STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 2/2022

July - December

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

Published by Babeş-Bolyai University

EDITORIAL BOARD

YEAR **Volume 67 (LXVII) 2022**
MONTH **DECEMBER**
ISSUE **2**

S T U D I A

UNIVERSITATIS BABEȘ-BOLYAI

INFORMATICA

2

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

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 2, 2022
DOI: 10.24193/subbi.2022.2.01

EXTENDED MAMMOGRAM CLASSIFICATION FROM TEXTURAL FEATURES

ADÉL BAJCSI, CAMELIA CHIRA, AND ANCA ANDREICA

Abstract. The efficient analysis of digital mammograms has an important role in the early detection of breast cancer and can lead to a higher percentage of recovery. This paper presents an extended computer-aided diagnosis system for the classification of mammograms into three classes (normal, benign and malignant). The performance of the system is evaluated for two different mammogram databases (MIAS and DDSM) in order to assess its robustness. We discuss the changes required in the system, particularly at the level of the image preprocessing and feature extraction. Computational experiments are performed based on different methods for feature extraction, selection and classification. The results indicate an accuracy of 66.95% for the MIAS dataset and 54.1% for DDSM obtained using genetic algorithm based feature selection and Random Forest classification.

1. Introduction

Computer-aided detection and diagnosis (CAD) relies on medical image processing, being used nowadays for a big variety of diseases, including breast cancer. In the current research, our aim is to create a CAD system able to detect breast cancer from mammograms. Input images are used from two different mammogram datasets (Mammographic Image Analysis Society – MIAS [15], Digital Database for Screening Mammography – DDSM [7]) in order to verify a stable performance for the stages of preprocessing, feature extraction and classification.

Received by the editors: 20 September 2022.

2010 *Mathematics Subject Classification.* 68T35.

1998 *CR Categories and Descriptors.* I.2.1 [**Artifical Intelligence**]: Applications and Expert Systems – *Medicine and science*; I.2.6 [**Artifical Intelligence**]: Learning – *Knowledge acquisition*; I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement – *Feature representation*;

Key words and phrases. Breast cancer detection, Mammogram classification, GLRLM, Feature selection, Random Forests, MIAS, DDSM.

A multi-class classifier is built to distinguish normal, benign and malignant classes. We start from a recently proposed CAD system [2], which consists of the following five steps:

(1) mammogram preprocessing, (2) segmentation of the image, (3) feature extraction (FE) by calculating characteristics from GLRLM (Gray-Level Run-Length Matrix [6]), (4) feature selection (FS) and (5) classification (CLS).

In reference [2], multiple experiments are reported with competitive results for medical images from the MIAS database [15] using Principal Component Analysis (PCA) for feature selection and Random Forest (RF) classifiers. As the system was created and tested for MIAS images, our main objective in this study is to determine if the system can be extended to obtain a good performance for other medical images. For DDSM images, the system is likely to generate errors because the intensity of the mammograms is different compared to images in MIAS. Also, images from DDSM have a high variance in their intensities and white patches could be found on DDSM entries, connected to the breast.

Considering the abovementioned issues, in the current paper we extend the previously proposed system mainly concerning the preprocessing and feature extraction steps. The preprocessing step is extended by

(1) removing white patches outside the breast, (2) setting the threshold to define the breast dynamically, instead of hard-coding it, (3) creating a more robust method to define the location of the pectoral muscle, (4) selecting the proper component overlapping with the pectoral muscle and (5) filling its concave parts.

Moreover, feature extraction considers characteristics calculated from Gray-Level Co-Occurrence Matrices (GLCM) with different distances and is reimplemented to use the graphical processing unit. For segmentation, feature selection, and classification, the same methods are used as in [2], namely, k-means, PCA and genetic algorithm (GA) based feature selection, and Decision Tree (DT) and RF respectively.

The rest of the article is organized as follows: in section 2 we will discuss existing solutions in the literature to solve the problem. In section 3 the proposed approach is presented in detail. Section 4 describes the data used and the results achieved. In section 5 we draw the conclusions of the conducted research and the future directions are defined.

2. Related work

With the development of image processing techniques and online (freely) accessible databases, experiments for image analysis are being conducted in

various fields. One of these fields is medicine. With the help of machines, illnesses can be easier detected and confirmed by doctors. Breast cancer is one of these illnesses being studied. The problem of detecting cancer from mammograms is highly important and complex because of the difference in the images created with different machines, the size of the mammograms, and the appropriate selection of the used features. In the following paragraphs, we will present existing methods in the literature to detect breast cancer.

In [13] a comprehensive review is presented on the methods used from 1998 to 2018. In the review 129 papers were included. As the review highlights, there are three major types of experiments:

(1) studies where only normal and abnormal classes are considered [16] (2) where benign and malignant classes are considered [1, 4, 4, 6, 8, 8] and (3) where all three classes (normal, benign and malignant) are considered [10, 12, 14].

The authors concluded that the most used classifiers were Artificial Neural Networks (ANN), Support Vector Machines (SVM) and K-Nearest Neighbors (k-NN). As for evaluation metrics most of the researchers used accuracy, area under curve, sensitivity and/or specificity.

Reference [5] conducted an experiment to compare several feature extraction methods, namely, First Order Statistic (FOS), Gray Level Co-Occurrence Matrix (GLCM), Gray-Level Run-Length Matrix (GLRLM), and Gray Level Difference Matrix (GLDM). The authors concluded that the best result (93.98% accuracy) was achieved using the GLRLM features for building ECOC SVM classifiers.

Another survey [6] focused on various feature extraction and selection methods. Besides the four features mentioned before, the authors analyzed Tamura features, Gabor features, Wavelet transform features, Hu's invariant moments features, and other shape features such as perimeter, area, compactness, aspect ratio, and so on. For feature selection, Tabu search, Genetic algorithm, ReliefF algorithm, and Sequential forward/backward selection are included. The authors [6] showed that for building a classifier, GLRLM features are the most appropriate. Using these features 66.66% and 90.9% respective to AUC and precision were achieved [6].

In [1] is reported 88% for both accuracy and AUC using classification with Neural Networks. As input the authors proposed using a concatenation of different Convolutional Neural Networks (AlexNet, VGG16, GoogLeNet, ResNet18, InceptionResNet). In reference [10] a novel approach is proposed to solve the problem of classifying the mammograms into normal, benign and malignant classes. Two types of features are extracted from the images:

(1) low-level features calculated using ResNet and transfer learning and (2) high-level features calculated using RNN-LSTM.

From the resulting feature set, a CNN selects the relevant features, then the final classification is made by an ensemble learning model (using RF learning and boosting). With this method, the authors achieved 96% accuracy. In [16] ANNs are used for feature extraction (DenseNet and MobileNet) and fed to a fully connected network. With this approach, accuracy of 96.34% was achieved using DenseNet features.

In [4] the application of super-resolution is presented to better distinguish abnormal masses. With this method, the authors boosted the result and achieved 96.7% accuracy. The authors used a combination of FOS and GLRLM features as input to nine different classifiers. Reference [14] proposes the extraction of features from the spatial pyramid and called Pyramid Histogram of Colors. Feeding an ANN with the calculated features, 82.1% AUC was achieved. In [12] a Forest Optimized Algorithm is proposed to select features from characteristics calculated from GLCM and wavelet transform. For classification, the authors used SVM, k-NN and Decision Trees (DTs).

Reference [8] presents a Fuzzy Rule-Based interpolative classifier for making differences between benign and malignant lesions. The input of the classifier consists of 18 features included shape features (shape and morphological characteristics), margin features (sharpness and roughness of the boundary), and density features. With this method, the authors get accuracy of 91.65%.

3. Proposed approach

In our research, we extended the approach presented in [2] in order to be able to apply it to other mammogram datasets. The system in [2] is not working accordingly if the breast tissue has too low intensity or if white patches appear on the image. In this section, we present the original CAD system proposed in [2] and then detail the method's extension and the main improvements proposed in the current paper.

3.1. **The base CAD system.**

As already mentioned, the CAD system introduced in [2] has five main steps: preprocessing, segmentation, feature extraction, selection and classification. Essential aspects for a successful computer-aided diagnosis include calculating the features and classification of the observations. The most important is to properly clear the image (remove the information – pixel –, which is not important from the problem's perspective). For instance, to diagnose breast cancer from medical images (such as mammograms), everything outside the breast tissue and the pectoral muscle is irrelevant. The pectoral muscle is not a possible location of a lesion, leading to its disregardment during the image analysis process.

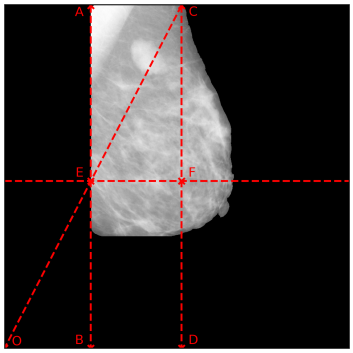


FIGURE 1. Defining the possible location of the pectoral muscle on image `mdb015` from MIAS.

In the first step, the tissue is separated from the rest of the image. To get a precise result, noise reduction (morphological opening) and emphasis (histogram equalisation) are applied. To remove the pectoral muscle, the Seeded Region Growing (SRG) algorithm, proposed by Maitra et al. [9], is used by selecting the pixels corresponding to the muscle. The requirement of this approach is that the pectoral muscle must be on the left side of the image. To decrease the image to the possible location of the muscle first, the left- (A) and rightmost bounds (C) of the breast at the top (vertical lines across the left-/rightmost points – lines AB and CD on image fig. 1) are defined, and then the top of the right bounding line is connected to the lower-left corner (O) of the image. As a result, a triangle should be formatted (on the top between the two bounding lines – ACE), containing the pectoral muscle. For this triangle, the SRG was applied using as seeds the pixels from the section between the right angle and the bisector of the hypotenuse. The final step of the preprocessing is to selecting a smaller bounding box surrounding the breast. Next (segmentation step), k-means was performed to segment the images into 12 clusters. During the feature extraction, from both the original and the segmented image GLRLMs are calculated and characteristics are extracted (resulting in 44 features per image – 11 features $\times$ 4 directions). Following with the feature selection step, which aims to reduce the size of the input. For this purpose, two algorithms are implemented: PCA and GA. Later, the result of the feature selection method is passed to the classification. To differentiate normal, benign and malignant classes, DT and RF models are built.

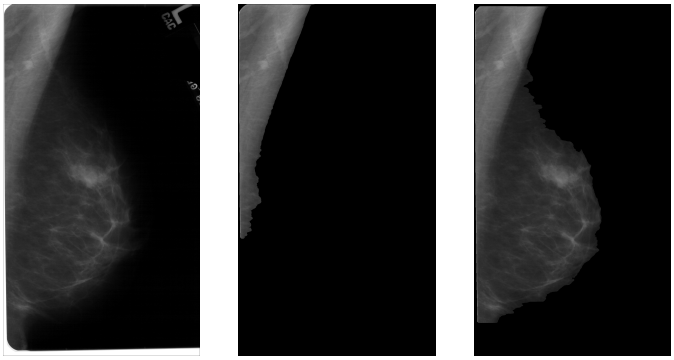
3.2. **Extended method.**

When introducing new images to the system, presented in the previous subsection, there could be cases when the execution differs from the expected behaviour because it was specified for mammograms from a specific dataset. For instance, referring to images from DDSM, there are cases when the intensity of the breast tissue is low, and the tissue disappears after the thresholding, or the position of the pectoral muscle is misdetected. The most vulnerable part is the preprocessing. This subsection details the changes made in the previously presented system.

On the side(s) of some images (from DDSM), a white column/row could appear. Previously, the position of the pectoral muscle was defined based on the longest vertical line found with Hough transform with an average intensity lower than 180 or if no line was found based on the pixel intensities in the upper corners of the mammogram. This method worked for images from MIAS. Even though MIAS mammograms could contain white columns, they are not connected to the breast and therefore they are removed together with the labels. Images from MIAS are squares and the actual mammogram is positioned in the centre. On the other hand, in DDSM, these columns could be mistaken for the pectoral muscle (because no lines will be detected due to the lack of surrounding black columns). Hence, before the preprocessing, every pixel with high intensity (> 240) on the margin of the image will be set to 0.

The first step of the preprocessing is the removal of the labels using thresholding. For images from MIAS the threshold value of 50 was defined experimentally. However, using the same value on the new images led to the disappearance of the breast tissue in some cases (due to its low intensity as on Fig. 2(b)). To overcome this problem, we decided to use a dynamic definition for the threshold values by calculating the median intensity value from the image – I (omitting pixels with 0 value). The result of the new thresholding is shown in Fig. 2(c). Next, we must ensure that the pectoral muscle is on the left side of the image (this is a prerequisite of the used SRG method).

To define the initial orientation of the breast, the intensities in the upper corners are used. We further break this down into two cases. First, we calculate the first and last non-zero positions from the first row. If one of the values is close enough to the side (within 100 pixels), then the location of the muscle is clear. Otherwise, we add zero padding around the image and apply Hough's line transform to determine vertical lines on the image. The longest line found specifies the position of the pectoral muscle. The zero-padding helps to define the border, even if this is exactly on the edge of the image.



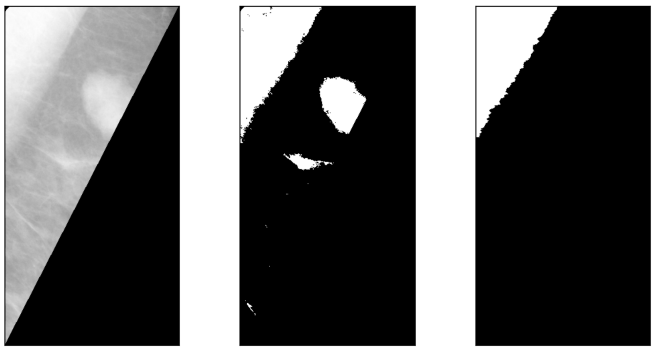
(a) Original (b) $\theta = 50$ (c) θ based on median

FIGURE 2. Separating foreground and background using thresholding presented on image `A-1833-1.LEFT_MLO` from DDSM.

The result of SRG, mentioned in the previous section, could contain more (unconnected) components (see Fig. 3(b)). Of these components, we should select the largest and closest to the top left corner of the image (Fig. 3(c)).

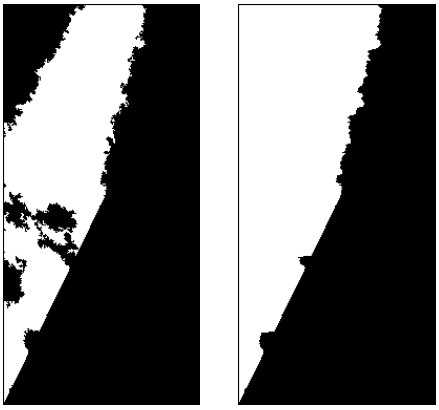
Moreover, we investigate the shape of the different resulting components and found that some of them are concave (Fig. 4(a)). For a more precise removal, we fill these holes in the component (Fig. 4(b)). To accelerate the pipeline, we propose a new feature extraction version that uses a graphical procession unit. In the current research, we had an Nvidia GeForce GTX 960M GPU and achieved $\approx 62\times$ acceleration. Heretofore, the GLRLM was calculated at $265.39s$. Even with parallel threads, it takes $14.52s$ to construct the matrix. However, using GPU, we can get the result matrix in $4.3s$ (depending on the image's size).

For steps (2), (4) and (5) the same methods are used as in the originally proposed method from [2]. Besides GLRLM features, GLCM features are calculated (for distances 1 - default - and 8) and used for classification. From the constructed (normalized) GLCM, the features are calculated using (3), (4) and (7) to (23), where *level* denotes the number of gray levels on the mammogram. The functions in (1) and (2) help the definition of sum average (18), sum variance (20), sum entropy (19) and difference average (21), difference variance (23), difference entropy (22) respectively.



(a) ROI of pectoral (b) Result of SRG (used in the original version) (c) Pectoral mask

FIGURE 3. Selecting the mask corresponding to the pectoral muscle from the result of the SRG [9] on image `mdb015` from MIAS.



(a) Result of SRG (used in the original version) (b) Final mask

FIGURE 4. Filling the wholes on the SRG [9] result on image `mdb017` from MIAS.

$$(1) \qquad p_{x+y}(k) = \sum_{i,j=0}^{levels-1} GLCM_{ij}, \quad \text{where } i+j=k \qquad (2) \qquad p_{x-y}(k) = \sum_{i,j=0}^{levels-1} GLCM_{ij}, \quad \text{where } |i-j|=k$$

(3) $$\mu_x = \sum_{i,j=0}^{levels-1} iGLCM_{ij}$$

(4) $$\sigma_x^2 = \sum_{i,j=0}^{levels-1} (i-\mu_x)^2 GLCM_{ij}$$

(5) $$\mu_y = \sum_{i,j=0}^{levels-1} jGLCM_{ij}$$

(6) $$\sigma_y^2 = \sum_{i,j=0}^{levels-1} (j-\mu_y)^2 GLCM_{ij}$$

(7) $$ent = -\sum_{i,j=0}^{levels-1} GLCM_{ij} \ln GLCM_{ij}$$

(8) $$cont = \sum_{i,j=0}^{levels-1} (i-j)^2 GLCM_{ij}$$

(9) $$ASM = \sum_{i,j=0}^{levels-1} GLCM_{ij}^2$$

(10) $$energy = \sqrt{ASM}$$

(11) $$diss = \sum_{i,j=0}^{levels-1} |i-j| GLCM_{ij}$$

(12) $$corr = \sum_{i,j=0}^{levels-1} \frac{(i-\mu_x)(j-\mu_y)}{\sqrt{\sigma_x^2 \sigma_y^2}} GLCM_{ij}$$

(13) $$IDM = \sum_{i,j=0}^{levels-1} \frac{1}{1+(i-j)^2} GLCM_{ij}$$

(14) $$sim = \sum_{i,j=0}^{levels-1} \frac{1}{1+|i-j|} GLCM_{ij}$$

(15) $$DM = \sum_{i,j=0}^{levels-1} \sqrt{\frac{|i-j|}{2} GLCM_{ij}}$$

(16) $$CP = \sum_{i=0}^{levels-1} (i+j-\mu_x-\mu_y)^4 GLCM_{ij}$$

(17) $$CS = \sum_{i=0}^{levels-1} (i+j-\mu_x-\mu_y)^3 GLCM_{ij}$$

(18) $$SA = \sum_{i=2}^{2levels} ip_{x+y}(i)$$

(19) $$SE = -\sum_{i=2}^{2levels} p_{x+y}(i) \ln p_{x+y}(i)$$

(20) $$SV = \sum_{i=2}^{2levels} (i-SA)^2 p_{x+y}(i)$$

(21) $$DA = \sum_{i=0}^{levels-1} ip_{x-y}(i)$$

(22) $$DE = -\sum_{i=0}^{levels-1} p_{x-y}(i) \ln p_{x-y}(i)$$

$$DV = \sum_{i=0}^{levels-1} (i - DA)^2 p_{x-y}(i) \tag{23}$$

4. Experiments and results

In this section, we present the details of our experiments, starting with the used datasets, the conditions of the splitting (into train and test) the datasets and the results of our research.

4.1. **Datasets.** MIAS [15] is a database of MLO (mediolateral oblique view) mammograms published in 1994. It contains 322 (161 pairs) images. Of these, 207 are from normal tissues, 64 are from tissues containing benign lesions and the remaining 51 contain malignant lesions.

DDSM [7] is another database that was first published in 1998. Over the years it was further developed. The database consists of 7808 mammograms from approximately 1950 patients. Compared to MIAS, DDSM contains both MLO and CC (craniocaudal view) view images. From the total images, we will use only the MLO ones, so 3904 images. The distribution of the observations is as follows: 1204 normal, 1342 benign and 1358 malignant.

4.2. **Experiment setup.** Each dataset is split into train and test sets containing 75% and 25% of the data. When creating the split, we maintain the same distribution in the result sets as in the original data (stratified sampling). Also, to reduce the possible bias of the classification, we place the observations from one person into the same set. Considering these, from MIAS, there will be 242 mammograms in the train set and 80 in the test set. As for the DDSM 2928 mammograms will be in the train set and the remaining 976 in the test set.

To define the optimal parameters, K-fold cross-validation was applied. The best parameter is defined based on the highest mean accuracy across the models built during cross-validation. In these experiments 5-folds are used.

As shown above, the datasets are imbalanced. Therefore, stratification is used to preserve the original distribution of the classes in each fold. With the use of stratified cross-validation, the average of the result metrics will be a close approximation to the result on the original sets.

For evaluation purpose accuracy (A), precision (P), recall (R) and f1-score (F_1) measures are used. As also shown in [13], the first three metrics are frequently used in studies related to breast cancer detection or diagnosis.

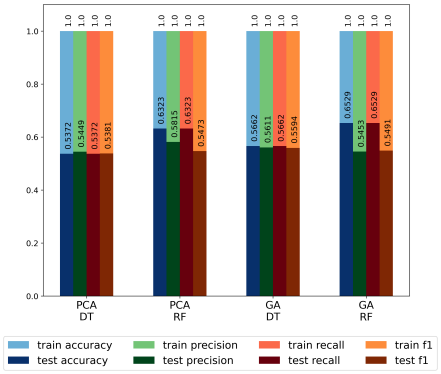


FIGURE 5. Result metrics using GLRLM features calculated from MIAS images.

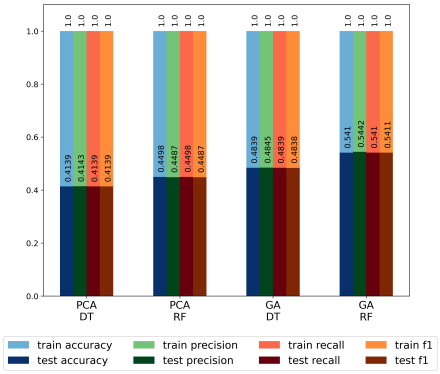


FIGURE 6. Result metrics using GLRLM features calculated from DDSM images.

4.3. **Results.** The results of the classifications are presented in Tables 1 and 2 and figs. (5) to (10). In columns A, P, R and F_1 of the tables, the mean values are reported over the folds with their corresponding standard deviation. In Table 1 and Table 2 only the test results are presented. During the training of the models, we achieved 100% (as show on Figs. (5) to (10)) with each parameter mentioned in the setup.

From Table 1, we can see that the results on MIAS are slightly different from the ones reported in [2]. This can be explained with the changes mentioned in Section 3.2. However, these modifications are necessary in order to use the proposed method on images from DDSM.

The best results for MIAS are obtained using $GLCM_8$ (for step (3)), GA feature selection (for step (4)) and RF classifier (for step (5)). With k-fold cross-validation we reached 66.95% average test accuracy. For the extended method, the average test accuracy using GLRLM, PCA and RF was 63.23%. The made in the preprocessing can cause the difference between the results of the original [2] and extended system. Now each image has the same orientation (previously the flip and the pectoral removal did not work on some images) and by applying CLAHE (contrast limited adaptive histogram equalization) filtering to the images. The different split, explained above, can also affect the result.

Table 2 presents the results obtained on DDSM images using the extended system proposed in this paper. The best classification result on DDSM is 54.1% for average test accuracy, when classifying the observations into three classes using features calculated from GLRLM, GA feature selection and RF classification.

FE	FS	Dir.[a]	CLS	A	P	R	F_1
GLRLM	PCA		DT	0.5372 ±0.0478	0.5449 ±0.0255	0.5372 ±0.0478	0.5381 ±0.0346
			RF	0.6323 ±0.0184	0.5815 ±0.0852	0.6323 ±0.0184	0.5473 ±0.0310
	GA		DT	0.5662 ±0.0469	0.5611 ±0.0478	0.5662 ±0.0469	0.5594 ±0.0433
			RF	0.6529 ±0.0239	0.5453 ±0.1135	0.6529 ±0.0239	0.5491 ±0.0319
GLCM	PCA		DT	0.5373 ±0.0641	0.5112 ±0.0818	0.5373 ±0.0641	0.5099 ±0.0543
			RF	0.5869 ±0.0497	0.5023 ±0.0721	0.5869 ±0.0497	0.5288 ±0.0545
	GA		DT	0.5949 ±0.0543	0.5880 ±0.0660	0.5949 ±0.0543	0.5785 ±0.0492
			RF	0.6491 ±0.0541	0.5573 ±0.0718	0.6491 ±0.0541	0.5681 ±0.0555
$GLCM_8$	PCA		DT	0.5327 ±0.0415	0.5326 ±0.0170	0.5327 ±0.0415	0.5301 ±0.0271
			RF	0.5910 ±0.0415	0.4733 ±0.0608	0.5910 ±0.0415	0.5143 ±0.0462
	GA		DT	0.5538 ±0.0569	0.5581 ±0.0564	0.5538 ±0.0569	0.5512 ±0.0553
			RF	**0.6695** ±0.0214	0.6318 ±0.0640	0.6695 ±0.0214	0.5938 ±0.0171

[a] the column marks the direction of the used features.

TABLE 1. Test result metrics for MIAS

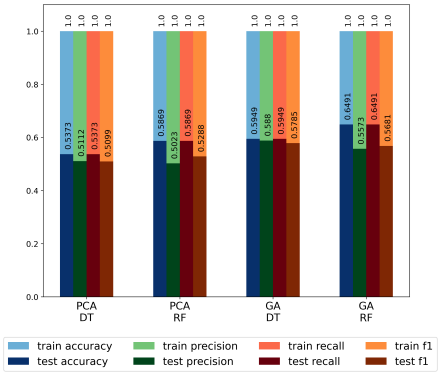


FIGURE 7. Result metrics using GLCM features calculated from MIAS images.

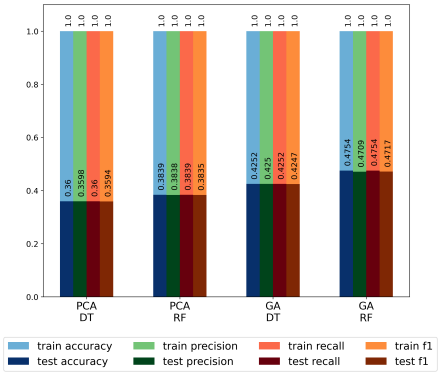


FIGURE 8. Result metrics using GLCM features calculated from DDSM images.

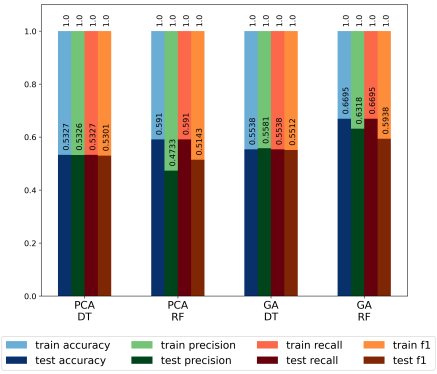


FIGURE 9. Result metrics using $GLCM_8$ features calculated from MIAS images.

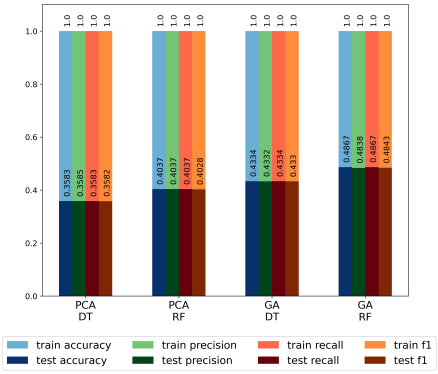


FIGURE 10. Result metrics using $GLCM_8$ features calculated from DDSM images.

From Table 1 (and Figures 5, 7 and 9) we can also see that the results using GA and RF are very close for all three feature extraction methods. However, in Table 2 (and Figures 6, 8 and 10) the results from GLCM and $GLCM_8$ are worse, hence, concluding the advantage of using GLRLM (run-length information) over GLCM (pixel correlation). When comparing the results of the two classifiers (DT/RF) we can see that a single DT obtains a lower accuracy value than RF. By using more models (DTs in the forest) we can increase the performance of the classification.

4.4. **Discussion.** In the previous subsection, the results of the proposed extended method have been compared to the system proposed in [2]. We now focus on a comparative analysis between the obtained results for MIAS and DDSM and relevant results from the literature.

In [3], the authors report 60.7% test accuracy using an approach based on GLCM features, correlation feature selection and RF classification (on MIAS). In our research, we achieved higher accuracy which can be explained by the different feature selection (GA) used in the proposed approach.

In [11] presents a comparison of different combinations of feature selection and classification (evaluated on MIAS). The best result reported was 70.53% using Local Binary Pattern and Deep Neural Network. This is comparable with our result. Results achieved with homogeneity, energy, HOG features and DNN, NB, NN, SVM classification are also reported in [11] with accuracies between 42.6% and 59.6%. These methods are outperformed by the proposed extended method.

FE	FS	Dir.[b]	CLS	A	P	R	F_1
GLRLM	PCA	↑↗	DT	0.4139 ±0.0255	0.4143 ±0.0252	0.4139 ±0.0255	0.4139 ±0.0254
		↑↗	RF	0.4498 ±0.0234	0.4487 ±0.0247	0.4498 ±0.0234	0.4487 ±0.0243
	GA	↗	DT	0.4839 ±0.0191	0.4845 ±0.0185	0.4839 ±0.0191	0.4838 ±0.0187
		↑→	RF	**0.5410** ±0.0248	0.5442 ±0.0225	0.5410 ±0.0248	0.5411 ±0.0239
GLCM	PCA	↗	DT	0.3600 ±0.0106	0.3598 ±0.0113	0.3600 ±0.0106	0.3594 ±0.0112
		↖↘	RF	0.3839 ±0.0175	0.3838 ±0.0182	0.3839 ±0.0175	0.3835 ±0.0180
	GA	→	DT	0.4252 ±0.0098	0.4250 ±0.0097	0.4252 ±0.0098	0.4247 ±0.0098
		↖↘	RF	0.4754 ±0.0124	0.4709 ±0.0123	0.4754 ±0.0124	0.4717 ±0.0127
$GLCM_8$	PCA	↖↘	DT	0.3583 ±0.0034	0.3585 ±0.0035	0.3585 x±0.0034	0.3582 ±0.0034
		↖	RF	0.4037 ±0.0231	0.4037 ±0.0233	0.4037 ±0.0231	0.4028 ±0.0234
	GA	→	DT	0.4334 ±0.0260	0.4332 ±0.0266	0.4334 ±0.0260	0.4330 ±0.0263
		↑→	RF	0.4867 ±0.0229	0.4838 ±0.0225	0.4867 ±0.0229	0.4843 ±0.0224

[b] the column marks the direction of the used features.

TABLE 2. Test result metrics for DDSM

In [10], the authors presented the results of CNNs and reported 95% accuracy for classifying images in MIAS into three classes and 96% on the DDSM. From this, we can see that with this basic classification further improvements are necessary to outperform ANNs.

5. Conclusions and Future Work

The scope of the current paper is to apply the CAD system presented in [2] to images from DDSM. However, some changes are needed to adapt the previously presented method to the new images. To determine the best parameters 5-fold cross-validation is applied. Based on the presented results we can see an overfitting of the classification (100% train metrics while around 60% test metrics). The extended system achieved 54.1% accuracy on DDSM using GLRLM, GA and RF. On MIAS, the best result is 66.95% from $GLCM_8$, applying GA

and RF. They are comparative with related results from the literature. However, the results are lower then expected, therefore the system needs further investigation.

As highlighted in [13] building a classifier using images from more datasets can lead to a more robust solution. Hence, in future work, we will investigate the result of the proposed solution on a combined input using images from MIAS and DDSM at the same time. Also, according to the same review, Random Forests and Decision Trees are least frequently used than other classifiers, such as Support Vector Machines and Artificial Neural Networks. Thus, we will dive deeper into these classifiers in order to explore their potential for breast cancer detection. Likewise, we will experiment building two separate classifiers that distinguish: (1) normal and abnormal and (2) benign and malignant tissues. In future work it would worth experimenting some typical failure cases and interpretation of behaviour.

Acknowledgment

This work was supported by a grant of the Romanian Ministry of Education and Research, CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2019-2607, within PNCDI III.

References

[1] Arora, R., Rai, P. K., and Raman, B. Deep feature–based automatic classification of mammograms. *Medical & Biological Engineering & Computing 58*, 6 (June 2020), 1199–1211.

[2] Bajcsi, A., Andreica, A., and Chira, C. Towards feature selection for digital mammogram classification. *Procedia Computer Science 192* (2021), 632–641.

[3] Bektas, B., Emre, I. E., Kartal, E., and Gulsecen, S. Classification of mammography images by machine learning techniques. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)* (2018), pp. 580–585.

[4] Boudraa, S., Melouah, A., and Merouani, H. F. Improving mass discrimination in mammogram-cad system using texture information and super-resolution reconstruction. *Evolving Systems 11*, 4 (Dec 2020), 697–706.

[5] Candra, D., Novitasari, R., Lubab, A., Sawiji, A., and Asyhar, A. H. Application of feature extraction for breast cancer using one order statistic, GLCM, GLRLM, and GLDM. *Advances in Science, Technology and Engineering Systems Journal 4*, 4 (2019), 115–120.

[6] Chaieb, R., and Kalti, K. Feature subset selection for classification of malignant and benign breast masses in digital mammography. *Pattern Analysis and Applications 22*, 3 (Aug. 2019), 803–829.

[7] Heath, M., Bowyer, K., Kopans, D., Moore, R., and Kegelmeyer, W. P. The digital database for screening mammography. In *Proceedings of the 5th International Workshop on Digital Mammography* (2001), M. Yaffe, Ed., Medical Physics, pp. 212—-218.

[8] Li, F., Shang, C., Li, Y., and Shen, Q. Interpretable mammographic mass classification with fuzzy interpolative reasoning. *Knowledge-Based Systems 191* (2020), 105279.
[9] Maitra, I. K., Nag, S., and Bandyopadhyay, S. K. Technique for preprocessing of digital mammogram. *Computer Methods and Programs in Biomedicine 107*, 2 (2012), 175–188.
[10] Malebary, S. J., and Hashmi, A. Automated breast mass classification system using deep learning and ensemble learning in digital mammogram. *IEEE Access 9* (2021), 55312–55328.
[11] Mathapati, M., Chidambaranathan, S., Nasir, A. W., Vimalarani, G., Rani, S. S., and Gopalakrishnan, T. An intelligent internet of medical things with deep learning based automated breast cancer detection and classification model. In *Cognitive Internet of Medical Things for Smart Healthcare*. Springer International Publishing, Oct. 2020, pp. 181–193.
[12] Mohanty, F., Rup, S., Dash, B., Majhi, B., and Swamy, M. N. S. Digital mammogram classification using 2D-BDWT and GLCM features with FOA-based feature selection approach. *Neural Computing and Applications 32*, 11 (June 2020), 7029–7043.
[13] Pedro, R. W. D., Machado-Lima, A., and Nunes, F. L. Is mass classification in mammograms a solved problem? - a critical review over the last 20 years. *Expert Systems with Applications 119* (2019), 90–103.
[14] Santos, G. PHOC descriptor applied for mammography classification. *Revista de Informática Teórica e Aplicada 27*, 1 (2020), 26–35.
[15] Suckling, J., Parker, J., and Dance, D. The mammographic image analysis society digital mammogram database. In *International Congress Series* (01 1994), vol. 1069, pp. 375–378.
[16] Xie, L., Zhang, L., Hu, T., Huang, H., and Yi, Z. Neural networks model based on an automated multi-scale method for mammogram classification. *Knowledge-Based Systems 208* (2020), 106465.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu, Cluj-Napoca 400084, Romania
Email address: adel.bajcsi@ubbcluj.ro
Email address: camelia.chira@ubbcluj.ro
Email address: anca.andreica@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 2, 2022
DOI: 10.24193/subbi.2022.2.02

FEASIBILITY OF USING MACHINE LEARNING ALGORITHMS FOR YIELD PREDICTION OF CORN AND SUNFLOWER CROPS BASED ON SEEDING DATE

ALINA DELIA CĂLIN, HOREA-BOGDAN MUREŞAN,
AND ADRIANA MIHAELA COROIU

Abstract. In this research, our objective is to identify the relationship between the date of seeding and the production of corn and sunflower crops. We evaluated the feasibility of using prediction models on a dataset of annual average crop yields and information on plant phenology, from several states of the US. After performing data analysis and preprocessing, we trained a selection of regression models. The best results were obtained for corn using HistGradientRegressor and XGBRegressor with $R^2 = 0.969$ for both algorithms and $MAE\% = 8.945\%$, respectively $MAE\% = 9.423\%$. These results demonstrate a good potential for the problem of yield prediction based on year, state, average plating day, and crop type. This model will be further used, combined with meteorological data, to build an agricultural crop prediction model.

1. Introduction

The problem of increasing crop yields and optimising agricultural production has become more relevant in the context of the growing population worldwide [27]. Recent years have added to this the rapid issues of climate change, which involve water shortages and soil erosion, which affect crop yield (with a projected decrease in corn production of 20-45% by 2100) [3].

Land degradation results in the reduction of available land for crops, unless it is rehabilitated in a sustainable way. Although many agronomic mitigating practises are being proposed, there must be an in-depth analysis as to which is

Received by the editors: 8 December 2022.

2010 *Mathematics Subject Classification.* 94A15, 94A99.

1998 *CR Categories and Descriptors.* H.1.1 [**Information Systems**]: MODELS AND PRINCIPLES – *Systems and Information Theory*; H.4.2 [**INFORMATION SYSTEMS APPLICATIONS**]: Types of Systems – *Decision support* I.2.1 [**ARTIFICIAL INTELLIGENCE**]: Applications and Expert Systems – *Medicine and science* .

Key words and phrases. regression, yield prediction, seeding date, agriculture, XGBoostRegressor.

the optimal approach, considering that they rely on economic viability, technical complexity, and the perception of the people involved [17]. Furthermore, it seems that several crops (for example corn), when using the *planting green* method, can become vulnerable to losses, while others are more stable (such as soybeans) [25].

Digital farming tools enhanced with artificial intelligence and machine learning models have the potential to help mitigate these issues and bring efficiency to crop management and protection. For example, they can reduce the usage of fungicides by up to 30% and the tank residues by up to 75%, through more precise calculations, thus reducing environmental pollution [28].

Furthermore, machine learning can be used successfully to identify factors that increase crop production under different environmental conditions, as well as model and predict future yields [20]. Many crops have a wide window of ideal plating date (60-90 days). However, crop success can also be influenced by changes in climate or soil composition; therefore, finding optimal planting windows in this context, with its associated risks, is a case-by-case problem for each crop and region [16]. The shortening of the planting window to a shorter optimum must be carried out for each region, according to the characteristics of the climate and hybrid type [5].

In this paper, our objective is to address the feasibility of using regression algorithms to predict corn and sunflower yields, based on the plating date and region, with limited available data. For this, we used historical crop data from several US states that were available online. These crops have been chosen as they are among the most widely cultivated in Romania. As algorithms have proven potential, our aim in future work is to gather Romanian specific crop data and apply machine learning algorithms for a more particular yield prediction.

In the following sections, we will present the related work for this specific problem, as well as the methods used in our experiments and the results obtained.

2. Related work

The problem of optimising yield based on seeding date is approached in many agricultural field researches with specific findings for each location, crop type, and climate particularities.

Patel et al. [24] analyse the effect of different sowing dates for rice crops, emphasising the importance of correlating the sowing date with the most favourable weather conditions of the region. The adaptation of the sowing date and the timing of management practises, as a result of climate change,

are also urged by looking at the effects of the sowing date of spring barley and maize in Germany and Poland [18].

For summer maize in irrigated crops, from the semi-arid Guanzhong region of China, an optimum planting date and water requirements for increased yield were modelled based on crop phenology, grain yield, above-ground biomass and leaf area index, using the Decision Support System for Agro Technology Transfer (DSSAT) Version 4.6 [26]. The results were statistically calculated on the observed data, obtaining a normalised root mean square error (nRMSE) of 9.91%, and $R^2 = 0.62$. An other study referring to summer maize in China regions argues that, given specific conditions of extreme heat, delaying the sowing date would improve the maize yield by 2–25% [15]. A late sowing date appears to be the best for overall yields for irrigated regions in the Mediteranean also [19], and a mid-to-late for the North China Plain [30].

When cultivating sunflower in Mediteranean climate conditions [12], the most effective was a later date, under rainfed conditions, but an early date in regions with little water availability. A study carried out on sunflower crops in Punjab, Pakistan [29], recommends earlier sowing dates for spring sowing and delayed sowing dates for fall, to mitigate the warming effects of climate change and ensure sustainable productivity.

A limitation of the studies presented above is that they only use a narrow data range of 1 to 3 years and a few plantations (1-5), which means that annual seasonal meteorological variations are not always accounted for [23]. Additionally, regional characteristics (soil type, climate) for the specific hybrids [2] will have an influence on yield, but field data is limited to one or two regions in most studies.

In this sense, machine learning can help model these problems, using different algorithms and techniques for data preprocessing and augmentation, and may leverage the effect of a single independent variable, which may not be obvious, in contrast to statistical models [20]. Algorithms such as Artificial Neural Networks, Support Vector Regression, k-Nearest Neighbour, Multiple Linear Regression, M5-Prime have been successfully applied with accurate results in estimating crop yield [13]. The accuracy metrics that are generally used for validation are Root Mean Square Error (RMS), Root Relative Square Error (RRSE), Normalised Mean Absolute Error (MAE) and Correlation Factor (R) or Coefficient of Determination (R^2 - basically, the square of the correlation coefficient) [7].

Yield prediction is a sub-field of crop management, and most research papers related to machine learning focus geographically on China, USA, India, and Brazil [4], with limited interest in European countries, especially Romania.

Alam et al. [1] use regression to determine the correlation between sowing dates and maize grain yield in Bangladesh, obtaining $R^2 = 0.972$. In Mourtzini et al. [20] process crop information for maize (n = 17,013) and soybean (n = 24,848) (including yield, crop management), and weather data involving 28 US states, between 2014 and 2018. The data was split in training (80%) and test (20%). The extreme gradient boosting (XGBoost) algorithm was trained to predict the yield based on the previous variables, resulting in a mean absolute error (MAE) of 4.7 with $R^2 = 0.94$ for maize, and MAE=6.4 with $R^2 = 0.92$ for soybean. Evaluation was performed using ten-fold cross-validation. A similar study uses the functional gradient descent algorithm on the data from the US corn and soybean, with a split of 85% training and 15% test [21]. The model predicts that early sowing dates can increase soybean yield by 10% in most US states, in the simulated context of climate change with a 30% reduction in precipitation during the summer months.

In the case of wheat yield prediction in Australia, Random Forest and Multiple Linear Regression models are used with meteorological data, identifying drought seasons as the main factor in yield losses [10]. The forecasts at 35 days before harvest were r = 0.85, MAPE = 17.6%, and 60 days before harvest r = 0.62, MAPE = 27.1%.

For the prediction of massive crop yields, Gonzalez-Sanchez et al. [13] analyse several algorithms on ten crop datasets. The M5-Prime Regression Trees and the k-Nearest Neighbour obtain the lowest average RMSE errors (5.14 and 4.91), the lowest average MAE errors (18.12% and 19.42%), and the highest average correlation (0.41 and 0.42), followed by Multiple Linear Regression, Multilayer Perceptron Neural Networks, and Support Vector Regression. In another study, Deshmukh et al. [9] analyse several algorithms (Random Forest, KNN, Naïve Bayes, XGBoost) for the top three crop recommendations for an optimised yield, of which XGBoost provides the best results.

3. Methods

Given the variety of approaches and results for different algorithms, we have selected the best potential for our specific data. Our approach focuses on the two types of crops: corn and sunflower.

3.1. **Data preprocessing.**

The data have been extracted from the US National Agricultural Statistics Service [31]. Although there are sufficient entries (with $n = 88808$ data instances) and many machine learning approaches for the corn data, as presented above, the sunflower data are limited to $n = 312$ instances for the sowing date and the corresponding yield, which is not sufficient for the accurate training of the regression model. Therefore, data preprocessing involved first the aggregation of yield data (in lb/ac) for each state and

year with the seeding date for both crops. The information available on the planting was in the format of percentage of the crop planted each week of the year. Using this information, a new feature was created as the average planting day of the year (D or AVGWeek), calculated as a weighted average based on the weekly percentages, with d_i the day of seeding and w_i the cumulative percentage planted up to that date d_i, with $w_0 = 0$ (see Equation (1) below). The resulting date was then translated into the day of year (with January 1 being the first day of year).

$$D = \frac{1}{n} \sum_{i=1}^{n} d_i (w_i - w_{i-1}) \tag{1}$$

Data cleaning was also performed to remove inconsistent field values or incomplete entries. As a result, we obtained $n = 88808$ instances for corn, for several years from 1979 to 2022. This is described in Figure 1.

For sunflower, given the available yield data, the information that was not available on the seeding date was completed using the forward fill method, which means similar planting dates for the same state, thus resulting $n = 1108$ instances, for several years in the range 1950 to 2021. The data description plots for sunflower are shown in Figure 2.

As a result, for each crop dataset, an entry contains 5 features: the year (type int), the state ANSI (type int), the crop type (Irrigated, Non-irrigated and Total, which have been encoded using the factorise method), the yield (of type float) and the planting day of year (of type float). From these data, we can extract and visualise the plating day-of-year interval window used in each state, across the reference years. In Figure 3, it can be observed that there are fewer states available for the sunflower dataset. The planting days are from day 80 to day 170 for corn and from day 135 to day 170 for sunflower, with the width of the planting windows ranging from 15 to 40 days.

Furthermore, the correlation matrices were also calculated and are shown in Figures 4 and 5. For corn, we note a 0.22 positive correlation between the yield and year, and a negative correlation between crop type and yield.

In the case of sunflower, we observe a 0.57 positive correlation between year and yield value, and 0.25 between year and average planting day of year. The correlation between the yield value and the average day of planting is 0.13, and there is a negative correlation between the type of crop and the yield value.

Given the correlation in both datasets between year and yield, we have analysed the line plots of crop yield for each state per year, in Figure 6. It can be observed that the overall tendency is that yield increases over time. As this might be due to several factors, including technological agricultural advancements (availability of machinery, fertilisers, pesticides, new hybrids,

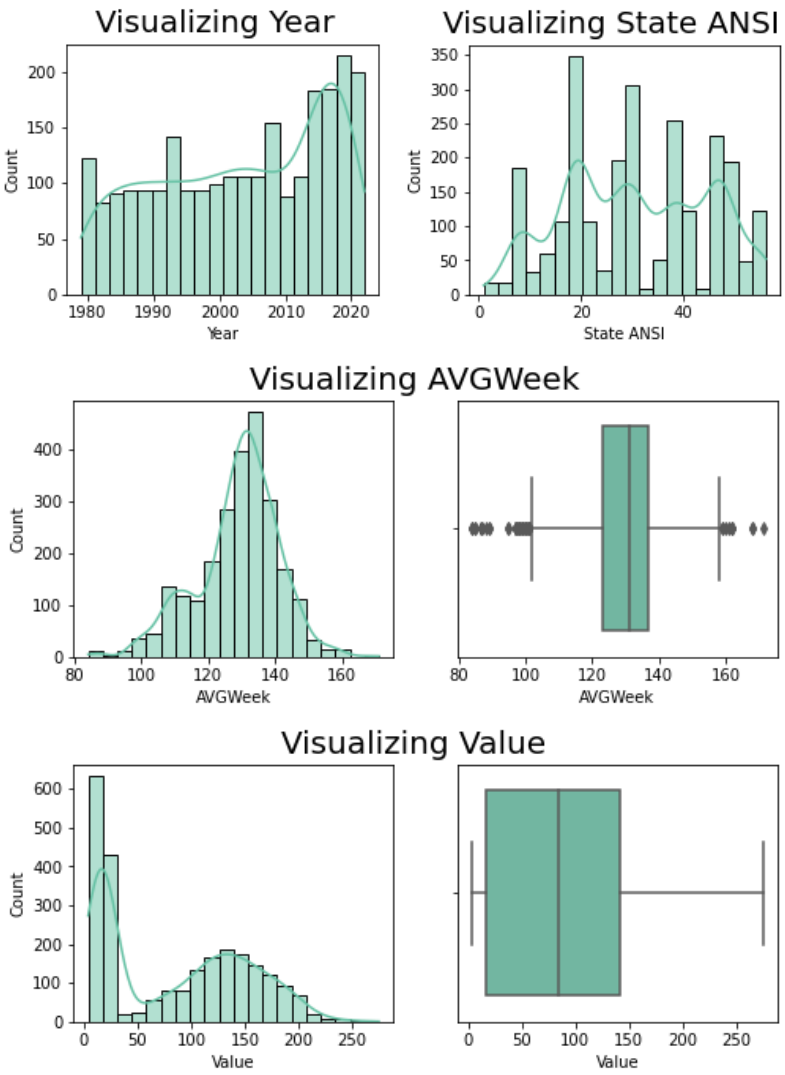


FIGURE 1. Feature description for the corn crop dataset features: Year, State ANSI (or state numerical code), AVGWeek (the average plating day of year) and Value (the yield value).

etc.), we decided to perform two experiments: one in which all features are involved in yield prediction, and one in which the year feature is removed, remaining only state code, crop type, and average plating day to be considered in the prediction of the crop yield.

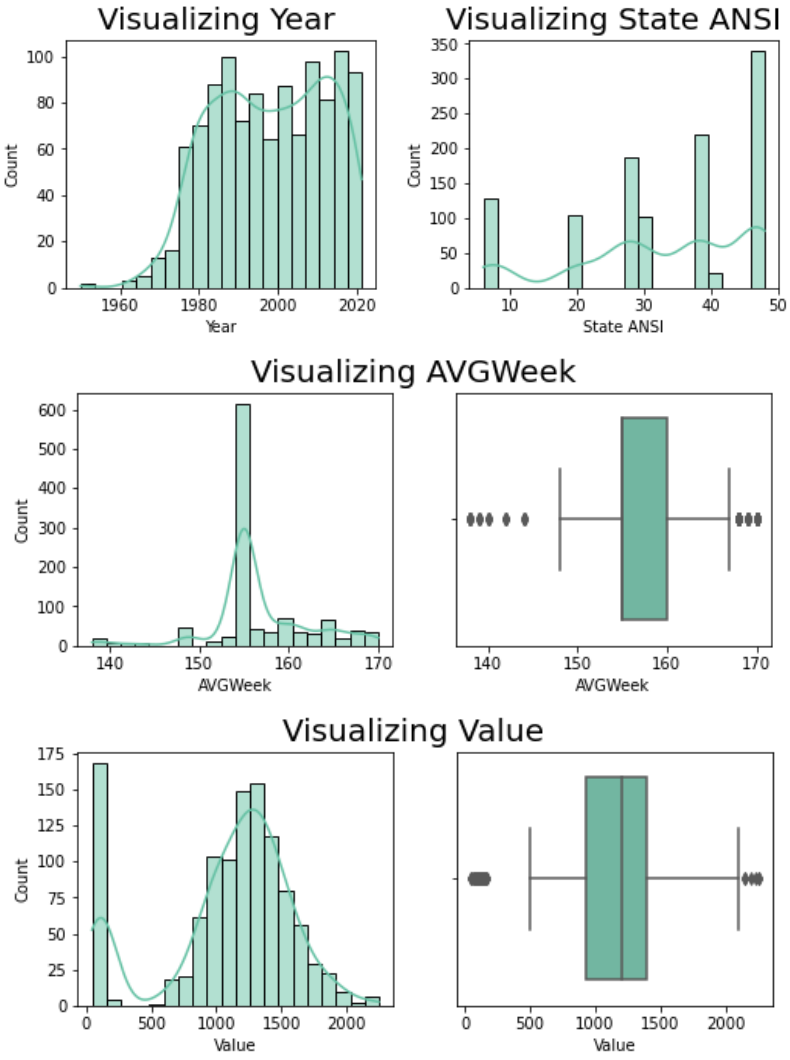


FIGURE 2. Feature description for the sunflower crop dataset: Year, State ANSI (or state numerical code), AVGWeek (the average plating day of year) and Value (the yield value).

3.2. **Models training.** Based on the literature review, we have selected several regression algorithms that are appropriate for our specific problem.

3.2.1. *DecisionTreeRegressor.* A Decision Tree is built in the form of a tree-like hierarchical structure, containing internal nodes (or decision nodes) and

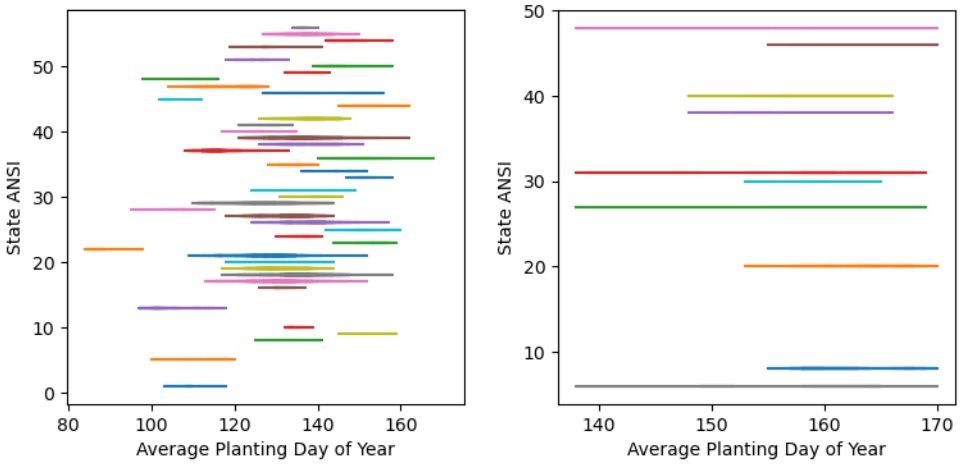


FIGURE 3. The average planting day of year window in each state (based on state ANSI) for corn (left) and sunflower(right).

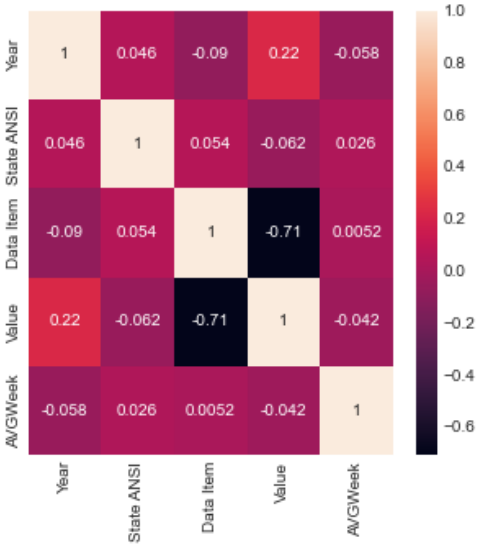


FIGURE 4. Correlation heat map for the corn crop dataset

leaf nodes (or prediction nodes). The height and width of such a tree depend on the data characteristics, amount, and algorithm configuration. In the case

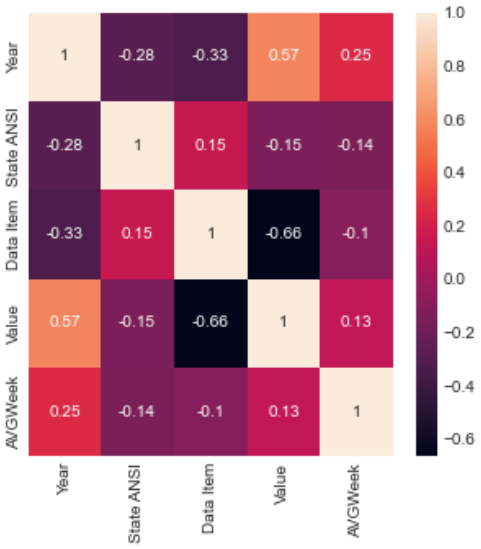


FIGURE 5. Correlation heat map for the sunflower crop dataset

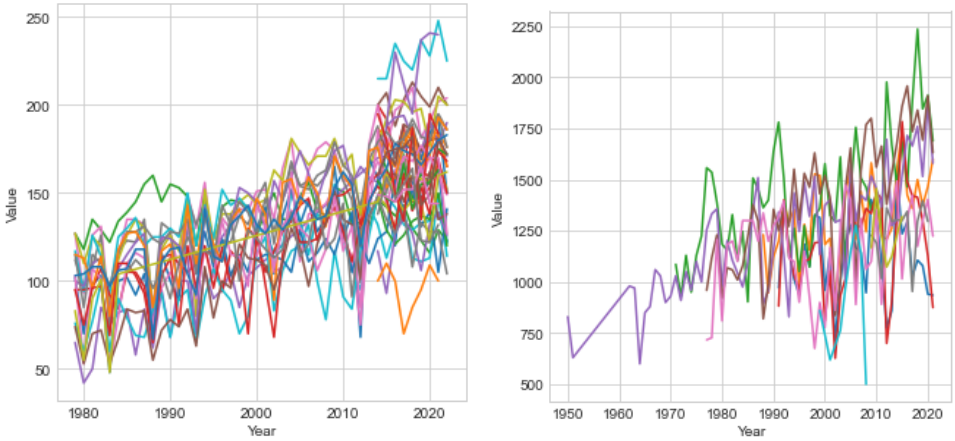


FIGURE 6. Yield value per year for each state for corn (left) and sunflower (right).

of prediction, we use an input x_i to go down the tree using decision nodes, up to a leaf that contains the predicted y_i value [8].

3.2.2. *RandomForestRegressor.* Random forest is an algorithm that groups an ensemble of growing decision trees, depending on a random vector $f(\phi)$. Thus,

the predictions are performed by aggregating the predictions of all the decision trees $h(x, \phi)$ [14].

3.2.3. *HistGradientBoostingRegressor.* The Histogram-based Gradient Boosting Regression Tree also uses ensemble decision trees. It improves performance by adding new corrective models in a greedy stepwise manner, with the aim of reducing the square error loss function until it is acceptable [11]. The histogram is an efficient data structure used by the tree-building algorithm to accelerate the process.

3.2.4. *XGBRegressor.* The XGBoost is a scalable end-to-end Gradient Boosting Tree system that is using cache access patterns, data compression, and block sharding to optimise resource use [6].

3.2.5. *Models training and hyperparameters tuning.* The algorithms presented above have been trained and tested using a data split of 20% for testing and 80% for training. Next, the parameters were fine-tuned; for XGBRegressor the number of estimators was set to 500, max depth to 8 and learning rate to 0.1. HistGradientBoostingRegressor used a learning rate of 0.01, with the maximum iterations set to 1000 and the loss function Poisson. RandomForestRegressor was set to use 25 estimators with a maximum of 4 features and at most 700 leaf nodes, with a random state of 45. For DecisionTreeRegressor the max depth was set to 10, the other parameters being as default by the Sklearn Python library implementation.

3.3. **Models evaluation.** For the model evaluation we used the k-fold cross-validation with $k = 10$. We note the mean absolute error (MAE), which is calculated as an average of the absolute prediction error as in Equation (2), where y_i is the observed value and $\hat{y}_i$ is the predicted value.

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

Another metric used is the coefficient of determination R^2, which represents how much of the variation in the y values (yield in our case) is taken into account by the involved features, computed as in Equation (3), where $\bar{y}$ is the mean of the observed values [22].

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

We also computed MAE%, computed by dividing MAE by the average yield value for corn and sunflower, respectively, as described by Formula (4):

$$MAE\% = \frac{MAE}{\frac{1}{n}\sum_{i=1}^{n} y_i} * 100 \tag{4}$$

4. Results and Discussion

The results obtained by each model are presented in Table 1 below, in which we provide the values for the mean (M) and standard deviation (SD), for the datasets including all features or without year (w/t year). The best yield prediction was obtained by the XGBRegressor (with $R^2 = 0.969$ for corn and $R^2 = 0.905$ for sunflower) and HistGradientBoostingRegressor ($R^2 = 0.969$, for corn and $R^2 = 0.884$ for sunflower), when all features were included.

While the R^2 score is lower when the year is removed, it is still greater than 0.9 for corn, with the best result for sunflower being 0.815.

The prediction plot is visible in Figure 7 for the corn dataset, including both experiments: using all features, and when removing the input feature that represents the year. Figure 8 presents the regression plot for sunflower with all features and, respectively, without year. In both figures, the x axis represents the actual yield, and the y axis is the predicted yield.

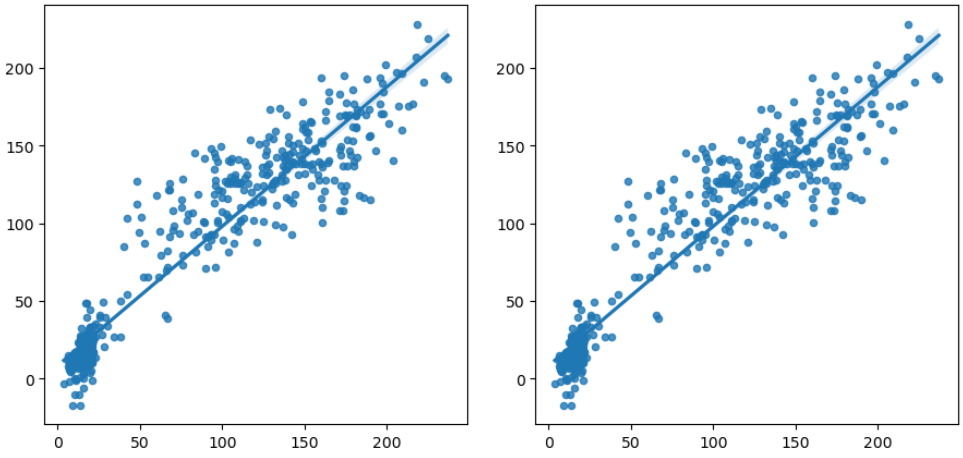


Figure 7. Corn crop yield prediction plot using all features (left), and without year (right), using the XGBRegressor model

Based on these results, we can state that the algorithms used in the prediction reveal models with a good correlation of the selected characteristics (year, state, plantation date, type of crop) with yield. From these, the year and the planting day appear to be both relevant features in predicting the yield for a

TABLE 1. The results on yield prediction for each algorithm

	XGBRegressor				
	R^2 **M**	R^2 **SD**	**MAE M**	**MAE SD**	**MAE%**
Corn	0.969	0.006	7.864	0.634	9.423
Sunflower	0.905	0.028	106.778	10.473	9.768
Corn (w/t Year)	0.907	0.013	14.299	0.976	17.134
Sunflower (w/t Year)	0.815	0.047	156.957	13.515	14.359

	HistGradientBoostingRegressor				
	R^2 **M**	R^2 **SD**	**MAE M**	**MAE SD**	**MAE%**
Corn	0.969	0.006	7.465	0.619	8.945
Sunflower	0.883	0.032	119.289	10.838	10.913
Corn (w/t Year)	0.909	0.015	12.736	1.187	15.261
Sunflower (w/t Year)	0.797	0.047	167.133	13.687	15.290

	DecisionTreeRegressor				
	R^2 **M**	R^2 **SD**	**MAE M**	**MAE SD**	**MAE%**
Corn	0.940	0.010	10.009	0.776	11.993
Sunflower	0.861	0.039	124.517	13.345	11.391
Corn (w/t Year)	0.870	0.021	14.587	1.139	17.479
Sunflower (w/t Year)	0.753	0.056	180.455	15.437	16.509

	RandomForestRegressor				
	R^2 **M**	R^2 **SD**	**MAE M**	**MAE SD**	**MAE%**
Corn	0.959	0.007	8.483	0.700	10.164
Sunflower	0.905	0.026	107.432	10.457	9.828
Corn (w/t Year)	0.891	0.018	13.779	1.235	16.510
Sunflower (w/t Year)	0.794	0.043	168.073	13.256	15.376

specific geographical location (state). These are in agreement with the results obtained and the findings described in the state-of-the-art literature presented in Section 2.

We also note the high metrics obtained for predicting the sunflower crop yield, where the available data were limited, which means that the total number of instances was $n = 1108$. Of these only $n = 312$ contained complete information regarding the plating day, the others were completed by our algorithms in the preprocessing phase. This is an important finding, because no other study has performed similar experiments on such a small number of

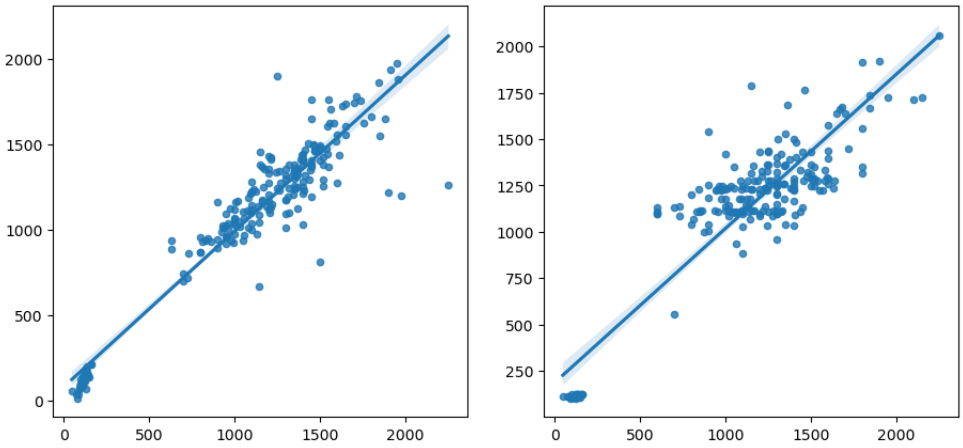


FIGURE 8. Sunflower crop yield prediction plot with all features (left), and without year (right), using the XGBRegressor model

instances for a crop dataset, considering our goal is to train models on Romanian crop datasets also, for which the data are being collected, and are expected to be reduced in size. This is due to the fact that the collection is not yet being centralised by a national statistical organisation, but privately gathered by smaller independent agricultural entities for their own research and seasonal activity.

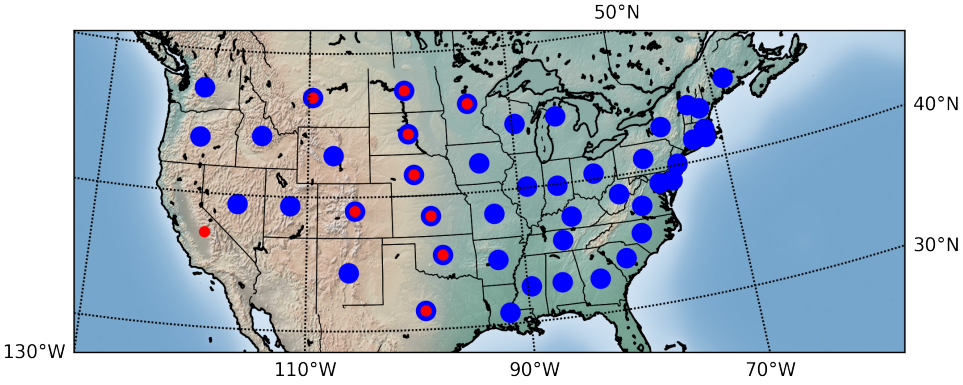


FIGURE 9. Geographical location of the available data. Blue dots represent states with corn crop data. Red dots are states with sunflower crop data.

5. Conclusions and future work

In this paper, we analysed two crop datasets for the purpose of predicting the yield based on the plating date. We obtained the best score of $R^2 = 0.969$ for the corn dataset ($n = 88808$) and $R^2 = 0.905$ for the sunflower dataset ($n = 1108$), using XGBRegressor.

To better isolate the effect of the plating date from other factors, such as technical advances throughout the years (especially given the wide range of years of available data and the tendency of yield increase throughout the years), we repeated the experiments without the year component. In this case, the best results were obtained using the HistGradientBoostingRegressor for the corn dataset ($R^2 = 0.909$).

Given the results obtained, we conclude that the plating day of year has a significant influence on crop yield prediction, for both corn and sunflower datasets. We also note that it is feasible to use regression algorithms to successfully predict crop yield even in cases where the available data are limited (as in the case of the sunflower crop), using adequate data preprocessing techniques. This finding is relevant for our planned work, because we expect the initial available data to be reduced and perhaps incomplete.

As next steps, we aim to collect several crop data sets specific for the Romanian agricultural sector and train predictive models adapted to geographical and crop particularities.

Also, considering the literature in the field stating that the seeding date might be in itself influenced by specific climate changes or meteorological seasonal variations, further work also involves correlating these parameters, in the context of global warming and its effects in agriculture.

6. Acknowledgement

This work was supported by the project "The Development of Advanced and Applicative Research Competencies in the Logic of STEAM + Health" /POCU/993/6/13/153310, project co-financed by the European Social Fund through The Romanian Operational Programme Human Capital 2014-2020.

References

[1] Alam, M. J., Ahmed, K. S., Nahar, M. K., Akter, S., and Uddin, M. A. Effect of different sowing dates on the performance of maize. *Journal of Krishi Vigyan 8*, 2 (2020), 75–81.

[2] Ali, W., Ali, M., Ahmad, Z., Iqbal, J., Anwar, S., and Kamal, M. K. A. Influence of sowing dates on varying maize (zea mays l.) varieties grown under agro-climatic condition of peshawar, pakistan. *European Journal of Experimental Biology 8*, 6 (2018), 36.

[3] Arora, N. K. Impact of climate change on agriculture production and its sustainable solutions. *Environmental Sustainability 2* (2019), 95–96.

[4] Benos, L., Tagarakis, A. C., Dolias, G., Berruto, R., Kateris, D., and Bochtis, D. Machine learning in agriculture: A comprehensive updated review. *Sensors 21*, 11 (2021), 3758.

[5] Cerioli, T., Gentimis, T., Linscombe, S. D., and Famoso, A. N. Effect of rice planting date and optimal planting window for southwest louisiana. *Agronomy Journal 113*, 2 (2021), 1248–1257.

[6] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 785–794.

[7] Chlingaryan, A., Sukkarieh, S., and Whelan, B. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. *Computers and electronics in agriculture 151* (2018), 61–69.

[8] Desai, A., and Chaudhary, S. Distributed decision tree. In *Proceedings of the 9th Annual ACM India Conference* (2016), pp. 43–50.

[9] Deshmukh, M., Jaiswar, A., Joshi, O., and Shedge, R. Farming assistance for soil fertility improvement and crop prediction using xgboost. In *ITM Web of Conferences* (2022), vol. 44, EDP Sciences, p. 03022.

[10] Feng, P., Wang, B., Li Liu, D., Waters, C., Xiao, D., Shi, L., and Yu, Q. Dynamic wheat yield forecasts are improved by a hybrid approach using a biophysical model and machine learning technique. *Agricultural and Forest Meteorology 285* (2020), 107922.

[11] Geurts, P., Wehenkel, L., and d'Alché Buc, F. Gradient boosting for kernelized output spaces. In *Proceedings of the 24th international conference on Machine learning* (2007), pp. 289–296.

[12] Giannini, V., Mula, L., Carta, M., Patteri, G., and Roggero, P. P. Interplay of irrigation strategies and sowing dates on sunflower yield in semi-arid mediterranean areas. *Agricultural Water Management 260* (2022), 107287.

[13] Gonzalez-Sanchez, A., Frausto-Solis, J., and Ojeda-Bustamante, W. Predictive ability of machine learning methods for massive crop yield prediction. *Spanish Journal of Agricultural Research 12*, 2 (2014), 313–328.

[14] Hsu, C.-C., Lee, Y.-C., Lu, P.-E., Lu, S.-S., Lai, H.-T., Huang, C.-C., Wang, C., Lin, Y.-J., and Su, W.-T. Social media prediction based on residual learning and random forest. In *Proceedings of the 25th ACM international conference on Multimedia* (2017), pp. 1865–1870.

[15] Lv, Z., Li, F., and Lu, G. Adjusting sowing date and cultivar shift improve maize adaption to climate change in china. *Mitigation and Adaptation Strategies for Global Change 25*, 1 (2020), 87–106.

[16] Ma, B., Zhao, H., Zheng, Z., Caldwell, C., Mills, A., Vanasse, A., Earl, H., Scott, P., and Smith, D. Optimizing seeding dates and rates for canola production in the humid eastern canadian agroecosystems. *Agronomy Journal 108*, 5 (2016), 1869–1879.

[17] Malhi, G. S., Kaur, M., and Kaushik, P. Impact of climate change on agriculture and its mitigation strategies: A review. *Sustainability 13*, 3 (2021), 1318.

[18] Marcinkowski, P., and Piniewski, M. Effect of climate change on sowing and harvest dates of spring barley and maize in poland. *International Agrophysics 32*, 2 (2018).

[19] Maresma, A., Ballesta, A., Santiveri, F., and Lloveras, J. Sowing date affects maize development and yield in irrigated mediterranean environments. *Agriculture 9*, 3 (2019), 67.

[20] Mourtzinis, S., Esker, P. D., Specht, J. E., and Conley, S. P. Advancing agricultural research using machine learning algorithms. *Scientific reports 11*, 1 (2021), 1–7.

[21] Mourtzinis, S., Specht, J. E., and Conley, S. P. Defining optimal soybean sowing dates across the us. *Scientific Reports 9*, 1 (2019), 1–7.

[22] Pan, B. Application of xgboost algorithm in hourly pm2. 5 concentration prediction. In *IOP conference series: earth and environmental science* (2018), vol. 113, IOP publishing, p. 012127.

[23] Partal, E. Sunflower yield and quality under the influence of sowing date, plant population and the hybrid. *ROMANIAN AGRICULTURAL RESEARCH 39* (2022), 463–470.

[24] Patel, A., Patel, M., Patel, R., Mote, B., et al. Effect of different sowing date on phenology, growth and yield of rice–a review. *Plant Archives 19*, 1 (2019), 12–16.

[25] Reed, H. K., Karsten, H. D., Curran, W. S., Tooker, J. F., and Duiker, S. W. Planting green effects on corn and soybean production. *Agronomy Journal 111*, 5 (2019), 2314–2325.

[26] Saddique, Q., Cai, H., Ishaque, W., Chen, H., Chau, H. W., Chattha, M. U., Hassan, M. U., Khan, M. I., and He, J. Optimizing the sowing date and irrigation strategy to improve maize yield by using ceres (crop estimation through resource and environment synthesis)-maize model. *Agronomy 9*, 2 (2019), 109.

[27] Sadigov, R. Rapid growth of the world population and its socioeconomic results. *The Scientific World Journal 2022:8110229* (2022).

[28] Shankar, P., Werner, N., Selinger, S., and Janssen, O. Artificial intelligence driven crop protection optimization for sustainable agriculture. In *2020 IEEE/ITU International Conference on Artificial Intelligence for Good (AI4G)* (2020), IEEE, pp. 1–6.

[29] Tariq, M., Ahmad, S., Fahad, S., Abbas, G., Hussain, S., Fatima, Z., Nasim, W., Mubeen, M., ur Rehman, M. H., Khan, M. A., et al. The impact of climate warming and crop management on phenology of sunflower-based cropping systems in punjab, pakistan. *Agricultural and Forest Meteorology 256* (2018), 270–282.

[30] Tian, B., Zhu, J., Nie, Y., Xu, C., Meng, Q., and Wang, P. Mitigating heat and chilling stress by adjusting the sowing date of maize in the north china plain. *Journal of Agronomy and Crop Science 205*, 1 (2019), 77–87.

[31] United States Department of Agriculture, U. National agricultural statistics service. https://quickstats.nass.usda.gov/. Accessed: 2022-10-17.

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

Email address: alina.calin@ubbcluj.ro

Email address: horea.muresan@ubbcluj.ro

Email address: adriana.coroiu@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 2, 2022
DOI: 10.24193/subbi.2022.2.03

COROUTINES COMUNICATIONS. DESIGN AND IMPLEMENTATION ISSUES IN C++20

RADU LUPŞA AND DANA LUPŞA

Abstract. This paper explores the communication mechanisms and patterns available to coroutines to cooperate with one another. It investigates the issues in designing and implementing a framework for using C++20 coroutines effectively, for generators, asynchronous function calls, and especially asynchronous generators.

1. Introduction

Coroutines are a programming concept that allows execution to be suspended and resumed. A coroutine can transfer the control to other piece of code and, when getting back the control at a later time, it continues from the next instruction and with all the data restored, including the execution stack.

Coroutines were originally proposed in 1963 and also were studied quite thoroughly in 1960's to beginning of 1980's. The interest to coroutines has resurfaced in recent years, with several mainstream languages offering some support to coroutines.

Even though coroutines dates back to 1963 [9] , they are still found useful in modern applications. For example, a recent work on coroutine study the use of corotuine for a web crawler [15]. Coroutines were proven to perform better than single-threaded and multi-threaded versions. Performance improvement was also demonstrated for coroutine use in `I/O` requests [14]. When used on Android, coroutines achieve better performance when compared to some existing concurrency frameworks [8]. Also, lot of recent academic works study the application of coroutines on resource-constrained platforms, in the Internet of Things and Embedded Systems [7].

Received by the editors: 4 December 2022.

2010 *Mathematics Subject Classification.* 68N19, 68Q85.

1998 *CR Categories and Descriptors.* D.3.3 [**Software**]: Programming Languages – *Language Constructs and Features*; D.1.3 [**Software**]: Programming Techniques – *Language Concurrent Programming*.

Key words and phrases. asynchronous programming, coroutines, language design.

While there are a lot of works exploring the low level aspects and the applicable aspects of using coroutine in asyncronous processing, not so much is studied about higher-level abstractions and patterns. However, in [13] there is a review about the following patterns: The Producer–Consumer Problem, Generator, Goal-oriented programming, Cooperative Multitasking. They also discuss Exception Handling, but this is more of CPS than coroutines. [10] also present an implementation approach to asynchronous programming and generator based on coroutines.

The coroutine primitives provided in C++20 are very powerful, but they are very complicated to use directly. In this paper, we explore how to use them to build the producer-consumer patterns and cooperative multitasking.

The rest of the paper is organized as follows. The next section makes an overview of the most known coroutine classification and presents the characteristics of coroutines existing in some programming languages. Section 3 details two main scenarios for inter-coroutine communication. Section 4 proposes a framework over C++20 coroutines allowing several use cases: to make an asynchronous call and switch to some other coroutine until the asyncronous call completes, to have a coroutine yielding a sequence of values (to implement a generator) and to combine those two features together. The paper ends with a short review over the main things that are presented in this paper.

2. What is a Coroutine

Essentially, a coroutine is an execute thread, together with the notion of current instruction, execution stack together with arguments, local and temporary variables. However, unlike threads, switching to or from a coroutine is done at the current coroutine request, instead of unpredictable, whenever the operating system decides to.

According to [12], the characteristic property of a coroutine is that it can transfer the control (yield) to other code and, when getting back the control at a later time, it continues from the next instruction and with all the data restored, including the execution stack.

There are 3 important classification criteria for coroutine support in a coroutine implementation [13], [10]:

symmetric vs asymetric: : in symmetric coroutine implementation, any coroutine can transfer control to any other coroutine; with asymmetric coroutines, a caller coroutine can transfer control to a subordinate coroutine, while the subordinate can only transfer back to the caller (or to own subordinates).

stackfull vs stackless: : a stackless coroutine can transfer control only from its main function; a stackfull one can transfer control from

within a called function at any depth. However, as a partial workaround, a stackless coroutine can create and call a child stackless coroutine, which can then transfer control to some other coroutine.

first class vs constrained: : a first-class coroutine is explicitly manipulated by the programmer, via handles that can be passed around and stored in variables; constrained coroutines exist only implicit, within some programming constructs such as a *generator* created based on *yield* statements and iterated within a *foreach* loop, or the *async-await* mechanism, both introduced in C# [4], [5] and also available in other languages such as Python.

2.1. **Existing coroutine support in programming languages.** *Windows Fibers* [6] are symmetric, stackfull, first-class coroutines. The API is constructed for the C language. The basic operations are: `CreateFiber()`, that creates a coroutine, given its main function, `SwitchToFiber()`, that suspends the current coroutine giving control to the specified coroutine, and `DeleteFiber()`, that deletes the specified coroutine.

Lua coroutines [11] are asymmetric, stackfull, first-class coroutines. The basic operations are: `coroutine.create()` that creates a child coroutine, `coroutine.resume()`, that transfers control to the specified child coroutine, and `coroutine.yield()`, that transfer control to the parent coroutine. An interesting feature is that an additional parameter given to `coroutine.resume()` (beside the coroutine identifier) gets retrieved inside the coroutine as the return value of `coroutine.yield()` and, vice-versa, any parameter given to `coroutine.yield()` can be retrieved in the parent coroutine as the returned value from `coroutine.resume()`.

C# coroutines [4], [5] are stackless, constrained coroutines (symmetry is a bit unclear). They occur under two forms: *generators* and the *async-await* mechanism. A generator looks like a function returning an `IEnumerable`. The parent coroutine transfers the control to the child by a call to `MoveNext()` on the corresponding iterator; the control is transferred back to the parent by a `yield return` statement in the generator. In an *async-await* scenario, the mechanism is more complex and involves threads and some thread pool mechanism in addition to the coroutines. A coroutine is marked with the `async` keyword and must return a `Task` or a `Task<T>`. An `await` statement inside a coroutine suspends it and may transfer control to a coroutine that is runnable at that time. The coroutine is resumed when the awaited future is completed.

C++20 coroutines [1] are symmetric, stackless, first-class coroutines, although the mechanism for controlling them is complex, poses some constraints

and offers distinct mechanisms for symmetric and asymmetric transfer of control. A coroutine main function is identified by having one or more of the `co_await`, `co_yield` or `co_return` statements within its body. The declared return type of a coroutine main function is a user-defined type that serves two purposes: on one hand, it is created when starting the coroutine and should be a wrapper over the coroutine handle, and, on the other hand, it must declare an inner class, `promise_type`, containing some member functions that control the behavior of the coroutine. It is those functions that decide if and to which coroutine should the control be transferred as a result of `co_await`, `co_yield` or `co_return`. These functions also allow data to be transmitted between coroutines in a user-customised way, allowing the construction of higher level mechanisms.

3. Communication between coroutines

Regardless of the lower level mechanisms of handling coroutines, there are two higher level patterns that cover most uses of coroutines:

- Producer-consumer scenarios, where one coroutine produces values for the consumption of another, with passing control together with the values.
- As "poor man's threads", that is to switch from a task, that cannot be continued because it depends on some external data, to another task, that can be continued.

3.1. **Producer-consumer scenario.** In this scenario, there is a producer and a consumer, and both are written as in full control, each having a main loop. Thus, the producer will have the main loop and will repeatedly execute a statement (usually called *yield*) that pushes a value to the consumer; the consumer also has the main loop and repeatedly pulls data from the producer, often just through a special form of a *for* loop. The producer push (*yield*) operation needs to both give data and switch control to the consumer coroutine; the consumer pull operation needs to give control to the producer and, when the control is transferred back, to return the data pushed by the producer.

3.2. **Poor man's threads scenario.** This scenario considers several independent operations. We want that, when one operation is blocked waiting for an asynchronous operation, to schedule another. Each operation has its own coroutine and decides when it can pass the control to another. This way, coroutines are used as a cooperative multitasking mechanism.

Each coroutine acts as a line of execution, similar to a thread. Its main advantage, though, is that it is handled in user space, which means that it is cheaper to switch from a coroutine to another. Also, since control is yielded

only at some specific points within the code, concurrency control can be relaxed.

When a coroutine executes an asynchronous operation (an operation that has to wait for an external event, such as read from input, from a socket, sleep for an amount of time, wait for a computation executed on some other thread), the coroutine must do three operations:

(1) start the asynchronous operation;
(2) arrange so that the completion notification of the operation marks the coroutine runnable again;
(3) invokes a scheduler that switches to a runnable coroutine.

It is important to note that, in this scenario, coroutines are mostly independent from one another, so, switching from a coroutine to the next is not accompanied with passing information. On the contrary, even deciding which coroutine is the next one is independent of the main bussines of the current coroutine, so, it is delegated to a scheduling mechanism. Furthermore, the scheduler can also move a coroutine from one thread to another.

4. Designing asynchronous generators

The basic usages of constrained coroutines in programming languages like C# and Python is for *generators* and for the *async-await* pattern.

We will look into creating these mechanisms with C++20 coroutines and combining them to form asynchronous generators.

Note that the C# language, that invented the async-await mechanism, does not support asynchronous generators yet, although the work in this direction is under way. Python has it implemented [3], and it comes in a straightforward way from the generators and the async-await. Both use a scheduler mechanism to decide which coroutine gets the control when the current one gets suspended, but neither gives the programmer explicit control to choose the scheduler.

There also exists an implementation for asynchronous generators using C++20 coroutines — the `CppCoro` project [2]. However, our framework has two important differences over what `CppCoro` provides:

(1) we provide primitives for conveying data from sources or to destinations that are not implemented as coroutines, while `CppCoro` only provides the possibility for a non-coroutine source to signal to a coroutine that it can proceed, and similar for a coroutine source to a non-coroutine consumer;
(2) we allow a scheduler to decide which coroutine would take over the current thread when the current coroutine gets suspended, thus using the symmetric coroutines mechanism in C++20 coroutines; `CppCoro`

returns the control to the parent (or latest resumer), using the asymmetric coroutine mechanism in C++20 coroutines.

4.1. **Implementing a *generator* mechanism.** The common way the communications and coroutines work together is the *generator* mechanism. We implemented a small C++20 framework allowing an easy way to write a generator — as a coroutine function that produces values and pushes them via the `co_yield` statement to the consumer — and the consumer code — that can consume the elements via the standard iterators mechanism, for example as a simple *foreach* style `for` loop.

The proposed implementation has the following elements:

- The returned type for the producer coroutines is a template over the produced objects, `Generator<T>`;
- The `Generator<T>` object contains only the coroutine handle;
- The `Generator<T>::promise_type` holds the value between the producer and the consumer, as an `std::optional<T>`. An empty optional signifies the end of data.
- The producer coroutine starts suspended (`initial_suspend()` in `promise_type` returns `std::suspend_always`). This way, we have a lazy evaluation mechanism — the values are produced on demand.
- The consumer calls a function `next()` defined in `Generator<T>`. This resumes the coroutine (suspending the caller).
- The producer coroutine sends the data items via the `co_yield`, invoking the `yield_value()` in the `promise_type`, which stores the value and suspends the producer coroutine, resuming the caller (consumer) code.
- The `next()` call returns in the value in the consumer code.
- At the end, the producer coroutine ends, leading to the runtime invoking `return_void()` in the `promise_type`. This stores a null optional, suspends the producer coroutine and resumes the consumer. The consumer must not invoke `next()` after a null was returned.
- The `next()` call is wrapped in a standard STL iterator.

The above mechanism allows, for instance, a straightforward way to implement a permutations generator. Note that the coroutine function here is recursive, so it acts both as a producer and as a consumer, so it demonstrates both usages:

```
Generator<std::vector<int> >
permutations_rec(std::vector<int> const& prefix, int n) {
    std::vector<int> newPrefix = prefix;
    newPrefix.push_back(0);
    for (int i = 0; i < n; ++i) {
```

```
        if (std::find(prefix.begin(), prefix.end(), i) ==
        prefix.end()) {
            newPrefix.back() = i;
            if (newPrefix.size() == n) {
                co_yield newPrefix;
            } else {
                for (auto perm : permutations_rec(newPrefix, n)) {
                   co_yield perm;
                }
            }
        }
    }
}
```

Like for other coroutine-based generators, the downside is that the generator is not copyable, so, the user cannot make a copy at a certain point and have both the original and the copy generate independently the remaining elements.

4.2. **Async-await pattern.** The next element we need is to implement a framework allowing to make an asynchronous call and switch to some other coroutine until the asyncronous call completes.

Its goal is to be able to write something like `v = co_await func()`, where `func()` is an asynchronous operation returning a *future*, with the following effect:

(1) `func()` is called, starting some asynchronous operation;
(2) our coroutine is suspended and the control is passed to some coroutine that is runnable;
(3) when the asynchronous operation completes, our coroutine is marked runnable again;
(4) eventually, when some other coroutine gets suspended or finishes, our coroutine is resumed;
(5) the result of the asynchronous operation is returned and assigned to the variable `v`.

Additionally, to make the mechanism composable, our framework allows the same `v = co_await func()` statement to be executed with `func()` being a coroutine returning a single value. In this case, the effect is similar to a single asynchronous call, but there will be multiple suspends and resumes between the initial call and getting the final result.

The implementation uses two auxiliary objects:

- a `CoroutineScheduler`, for keeping track of runnable coroutines.
- a `PromiseFuturePair`, that will hold the return value from the asynchronous call.

The `CoroutineScheduler` needs to offer two functions, `markRunnable()`, that puts the specified coroutine into a set of coroutines ready to be executed, and `schedule()`, which picks and returns a runnable coroutine or waits until such a coroutine exists.

The `PromiseFuturePair` offers three operations: `set()`, that can be called only once and sets the return value of the asynchronous operation, `get()`, that waits for the result and returns it, and `addCallback()`, that sets a callback to be called when the operation completes; this is needed to mark the coroutine that waits for the result runnable.

With the above, the PromiseFuturePair can be wrapped into an *awaiter* object. The `await_suspend()` will set the callback for the `PromiseFuturePair` to mark the awaiting coroutine runnable again and will invoke the scheduler to schedule some other coroutine. The `await_resume()` will return the value from the future.

Note that the PromiseFuturePair awaiter needs to be linked to the CoroutineScheduler that will provide the next coroutine for the current thread and will also schedule the current coroutine when the awaited condition is fullfilled. The way it is done is by having the coroutine promise object holding a pointer to the CoroutineScheduler and the `await_transform()` that creates the awaiter out of the PromiseFuturePair embed the CoroutineScheduler into the awaiter object. We took two basic assumptions behind this design:

(1) each coroutine is handled by a single scheduler (it cannot go from one scheduler to another);
(2) no coroutine may outlive its scheduler.

The coroutine return object is also an awaiter. Its `await_suspend()` operation, invoked by the caller coroutine, makes the called coroutine runnable, so that the called coroutine will eventually run. It also memorizes the handle of the coroutine invoking it (that gets suspended) as well as its scheduler, so that the caller coroutine is marked ready in its scheduler when the called coroutine returns a value. The `return_value()` of the `promise_type` object marks runnable the coroutine mentioned above.

This design allows control of the scheduler for each coroutine, allowing, among other use cases, to control the thread used by each coroutine. This is important for some GUI frameworks that insist that GUI related functions can be called only on the UI thread.

4.3. **A pipe mechanism.** To go from asynchronous operations returning single values to asynchronous operations returning multiple values, the mechanism for conveying the result must be changed from `PromiseFuturePair` to a pipe (queue).

While the basics of a pipe are very simple, with the two basic operations, `send()` and `recv()`, there are a few needed additions and clarifications needed.

First, there is the issue of how to signal the end of communication. On the producer end, we have to either call a different function (say, `pipe.close()`) or send a special *EOF* data value. On the consumer end, since the consumer cannot usually know beforehand when the communication will end, the only possibility is to have `pipe.recv()` return the special *EOF* value.

Second, like for the `PromiseFuturePair`, we need a callback to be called when an element is put into the queue. As the pipe is fed from an asynchronous operation, setting the callback can be at race with putting an element into the queue. To simplify the operations, we make the following assumptions:

- When setting a callback to be called for enqueued elements, the callback will be called on each element that is not consumed yet (via `recv()`).
- For this to not create a race condition, the functions `recv()` and `setDataAvailableCallback()` must not be called concurrently.
- To simplify the behavior of `setDataAvailableCallback()`, the capacity of the pipe will be 1 element.

4.4. **Asynchronous generator.** We combined the elements described above to enable the creation of *asynchronous generators*.

Our design offers the user the possibility to implement an asynchronous generator as a coroutines function having the following elements:

- like a regular generator, it returns multiple values via `co_yield`;
- like an async-await coroutine, when it executes `co_await`, it gets suspended until the result of the function invoked with `co_await` is available;
- the argument of `co_await` may be an asynchronous function producing multiple values. In this case, the calling coroutine gets suspended until the next value is generated and `co_await` returns that value;
- the argument of `co_await` may be another asynchronous generator, in which case `co_await` returns the next generated value. Note that this may lead to an arbitrary sequence of nested asyncronous generators calls and the use of `co_await` leads to a potentially complex sequence of suspending and resuming of their coroutines.

To support the above functionalities, our framework defines a class template, `AsyncGenerator<T>`, that represents an asynchronous generator coroutines that generates objects of type `T`. Given the way coroutines work in C++20, the coroutine function that acts as an asynchronous generator should have a declared return type which is `AsyncGenerator<T>`

`AsyncGenerator<T>` is designed to be usable in two contexts:

- in a coroutine, as an argument of a `co_await` statement. Here, it must act as an awaiter.
- in a regular function. Here, it must offer a blocking receive function.

To implement that, the `AsyncGenerator<T>::promise_type` acts as pipe. The pipe is fed by the `yield_value()`, which puts a value, and `return_void()`, which closes the pipe (that is, it adds a special EOF object). After feeding the pipe, the control is transmitted back to the calling coroutine.

For the case `AsyncGenerator<T>` is the operand of `co_await`, its method `await_suspend()` memorizes the handle of the calling coroutine and resumes the own (called) coroutine. This is done so that the owned coroutine can produce the next value. When the next value is produced (via `yield_value()` or `return_void()` of `AsyncGenerator<T>::promise_type`), the memorized calling coroutine is set to runnable again.

For the case `AsyncGenerator<T>` is used from a regular function, it offers a blocking `recv()` function that repeatedly calls `resume()` on the owned coroutine, until a value is made available by the coroutine.

To demonstrate the capabilities of our framework, we present below an implementation of an asynchronous generator that uses coroutines. It takes a sequence of characters produced by another asynchronous generator named `source`, parses it as a sequence of numbers and returns them to its caller.

```
AsyncGenerator<unsigned> parseFlow(CoroutineScheduler* pScheduler,
        AsyncGenerator<char>& source) {
    unsigned v = 0;
    bool parseStarted = false;
    while (true) {
        ElementOrEof<char> el = co_await source;
        if (el.isEof()) {
            if (parseStarted) {
                co_yield v;
            }
            co_return;
        }
        char c = el.value();
        if (c >= '0' && c <= '9') {
            parseStarted = true;
            v = 10 * v + (c - '0');
        } else if(c == ' ' || c == 10 || c == 13) {
            co_yield v;
            v = 0; parseStarted = false;
        }
    }
}
```

It is worth noting that the above example shows a possible typical usage of our framework, for instance in some networked server or client. The data source in the example would encapsulate an asynchronous read operation from a network connection, the coroutine in the example would do some parsing or preprocessing of the requests, and the user of the data would contain the main processing loop. Classical alternative mechanisms would be threads or callbacks. Threads consume more resources than coroutines. Callbacks are harder to understand because of the inversion of control — in the callback, the programmer needs to keep track of a state and to update it at each incoming callback; with coroutines, the programmer writes the main processing loop.

5. Conclusions

The asynchronous coroutines, proposed in this paper, allow a very natural way of writing code that processes a flow of data coming asynchronously, from some external source, without resorting to threads for this purpose.

This paper also demonstrates how to create, each by itself, the generators and the async-await mechanism, which exist in other languages, but only at its beginning using C++20 coroutines.

It also shows that the C++20 coroutines mechanism, while quite a bit hard to use directly, is very powerful and allows very diverse use scenarios.

References

[1] C and c++ reference, coroutines. `https://en.cppreference.com/w/cpp/language/coroutines`. Accessed: 2022.

[2] A library of c++ coroutine abstractions for the coroutines ts. `https://github.com/lewissbaker/cppcoro`. Accessed: 2022.

[3] Pep 525 – asynchronous generators. `https://peps.python.org/pep-0525/` . Accessed: 2022.

[4] `.NET/C#` guide/language reference. await operator - asynchronously await for a task to completes. `https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/await`. Accessed: 2022.

[5] `.NET/C#` guide/language reference. yield statement - provide the next element. `https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/yield`. Accessed: 2022.

[6] Windows app development documentation. processes and threads. fibers. `https://learn.microsoft.com/en-us/windows/win32/procthread/fibers`. Accessed: 2022.

[7] Belson, B., Xiang, W., Holdsworth, J. J., and Philippa, B. W. C++20 coroutines on microcontrollers—what we learned. *IEEE Embedded Systems Letters 13* (2021), 9–12.

[8] Chauhan, K., Kumar, S., Sethia, D., and Alam, M. N. Performance analysis of kotlin coroutines on android in a model-view-intent architecture pattern. In *2021 2nd International Conference for Emerging Technology (INCET)* (2021), IEEE, pp. 1–6.

[9] Conway, M. E. Design of a separable transition-diagram compiler. *Commun. ACM 6*, 7 (jul 1963), 396–408.

[10] Elizarov, R., Belyaev, M., Akhin, M., and Usmanov, I. Kotlin coroutines: Design and implementation. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (2021), Association for Computing Machinery, p. 68–84.

[11] Ierusalimschy, R. Programming in `Lua`, 2003.

[12] Marlin, C. D. *Coroutines: A Programming Methodology, a Language Design and an Implementation*, vol. 95 of *Lecture Notes in Computer Science*. Springer, 1980.

[13] Moura, A. L. D., and Ierusalimschy, R. Revisiting coroutines. *ACM Trans. Program. Lang. Syst. 31*, 2 (feb 2009).

[14] von Merzljak, L., Fent, P., Neumann, T., and Giceva, J. What are you waiting for? use coroutines for asynchronous `I/O` to hide `I/O` latencies and maximize the read bandwidth! In *International Workshop on Accelerating Data Management Systems (ADMS)* (2022).

[15] Wang, Z. Web crawler scheduler based on coroutine. *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)* (2019), 540–543.

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

Email address: `radu.lupsa@ubbcluj.ro, dana.lupsa@ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 2, 2022
DOI: 10.24193/subbi.2022.2.04

EXAMINING THE SOCIAL BEHAVIOR OF ANT COLONIES USING COMPLEX NETWORKS

BOGDAN MURSA

Abstract. This paper proposes the use of Complex Network Theory to model the interactions between ants and analyze their social behavior. Specifically, the study focuses on six colonies of ants to investigate whether their behavior is community-oriented or individual-oriented. The research employs various nodes properties that define nodes' importance to quantify the existence of a social or individual-oriented behavior. The results aim to provide insights into the social behavior of ants and may have implications for understanding other complex social systems.

1. Introduction

In nature, a variety of species exhibit a pronounced social behavior, whereby members of the same species tend to interact in order to increase their chances of survival. Such behavior is not restricted to mammals alone, but also found in insect colonies, schools of fish, and, to a lesser extent, reptiles. Certain species of lizards have been observed to display social behavior and organize themselves into complex social structures [6].

In species with smaller members, individual survival rates tend to be lower, leading to the formation of complex communities characterized by homogenity and defined roles. Ant colonies are a prime example of such communities, often comprising millions of members [17, 2]. These colonies exhibit remarkable synchronization in tasks such as food gathering, cleaning, and protection. Ant behavior has been widely studied in numerous experiments [10, 25, 19], although logistical challenges often pose a significant obstacle to researchers.

As the complexity of interactions between members within a system is difficult for human observation and measurement in real-time, researchers have

Received by the editors: 17 March 2023.

2010 *Mathematics Subject Classification.* 05Cxx, 93Bxx, 92-xx.

1998 *CR Categories and Descriptors.* E.1 [**Data**]: Data Structures – *Graphs and networks*; G.2.2 [**Discrete mathematics**]: Graph Theory – *Network problems (G.2.2)*.

Key words and phrases. complex networks, centrality measurements, social behaviours.

sought frameworks to model the underlying dynamic system. Complex Network Theory (CNT) has emerged as an increasingly popular framework in recent years [7], as it enables the modeling of complex systems as topological spaces, specifically as graphs, where the members are represented as nodes and their interactions as edges.

By studying a modeled network, it is possible to gain insight into the behavior of the network's members using a range of properties. Some properties provide information at the level of individual nodes, such as node degree, assortativity degree, and centrality measurements [22, 4, 21]. Other properties are oriented towards characterizing the network as a whole, such as clustering coefficient [16], communities [9, 8], and network motifs [1, 13, 14].

Various studies have demonstrated the advantages of utilizing CNT to model ant colonies as intricate networks, enabling researchers to investigate ant social behavior and dynamic processes such as food collection and communication. However, most of these studies have focused on the colony as a whole rather than individual ants. Given the existence of well-defined structure and organization in ant colonies, the lack of research on individual behavior is notable. Therefore, the paper aims to explore whether individual ants prioritize their well being instead of their defined role in the organization or strive to exhibit purely altruistic behavior by acting for the benefit of the community without expecting anything in return.

A set of research inquiries will be developed to steer an experimental analysis of six ant colonies monitored for a duration of 41 days [18]. The objective is to employ properties at the node and network levels derived from CNT to measure and confirm the central query: do ants act altruistically or selfishly? The objective is to distinguish between selfish and altruistic behavior in ants by employing centrality measurements, which are commonly used to evaluate the significance of a node in complex networks. There will be utilized network properties that optimize the exchange of information within the colony, such as maintaining a low average shortest path, to identify altruistic behavior. Ultimately, a qualitative analysis will be conducted to compare the two methods of quantifying altruistic and selfish behavior.

The following sections aim to provide an overview and clarification on the challenges that arise when studying small-sized creatures, as well as how CNT can help overcome these challenges based on the existing state-of-the-art. Additionally, there will be detailed each of the metrics employed in the experiment, explaining their relevance and how they will be applied. There will be introduced the dataset and the research questions that will guide the experiment, with the results being validated against the network properties obtained.

Finally, the paper will conclude with a discussion that will confirm or reject the initial assumptions regarding the research questions posed.

1.1. **Problem definition.** From a logistical standpoint, studying ant colonies is challenging due to the small size of the ants, making them difficult to observe by the naked eye. Additionally, the high number of members that share a similar appearance can complicate the study of their behavior, as they move quickly and often exhibit chaotic movement patterns. Manually observing the behavior of ants in such a scenario would be time-consuming and exhaustive, requiring video recording and subsequent frame-by-frame tracking by one or more researchers [20, 26].

Following the logistical difficulties of studying ant colonies, there is a secondary issue that arises during the process of analyzing the interactions between ants. This issue is caused by the need for a framework to facilitate the entire process, starting from defining interactions and culminating with the impact a group of interactions has on the dynamic of the colony. One solution to this issue is to model ant colonies as systems that can be studied using mathematical and statistical approaches. This allows for the numerical quantification of ant behavior, leading to a feasible and valid way of answering research questions. However, even with a colony modeled as a system, it can be difficult to understand the apparent chaotic behavior between ant interactions. To address this challenge, a set of tools is required to extract properties from the system that may lead to paths not initially intended by the researchers. Complex Network Theory is one such framework that provides a wide range of tools to extract properties about the system modeled as a complex network. By modeling the colony's system as a network, where ants are nodes and their interactions are edges, it can be explicitly analyzed the interactions a given ant or group of ants have, defining statements about their social or individual behavior [24].

Complex Network Theory has emerged as a valuable analytical tool for studying the organizational dynamics of ant colonies. Its applications in this field are numerous, including investigations into the community structures of colonies [12] and the role of information flow in collective decision-making [5]. By modeling the networks underlying ant colonies, researchers aim to gain insights into the structural organization of the colonies, the development of modular structures, and the resilience and optimality of information flows among colony members. These studies also provide insights into the potential impact of member loss in the event of a disaster [3, 11].

In the subsequent section, there will be provided an overview of the fundamental properties existing in complex networks that are pertinent to the examination of social interactions among ants belonging to the same colony, accentuating the practical implementation of the theoretical aspects in the context of a real-world ant colony.

2. Theoretical insights

Any complex network is characterized by a collection of nodes and a set of edges that connect them. Although these two fundamental components are simple, they give rise to multidimensional complex topologies with unique properties that can be explored, underscoring the advantages of representing real-world systems as complex networks. One of the most widely researched concepts, particularly in social networks, is the definition of critical, important, or popular nodes. However, this is more of a philosophical question that has been debated extensively in the literature [15]. Nevertheless, the literature proposes a group of metrics known as centrality measures that aim to offer various ways of characterizing important nodes.

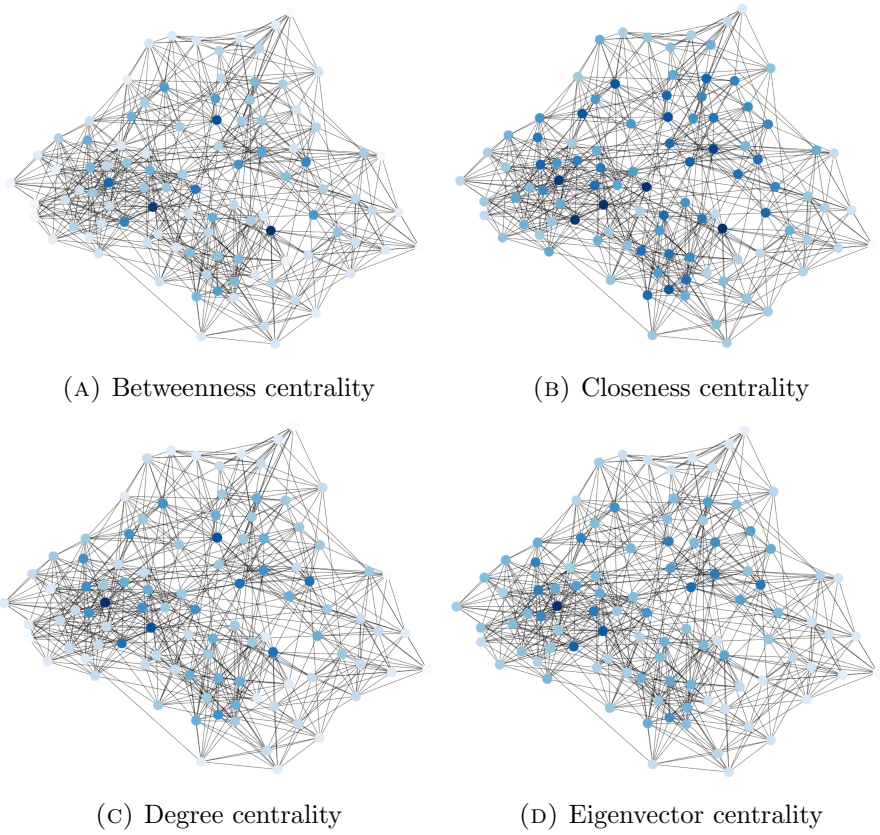
In this section, there will be provided detailed descriptions of the graphs and their respective nodes' centrality measures, as shown in Figure 1. The figures illustrate that different nodes are identified as "important" by each centrality measure, highlighting that each metric has a distinct approach to determining a node's significance.

Degree centrality (D_c) is one of the fundamental measures of centrality in complex networks. It quantifies the importance of a node based on the number of edges it has with other nodes in the network. Nodes with a high number of edges have higher degree centrality and are considered more important. The mathematical formula for degree centrality is as follows:

$$D_c(x) = \frac{d_x}{n-1}$$

where $D_c(x)$ is the degree centrality of node x, d_x is the degree of node x, and n is the total number of nodes in the network [22].

In addition to degree centrality, another centrality measure that takes into account the number of links and goes further in assessing a node's importance is *eigenvector centrality* (E_c) [22]. This property evaluates a node's influence in the network based on the degree centrality of its neighboring nodes. From a real-world perspective, a node with a high eigenvector centrality is connected to other nodes that are also important, meaning that being connected to popular nodes increases one's own popularity. The eigenvector centrality formula



(A) Betweenness centrality (B) Closeness centrality

(C) Degree centrality (D) Eigenvector centrality

FIGURE 1. Centrality measures in a Newman Watss Strogatz graph with 100 nodes. Light blue - low value, Dark blue - high value.

follows a recursive approach that calculates a value for a node by using the values computed for its neighboring nodes, as follows:

$$E_c(x) = \frac{1}{\lambda} \sum_{u \in N(v)} E_c(u)$$

where $E_c(x)$ is the eigenvector centrality of node x, $N(x)$ is the set of nodes that are connected to node x, and λ is a constant called the leading eigenvalue of the adjacency matrix of the network [15].

A distinct perspective on defining node importance or popularity is based on the flow of information in the network, whereby nodes that enable the flow

of information in the network tend to be more important or popular to other nodes. In this regard, the next group of centrality measures uses the path between the nodes and their lengths to evaluate the significance of the nodes. *Betweenness centrality* (B_c) assesses the importance of a node by its ability to control the flow of information through the number of shortest paths between any two nodes that pass through it. Having more shortest paths passing through it means that the node is a hub facilitating the information flow in the most rapid manner through the shortest paths it is part of. The formula below depicts the quantification of this centrality measure:

$$B_c(x) = \sum_{s \neq x \neq t \neq x} \frac{\sigma_{st}(x)}{\sigma_{st}}$$

where x is the node for which there was computed the metric, σ_{st} is the total number of shortest paths from any node s to any node t and $\sigma_{st}(x)$ is the number of those paths that pass through x (not where x is an end point) [22].

Closeness centrality (C_c) is another measure that quantifies the importance of a node in a network. Unlike betweenness centrality, C_c considers how quickly a node can be reached by all other nodes in the network. A node with a high C_c is considered to be "close" to all other nodes in terms of its shortest paths, making it an important hub that facilitates the flow of information in the network:

$$C_c(x) = \frac{1}{\sum_{u \neq x} d(u, x)}$$

where $C_c(x)$ is the closeness centrality of node x, $d(u, v)$ is the shortest path distance between nodes u and v, and the summation is taken over all nodes $u \neq v$ in the network [22].

In recent studies, a new approach to defining the significance of nodes has emerged, focusing on their role in maintaining network integrity. *Articulation points* (AP) are nodes that, when removed, divide the network into two or more connected components, acting as bridges between isolated groups of nodes (Figure 2). In social networks, an AP could be a social media influencer or politician, while in an ant colony, the queen can be an AP. This concept is gaining popularity as it provides insights into the structure of networks and can inform strategies for improving network efficiency and stability [23].

The centrality measures detailed earlier are indicative of macro-level characteristics of a node, which describes its role in influencing the overall dynamics

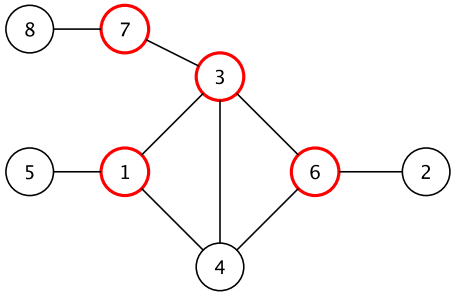


FIGURE 2. Example of Articulation Points (in red)

of the network. Conversely, there are other properties that are more relevant to micro-level behavior, which tend to be oriented towards the individual node itself. For instance, the local clustering coefficient (CC_l) is used to quantify the tendency of nodes to form communities with other nodes that share similar characteristics or interests (e.g., ants performing similar tasks). The following formula expresses CC_l as the ratio of the actual number of links (E_i) connecting the vertices within a node (i) neighbors to the maximum number of possible links that could exist among them ($k_i(k_i - 1)$)).

$$CC_{li} = \frac{2E_i}{k_i(k_i - 1)}$$

Assortativity degree (ρ) measures the tendency of nodes to connect with other nodes that share similar degrees (e.g., popular individuals preferring to associate with others who share a similar level of popularity). In this sense it can defined ρ as the correlation between the degrees of connected nodes in a network, with values in interval -1 to 1 computed with the following formula:

$$\rho = \frac{\sum_{jk} jk(e_{jk} - q_j q_k)}{\sigma_q^2}$$

where e_{jk} is the join probability between excess degree of j and k (excess degree, also known as remaining degree, is computed by subtracting one from the degree of a given node), $q_k = \frac{(k+1)p_{k+1}}{\sum_{j\geq 1} jp_j}$ is the normalized distribution of the excess degree of a randomly chosen node, respectively σ_q is standard deviation of q_k, used to normalize ρ in interval [-1, 1].

Positive values indicate nodes tend to connect with others with similar degrees, while negative values indicate nodes tend to connect with nodes with different degrees. An r value of 1 indicates perfect assortative mixing, 0 indicates non-assortative mixing, and -1 indicates completely disassortative mixing.

In the experiment described in the following section, the aim is to investigate the social behavior of ants in their colonies by analyzing their network properties. Specifically, it will be explored whether nodes with high values for centrality measures, such as degree centrality, eigenvector centrality, betweenness centrality, and closeness centrality, respectively local clustering coeffient and assortativity degree, but also existence of articulation points, are associated with the emergence of individual behavior among ants that is not aligned with the organization of the colony as a whole.

3. Experiment

The following section will overview the experiment proposed to analyze the social behaviour of ants, making use of a comprehesive dataset for which there were validated a series of research questions that will be evaluated using a defined methodology. The results of the extracted properties (detailed in Section 2) obtained in the experiment will be analyzed and a series of conclusions will be drawn to conclude the formulated research questions.

3.1. **Dataset.** To conduct a robust and meaningful experiment, it was deemed necessary to utilize a diverse dataset that is both horizontally and vertically scaled. To this end, there will be employed a complex dataset of complex networks that consist of six ant colonies that were completely isolated, as proposed by Mersch D. et al. in their research paper [18]. The complex networks were modeled by observing each ant colony over a timespan of 41 days, utilizing a video tracking system that was based on fiducial identification labels. Each ant's position was tracked twice per video frame, resulting in a vast amount of data - 2,433,250,580 ant positions and 9,363,100 social interactions. Social interactions were defined as instances where one ant's front end was within the trapezoidal shape representing another ant. From this data, a total of 246 networks were modeled, utilizing both the ant positions and the tracked interactions.

Table 1 provides a comprehensive summary of the modeled networks based on their nodes and edges - ants are represented as nodes, and the interactions between them are represented as undirected edges. The weight of each edge is determined by the number of interactions between the same ants. Since the ant colonies were observed for a period of 41 days, changes in the number

TABLE 1. General properties of the studied networks [18]

colony id	metric	mean	95% percentile	Stdev
1	nodes	89.21	[82.35, 96.06]	± 21.15
	edges	2803.97	[2429.41, 3178.54]	± 1155.48
2	nodes	100.13	[91.47, 108.79]	± 26.71
	edges	3541.03	[2909.48, 4172.57]	± 1948.25
3	nodes	130.05	[122.01, 138.09]	± 24.80
	edges	6036.79	[5274.19, 6799.40]	± 2352.55
4	nodes	68.77	[60.73, 76.80]	± 24.79
	edges	2067.03	[1640.79, 2493.26]	± 1314.88
5	nodes	113.31	[102.26, 124.35]	± 34.07
	edges	4905.05	[4034.66, 5775.44]	± 2685.03
6	nodes	131.85	[123.07, 140.63]	± 27.09
	edges	6338.82	[5451.44, 7226.20]	± 2737.44

of individuals within each colony could occur, resulting in variations in the network's characteristics. These changes were captured by the mean, standard deviation, and 95% percentile values, enabling the dynamics of the colonies to be analyzed.

The metrics obtained from the modeled networks exhibit a significant degree of diversity with regards to the number of members in each colony, which is reflected in the number of edges present in each network. This phenomenon can be attributed to the fact that larger colonies tend to have more interactions among members, resulting in a higher number of edges.

3.2. **Research questions.** The experiment is designed to address two research questions, a primary one (**RQ1**) and a secondary one (**RQ2**):

RQ1 - What is the extent of variation in individual behavior within ant colonies, and does this variation lead to the presence of outliers exhibiting selfish behavior or does the colony exhibit a predominantly homogeneous altruistic behavior?

RQ2 - Does the dynamics of ant colonies optimize the flow of information through interactions among the ants?

3.3. **Methodology.** In accordance with the formulated research question, the methodology of the experiment will involve the extraction of centrality measures and articulation points from each network to investigate the occurrence

of nodes with abnormally high values for these metrics, indicating their inclination to strategically position themselves to maximize their valuable connections. In addition to the previously mentioned topological indicators, there will be also extracted two additional measures, namely CC_l and ρ, to gain a deeper understanding of the global social behaviour of ants and to investigate if any specific behaviour emerges. These measures will aid in addressing the research questions posed in this experiment.

By quantifying and validating **RQ1** through this approach, it can be employed the concept of popularity, which is defined in various ways as discussed in Section 2.

To investigate whether the networks and colonies aim to optimize the performance of the information flow among their members, there will be used two topological indicators: density and average shortest path. Density (d) is a measure of how close a network is to being fully connected, where all ants interact with one another. Networks with high density tend to optimize the information flow by ensuring that all members are easily reachable through a high density of edges. The average shortest path is a measure of the average number of steps needed to travel between any two ants in the network. There will be computed and used these indicators to demonstrate the colonies' dynamics and to answer the research question. The calculation of the average shortest path (AVG_{sp}) can aid in determining if ants strive to optimize the formation of valuable links within the context of an efficient information flow. A low average shortest path, which denotes the average number of links needed to create an optimal path between two members, suggests that ants can easily reach each other. By analyzing this metric, it can be provided a formal response to **RQ2**.

3.4. **Results.** In accordance with the methodology outlined in the preceding section, there were derived all the centrality measures outlined in Section 2 as well as other topological properties, including CC_l, d, AVG_{sp}, and ρ (Table 2). This was done to enable the quantification of the propositions that could potentially serve as answers to the formulated research questions.

Ants are social animals with highly organized behavior and specialized roles throughout their lifetime [18]. Their division into groups is strictly task-oriented, which may result in a large number of interconnections between ants within the same group. This behavior is well-reflected in the high values of CC_l observed in all the studied colonies.

It is well known that each of the examined ant colonies has a queen, whose exclusive responsibility is to lay eggs. Given its role as being one that it is

unique in the structure of the colony (only one ant, namely the queen, lays eggs), it is reasonable to classify the queen as an articulation point (AP). However, the analysis showed that the queen is not often an articulation point in any of the modeled networks. This indicates that despite the queen's critical role, it is part of a well-connected community, and her death would not necessarily lead to the colony's disintegration.

At a macro level, the social behavior of ants does not exhibit a particular tendency to create links with new ants, as indicated by the value of ρ which is equal to 0. This suggests that the creation of links is not driven by individual preferences, but rather by the collective behavior of the group as a whole.

This hypothesis is strongly supported by all of the computed centrality measures, which do not identify any notable group of nodes that exhibit exceptional values for their centrality measures. By computing the 95% percentile interval and mean values, it can be observed the homogeneity of centrality measure values across all nodes, with insignificant standard deviation. Even though the standard deviation of the centrality measures' computed values is insignificant, the real-world implications of each metric should be taken into consideration while interpreting their mean values. It is noticeable that C_c has a substantially high value, reaching the maximum value of 1, which suggests that the colony's organization is optimized for efficient information flow, with every ant only a few connections away from any other ant. A similar observation can be made for D_c, which approaches a value of 1, indicating that all ants have nearly equal connectivity with each other. These findings reinforce the conclusions drawn regarding the ρ property.

The B_c value in the colony's centrality measures shows a relatively low value from the range of [0, 1]. Typically, a higher B_c value indicates that a network node has a greater ability to control information flow. This low value may, therefore, suggest that the colony is organized dynamically and oriented toward the collective benefit. Although members with high B_c values are essential to the network, they are also critical points whose loss could disrupt the system's functionality. This robust organization enables the colony to sustain itself even if some members are lost. Additionally, the centrality measure E_c shows a low value, suggesting that the ants' interactions are task-oriented, with little emphasis on creating new links based on other ants' links, resulting in an organization geared towards the benefit of the system as a whole.

Using these two observations it can be concluded the altruistic behaviour of the ants (**RQ1**), meaning their interactions are ones established purely

on their role in the community they are part of and the only thing that is important to them is to succesfully complete their task.

TABLE 2. Network properties extracted for each of the six ant colonies

ID	Metric	Mean	95% percent.	Stdev	ID	Mean	95% percent.	Stdev
1	AVG_{sp}	1.31	[1.30, 1.33]	0.05	2	1.36	[1.34, 1.38]	0.07
	CC_l	0.79	[0.78, 0.80]	0.03		0.77	[0.76, 0.78]	0.03
	ρ	0.03	[0.02, 0.04]	0.03		0.09	[0.07, 0.11]	0.06
	d	0.69	[0.67, 0.70]	0.05		0.64	[0.62, 0.66]	0.06
	AP	0.08	[0.00, 0.16]	0.27		0.03	[0.03, 0.08]	0.16
	C_c	0.77	[0.74, 0.80]	0.08		0.75	[0.72, 0.77]	0.08
	B_c	0.0035	[0.003, 0.005]	0.0027		0.0041	[0.003, 0.006]	0.0038
	D_c	0.69	[0.64, 0.73]	0.15		0.64	[0.59, 0.70]	0.16
	E_c	0.11	[0.10, 0.12]	0.02		0.10	[0.09, 0.11]	0.03
3	AVG_{sp}	1.31	[1.30, 1.32]	0.04	4	1.20	[1.19, 1.21]	0.04
	CC_l	0.79	[0.78, 0.80]	0.03		0.86	[0.85, 0.87]	0.03
	ρ	0.01	[0.00, 0.02]	0.02		0.03	[0.00, 0.05]	0.04
	d	0.69	[0.68, 0.71]	0.04		0.80	[0.79, 0.82]	0.04
	AP	0.15	[0.04, 0.27]	0.37		0.00	[0.00, 0.00]	0.00
	C_c	0.77	[0.75, 0.80]	0.08		0.85	[0.82, 0.88]	0.08
	B_c	0.0025	[0.002, 0.003]	0.0015		0.0032	[0.002, 0.004]	0.0020
	D_c	0.69	[0.65, 0.73]	0.15		0.80	[0.76, 0.85]	0.14
	E_c	0.09	[0.08, 0.09]	0.02		0.13	[0.11, 0.14]	0.02
5	AVG_{sp}	1.29	[1.28, 1.30]	0.04	6	1.31	[1.29, 1.32]	0.05
	CC_l	0.80	[0.79, 0.81]	0.03		0.79	[0.78, 0.80]	0.03
	ρ	0.01	[0.00, 0.02]	0.05		0.03	[0.02, 0.03]	0.02
	d	0.71	[0.70, 0.72]	0.04		0.69	[0.68, 0.71]	0.05
	AP	0.00	[0.00, 0.00]	0.00		0.03	[0.00, 0.08]	0.16
	C_c	0.78	[0.76, 0.81]	0.08		0.78	[0.75, 0.80]	0.08
	B_c	0.0029	[0.002, 0.004]	0.0021		0.0025	[0.002, 0.003]	0.0015
	D_c	0.71	[0.67, 0.75]	0.15		0.69	[0.65, 0.74]	0.15
	E_c	0.10	[0.09, 0.11]	0.02		0.09	[0.08, 0.09]	0.02

The current findings strongly indicate the well-organized nature of ant colonies. The AVG_{sp} metric is another measure demonstrating that colonies are optimized not only in terms of creating specific groups for efficient task

completion, but also for the rapid flow of information among members. According to this metric, it takes approximately 1.30 edges for any ant to reach another ant using the shortest path in the modeled network. This value is remarkably low for any network, highlighting the communication efficiency present in ant colonies and confirming the answer to **RQ2**.

Another notable finding is that the ant colonies exhibit both a low AVG_{sp} and a high CC_l, which are characteristic features of small-world networks. These networks are commonly observed in various real-world systems, such as transportation networks, and have been the subject of many studies aimed at understanding their effective information flow and how to replicate it in other contexts. Given the well-organized structure of ant colonies, it is not surprising to find that the networks derived from observing them exhibit small-world characteristics.

4. Conclusions

Ants have been a subject of fascination for the scientific community for a long time due to their ability to develop highly intricate social structures organically, which enables them to efficiently accomplish tasks such as foraging, cleaning, and defense. However, research on ants has largely focused on their collective behavior rather than individual behavior. While it is widely acknowledged that ants exhibit altruistic behavior at the group level, it remains to be seen whether this behavior is universal across all members of the colony or if some individuals display a more self-centered approach aimed at maximizing their own benefit.

Continuing this line of inquiry and utilizing a set of intricate network models based on observations of six distinct ant colonies over a span of 41 days, our research aimed to address two fundamental questions. These questions were formulated to resolve the previous uncertainties:

RQ1 - What is the extent of variation in individual behavior within ant colonies, and does this variation lead to the presence of outliers exhibiting selfish behavior or does the colony exhibit a predominantly homogeneous altruistic behavior?

RQ2 - Does the dynamics of ant colonies optimize the flow of information through interactions among the ants?

During the study experiment, there were obtained various topological properties and centrality measures of the examined networks. The analysis revealed that there is a considerable consistency among the centrality values of the individual nodes, with insignificant standard deviation. Furthermore, based on

the property ρ, it was observed that the ants do not have any macro-level inclination towards preferential attachment, but rather establish connections through their task-based activities. These two observations led us to confirm the altruistic behavior of ants as an answer to **RQ1**.

In addition to ants' ability to naturally and organically evolve complex groups that optimize task performance, the observations indicate that the information flow within their networks is highly efficient, confirming **RQ2**. The average shortest path between any two ants in the network is close to one edge, indicating that every ant is almost directly connected to every other ant. Furthermore, all studied networks show a high CC_l and exhibit characteristics of small-world networks, which optimize information flow performance and evolve strong and complex structures, as commonly observed in other real-world systems such as transportation networks.

The experiment corroborates the widely accepted behavior of ants as altruistic individuals that prioritize the collective good over individual interests, while also demonstrating their capacity to naturally develop sophisticated systems that optimize task performance for the group.

In future studies of this paper, a significant enhancement would entail exploring other network properties, including communities and network motifs, to gain a deeper understanding of the organization structure and how ants interact in small modules. Such an approach would provide further insights into the optimized interactions that drive task-oriented actions.

References

[1] Angulo, M., Liu, Y.-Y., and Slotine, J.-J. Network motifs emerge from interconnections that favor stability. *Nature Physics 11* (11 2014), 848–852.

[2] Aron, S., Beckers, R., Deneubourg, J. L., and Pasteels, J. M. Memory and chemical communication in the orientation of two mass-recruiting ant species. *Insectes Sociaux 40*, 4 (Dec 1993), 369–380.

[3] Backen, S. J., Sendova-Franks, A. B., and Franks, N. R. Testing the limits of social resilience in ant colonies. *Behavioral Ecology and Sociobiology 48*, 2 (2000), 125–131.

[4] Bell, D. C., Atkinson, J. S., and Carlson, J. W. Centrality measures for disease transmission networks. *Social Networks 21*, 1 (1999), 1–21.

[5] Blonder, B., and Dornhaus, A. Time-ordered networks reveal limitations to information flow in ant colonies. *PLOS ONE 6*, 5 (05 2011), 1–8.

[6] Burghardt, G. M., Greene, H. W., and Rand, A. S. Social behavior in hatchling green iguanas: Life at a reptile rookery. *Science 195*, 4279 (1977), 689–691.

[7] Erdös, P., and Rényi, A. On the evolution of graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences 5* (01 1960), 17–61.

[8] Fani, H., Jiang, E., Bagheri, E., Al-Obeidat, F., Du, W., and Kargar, M. User community detection via embedding of social network structure and temporal content. *Information Processing and Management 57*, 2 (2020), 102056.

[9] Girvan, M., and Newman, M. E. J. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences 99*, 12 (11 2001), 7821–7826.

[10] Gordon, D. M., Guetz, A., Greene, M. J., and Holmes, S. Colony variation in the collective regulation of foraging by harvester ants. *Behavioral Ecology 22*, 2 (02 2011), 429–435.

[11] Hoenle, P. O., Staab, M., Donoso, D. A., Argoti, A., and Blüthgen, N. Stratification and recovery time jointly shape ant functional reassembly in a neotropical forest. *Journal of Animal Ecology 32*, 4 (2022), e2559.

[12] Ivens, A. B., von Beeren, C., Blüthgen, N., and Kronauer, D. J. Studying the complex communities of ants and their symbionts using ecological network analysis. *Annual Review of Entomology 61*, 1 (2016), 353–371.

[13] Kashani, Z. R. M., Ahrabian, H., Elahi, E., Nowzari, A., Ansari, E. S., Asadi, S., Mohammadi, S., Schreiber, F., and Masoudi-Nejad, A. Kavosh: A new algorithm for finding network motifs. *BMC bioinformatics 10* (10 2009), 318.

[14] Kashtan, N., Itzkovitz, S., Milo, R., and Alon, U. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics 20*, 11 (08 2004), 1746–1758.

[15] Landherr, A., Friedl, B., and Heidemann, J. A critical review of centrality measures in social networks. *Business & Information Systems Engineering 2*, 6 (Dec 2010), 371–385.

[16] Lee, D.-S., Chang, C.-S., Ye, W.-G., and Cheng, M.-C. Analysis of clustering coefficients of online social networks by duplication models. In *2014 IEEE International Conference on Communications (ICC)* (2014), Institute of Electrical and Electronics Engineers, pp. 4095–4100.

[17] Mersch, D. P. The social mirror for division of labor: what network topology and dynamics can teach us about organization of work in insect societies. *Behavioral Ecology and Sociobiology 70*, 7 (Jul 2016), 1087–1099.

[18] Mersch, D. P., Crespi, A., and Keller, L. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science 340*, 6136 (2013), 1090–1093.

[19] Middleton, E. J. T., Reid, C. R., Mann, R. P., and Latty, T. Social and private information influence the decision making of australian meat ants (iridomyrmex purpureus). *Insectes Sociaux 65*, 4 (Nov 2018), 649–656.

[20] Mondéjar-Guerra, V., Garrido-Jurado, S., Muñoz-Salinas, R., Marín-Jiménez, M. J., and Medina-Carnicer, R. Robust identification of fiducial markers in challenging conditions. *Expert Systems with Applications 93* (2018), 336–345.

[21] Noldus, R., and Van Mieghem, P. Assortativity in complex networks. *Journal of Complex Networks 2015 3*, 4 (03 2015), 507–542.

[22] Saxena, A., and Iyengar, S. Centrality measures in complex networks: A survey. *arXiv: Social and Information Networks arXiv/2011.07190* (2020).

[23] Tian, L., Bashan, A., Shi, D.-N., and Liu, Y.-Y. Articulation points in complex networks. *Nature Communications 8*, 1 (08 2016), 1–9.

[24] Toth, J. M., Fewell, J. H., and Waters, J. S. Scaling of ant colony interaction networks. *Frontiers in Ecology and Evolution 10* (2023), 993627.
[25] Vilela, E. F., Jaffé, K., and Howse, P. E. Orientation in leaf-cutting ants (formicidae: Attini). *Animal Behaviour 35*, 5 (1987), 1443–1453.
[26] Zhang, Z., Hu, Y., Yu, G., and Dai, J. Deeptag: A general framework for fiducial marker design and detection. *IEEE Transactions on Pattern Analysis; Machine Intelligence 45*, 03 (2023), 2931–2944.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu, Cluj-Napoca 400084, Romania
Email address: bogdan.mursa@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 2, 2022
DOI: 10.24193/subbi.2022.2.05

COMPARISON OF DATA MODELS FOR UNSUPERVISED TWITTER SENTIMENT ANALYSIS

SERGIU LIMBOI

Abstract. Identifying the sentiment of collected tweets has become a challenging and interesting task. In addition, mining and defining relevant features that can improve the quality of a classification system is crucial. The data modeling phase is fundamental for the whole process since it can reveal hidden information from the textual inputs. Two models are defined in the presented paper, considering Twitter-specific concepts: a hashtag-based representation and a text-based one. These models will be compared and integrated into an unsupervised system that determines groups of tweets based on sentiment labels (positive and negative). Moreover, word-embedding techniques (TF-IDF and frequency vectors) are used to convert the representations into a numeric input needed for the clustering methods.

The experimental results show good values for Silhouette and Davies-Bouldin measures in the unsupervised environment. A detailed investigation is presented considering several items (dataset, clustering method, data representation, or word embeddings) for checking the best setup for increasing the quality of detecting the sentiment from Twitter's messages. The analysis and conclusions show that the first results can be considered for more complex experiments.

1. Introduction

In the last years, social media has gained ground based on the fact that people can express their feelings, ideas, and attitudes regarding almost everything. An interesting platform is Twitter, where users can write short messages (of maximum 280 characters), called tweets, and can follow other users and observe opinion trends or reviews about politics, social events, pop stars, etc. According to these new tendencies, the Sentiment Analysis domain becomes suitable for analyzing and detecting the hidden sentiment from messages of short lengths. Hence, the main goal of a lot of designed systems was

Received by the editors: 4 April 2023.

2010 *Mathematics Subject Classification.* 68T30, 68T50.

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

Key words and phrases. Sentiment Analysis, Twitter, Data Representation, Hashtags, Clustering.

to identify if a given piece of information reflects a specific sentiment (e.g., positive, negative, neutral). Exploring and handling texts is a difficult task due to the free style of writing, colloquial language, and the use of abbreviations. Besides the preprocessing step, an essential phase is represented by the way researchers model the data based on the collected messages. So, it could be the case that unrevealed features (attributes that are not part of the actual text- e.g., metadata, emoticons, etc.) can have a high impact on the sentiment detection process and can be related to other built attributes. According to [3], several features are used in literature: lexicon-based (derived from the use of sentiment lexicons), linguistic attributes (number of nouns, adverbs, adjectives), part of speech tagging or others like number of curse words, greeting words, or question marks. All these new features can be combined and define more complex models (e.g. a model has both linguistic features and part of speech tagging) that are very important for the entire process that handles textual information.

Our designed system has the goal of determining if different models or data representations are suitable to identify the sentiment of tweets in the unsupervised context. It is well known that clustering techniques aim to determine groups of instances (in this case, messages) where objects from the same group are very similar and different from the objects of the other groups. Therefore, an analysis of the relevance of the two types of features (hashtag and tweet text) using unsupervised techniques is proposed. Applying different clustering algorithms, we want to determine two groups of messages (one positive and one negative) by using two new models. Bearing in mind that in Twitter's world, hashtags represent an important feature since they are indicators of the message, we define a hashtag representation that will use this concept determined from the tweet. On the other hand, a text-based representation is built based on the idea that maybe the text (without hashtags) composes a relevant input for the sentiment detection problem. Furthermore, in the numerical experiments, we will determine which model is better for sentiment classification in the unsupervised context.

Finally, the original contributions of this paper are the following. Two data representations are defined based on standalone features extracted from tweets: text-based and hashtag-based. The defined representations are applied in the unsupervised context for detecting two groups of messages: one with positive tweets and one with negative messages. The main contribution is represented by comparing the two representations to determine which fits best in the unsupervised scenario. According to our previous experiments for the supervised approach [6], the presented representations are not new but crucial for the starting point of defining more complex and interesting features based

on tweets. Moreover, a detailed analysis is conducted to check if the clustering technique impacts the process in combination with two word-embedding methods (TF-IDF and frequency vector).

The remainder of the paper presents the related work in Chapter 2 and the whole methodology (architecture, steps, data models) in Chapter 3. The experimental setup is highlighted in Chapter 4, focusing on the analysis of results. In the end, conclusions and future work are specified in Chapter 5.

2. Related Work

In literature, various approaches define new features or models for detecting the sentiment of a collection of tweets. For example, one of the novel features is the one described in [7]. A flexible feature is built considering its proximity words extracted from a given tweet. An interesting survey is the one of Zhang [13] that presents different features for detecting mental illnesses from textual input, especially if people can present depression symptoms. Various attributes can be handled for this context, from depressive symptoms lexicons, emotion lexicons, and mood emoticons to emotion variability features. An attractive model is OL-DAWE presented in [10], where a tweet's sentiment is reversed if there are many negative words in the message. The system proposed by Chiong [2] uses three groups of features considering sentiment lexicons and platform-specific features for depression detection for tweets input. Therefore, several features are explored: the number of positive, negative, and neutral words, the number of links, negative terms or retweets, or linguistic attributes (e.g., the ratio of adverbs and adjectives). The system of [5] uses hashtags to detect different emotions (e.g., sadness, joy, etc.). A term frequency is computed for each hashtag, and four hashtag-based emotion lexicons are built and applied for the whole process.

The presented paper explores two basic features from collected texts and compares them to determine which fits best for the sentiment detection of tweets. The following section will describe the entire methodology of our approach.

3. Methodology

3.1. **Architecture.** The architecture of the entire system is illustrated in **Figure 1**. In the initial point, the relevant tweets are collected for the experiments. This step is enhanced with the sentiment label provided by the Vader lexicon (if the label is not already present in the dataset). Then, the data is pre-processed and modeled based on two representations: text-based and hashtag-based. These representations extract the relevant aspects from a tweet and pass the outcome to a word embedding step where the models are

converted into numerical representations. These inputs will be handled by a clustering algorithm to determine groups of similar data. Next, the clusters can be evaluated and visualized. In addition, the initial data can be visualized for further comparisons. In the following subsections, every phase will be explained.

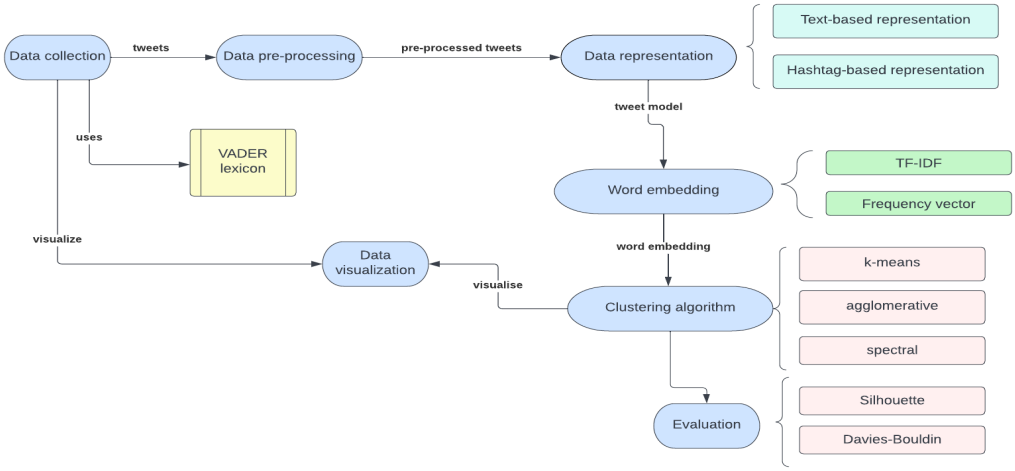


FIGURE 1. System Overview

3.2. **Pre-processing.** The data pre-processing phase is very important in the Sentiment Analysis process because the system handles textual information. Thus, some cleanup mechanisms are needed to provide the proper input for the data modeling phase. The following operations are used to pre-process the collected tweets: lowercasing, removal of punctuation, stop words, and special characters, and stemming by using the Porter Stemmer [1].

3.3. **Data Representation.** After the pre-processing of tweets, there is a need to extract valuable information from the data. So, various features can be built to handle tweets and to find a proper representation that can be converted into numerical input for the clustering algorithms.

We define two data representations that will be analyzed during our experiments: **text-based** and **hashtag-based** representation. Since a hashtag is an important indicator for a tweet, highlighting the keywords of the short message, we consider that we can design a representation that will take into

[1]http://snowball.tartarus.org/algorithms/porter/stemmer.html

account this aspect. On the other hand, we want to observe the impact of removing these keywords from the tweet and keep only the text for further stages.

3.3.1. *Hashtag Representation.* Considering a collection of tweets $T = \{tweet_1, tweet_2, ..., tweet_t\}$ where t is the length of the tweet, and $tweet_i$ is a message that contains hashtags and text, the hashtag representation will be defined for a $tweet^i_{hash}$:

$$tweet^i_{hash} = \{hash_1, hash_2, ..., hash_h\} \tag{1}$$

where $hash_i$ is the *i-th* hashtag of the tweet from the collection T and h is the number of extracted hashtags from the tweet $tweet^i$,

For example, if there is the tweet "Donald Trump will be the next #president #Trump for 2016 #victory", the hashtag-based representation will contain the corresponding list of hashtags: {president, Trump, victory}.

3.3.2. *Text-based Representation.* The text-based representation will start from the collection of tweets T where every tweet will contain the textual information without the hashtags. So, a $tweet^i_{text}$ will be specified in the next way:

$$tweet^i_{text} = \{word_1, word_2, ..., word_w\} \tag{2}$$

where $word_i$ is the *i-th*word of the tweet from the collection T (it cannot be a hashtag) and w is the number of words that compose the tweet.

If we use the same example as in the hashtag-based representation, the corresponding list of words will be: {Donald, Trump, will, be, the, next, for, 2016}. Of course, if we apply the pre-processing rules, the list will be shortened.

3.4. **Word embedding Representation.** The next step is representing by the conversion of the previously defined models into a numerical representation, technique called **word embedding**. Two methods are used for the experiments: **TF-IDF** and **Count vectorizer**.

TF-IDF (term frequency-inverse document frequency) [1] is defined based on the next formulas

$$TF(term) = \frac{m}{M} \tag{3}$$

and

$$IDF(term) = log(\frac{N}{n}) \tag{4}$$

where m is the number of times the term (word/ hashtag, in our case) appears in the tweet, M is the number of terms in the tweet, N is the number of tweets and n is the number of tweets where the term appears in the collection T.

The **Count vectorizer** is the Python naming [2] for a frequency vector. Basically, it determines for every word/ hashtags the number of appearances. If we have the tweet "Donald Trump will be the next president. He is the best president", the words will have the frequencies: Donald-1, Trump-1, will-1, be-1, the-2, next-1, president-2, he-1, is-1, best-1.

3.5. **Clustering Algorithms and Evaluation Measures.** The next stage will be represented by the clustering algorithm that will use the numerical input modeled in the previous phase to determine relevant groups of data. The main idea is that information from the same group is very similar and different from data from other clusters. For our experiments, the goal is to determine two clusters: one with positive tweets and the other one with negative messages. As algorithms, three techniques are used: **k-means** [12], **agglomerative** [12] and **spectral clustering** [9].

The result of the clustering is evaluated via internal measures like **Silhouette** and **Davies-Bouldin** indices [11].

3.6. **Vader Lexicon.** We label the datasets, in case the sentiment is missing, to visualize the information and have a better view of tweets, with the corresponding polarity by using the Vader lexicon [4]. **Vader (Valence Aware Dictionary and Sentiment Reasoner)** lexicon determines a compound value for every word. Then, a so-called sentiment score of a message *tweet* will be the sum of the sentiment scores of the corresponding terms (word or hashtag):

$$score(tweet) = \sum_{i=1}^{q} score_{Vader}(term_i), \tag{5}$$

where q is the length of tweet *tweet* and $score_{Vader}(term_i)$ is the sentiment score of the i^{th} word.

All in all , the sentiment label of tweet *tweet* is determined as follows:

$$sentiment_{label}(tweet) = \begin{cases} positive, & \text{if } score(tweet) > 0.05 \\ negative, & \text{otherwise} \end{cases} \tag{6}$$

where 0.05 is a threshold computed taking into account different experiments from the literature. So, the dataset that does not contain the sentiment label will be enriched with this information via the Vader lexicon.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.h

3.7. **Data Visualization.** Data visualization is an important phase during the experiments. There is a need to visualize data before and after the clustering process in order to proceed with detailed analysis and comparison between representations (text and hashtag-based). So, the t-SNE technique is used for dimension reduction of the dataset [8]. In other words, each collection of tweets is reduced from the high-dimensional representations to two dimensions and visualized according to the two sentiments (positive and negative).

4. Experimental setup

The experiments will cover all previously mentioned phases focusing on the datasets, results, and comparison between different experiments in order to highlight which model or representation fits properly for tweets.

4.1. **Data sets.** Three data sets are used for the numerical experiments. The first one contains tweets related to president Joe Biden [3]. This textual input will be referenced as **Joe Biden data set** in the experiment's details. The data collection consisted of 357 tweets that contain hashtags, divided into 298 positive messages and 59 negatives. As a remark, this is an unlabelled data set, so the Vader lexicon will be used to determine an initial sentiment for each message. The second data set is composed of 4.316 tweets with hashtags (3219 positive and 1097 negative) related to COVID-19, messages collected from the April-June period of 2020 [4]. In the experiments, this collection will be called **COVID-19 data set**.

The third data set has tweets from the 2016 USA presidential debate of the Republican Party [5]. It consists of 13.871 labeled messages that are positive, negative, or neutral. Since the focus is also on tweets that have hashtags, 10.323 are kept for the experiments, and only the positive and negative ones are used (the focus is on a binary sentiment). Hence, 2.180 messages are positive, and 8142 are negative. This fact leads to the idea that the data set is quite unbalanced. Therefore, we took 30% as the testing dataset (3097 tweets where 2180 are labeled as positive and 917 as negative messages). In addition, this collection will be called **Republican Presidential debate data set**.

4.2. **Experiments for Joe Biden Data Set.** Before applying the clustering techniques to the chosen dataset, a visualization step is used for checking how data is distributed according to the target classes (positive and negative tweets). Vader lexicon is used for determining the label since the collection does not have the sentiment. T-SNE is used for both previously defined data

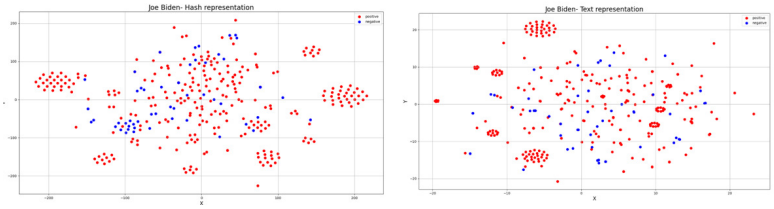
[3]https://www.kaggle.com/ibrahimrrz/tweeter-nlp

[4]https://www.kaggle.com/arunavakrchakraborty/covid19-twitter-dataset

[5]https://www.kaggle.com/datasets/crowdflower/first-gop-debate-twitter-sentiment

representations (hashtag-based and text-based) and word embeddings (TF-IDF and Count Vectorizer). This is a mandatory phase for proceeding with a detailed comparison and drawing relevant conclusions.

4.2.1. *Experiments using the TF-IDF embedding.* Figure **2** presents the dataset before the clustering process for both models. Then, the three algorithms (k-means, agglomerative and spectral) are used for determining two groups of tweets: one with positive messages and one with negative ones.



(A) (B)

FIGURE 2. Initial Joe Biden dataset (A)Hashtag (B)Text

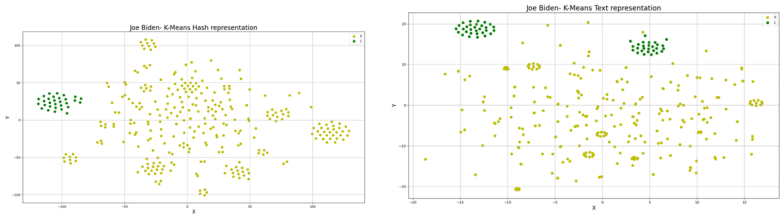
The results of the grouping are reflected in Table **1**, presenting the Silhouette and Davies-Bouldin values. A value closer to 1 is a better clustering for the Silhouette measure and a lower value is a good indicator for the Davies-Bouldin metric.

TABLE 1. Joe Biden dataset- Clustering results for TF-IDF

Clustering algorithm	Hashtag		Text	
	Silhouette	Davies-Bouldin	Silhouette	Davies-Bouldin
K-Means	**0.215**	**0.761**	**0.142**	**0.877**
Agglomerative	0.214	0.788	0.138	0.982
Spectral	0.168	0.788	0.138	0.982

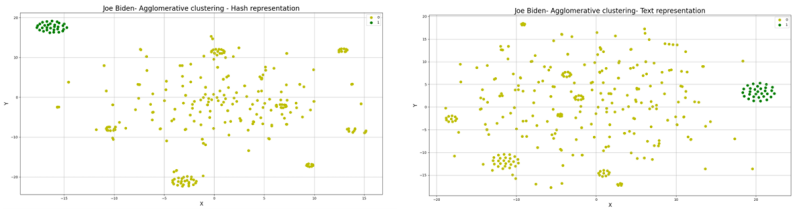
The figures 3 , 4 and 5 highlight the t-SNE representation of the clustering results for the defined models considering the three mentioned techniques.

4.2.2. *Experiments using the frequency vector/Count Vectorizer embedding.* The next word embedding used in the experiments is the frequency vector implemented via the Count Vectorizer library from Python. Figure **6** presents the initial dataset after the modeling with the hashtag and text representation and converting the representations into frequency vectors.



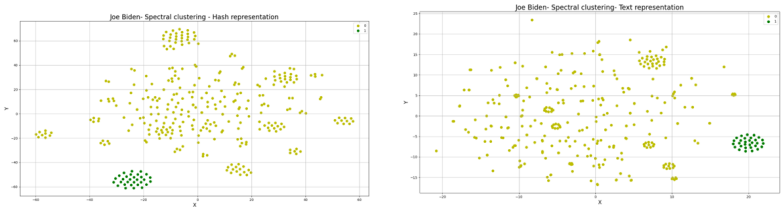
(A) (B)

FIGURE 3. K-Means TF-IDF (A)Hashtag (B)Text



(A) (B)

FIGURE 4. Agglomerative TF-IDF (A)Hashtag (B)Text



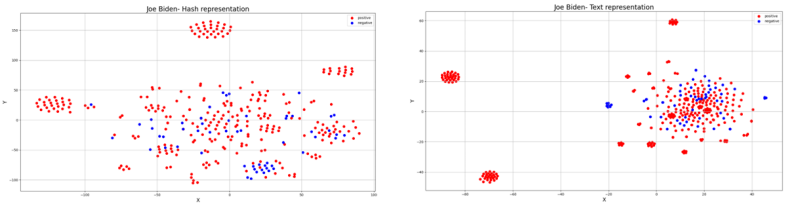
(A) (B)

FIGURE 5. Spectral TF-IDF (A)Hashtag (B)Text

Then, the clustering results for all the techniques are briefly described in Table **2**. It is noteworthy that there is no significant distinction between the used clustering algorithms.

In the end, the clustered data is illustrated in figures 7, 8, and 9.

4.3. **Analysis for Joe Biden dataset experiments.** From the t-SNE visualization of the Joe Biden dataset, we can notice that the collection is quite unbalanced. There are a lot of positive messages (marked with red color) and

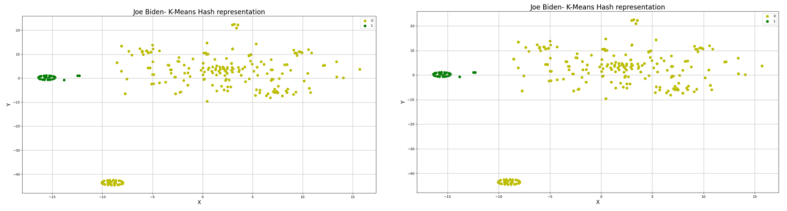


(A) (B)

FIGURE 6. Initial Joe Biden dataset with frequency vector: A)Hash B) Text

TABLE 2. Joe Biden dataset- Clustering results for Count vectorizer

Clustering algorithm	Hashtag		Text	
	Silhouette	Davies-Bouldin	Silhouette	Davies-Bouldin
K-Means	**0.102**	**0.971**	**0.095**	**0.998**
Agglomerative	0.101	0.982	0.079	1.123
Spectral	0.101	0.982	0.079	1.123

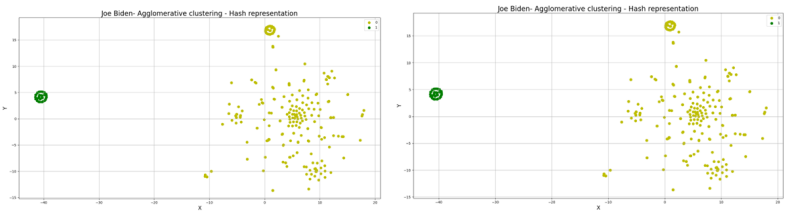


(A) (B)

FIGURE 7. K-Means A)Hashtag (B)Text

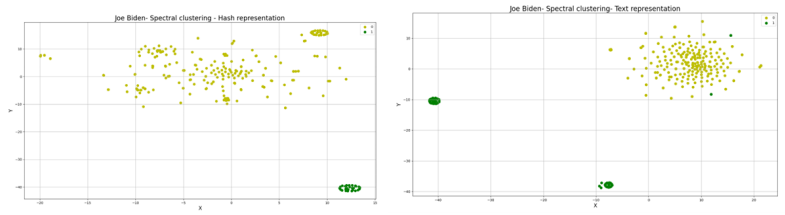
only a few negative tweets (the blue color indicates the negative sentiment). In addition, from the initial visualizations, we can notice that for the hashtag-based model, several sub-groups/ sub-clusters can indicate potential hashtag-based clusters. In other words, from the big group of positive messages, we can deduce small groups that have as highlights some relevant hashtags.

4.3.1. *Clustering algorithm analysis.* From the experiments, we can observe that the best values for the hashtag-based and text-based experiments, for both word embeddings, are the ones produced by the K-means algorithms.



(A) (B)

FIGURE 8. Agglomerative clustering A)Hashtag (B)Text



(A) (B)

FIGURE 9. Spectral clustering A)Hashtag (B)Text

Analyzing the values for the Silhouette and Davies-Bouldin indexes, there is no significant difference between the used clustering methods (the values are quite similar with only small variations). Therefore, the added value is not represented by the technique and the enhancement is reflected by how data is modeled (the defined representations). Also, regarding the t-SNE visualization of the clustering results, it can be highlighted the idea that there are a lot of positive messages and only a spot of negative tweets.

4.3.2. *Word embeddings analysis.* Comparing the results from TF-IDF and frequency vector embeddings, the best values, in terms of Silhouette and Davies-Bouldin, are the ones when the representations are converted into TF-IDF. This embedding brings relevance to our context since we work with texts of small lengths. When we convert textual input into frequency vectors we can face the issue that text is really small after the preprocessing phase. The evaluation in terms of embedding techniques is supported also by the t-SNE visualization. For the TF-IDF case, we distributed data in small sub-groups, but for the count vectorizer situation, after the clustering, there are a lot of dots (instances) distributed randomly and only two or three subgroups can be identified (so no semantic relevance can be deduced from the visualizations).

4.3.3. *Data representation analysis.* The best results are achieved by the hashtag-based representation in the unsupervised context (determining groups of similar tweets). This case is reflected also in the visualization part where we can identify subgroups from the dataset. All these things highlight the idea that hashtags are relevant concepts for the Twitter world and they can be drivers for defining the groups.

4.3.4. *Summary of the analysis.* **Table 3** sums up the conclusions of the analysis for the experiments conducted on the Joe Biden dataset.

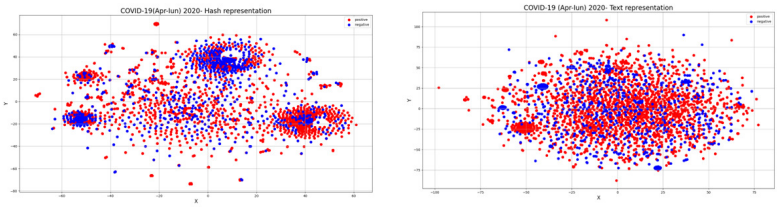
TABLE 3. Joe Biden dataset- Analysis summary

Concept	Conclusions
The clustering algorithm	K-Means produces the best values, but there is no significant difference between the used clustering algorithms
Word embedding	TF-IDF has better results than frequency vector
Data representation	For the unsupervised context, the hashtag-based is more relevant than the text-based model
Dataset	It is quite unbalanced: 298 positive tweets and 59 negatives.
Data visualization	Better visualization for the hashtag-based representation since we can identify relevant subgroups (conceptual/semantical clusters) in comparison with the text-based model.

4.4. **Experiments for COVID-19 Data Set.** Considering the previous experiments and the ones conducted on the second dataset (COVID-19), we will present only the results for one clustering algorithm (K-Means) and the TF-IDF embedding. The initial visualization of the dataset for both representations (text-based and hashtag-based) is presented in **Figure 10**.

The clustering evaluation is given in the **Table 4** that illustrates the Silhouette and Davie-Bouldin values for the K-Means algorithm for both representations. The clustering visualization is presented in **Figure 11**.

4.4.1. *Analysis and conclusions.* The COVID-19 dataset is bigger than the previous one and data is more balanced than the Joe Biden set, but still quite unbalanced in terms of positive and negative sentiments. The clustering evaluation reflects the idea that the hashtag representation is better than the

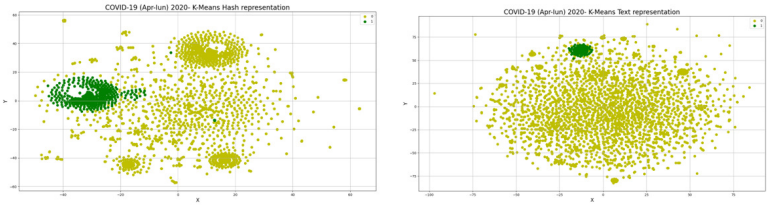


(A) (B)

FIGURE 10. Initial COVID-19 dataset A)Hashtag (B)Text

TABLE 4. COVID 19 dataset- Evaluation for TF-IDF embedding

Representation	Silhouette	Davies-Bouldin
Hashtag-based	0.166	1.030
Text-based	0.022	1.044

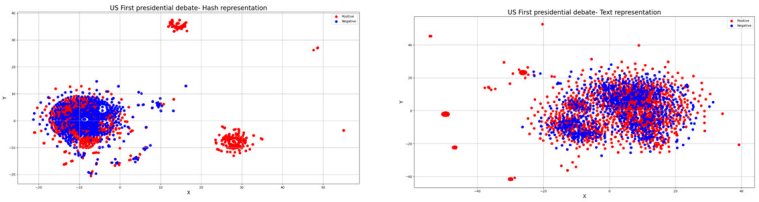


(A) (B)

FIGURE 11. K-Means A)Hashtag (B)Text

text-based one. This conclusion is strengthened by the data visualization since we have grouped data where we can identify more subgroups. The text-based visualization is a big cluster of positive tweets and a smaller group of negative instances.

4.5. **Experiments for 2016 Republican Presidential Debate Data Set.** The visualization of the initial dataset is presented in **Figure 12**. The test data is more balanced (2180 positive and almost 1000 negative) than the others, an idea reflected also in the t-SNE visualization. The experiments are conducted using the K-Means algorithm and TF-IDF embedding for text-based and hashtag-based representations.



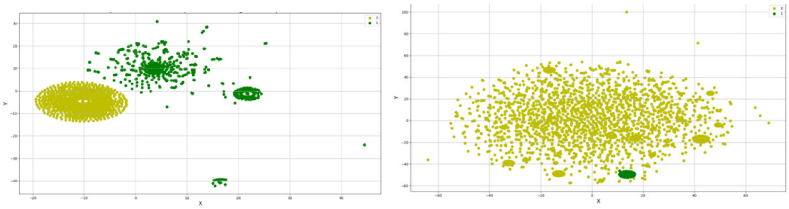
(A) (B)

FIGURE 12. Initial Republican Debate dataset A)Hashtag (B)Text

Table 5 defines the Silhouette and Davies-Bouldin values for the defined representations, using the K-Means technique. The clustering visualization is drawn in **Figure 13**. As evident from the evaluation measures' values and t-SNE visualization, the text representation gives pretty poor outcomes. Almost all instances are labeled as one class, and only a few are marked as the opposite one.

TABLE 5. COVID 19 dataset- Evaluation for TF-IDF embedding

Representation	Silhouette	Davies-Bouldin
Hashtag-based	0.445	0.566
Text-based	0.011	1.099



(A) (B)

FIGURE 13. K-Means A)Hashtag (B)Text

Based on the analysis, we can conclude that also for bigger datasets the hashtag-based representation is better than the text one, in the unsupervised

context. Moreover, we notice that we do not have the small subgroups presented in the smaller datasets. Thus, the models tend to evolve into two main clusters with the defined labels: positive and negative.

4.6. **Comparisons and Summary. Table 6** presents the final conclusions of the conducted experiments on the three datasets: Joe Biden, COVID-19 and 2016 Republican Presidential Debate.

Table 6. Conclusions and summary

Concept	Conclusions
Clustering method	No significant difference between clustering technique
Word embedding	TF-IDF has better results than frequency vector
Data representation	For the unsupervised context, the hashtag-based is the relevant mode
Dataset	The Joe Biden dataset has 298 positive tweets and 59 negatives. The COVID-19 collection has 3219 positive and 1097 negative. The last one contains 2180 positive and 917 negative

5. Conclusions and Future Work

In the Sentiment Analysis area, there is a need to define new data representations and explore the valuable information the collected input offers. In this paper, we used two data representations for textual information of short lengths, in this case, tweets, that use the whole text or extract relevant platform-specific features: hashtag-based and text-based representations. Moreover, several clustering algorithms apply these two in the unsupervised learning context. The goal is to determine two main groups of tweets according to two sentiment labels: positive and negative. The experimental results reveal only a slight difference between the used clustering techniques. Therefore, the data representations bring the main enhancement. Regarding evaluation and data visualization, hashtag representations handle short messages better than text ones. Even though this is a simple methodology that uses two existing representations and analyzes which one fits better in the unsupervised context, our plan involves more interesting and complex work. The plan is to define topic-driven clusters based on the most popular and relevant hashtags

collected from the data, exploring bigger datasets. Also, using the models defined in [6] and combined with more complex ones (e.g., BERT-based models) will be quite interesting. Overall, the first results are encouraging and design the steps for more exploratory and extensive experiments.

References

[1] Baeza-Yates, R., Ribeiro-Neto, B., et al. *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[2] Chiong, R., Budhi, G. S., and Dhakal, S. Combining sentiment lexicons and content-based features for depression detection. *IEEE Intelligent Systems 36*, 6 (2021), 99–105.

[3] Hung, L. P., and Alias, S. Beyond sentiment analysis: A review of recent trends in text based sentiment analysis and emotion detection. *Journal of Advanced Computational Intelligence and Intelligent Informatics 27*, 1 (2023), 84–95.

[4] Hutto, C., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media* (2014), vol. 8, pp. 216–225.

[5] Koto, F., and Adriani, M. Hbe: Hashtag-based emotion lexicons for twitter sentiment analysis. In *Proceedings of the 7th Annual Meeting of the Forum for Information Retrieval Evaluation* (2015), pp. 31–34.

[6] Limboi, S., and Dioşan, L. Hybrid features for twitter sentiment analysis. In *Artificial Intelligence and Soft Computing: 19th International Conference, ICAISC 2020, Zakopane, Poland, October 12-14, 2020, Proceedings, Part II 19* (2020), Springer, pp. 210–219.

[7] Pilar, G.-D., Isabel, S.-B., Diego, P.-M., and Luis, G.-Á. J. A novel flexible feature extraction algorithm for spanish tweet sentiment analysis based on the context of words. *Expert Systems with Applications 212* (2023), 118817.

[8] Van der Maaten, L., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, 11 (2008).

[9] Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing 17* (2007), 395–416.

[10] Wang, W., Li, B., Feng, D., Zhang, A., and Wan, S. The ol-dawe model: tweet polarity sentiment analysis with data augmentation. *IEEE Access 8* (2020), 40118–40128.

[11] Xu, D., and Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science 2* (2015), 165–193.

[12] Xu, R., and Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on neural networks 16*, 3 (2005), 645–678.

[13] Zhang, T., Yang, K., Ji, S., and Ananiadou, S. Emotion fusion for mental illness detection from social media: A survey. *Information Fusion 92* (2023), 231–246.

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

Email address: sergiu.limboi@ubbcluj.ro