assessed two more automatic approaches: genre recognition by analyzing the frequency distribution of parts of speech (POS) tags, and a text classification method which enables them to model both content and context with n-gram features. Their final goal was to identify fraudulent opinion spam, which is a fundamentally different challenge from the problem of identifying dishonest language. When it comes to detection, findings reveal that text classification based on n-grams is the best technique; nevertheless, combining LIWC features and n-gram features is the solution in order to achieve somewhat superior results.

Similar scientific endeavors as [8] were made by [1]. The importance of this paper comes from the novelty of exploring deception in the Spanish language and creating a comparison with similar studies that follow English as the main focus in order to uncover structural and lexical variations in the linguistic manifestation of deception in both languages. This paper describes an artificial intelligence model based on a Support Vector Machine (SVM) for detecting dishonesty in an ad hoc opinion corpus composed of various Spanish written communication texts. The questionnaire for the corpus compilation was designed similarly to that used by [8]. The created framework tests the effectiveness of the LIWC2001 categories in Spanish compared with a Bag-of-Words (BoW) model. The results emphasize the discriminatory power of the variables, the two first dimensions, linguistic and psychological processes, being the most relevant ones from the LIWC categories.

These investigations manage written language as utilized in asynchronous methods of communication, while Hancock and his research group investigated deceptive language in real-time computer-mediated communication (CMC), in which all members are online simultaneously using chat rooms. [6] explored dissimilarities between the transmitter's and the recipient's linguistic way across honest and deceptive communication in their initial research based on LIWC. They picked the elements thought to be important to the hypotheses for this study, which were word counts, pronouns, emotion words, sense terms, exclusive words, negations, and inquiry frequency. The findings revealed that when respondents lied, they were more chatty, using more words, more allusions to others, and more sense-related vocabulary.

3. Methodology

It is worth mentioning that during this study, we also created our own deception dataset of autobiographical narratives which was a non-topic-specific

dataset. All the experiments and data analysis we carried out were also done on that set of data but the results were not competitive further proving the difficulty of this classification problem, especially in the context of diverse narrative settings where a common subject of discussion is absent.

3.1. **Dataset.**

To study the distinction between true and deceptive statements, we used the only such data set, to the best of our knowledge, which is the deception dataset mentioned in [13], which covers four distinct cultures: the US, India, Mexico, and Romania. Each part of this dataset comprises short deceptive and truthful statements on three subjects of discussion: beliefs on abortion, views on capital punishment, and sentiments about a best friend. In this research, we used only the ones related to best friends as this topic is the most generic one and can replicate better how people lie on common topics.

The data extracted from the [13] dataset were gathered from native Romanian speakers using a web interface. The respondents have been enlisted through contacts of the paper's authors [11]. For the third subject (best friend), the participants in that study were first asked to meditate about their best friend and detail the motives behind their fellowship (incorporating facts and stories considered important for their relationship). Accordingly, for this situation, they were requested to express their true sentiments about how they felt about their best friend. Next, they were required to imagine an individual they could not stand, and depict their relationship with that person as if they were their best friend. In this subsequent case, they needed to lie about their emotions towards this individual.

In all cases, the instructions requested no less than 4 to 5 sentences and as numerous details as possible. Altogether, there were gathered 149 true and 149 false testimonies about best friends with an average of 78 words per statement. Furthermore, manual verification of the quality level of the input was made.

For ease of understanding and explanations, we decided to use a suggestive name for the dataset. As it is a topic-specific dataset, focused on the subject of best friends, the dataset will be from now on referred to as the BestFriend dataset. In Table 1, we included some examples from the BestFriend dataset, divided by class, which in this context is the level of truthfulness.

3.2. **Data preprocessing and representation.**

3.2.1. *Preprocessing.*

This stage is concerned with the preparation of deceptive and true texts before extracting relevant features. As part of the data preparation, several operations were performed. To begin with, we converted all the capital letters

TABLE 1. Dataset examples

Deceptive statement	**Truthful statement**
Mereu mă ajută când am nevoie. Dacă nu înțeleg ceva este foarte răbdătoare și îmi explică până la capăt. Nu este niciodată invidioasă pe mine. Ne înțelegem de minune.	Suntem cei mai buni prieteni deoarece ne putem spune orice în fata fără sa ne deranjeze, avem aceleași concepții și idei, ne ajutam la greu și petrecem la bine. Putem discuta o problema personala fără sa afle încă 10 oameni.

to lowercase and all punctuation marks were eliminated (they are always used in any correctly written text, but they do not carry any specific information required to train the model for this problem). For the next operation, we used the LIWC lexicon and a dictionary with Romanian words and their lemmas. We used either the word or its lemma if the word did not exist in the LIWC lexicon. After all the above-mentioned preprocessing had been done, we reconstructed the phrase with space as a separator between each word.

3.2.2. *Representation.*

Our study is based on a textual representation that is somehow different from the general models that are used in NLP, such as TF-IDF or BoW representations, but it preserves their intuition. This representation is based on the Linguistic Inquiry and Word Count lexicon.

Linguistic Inquiry and Word Count or LIWC, is a tool for textual examination where words are divided into psychologically relevant groups. The Romanian version of this lexicon incorporates 47,825 entries and is organized into 73 categories related to psychological processes. This taxonomy offers an effective technique for examining the emotional, cognitive, and structural components contained in language on a word-by-word basis. Words and word stems are classified in the LIWC internal lexicon along four broad dimensions: standard language processes, psychological processes, relativity, and personal concerns [4]. Each word or word stem is characterized by at least one of the 73 default word categories.

From all the categories, we chose the most relevant classes according to different studies that investigated a similar problem as the one stated in our research: [8], [7], [1] and [9]. These categories would be: **self-reference terms**, **references to others**, **negative emotion elements**, **motion words**, **belief-oriented vocabulary**, **words related to certainty**, **negation terms**,

sense terms and **positive emotion elements**. Table 2 contains examples of relevant instances of words that belong to the LIWC categories.

TABLE 2. LIWC categories and relevant examples

LIWC category	**Examples**
self-reference terms	"eu", "îmi", "înșine", "mi-"
references to others	"însăți", "își", "l-", "le"
negative emotion elements	"panicat", "neliniștit", "smiorcăi", "amărât"
motion words	"fugi", "prăbușire", "împiedicat"
belief-oriented vocabulary	"bănui","gândire", "reține"
words related to certainty	"bineînțeles", "categoric", "iminent"
negation terms	"fără", "n-aș", "nicăieri"
sense terms	"privitor", "pălăvrăgea", "înșfăca"
positive emotion elements	"acceptat", "mulțumire", "valoros"

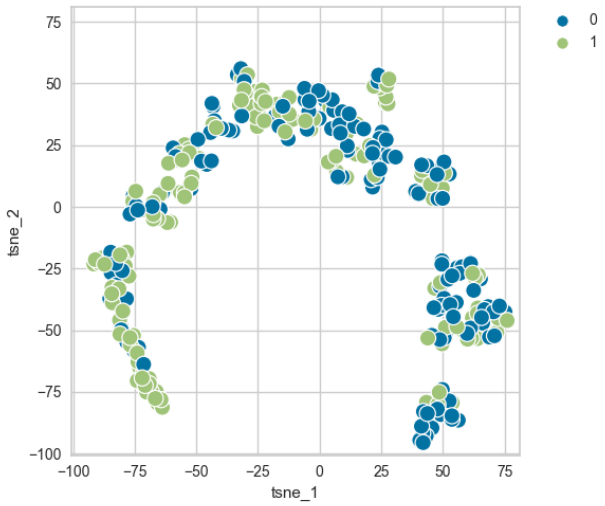
These categories are used as they are considered related to deception, for instance, an increased sense terms linguistic variable indicates deception, as liars attempt to create a detailed story. To give another example, there is less self-reference in false narratives and more frequent references to third parties and objective elements. This suggests impersonality as the liar tries to increase the narrative distance [2].

Moreover, we added two more relevant values in the feature vector of each text, more specifically **the number of words** presented in each text and the **Type-Token Ratio** (TTR). The TTR is defined as the ratio of unique tokens (types) divided by the total number of tokens. We added these measurements as it is generally considered that liars tend to produce more words during deceptive discussions [7] and the deceptive narratives are expressed with a higher syntactical simplicity [3], thus a lower Type-Token Ratio that evaluates a person's verbal diversification and asses textual richness.

For each text that was provided, we created a count vector with eleven values in which the first nine represents a category of words, more specifically the number of words that fit into this category and are included in the text and the last two represent the number of words and the TTR.

3.3. **Data analysis.**

Before proceeding to create the machine learning models we wanted to study the difficulty of the classification task that we are trying to solve and in order to do that we used a number of techniques.



0: Deceptive class 1: Truthful class

FIGURE 1. t-SNE algorithm applied on the BestFriend Dataset

3.3.1. *Investigate difficulty of the classification.*

To determine the difficulty of the classification task, we employed **t-distributed Stochastic Neighbor Embedding** (t-SNE) algorithm. It computes a non-linear dimensionality reduction which allows us to separate and visualize data that cannot be separated by a straight line. After we ran different tests, with different parametrizations (such as different perplexity and number of iterations) of the algorithm, we concluded that there is a tendency towards clearer shapes as the perplexity value increases. Applying the t-SNE algorithm, we obtained a semicircle shape (Figure 1), the graphic having a tendency to differentiate the two classes at the opposite poles of the figure.

3.3.2. *Feature Relevance.*

As we previously mentioned in the RG1 goal, we want to explore the quality of linguistic markers in deception detection. In order to do that, another approach that we considered was the investigation of the relevance of the extracted features.

Pearson Correlation Coefficient

More precisely, we computed the Pearson Correlation Coefficient [5] to examine the relationships between all features within the dataset. Additionally,

we also calculated the coefficients between the features and the labels to assess their predictive power. Given the values of the correlations between each feature and the set of labels, computing the Pearson Correlation Coefficient on the BestFriend Dataset, provided us with values between -0.183703 and 0.350826. By sorting the values of the coefficient we deduced that the most relevant features are: self-reference words, the number of words, belief-oriented vocabulary, sense terms, vocabulary related to positive emotions, references to others, and lastly, motion words. When it comes to features, there is a fairly strong positive relationship (correlation coefficient with a value above 0.7) between the number of words and the following features: self-reference terms, references to others, and belief-oriented vocabulary. This is expected as the dataset focuses on autobiographical stories, relationships with people, and opinions on them. Moreover, a moderate positive correlation (value above 0.6) exists between the number of words and both words related to certainty and positive emotion elements. This would suggest that people talk more when they experience positive emotions but also when they are or try to emulate a sense of certainty.

Relief Algorithms

Correspondingly to what we expressed in the latter paragraph, we wanted to deepen our analysis and we also applied feature selection using Relief algorithms [15]. Relief calculates a score for each feature expressing the relevance of that feature for the output label. The scores are used to rank and choose top-scoring features for feature selection. For our analysis, we looked at features based on their relief scores, prioritizing those with higher values such as self-references, insight vocabulary, the number of words, sense terms, positive emotion words, words related to certainty, negative emotion terms, and lastly, the TTR. Comparing these results with the ones obtained via the Pearson Coefficient, we can conclude that both algorithms found relevant five categories: self-references, belief-oriented vocabulary, sense terms, positive emotion terms, and the number of words.

4. Developing the Classification Model

After pre-processing and feature extraction, we wanted to evaluate three different classifiers: Support Vector Machine(SVM), Random Forest (RF), and Artificial Neural Network (ANN).

To develop the deception classifiers of the first two above-mentioned classification algorithms, we used the Scikit-learn (Sklearn) library. For all the classifiers we used the default parametrization given by the library. More precisely, for RF the criterion is set on *gini* and the $n\_estimators$ is 100. For SVM we did not modify the kernel function from the default value of Radial Basis

Function Kernel or rbf. For the ANN we employed the Keras deep learning framework for constructing our neural network architecture. The model comprises two Dense layers: the first layer consists of 64 neurons with a rectified linear unit (ReLU) activation and the second layer is a single neuron output layer with a sigmoid activation function. The model was compiled using the Adam optimizer and the binary cross-entropy loss function to optimize the network's performance in the binary classification task. Additionally, we utilized accuracy as the evaluation metric. The training process involved fitting the classifier to the training data using a $batch\_size$ of 32 and training for 25 epochs, with a validation split of 0.2.

For the models' training and testing, we did the experiments using 5-fold Stratified Cross-Validation. The 5-fold Stratified Cross-Validation ensures that each fold is then used once as validation while the four remaining folds form the training set and that each is made by maintaining the percentage of samples for each class. This way we have a division of 80% of the data being used in the training process and 20% for testing. The next section presents the results we obtained by our three classifiers during a number of different experimental setups.

5. Results and discussions

Consistent with what we expressed in the **RG2** research goal, we created and evaluated a number of binary classifiers along with linguistic models. The results that we obtained and a conclusive discussion based on the outcomes of our experiments are presented in the current section.

5.1. **Experimental results.**

To implement our machine learning models, we employed Python 3.7 and the Windows operating system. The tables in this section summarize the results of our experiments in terms of accuracy and F1-Score in the testing step for the three various classifiers that have been utilized on the linguistic models. These metrics are expressed in the form of confidence intervals (CI) that have a 95% confidence level. For these calculations the next formula was used:

$$CI = \bar{x} \pm \frac{z * \sigma}{\sqrt{n}}$$

where:

- $\bar{x}$ is the mean of the testing accuracies
- n is the sample size
- σ is the standard deviation of the testing accuracies
- z is the confidence coefficient, which is 1.96 for a 95% confidence level

For the experiments that we conducted, we used different linguistic models that will be presented in the following.

Experiment 1. LIWC-based model: 11 features

The first linguistic model uses in the feature representation all the initial attributes that were selected, 9 characteristics computed based on the LIWC lexicon (the number of: **self-reference terms**, **references to others**, **negative emotion elements**, **motion words**, **belief-oriented vocabulary**, **words related to certainty**, **negation terms**, **sense terms**, **positive emotion elements**), to which we added the **number of words** and the **TTR**. The results obtained for the LIWC-based model, using all three classifiers, can be consulted in Table 3. Along with this experiment we tried to use all of the 73 categories of the LIWC lexicon, along with the two features added by us (the number of words and the TTR), but the results were extremely similar to the ones obtained via our linguistic model with only 11 features.

TABLE 3. Testing accuracy and F1-Score for the LIWC-based model on the BestFriend dataset

Classifier	**Accuracy (CI%)**	**F1-Score (CI%)**
SVM	0.658±0.099	0.622±0.126
RF	0.715±0.065	0.707±0.055
ANN	0.731±0.034	0.735±0.072

During the tests, we took into consideration the analysis we conducted on the dataset and the results presented in Subsection 3.3, and we created some simplified linguist models, with only the features that were found to be qualitative attributes in our study. We tried several set-ups such as using only the features found relevant by either the Pearson Coefficient or by the Relief Algorithms and also tried creating a model with features from both. Given this context, we retrained and tested these leaner models, however the results we obtained showed a decrease compared to the initial LIWC-based model with 11 features.

Experiment 2. TF-IDF model

To draw a comparison between the linguistic model based on the LIWC lexicon and general representations used in Natural Language Processing, we trained the classifiers that use a TF-IDF representation. This representation was obtained by utilizing the $TfidfVectorizer$ with $smooth\_idf$ on $True$ to prevent zero divisions and the min_df on 0.001 to ignore terms that have a document frequency strictly lower than the given threshold.

Experiment 3. LSA model

As previously stated in our **RG3** goal we wanted to draw a comparison between a TF-IDF representation and a LSA one. Latent Semantic Analysis or LSA is a technique that learns latent topics by decomposing or factorizing the document-term matrix such as the TF-IDF matrix using a mathematical technique known as Singular Value Decomposition or SVD. The purpose of Latent Semantic Analysis is to reduce the dimensionality of the corpus vector space while detecting higher-order patterns within it.

For the LSA representation, the TF-IDF vectors were mapped by calling $TruncatedSVD$ with a number of 300 topics that have a variance of 99%. The topic value was chosen in regard to the variance ratio graph that we plotted for the dataset based on the TF-IDF representations and we chose the value that presented the highest value. The plot can be visualized in Figure 2 and was created by calculating the total variance ratio as the sum of the variances explained by each of the selected components for all the possible values (from one to the total length of the vocabulary).

For the TF-IDF and LSI representations we experimented with various token N-gram sizes (from 1-gram to 5-grams), but we decided to utilize a smaller subset, just from unigrams and 2-grams, as the discriminating capability of the other values as types of N-grams proved to be extremely limited, and as a result, these findings were omitted. Additionally, we also experimented with Principal Component Analysis (PCA) representations but found results similar to those obtained with LSA.

TABLE 4. Testing accuracy and F1-Score for the TF-IDF and LSA models on the BestFriend dataset

Classifier	**N-grams**	**TF-IDF representation**		**LSA representation**	
		Accuracy	F1-Score	Accuracy	F1-Score
SVM	1-gram	0.785±0.082	0.791±0.095	0.789±0.086	0.797±0.08
	2-grams	0.718±0.092	0.726±0.074	0.678±0.137	0.669±0.153
RF	1-gram	0.755±0.068	0.753±0.082	0.678±0.038	0.675±0.062
	2-grams	0.715±0.021	0.665±0.056	0.668±0.088	0.673±0.087
ANN	1-gram	0.913±0.069	0.914±0.064	0.89±0.141	0.88±0.165
	2-grams	0.896±0.15	0.904±0.127	0.849±0.185	0.837±0.215

The values in the tables confirm that the classification task is solved more successfully in the case of the neural network-based classifier across all of the linguistic models. This suggests the potential of neural network architectures in similar classification tasks such as deception detection in legal contexts as courtroom cases would represent. Secondly, despite employing LSA (Latent

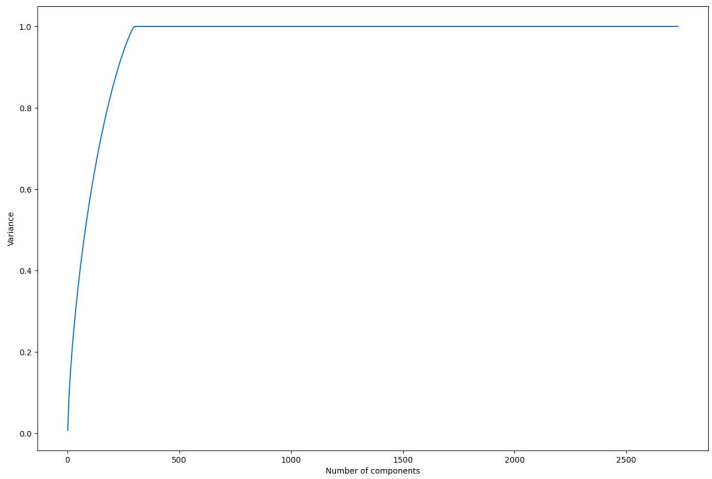


FIGURE 2. Graphic of the coherence score depending on the number of topics for the Best Friend dataset

Semantic Analysis) dimensionality reduction, we did not observe improvements in the studied metrics. This indicates that LSA may not enhance the performance of the classifiers in this context nor capture the semantic complexity of the specificities of deceit. Furthermore, increasing the size of N-grams did not result in improved performance metrics. This finding suggests that simply expanding N-grams may not necessarily enhance the classifier's performance as it might introduce data sparsity. This might also be a result of the fact that Romanian, even though it generally follows a Subject-Verb-Object (SVO) word order, is more flexible than English in terms of word order variation meaning that two n-grams could contain the same words but in a different order. Another conclusion that we draw is related to the linguistic models that we designed, especially the ones that do not use LIWC as a base for feature vector creation. Even though TF-IDF and LSA are considered to be general models, they were able to achieve in most cases better results than the models trained with psychologically relevant attributes. This made us conclude that a major part of the deceptive process in the case of topic-specific statements is not related to which category of words we use, but which terms we utilize.

5.2. **Comparison with SOTA.**

Even though our study is considered to be a novelty, from the point of view of experimenting on Romanian datasets, we tried to draw a comparison between the results that we obtained and the performance of other similar approaches 5, even though they are not all implemented on a topic specific dataset or they are following different languages. Considering this aspect, we chose a sample of researches to compare their results with the ones we achieved.

Firstly, [14] created a new open-domain deception dataset that also includes demographic data such as gender and age. Even though the methods that obtained the best performance are not similar to the ones conducted by us, and the dataset has a somewhat autobiographical topic, the authors tried several sets of features, including semantic features based on the LIWC lexicon. This approach had only an accuracy of 60.21% compared to the maximum of 69.50% obtained via part-of-speech (POS) tags.

Next, a more similar approach to ours in terms of the classification algorithms that have been used, the selected features, and the data utilized for research is presented in [13]. Even though the study presents experiments made on a cross-cultural dataset, a comparison deserves to be done as our experiments were made on the Romanian version of the dataset collected for the mentioned paper.

From all the research we evaluated, [1] is the closest one to our approach in terms of methodology, dataset, and target language. The research is the exploration of a non-English language, more precisely on Spanish written communication. They have designed an automatic classifier based on SVM and the dataset is created similarly to the one mentioned in [8]. We consider this comparison to be the most relevant one as it is done based on a language close to Romanian, the topic of the dataset is the same as ours and the methodology is similar.

6. Conclusions and Future Work

Although many artificial intelligence models for automatic deception detection were implemented, most of them were for English texts, the Romanian Language being somehow neglected. In this paper, we researched an important Natural Language Processing task, analyzed a topic-specific dataset, and investigated automatic methods for the identification of deceptive language in written Romanian statements on the topic of friendship, using several representations for their training such as the LIWC psycho-linguistic categories, TF-IDF and LSA. By comparing different algorithms and evaluating their output we achieved a 91.3% accuracy in terms of detecting deception, which

TABLE 5. Comparison between our models and relevant research

Dataset	Features	Classifier	Classification performance
Open Domain Deception Dataset [14]	All categories from the LIWC lexicon	SVM	Accuracy 60.21%
Open Domain Deception Dataset [14]	POS tags	SVM	Accuracy 69.50%
Best Friend Spanish Dataset [1]	All categories from the LIWC lexicon	SVM	F1-Score 84.5%
Best Friend English Dataset [13]	Linguistic categories from the LIWC lexicon	SVM	Accuracy 75.98%
Best Friend Romanian Dataset	**TF-IDF unigram representation**	**ANN**	**Accuracy 91.3% F1-Score 91.4%**
Best Friend Romanian Dataset	LSA unigram representation	ANN	Accuracy 89%
Best Friend Romanian Dataset	Selected categories LIWC representation	ANN	Accuracy 73.1%

represents competitive results that outperform similar methodologies we used as state-of-the-art approaches for this task.

As for future work, although the classification algorithms provided positive results, there are a number of improvements that can be mentioned. One development can be made in terms of the datasets that are used as there is a lack of data for this type of task, especially in less studied languages such as Romanian. Furthermore, datasets that explore different types of deception and contexts where people lie would be helpful in creating more accurate textual lie detectors.

Moreover, even though every language has its individuality and the deception process should be, from a certain point, particular for the language, the proposed approach could be extended to different languages. This direction could be the one of studying possible structural and lexical dissimilarities between the linguistic manifestation of deceit in languages from different families (i.e. Romance languages and Germanic languages).

Additionally, more features can be added to the classification algorithm for an improvement in deception detection. We plan to explore further the implication of affect and the possible inclusion of automatic emotion analysis into the identification of deceptive language. Moreover, different representations of

the texts are to be considered, as word embeddings are a very versatile method in problems of text analysis and classification.

Finally, another advancement that can be made is in the algorithms that we studied. Given the fact that the best results were obtained with an artificial neural network, a dive into some Deep Learning architectures would bring a new perspective on deception detection. For example, an architecture based on Transformers might help with the difficulty of the classification task and bring better outcomes.

References

[1] Ángela Almela, Rafael Valencia-García, and Pascual Cantos. Seeing through deception: A computational approach to deceit detection in written communication. In Eileen Fitzpatrick, Joan Bachenko, and Tommaso Fornaciari, editors, *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 15–22, Avignon, France, April 2012. Association for Computational Linguistics.

[2] Luigi Anolli, Michela Balconi, and Maria Ciceri. Linguistic styles in deceptive communication: Dubitative ambiguity and elliptic eluding in packaged lies. *Social Behavior and Personality: an international journal*, 31:687–710, 01 2003.

[3] Jeffrey S. Bedwell, Shaun Gallagher, Shannon N. Whitten, and Stephen M. Fiore. Linguistic correlates of self in deceptive oral autobiographical narratives. *Consciousness and cognition*, 20(3):547–555, 2011.

[4] Diana Paula Dudău and Florin Alin Sava. Performing multilingual analysis with linguistic inquiry and word count 2015 (liwc2015). an equivalence study of four languages. *Frontiers in Psychology*, 12:570568, 2021.

[5] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[6] Jeffrey T. Hancock, Lauren E. Curry, Saurabh Goorha, and Michael T. Woodworth. Lies in conversation: An examination of deception using automated linguistic analysis. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26, 2004.

[7] Saurabh Goorha Jeffrey T. Hancock, Lauren E. Curry and Michael Woodworth. On lying and being lied to: A linguistic analysis of deception in computer-mediated communication. *Discourse Processes*, 45(1):1–23, 2007.

[8] Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In Keh-Yih Su, Jian Su, Janyce Wiebe, and Haizhou Li, editors, *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[9] Matthew L. Newman, James W. Pennebaker, Diane S. Berry, and Jane M. Richards. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin*, 29(5):665–675, 2003. PMID: 15272998.

[10] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[11] Katerina Papantoniou, Panagiotis Papadakos, Theodore Patkos, Giorgos Flouris, Ion Androutsopoulos, and Dimitris Plexousakis. Deception detection in text and its relation to the cultural dimension of individualism/collectivism. *CoRR*, abs/2105.12530, 2021.
[12] James W. Pennebaker and Martha E. Francis. *Linguistic Inquiry and Word Count*. Lawrence Erlbaum Associates, Incorporated, 1999.
[13] Verónica Pérez-Rosas and Rada Mihalcea. Cross-cultural deception detection. In Kristina Toutanova and Hua Wu, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 440–445, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
[14] Verónica Pérez-Rosas and Rada Mihalcea. Experiments in open domain deception detection. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1120–1125, Lisbon, Portugal, September 2015. Association for Computational Linguistics.
[15] Marko Robnik-Sikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53:23–69, 10 2003.
[16] Aldert Vrij, Pär Anders Granhag, and Stephen Porter. Pitfalls and opportunities in nonverbal and verbal lie detection. *Psychological Science in the Public Interest*, 11(3):89–121, 2010. PMID: 26168416.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş -Bolyai University, Mihail Kogălniceanu 1, 400084, Cluj-Napoca, Romania

Email address: malina.crudu@stud.ubbcluj.ro