Avrupa Bilim ve Teknoloji Dergisi
Sayı 31 (Ek Sayı 1), S. 991-1003, Aralık 2021
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 31 (Supp. 1), pp. 991-1003, December 2021
Copyright © 2021 EJOSAT
Research Article

Comparison of Different Classification Algorithms for Extraction Information from Invoice Images Using an N-Gram Approach

Resmiye Nasiboglu[1*], Adem Akdogan[2]

[1*] Dokuz Eylul University, Faculty of Science, Departmant of Computer Science, Izmir, Turkey, (ORCID: 0000-0003-1739-1469), resmiye.nasiboglu@deu.edu.tr
[2] Dokuz Eylul University, The Graduate School of Natural and Applied Sciences, Izmir, Turkey, (ORCID: 0000-0002-4236-1563), adem.akdogan92@gmail.com

(First received 24 December 2020 and in final form 6 October 2021)

(**DOI:** 10.31590/ejosat.844862)

ATIF/REFERENCE: Nasiboglu, R., & Akdogan, A. (2021). Comparison of Different Classification Algorithms for Extraction Information from Invoice Images Using an N-Gram Approach. *European Journal of Science and Technology,* (31), 991-1003.

Abstract

Artificial intelligence (AI) has started to be used in many areas today. One of these areas is the accounting sector. Accounting companies may sometimes be inadequate especially in the face of intense invoicing transactions of large companies. This problem raised the need to process invoices by an Artificial Intelligence powered system. The goal of this work is to determine the best machine learning model to extract information such as invoice number, invoice date, due date, delivery date, total gross, total net, vat amount and IBAN from the invoice image files. Information obtained by the Tesseract Optical Character Recognition (OCR) system has been converted into n-gram format. A number of attributes of the n-gram are calculated such as the coordinates, the length, the width, the line number, the template information of n-grams, the Levenshtein and the Jaro-Winkler distances between the candidate n-grams and the keywords in the control keywords list. The use of the Levenshtein distance between candidate n-grams and the control keywords has resulted in a sufficiently high predictive rate. The most appropriate model and features are determined for the training. Algorithms such as Random Forest, Gradient Boosting Machine, Extreme Gradient Boosting, K-Nearest Neighbors, AdaBoost and Decision Tree were compared as prediction models. A total of 9910 invoices were used by splitting 80% for training and 20% for testing. It was observed that the Random Forest model using the Levenshtein distance is the best model with an average F1 score of 0.9137.

Keywords: Machine learning, Information extraction, N-gram, Levenshtein distance, Jaro-Winkler distance.

N-Gram Yaklaşımı Kullanılarak Fatura Görüntülerinden Bilgi Çıkarımında Farklı Sınıflandırma Algoritmalarının Karşılaştırılması

Öz

Yapay Zeka (AI) günümüzde birçok alanda kullanılmaya başlanmıştır. Bu alanlardan biri de muhasebe sektörüdür. Özellikle büyük firmaların yoğun faturalama işlemleri karşısında muhasebe firmaları bazen yetersiz kalabilmektedir. Bu sorun, faturaların Yapay Zeka destekli bir sistemle işlenmesi ihtiyacını ortaya çıkarmıştır. Bu çalışmanın amacı, fatura görüntü dosyalarından fatura numarası, fatura tarihi, vade bitiş tarihi, teslim tarihi, toplam brüt, toplam net, kdv tutarı ve IBAN gibi bilgileri çıkarmak için en iyi makine öğrenme modelini belirlemektir. Çalışmada, Tesseract Optik Karakter Tanıma sistemi ile elde edilen bilgiler n-gram formatına dönüştürülmüştür. N-gramların koordinatları, uzunluk, genişlik, satır numarası gibi şablon bilgileri, aday n-gramlar ile kontrol anahtar kelimeler listesindeki anahtar kelimeler arasındaki Levenshtein ve Jaro-Winkler mesafeleri gibi bir dizi öznitelikleri hesaplanmıştır. Aday n-gramlar ile kontrol anahtar kelimeler arasındaki Levenshtein mesafesinin kullanılması, yeterince yüksek bir tahmin oranıyla sonuçlanmıştır. Eğitim için en uygun model ve özellikler belirlenmiştir. Tahmin modelleri olarak Rassal Orman (Random Forest), Gradyan Yükseltme Makinesi (Gradient Boosting Machine), Aşırı Gradyan Yükseltme (Extreme Gradient Boosting), K-En Yakın Komşu (K-Nearest Neighbors), AdaBoost ve Karar Ağacı (Decision Tree) gibi algoritmalar karşılaştırılmıştır. Çeşitli firmalardan toplanan 9910 adet fatura, %80'i eğitim ve %20'si test olacak şekilde bölünerek kullanılmıştır. Levenshtein mesafesini kullanan Rassal Orman modelinin ortalama 0,9137 olan F1 puanı ile en iyi model olduğu görülmüştür.

Anahtar Kelimeler: Makine öğrenimi, Bilgi çıkarımı, N-gram, Levenshtein uzaklığı, Jaro-Winkler uzaklığı.

[*] Corresponding Author: The Department of Computer Science, Dokuz Eylul University, Izmir, Turkey, ORCID: 0000-0003-1739-1469, resmiye.nasiboglu@deu.edu.tr

1. Introduction

Digitization of invoices, receipts and similar company documents has become very common with the rapid development in technology. It is a very labor intensive and expensive process to obtain and manage the information in these documents using traditional methods that depend on manpower (Klein et al., 2004). In addition, manual processing of these documents poses a serious timing problem. In particular, invoices are documents that need to be processed and managed continuously. As many companies use different types of invoices, we can see that the invoices have a very variable structure. Although we have strong estimation methods and techniques for specific templates, these methods do not work for uncertain documents. Therefore, these methods can only be applied to a small number of limited problems.

The main goal of our study is to compare machine learning models to extract information from invoices without a specific template. First of all, text information is extracted from the invoice images. For this, Tesseract is used that is an optical character recognition (OCR) engine. After obtaining the text information, the stage of creating useful features is performed. The location of words is very important in structural documents such as invoices and receipts. Especially, other words that are around the target word can give us remarkable information abaout our target. These words are vectorized by using distance values from potential words that can be keywords determined according to frequency.

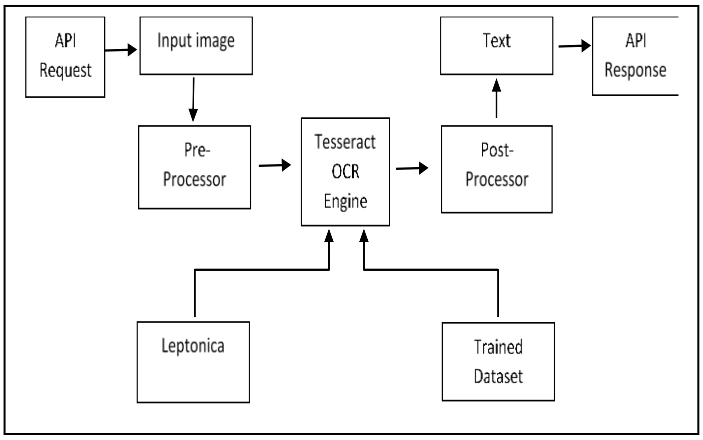


Figure 1. Tesseract OCR Process Scheme (Zelic and Sable 2020)

Two distances such as the Levenshtein distance and the Jaro-Winkler distance are used for measuring of word similarities. Various machine learning models such as Random Forest, Gradient Boosting Machine, Extreme Gradient Boosting, K-Nearest Neighbors, AdaBoost and Decision Tree were compared as prediction models. There are eight fields that we want to predict on the invoice such as invoice number, invoice date, delivery date, due date, total gross, total net, vat amount and IBAN. Instead of using a general trained model for predicting of all fields, different models are used for each field. Thus, the attributes that best represent that field are used in the training of each field. A dataset containing total 9910 invoice images collected by authors from different firms is used in training and testing process by dividing 80% and 20% as appropriate. Since the invoices contain sensitive financial information, the dataset cannot be presented as an open source. The Random Forest model with Levenshtein distance is obtained as the best prediction model for all fields.

The rest of the article is organized as follows. Information about the tools, methods and models used in this work are given in chapter 2. Related work and the proposed approach are given in chapter 3. Chapter 4 provides information about the data set. Explanation and discussion of obtained results are also included in this chapter. Finally, the concluding remarks are given in chapter 5.

2. Methodology

2.1. Tesseract OCR

The first tool we used in the document processing is an Optical Character Recognition (OCR) engine called Tesseract, which is supported by the Google, in order to convert documents in image format to text format (Smith, 2007). The scheme of the Tesseract OCR is given in Fig. 1.

While the documents are converted to text via Tesseract, information of the words such as coordinate information, line number, height and width can also be obtained (Table 1).

Line-based n-gram structure is created from the list of words generated by Tesseract OCR. While creating the n-gram structure, besides some features used in the study (Palm et al., 2017), other effective features can be calculated (Table 2). The most suitable features are obtained by processing the attributes of n-grams in different combinations for training.

Table 1. Attributes Obtained with Tesseract OCR.

Attribute name	**Description**	**Attribute name**	**Description**
level	Level of the detected unit	*left*	X coordinate of the upper-left corner of the detected unit
page_num	Page number of the detected unit	*top*	Y coordinate of the upper left corner of the detected unit
block_num	The block number of the detected unit	*width*	Width of the detected unit
par_num	Paragraph number of the detected unit in the block	*height*	Height of the detected unit
line_num	Line number of the detected unit in the paragraph	*conf*	Recognition accuracy percentage of detected unit
word_num	The word number of the detected unit on the line	*text*	Text of the detected unit

Table 2. Attributes Created for the N-gram Structure.

The attribute of n-gram	**Description**	**The attribute of n-gram**	**Description**
RawText	Text of n-gram	*Left*	Left margin of n-gram (X coordinate)
TextPattern	Template of n-gram	*Top*	Top margin of n-gram (Y coordinate)
IsFirstStr	Is the first character of n-gram an alphabetical one?	*LeftMargin*	Proportional left margin to the page width of n-gram
IsFirstInt	Is the first character of n-gram a digit?	*TopMargin*	Proportional right distance to the page width of n-gram
IsFirstSpc	Is the first character of n-gram a special character?	*HasDigit*	Whether there is a digit in the n-gram
IsLastStr	Is the last character of n-gram an alphabetical one?	*FirstQuarter*	The n-gram is in the first quarter of the page
IsLastInt	Is the last character of n-gram a digit?	*SecondQuarter*	The n-gram is in the second quarter of the page
IsLastSpc	Is the last character of n-gram a special character?	*ThirdQuarter*	The n-gram is in the third quarter of the page
StrCount	Total number of alphabetical characters in n-gram	*FourthQuarter*	The n-gram is in the fourth quarter of the page
IntCount	Total number of digits in n-gram	*LineNo*	Number of the line of the n-gram
SpcCount	Total number of special characters in n-gram	*PageNo*	Number of the page of the n-gram
WordCount	Total number of words of n-gram	*PageWidth*	Width of the page containing n-gram
CharCount	Total number of characters of n-gram	*PageHeight*	Height of the page containing n-gram

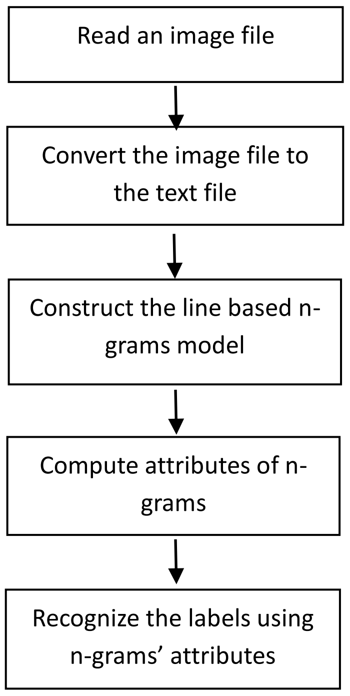

Read an image file
Convert the image file to the text file
Construct the line based n-grams model
Compute attributes of n-grams
Recognize the labels using n-grams' attributes

Figure 2. The Process of Recognizing Certain Fields from the Image File.

The process of recognizing certain labels on invoices is shown in Fig. 2. First of all, string expressions that are the result of the Tesseract OCR are converted into n-grams. Although some information such as invoice number and postal code on the invoice consists of one piece of string expression ("12345", "A1245"), other information such as address and IBAN number can have multiple string expressions ("TR 2600 1200 0000 0000 0000", "110 St No: X Ankara”). When the field we are predicting has more than one string expression, the single string expressions that Tesseract OCR outputs are insufficient. In this case, we reach the word sequence that we are looking for by using multiple n-grams (Watanabe et al., 2009).

2.2. The n-gram structure

Let's examine the word sequence "Dokuz Eylul University". An example of the creation of n-grams is given in Table 3

Table 3. Example N-gram Structure

1-grams	“Dokuz”,”Eylul”,” University”
2-grams	“Dokuz Eylul”,”Eylul University”
3-grams	“Dokuz Eylul University”

As can be seen in the example in Table 3, 1-gram and 2-gram sequences must be created in order to reach the 3-gram sequence. The following formula can be used to calculate the total number of n-grams that can be generated with maximum length w from a sequence of k numbers of words:

$$C_{n-gram} = \sum_{i=1}^{w} \big(k-(i-1)\big) =$$

$$= k \cdot w - \frac{(w-1)\cdot w}{2} = \frac{2kw - w^2 + w}{2} \qquad (1)$$

For example, the total number of n-grams is calculated as 6 by applying the Eq. (1) for Table 3. When we want to apply the same method for invoices, every n-gram sequence is created starting from 1-gram up to the specified maximum n-gram value. Thus, the n-gram sequences are achieved like the invoice date (the n-gram value must be at least 3 to detect the date "20 June 2022”). The maximum n-gram value, n was determined as 6 considering the long expressions such as address and IBAN number. Let L be the number of lines in the document and k be the number of words in each line. The total number of n-grams in the document will be calculated as follows:

$$\sum_{i=1}^{L} C_{n-gram}^{i} \le L\,\frac{2kw - w^2 + w}{2} \qquad (2)$$

In Eq. (2), C_{n-gram}^{i}, shows the number of n-grams to be formed in the line i. Let k be the number of words in the line. In the case where $w \ge k$, the constraint as $w = k$ should be considered.

The Tesseract OCR, where we receive the necessary information for the n-gram design, also offers some features that would be useful in machine training. The information of each string expression can be obtained from Tesseract OCR such as width, height, number of lines, coordinate information of the expression, page number, page width, and page height. Then, the pattern of each n-gram sequence is extracted. The pattern of the text of the n-gram is formed via replacing all lowercases with “x”, all uppercases with “X”, all numbers with “0” and all special characters with “?”. The purpose of this pattern is to provide the model with the best learning of the structure of each field to predict. For example, it may be not every expression as an alpha numerical one in the invoice date. Although it usually has a special pattern (eg, 13.09.2018), in some cases only part of the sequence may be a string expression (eg, 8 October 1993). In our work, we used information such as total letters, total digits, total special characters, and the type of the first and last character of the pattern as an attribute. Although the pattern is used in this way in this work, the pattern can also be used by making One Hot Encoding, which means that categorical variables are represented using a binary vector. In One Hot Encoding firstly, we create vectors consisting of 0 or 1 values as much as the number of categorical values for this operation. Then we create a vector by assigning 1 to the value corresponding to its index and 0 to the others for each categorical data. For example, there are data in three categories as “blue”, “red” and “green”. When these fields are decomposed into binary, the value “red” is converted to the vector (0, 1, 0).

In addition to the features mentioned above, we also used features such as the number of words in each n-gram, the number of characters in each n-gram, left margin, right margin, left alignment, right alignment, the closest 1-gram on the left, top, right and bottom. The nearest left and the nearest top n-grams are most likely to be keywords. It is conceivable that these n-grams will increase the effectiveness of the training, considering that each field has different keywords. However, there are some problems that need to be solved at this point. There are many different types of invoices. For this reason, the keywords can be a variable structure rather than a fixed one. For example, the keywords can be “invoicenummer”, “belegnummer” or “rechnungnummer” on an invoice where the invoice number is to be predicted. Also another problem is that these keywords can be shortened in some invoices (for ex. “rech.num.“) or that may be misspelled due to human errors (for ex. “recnhugnummr”). At this point, the Levenshtein and the Jaro-Winkler methods of similarity (distance) can solve these problems instead of exact matching. These methods provide us calculation of the distances between the strings from two existing sequences (Schulz and Mihov 2002). The first sequence is potential keywords that we have already obtained from n-gram. The second sequence is a specified keyword checklist. The frequencies of the potential keywords of all the invoices are calculated to create the specified keyword checklist. Keywords that exceed a specified threshold value of similarity are added to the result list. The distance between the keyword with the highest similarity is added as a feature.

2.3. The Levenshtein distance

There are three basic operations in calculating Levenshtein distance such as replace, insert and delete. The goal is to transform one string into another using the minimum number of basic operations (Haldar and Mukhopadhyay, 2011). The Levenshtein distance between any strings a and b is calculated as a recursive function as follows:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & , if \min(i,j)=0, \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & , otherwise. \end{cases} \quad (3)$$

In Eq. (3), the Levenshtein distance is calculated between the first i character of the sequence a and the first j character of the sequence b. $1_{(a_i \neq b_j)}$ is the token function, i.e. it equals 0 when $a_i = b_j$, otherwise equals to 1. The first row corresponds to the deletion, the second row to the insert, and the third row corresponds to match operations in section min in the Eq. (3).

Let's look at the calculation of the Levenshtein distance between the words "PAIN" and "CHAIN" as an example. If we use the Eq. (3), the first thing to do to convert the word "PAIN" to "CHAIN" is to add the letter "C". Subsequently, the letter "P" is converted to the letter "H" by making a replacement operation. Since the remaining three letters are the same, no action is taken on them. Thus, the Levenshtein distance between these two words is equal to 2.

2.4. The Jaro-Winkler distance

Another method to measure the similarity between two strings is the Jaro-Winkler distance (Wang et al., 2017). In order to calculate this distance between two given strings s_1 and s_2, the Jaro similarity value must be found before as follows:

$$sim_j = \begin{cases} 0 & , if\ m = 0, \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & , otherwise \end{cases} \quad (4)$$

where, $|s|$ is the length of the string, m is the number of matching characters and t is half the number of transpositions. Two characters from s_1 and s_2 respectively, are considered matching only if they are the same and not farther than

$$\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1 \quad (5)$$

positions, where $\lfloor . \rfloor$ denotes the lower bound integer value of the operand. For example, the matching range value of a 6-character word is 2. A match is accepted for the character "B" between the words "BXXXXX" and "XBXXXX", while the match is not accepted between the words "BXXXXX" and "XXXXBX". The number of matching (but different sequence order) characters divided by 2 defines the number of transpositions. For example, in comparing CRATE with TRACE, only 'R', 'A', 'E' are the matching characters, i.e. m=3. Although 'C', 'T' appears in both strings, they are farther apart than 1 (the result of $\left\lfloor \frac{5}{2} \right\rfloor - 1$), so t=0. In "DwAyNE" versus "DuANE" the matching letters are already in the same order D-A-N-E, so no transpositions are needed (Jaro, 1989). So, the Jaro-Winkler similarity value (that is used for distance calculation) can be calculated as follows

$$sim_w = sim_j + l \cdot p(1 - sim_j). \quad (6)$$

In Eq. (6), the sim_j is the Jaro similarity value (we got in Eq. (4)), l is the length of common prefix at the start of the string up to maximum of 4 characters and p is scaling factor (default value 0.1) should not exceed 0.25 (i.e., 1/4, with 4 being the maximum length of the prefix being considered), otherwise the similarity could become larger than 1. Then the Jaro-Winkler distance d_w is defined as

$$d_w = 1 - sim_w \quad (7)$$

The rest of the calculation on the invoice data according to the Jaro-Winkler similarity is made similar to the Levenshtein calculation given in section 2.4 and the calculated Jaro-Winkler similarity values are added into attributes list.

2.5. Prediction models

The Random Forest model that was concluded as the best model in this work, is based on the Decision Tree approach (Quinlan, 1986). The idea in Decision Trees is to create the decision structure by using all available attribute values in the most efficient way (Mashat et al., 2012). In this way we can achieve high success rates very quickly. One of the most used decision tree algorithms is the C4.5 algorithm to create trees (Xiaoliang et al., 2009).

The selection of attributes used in decision nodes is based on "information gain" in the C4.5 algorithm. The concept of entropy (uncertainty level) is used in determining the information gain. The following calculations to determine information gain of a given attribute A must be performed:

$$info(D) = -\sum_{i=1}^{m} p_i \log p_i\ , \quad (8)$$

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} info(D_j), \quad (9)$$

$$Gain(A) = info(D) - info_A(D), \quad (10)$$

where

p_i : The ratio of the number of labels with class i to the total number of labels,

D_j : Subset that contains only the j.th value of a given attribute A in the dataset,

$info(D)$: Total entropy value in the data set,

$info_A(D)$: Total entropy value after splitting according to different values of the attribute A,

$Gain(A)$: The information gain of the attribute A.

The total entropy value of the system is calculated with the Eq. (8) to calculate the information gain with a certain attribute. The entropy value of each attribute in the system is calculated separately with the Eq. (9). Then, the information gain provided by the value of that attribute to the system is obtained with the Eq. (10). When the calculations are completed, the gain values calculated for each attribute are compared. The attribute that provides the highest value is determined as the decision node to the next split. These nodes are combined to create an optimal decision tree structure.

Another method used to create decision trees is the Gini index method developed by IBM (Gelfand et al., 1991). In this method, calculations are made similar to the Eqs. (8) - (10). The amount of information is calculated with the help of the Gini index determined in Eq. (11) instead of Eq. (8):

$$gini(D) = 1 - \sum_{i=1}^{m} p_i^{\,2} \, . \tag{11}$$

One of the biggest problems in machine learning is overfitting. This problem is that the system loses its flexibility as a result of over-training. In the training of the decision tree model, the system creates a special structure that takes into account the details. Outlier observations may also be included in these details as a result of the model's over-training. For example, after the required eliminations are made in the decision tree, 99.990 of 100.000 data return True and the remaining 10 return False value. A separate branch can be created for these 10 data, which is inconsistent observation with the system's overfitting. This negatively affects the optimum number of branches in the system.

The Random Forest model is the solution to the above-mentioned overfitting problem of decision trees. The random forest model consists of a random combination of multiple decision trees (Aydın, 2018). This method was developed by Leo Breiman in 2001 (Breiman, 2001). The chosen decision trees should be as different as possible. The low correlation between decision trees will allow the system to yield more accurate results by lowering the amount of deviation in the overall prediction mechanism. The Model does not give equal weight to each tree that makes up itself. The model uses out-of-Bag (OOB) error rate to determine the weight of these trees. The data set is divided into 2/3 of the training and 1/3 of the test. As a result of this process, the tree with the lowest error is given the highest weight, while the tree with the highest error is given the lowest weight. In the classification problem, each tree in the forest makes a prediction. The prediction value with the majority is determined as the overall prediction value of the model.

2.6. Evaluation metrics

For the evaluation of binary classification models, various metrics are used based on confusion matrix using TP (True Positive), FP (False Positive), TN (True Negative), and FN (False Negative) values (Yıldız and Karadeniz, 2019):

$$TP = The\ number\ of\ correctly\ comfirmed\ cases,$$

$$TN = The\ number\ of\ correctly\ rejected\ cases,$$

$$FP = The\ number\ of\ vrongly\ comfirmed\ cases,$$

$$FN = The\ number\ of\ vrongly\ rejected\ cases.$$

The most used model evaluation metrics are precision, recall and F1-score values, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive:

$$precision = \frac{TP}{TP+FP} \ , \tag{12}$$

$$recall = \frac{TP}{TP+FN} \ . \tag{13}$$

The F1-score value is a more accurate evaluation metric, especially in cases where the number of positive and negative data in the training set is unbalanced. The F1-score is calculated from the precision and recall as their harmonic mean as follows:

$$F_1 = 2 \cdot \frac{precicion \cdot recall}{precision + recall} \, . \tag{14}$$

In our study, precision, recall and F1-score metrics were also used to evaluate the models.

3. Proposed Method

3.1. Related work

There are some studies in the literature to predict invoice fields from image files. Although some of them resemble our work in terms of ideas, there are differences with some critical points. The following are some of these works identified as a result of the literature review. An automatic indexing approach was used in the study (Esser et al., 2012). The desired inputs must be in precisely determined positions in this approach. Results are obtained by making index assignments to the information in this position. Although correct results can be obtained, changing the positions of the information on the invoices negatively affects the results. Therefore, only invoices in allowed templates can give good results. The Intellix is another work with similarities to the automatic indexing approach (Schuster et al., 2013). In the Intellix method proposed in the work, the classification process is performed first. Then, according to the rules, the information is extracted from the invoice. The positions in the invoice information must always be the same and precise to make the results accurate. It is necessary to use templates to get the correct results. The approach in the study (Liu et al., 2019) is to ensure that the 2-dimensional structures of the documents are preserved. It is aimed to transfer information without the need for any template. Each text has been converted into graph format and the position information of the words that they represent has been assigned to the vertexes in this graph. Thus, each word has become a vector whose coordinates are determined in space. The training was carried out with BiLSTM - CRF model. The main purpose of the work (Katti et al., 2018) is to create the model that best represents the 2-dimensional structure in the documents as in the work (Liu et al., 2019). The information is used to describe each letter in the words and their position on the document. The model consists of two main parts as encoder and decoder. As a result of the training, they achieved good results in sections such as invoice number and invoice date, unlike in sections such as seller name and address. The CloudScan approach presented in the study (Palm et al., 2017) does not require a template, unlike automatic indexing and Intellix. The working principle is based on the n-gram system. Each sentence is evaluated based on the sequence of words. The information that matches the correct format has been transferred to the system. In addition to the Logistic Regression model used as baseline, LSTM (Long Short-Term Memory) model was also used. The CloudScan approach is the most related work to the appoach proposed in this work.

3.2. Proposed Approach

The proposed in this study approach does not require any invoice document template, unlike automatic indexing and

Intellix. The working principle is also based on the n-gram analysis as in the CloudScan approach. Nevertheless, there are some important differences between CloudScan and the proposed approach. Both the training models and vectorization techniques used are different. The main difference of the proposed approach is using of the word distance measurement methods such as Levenshtein and Jaro-Winkler for vectorization technics. CloudScan uses Logistic Regression and LSTM models while the Random Forest model as the main training model is used in this work. Contrary to these differences, N-gram creation and feature calculation methods are similar to those in the CloudScan aproach.

```
def calc_levenshtein(i,j):
    rows = len(i) + 1
    cols = len(j) + 1
    dist = create_zeros_matrix(i,j)
    for col in range(1,cols):
        for row in range(1,rows):
            if i[row-1] == j[col-1]:
                cost = 0
            else:
                cost = 1
            dist[row][col] = min(dist[row-1][col]+1,
                                 dist[row][col-1]+1,
                                 dist[row-1][col-1] + cost)
    ratio = ((len(i)+len(j) – dist[row][col])/ (len(i) + len(j))
    return ratio
pot_keys = list of potential keys
control_keys=list of control keys
result_list = []
for pot_key in pot_keys:
    ratio_list = []
        for control_key in control_keys:
            ratio = calc_levenshtein(pot_key, control_key)
            ratio_list.append(ratio)
        result_list.append(max(ratio_list))
```

Figure 3. Pseudocode of the algorithm using the Levenshtein Distance

In the proposed approach, firstly Levenshtein distances are calculated between potential keywords and members in the control keywords list. In order the method to work properly, the keyword checklist, which are compared with the potential keywords, should be carefully selected. When the number of elements in this list is more than one, the distance of the keyword closest to the potential keyword is taken and processed in the system. The pseudocode of the algorithm is used is given in Fig. 3.

Since the invoices used in this work are in German, it was sufficient to use 1-gram to identify potential keywords. It would be better to use 2-gram in case of working on a Turkish or English invoice. Because potential keywords consist of 2 words in these languages such as “Fatura Numarası”, “Fatura Tarihi” or “Invoice Number”, “Invoice Date”. However, these structures can be expressed in a single word in German, such as "Rechnungsnummer", "Rechnungsdatum", “Belegdatum". After the 1-gram structure of the words on the invoice is created, Levenshtein distances are calculated by comparing them with each element in the keyword checklist. Additional attributes used in the proposed approach reflecting Levenshtein distances of the n-gram to the potential keywords used for training are given in Table 4.

Table 4. Additional Levenshtein Distance Attributes Used in Training

Attribute	**Description**
InvoiceDateLevDistance	Levenshtein Distance value used for Invoice Date
DeliveryDateLevDistance	Levenshtein Distance value used for Delivery Date
DueDateLevDistance	Levenshtein Distance value used for Due Date
InvoiceNoLevDistance	Levenshtein Distance value used for Invoice No
TotalGrossLevDistance	Levenshtein Distance value used for Total Gross
TotalNetLevDistance	Levenshtein Distance value used for Total Net
VatAmountLevDistance	Levenshtein Distance value used for Vat Amount
IbanLevDistance	Levenshtein Distance value used for IBAN

4. Computational Results

4.1. Dataset

A dataset containing of 9910 invoice images collected by authors from different firms is divided into 80% training and 20% test sets. Since the invoices contain sensitive financial information, the dataset cannot be presented as an open source. A total of 34 features were used given in Tables 2 and 4. There are 26 features in Table 2 and 8 features in Table 4. An example of invoice image that is used in training is shown in Fig. 4. The invoice can be seen in Table 5, where all processes are completed and ready for training. The labeling process was done by two experts. While one expert performed the labeling, the other one checked the labelled fields.

The machine learning models are used such as Random Forest, Gradient Boosting Machine, Extreme Gradient Boosting, K-Nearest Neighbors, AdaBoost and Decision Tree to predict the labels for all fields. There is an imbalance between labeled and unlabeled data used in training. Although there is single field (for example, invoice number) on a single invoice, there is an average of 1500 to 2000 unclassified data (n-grams). This situation causes unbalanced data problem (Chawla et al., 2002). If the unbalanced data problem is not solved, the model may qualify very few classes of observations as completely outlier. For this reason, whole data in the minor class (labeled) is taken, while only a certain part of the data in the major class (unlabeled) is taken. As a result of the training, the best ratio of the number of unlabeled data to the number of labeled data was determined as 15. Its effect on training may be different each time since the unlabeled data will be chosen randomly. Apart from these, the overfitting problem has been prevented by limiting the number of invoices of the same type. The training has been repeated 500 times because of this randomness.

4.2. Results and Discussion

Computational experiments were performed on the computer with 2.3 Ghz 8-core 9th-generation Intel Core i9 processor, AMD Radeon Pro 5500M with 4GB of GDDR6 memory and 16GB of 2666 MHz DDR4 memory. The training process took 164 minutes for all invoices. Python was used as the programming language, numpy and pandas were used as a library in this work. The scores for labeled data such as "invoice date", "invoice number", "due date", "delivery date", "total gross", "total net", "vat amount" and "IBAN" are given in tables 6-13. The models generally using the F1-scores because of the unbalanced data are compared. The bold numbers represent the best F1-scores in the tables. In the Random Forest, which is concluded as the best prediction model, we achieve F1 scores as 0.97 for the “invoice number”, 0.97 for the “invoice date”, 0.88 for the “due date”, 0.76 for the “delivery date”, 0.93 for “total gross”, 0.89 for “total net”, 0.92 for “vat amount” and 0.99 for “IBAN”. When the values are analyzed, the Extreme Gradient Boosting model achieved the second closest results to the Random Forest. The other models did not show any obvious superiority over each other.

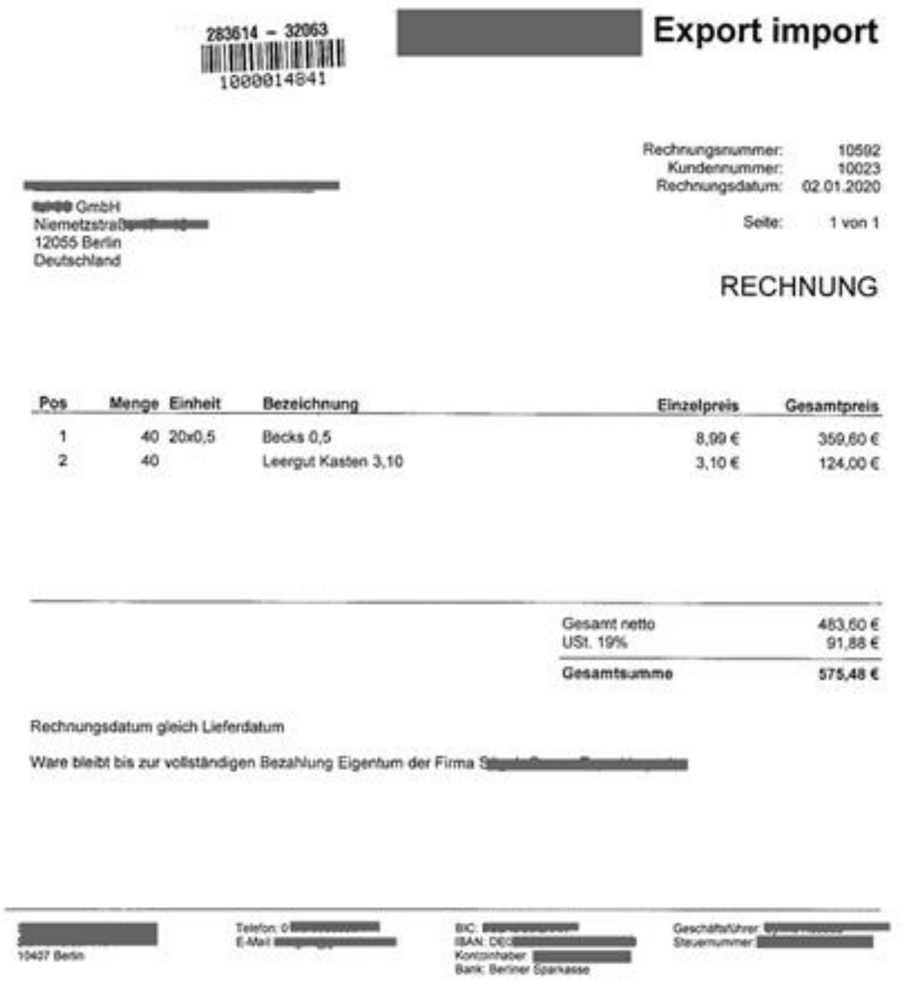
The training results of all labeled fields to be predicted were obtained separately. The results obtained for each model and each distance measurement technique can be seen in Tables 6-13. First of all, there are two main reasons for special training for each field. First reason is ensuring maximum score by choosing different attributes that best represent each tag. The second is to be able to choose different models that provide the best success for the field. Note that, it may not always be possible to achieve the best score in all fields with a single model (Nasiboğlu and Akdoğan, 2020).

According to the Tables 6-13, it can be seen that the best model is Random Forest for all fields. However, as can be seen in Table 6, Table 7 and Table 13, the Extreme Gradient Boosting model was able to produce results close to Random Forest. In the future, it can be thought that these two models will be successful in the estimation of other fields such as address, tax number. At the same time, the values of two different distance measurement methods can be evaluated in the tables. Although Jaro-Winkler and Levenshtein methods obtain very close scores, we can say that Levenshtein method gives a better result.

If we evaluate the results on the basis of fields, high scores were obtained on invoice number, invoice date and IBAN. The reason for this success is that these tagged data, coordinates and potential words around them are similar in many invoices. The reason for not achieving such high success in the amounts section is because the labeled data can be located in very variable regions on the page. For example, total amount may be at the top on the second page in a multi-page invoice, while it is in the middle on the page of single-page invoice. It can also be located on both the right and left side on the page. Despite these variations, a satisfactory success has been achieved for the amount fields. Due to the insufficient amount of labeled data in the delivery date and the due date fields, the desired success can not be achieved for this fields.

Table 5. Part of the Sample Invoice Ready for Traning

RawText	Pattern	IsFirstStr	IsFirstInt	…	InvoiceDateLevDistance	TotalNetLevDistance
Gesamt	Xxxxxx	1	0	…	0	0
Gesamt netto	Xxxxxx xxxxx	1	0	…	0	0
Gesamt netto 483,60	Xxxxxx xxxxx 000?00	1	0	…	0	0
Gesamt netto 483,60 €	Xxxxxx xxxxx 000?00 ?	1	0	…	0	0
netto	xxxxx	1	0	…	0.35	0.65
netto 483,60	xxxxx 000?00	1	0	…	0.35	0.65
netto 483,60 €	xxxxx 000?00 ?	1	0	…	0.35	0.65
483,60	000?00	0	1	…	0.25	1



283614 – 32063
1000014841

[redacted] **Export import**

Rechnungsnummer:	10592
Kundennummer:	10023
Rechnungsdatum:	02.01.2020
Seite:	1 von 1

[redacted]
[redacted] GmbH
Niemetzstraß[redacted]
12055 Berlin
Deutschland

RECHNUNG

Pos	Menge	Einheit	Bezeichnung	Einzelpreis	Gesamtpreis
1	40	20x0,5	Becks 0,5	8,99 €	359,60 €
2	40		Leergut Kasten 3,10	3,10 €	124,00 €

Gesamt netto	483,60 €
USt. 19%	91,88 €
Gesamtsumme	**575,48 €**

Rechnungsdatum gleich Lieferdatum

Ware bleibt bis zur vollständigen Bezahlung Eigentum der Firma S[redacted]

[redacted]
10407 Berlin

Telefon: 0[redacted]
E-Mail: [redacted]

BIC: [redacted]
IBAN: DE[redacted]
Kontoinhaber: [redacted]
Bank: Berliner Sparkasse

Geschäftsführer: [redacted]
Steuernummer: [redacted]

...

Figure 4. A Sample Invoice image from the Dataset

Table 6. Precision, Recall and F1 Scores for "invoice number"

Model	Precision Lev.	Recall Lev.	F1 Lev.	Precision J-Win	Recall J-Win.	F1 J-Win
Random Forest	0.99	0.96	**0.97**	0.99	0.96	0.97
Gradient Boosting Machine	0.96	0.91	0.93	0.96	0.90	0.93
Extreme Gradient Boosting	0.98	0.97	0.97	0.98	0.97	0.97
K-Nearest Neighbors	0.93	0.95	0.94	0.90	0.94	0.92
AdaBoost	0.94	0.89	0.92	0.94	0.88	0.91
Decision Tree	0.95	0.96	0.95	0.95	0.96	0.95

Table 7. Precision, Recall and F1 Scores for "invoice date"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.97	0.97	**0.97**	0.97	0.97	0.97
Gradient Boosting Machine	0.93	0.95	0.94	0.93	0.94	0.94
Extreme Gradient Boosting	0.96	0.97	0.96	0.96	0.97	0.96
K-Nearest Neighbors	0.91	0.93	0.92	0.92	0.93	0.93
AdaBoost	0.90	0.94	0.92	0.90	0.93	0.92
Decision Tree	0.97	0.96	0.96	0.97	0.96	0.96

Table 8. Precision, Recall and F1 Scores for "due date"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.97	0.81	**0.88**	0.95	0.81	0.87
Gradient Boosting Machine	0.90	0.64	0.75	0.87	0.65	0.74
Extreme Gradient Boosting	0.91	0.80	0.85	0.91	0.81	0.86
K-Nearest Neighbors	0.66	0.61	0.63	0.64	0.56	0.60
AdaBoost	0.85	0.53	0.65	0.69	0.56	0.61
Decision Tree	0.78	0.80	0.79	0.74	0.78	0.76

Table 9. Precision, Recall and F1 Scores for "delivery date"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.93	0.64	**0.76**	0.89	0.63	0.74
Gradient Boosting Machine	0.83	0.44	0.58	0.76	0.52	0.62
Extreme Gradient Boosting	0.84	0.61	0.71	0.81	0.57	0.67
K-Nearest Neighbors	0.65	0.50	0.56	0.70	0.57	0.63
AdaBoost	0.65	0.34	0.45	0.83	0.34	0.48
Decision Tree	0.63	0.64	0.64	0.64	0.62	0.63

Table 10. Precision, Recall and F1 Scores for "total gross"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.97	0.88	**0.93**	0.98	0.88	0.93
Gradient Boosting Machine	0.93	0.69	0.79	0.93	0.68	0.79
Extreme Gradient Boosting	0.96	0.88	0.92	0.96	0.88	0.92
K-Nearest Neighbors	0.87	0.78	0.82	0.84	0.77	0.80
AdaBoost	0.86	0.67	0.76	0.85	0.64	0.73
Decision Tree	0.86	0.88	0.87	0.85	0.88	0.86

Table 11. Precision, Recall and F1 Scores for "total net"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.96	0.83	**0.89**	0.96	0.83	0.89
Gradient Boosting Machine	0.90	0.55	0.68	0.89	0.55	0.68
Extreme Gradient Boosting	0.93	0.81	0.86	0.93	0.81	0.86
K-Nearest Neighbors	0.79	0.70	0.74	0.76	0.69	0.73
AdaBoost	0.78	0.52	0.62	0.76	0.52	0.62
Decision Tree	0.80	0.83	0.82	0.81	0.82	0.82

Table 12. Precision, Recall and F1 Scores for "vat amount"

Model	**Precision Lev.**	**Recall Lev.**	**F1 Lev.**	**Precision J-Win**	**Recall J-Win.**	**F1 J-Win**
Random Forest	0.96	0.88	**0.92**	0.96	0.87	0.92
Gradient Boosting Machine	0.91	0.63	0.75	0.91	0.62	0.74
Extreme Gradient Boosting	0.95	0.88	0.92	0.94	0.87	0.91
K-Nearest Neighbors	0.85	0.80	0.82	0.85	0.78	0.81
AdaBoost	0.80	0.59	0.68	0.82	0.57	0.67
Decision Tree	0.85	0.85	0.85	0.82	0.85	0.83

Table 13. Precision, Recall and F1 Scores for "IBAN"

Model	Precision Lev.	Recall Lev.	F1 Lev.	Precision J-Win	Recall J-Win.	F1 J-Win
Random Forest	0.99	0.99	**0.99**	0.99	0.99	0.99
Gradient Boosting Machine	0.98	0.97	0.98	0.98	0.97	0.98
Extreme Gradient Boosting	0.99	0.99	0.99	0.99	0.99	0.99
K-Nearest Neighbors	0.97	0.99	0.98	0.98	0.99	0.99
AdaBoost	0.97	0.97	0.97	0.97	0.97	0.97
Decision Tree	0.99	0.99	0.99	0.99	0.99	0.99

5. Conclusion

In this work, the machine learning models based on n-gram structure were discussed to recognize information on the invoice images. The conversion of the words on the invoice into n-grams and the using of Levenshtein and Jaro-Winkler distances between words were effective in the training of the models. Although there is not a big difference between Jaro-Winkler and Levenhstein distances' calculation methods, the Levenshtein distance is slightly better, as seen in the results. The effective attributes' list was discovered to increase the performance values of the models. The data set, collected by authors from different firms, containing 9910 invoice images were divided into 80% training and 20% test sets. The labeling process performed by two expert. While one expert performed labeling for the fields, the other one checked the labeled fields. The F1 scores were used to compare the models because of the unbalanced data.

When the performances of the models were compared, it was observed that the Random Forest model was superior to other models with a value of 0.9137 average F1 score. In addition, the Random Forest model was found to be robust to the overfitting problem. The average scores of the other models were determined as Gradient Boosting Machine with 0.80, Exreme Gradient Boosting with 0.8975, K-Nearest Neighbour with 0.8012, AdaBoost with 0.7462 and Decision Tree with 0.8587 F1-scores.

In the future studies, it is aimed to construction of the models and the additional attributes for effective extraction of invoice information. Also, it is planned to construct effective approaches to extract other information such as address, postal code and customer number on the invoice images.

References

Aydın, C. (2018). Makine Öğrenmesi Algoritmaları Kullanılarak İtfaiye İstasyonu İhtiyacının Sınıflandırılması. *Avrupa Bilim ve Teknoloji Dergisi*, 14(4), 169–175.

Breiman, L. (2001). Random Forests. *Machine learning*, 45(1), 5–32.

Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

Esser, D., Schuster, D., Muthmann, K., Berger, M., & Schill, A. (2012). Automatic indexing of scanned documents: a layout-based approach. *Document Recognition and Retrieval*, XIX, 8297, 82970H.

Gelfand, S.B., Ravishankar, C.S., & Delp, E.J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(2), 163-174.

Haldar, R., & Mukhopadhyay, D. (2011). *Levenshtein distance technique in dictionary lookup methods: an improved approach.* https://arxiv.org/abs/1101.1232. Accessed 23.05.2020.

Jaro, A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa. *Journal of the America Statistical Association*, 84(406), 414-420.

Katti, A., Reisswig. C., Guder, C., Brarda, S., Bickel, S., Hohne, J., & Faddoul, J. (2018). Chargrid: Towards understanding 2d documents. *Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, pp. 4459-4469.

Klein, B., Agne, S., & Dengel, A. (2004). Results of a Study on Invoice-Reading Systems in Germany. In: Marinai S, Dengel AR (eds.), *Document Analysis Systems VI, ser. Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 451–462.

Liu, X., Gao, F., Zhang, Q., & Zhao, H. (2019). Graph convolution for multimodal information extraction from visually rich documents. *Proceedings of the 2019 Conference of the North. Minnesota. NAACL.*

Mashat, A., Fouad, M., Yu, P., & Gharib, T. (2012). A decision tree classification model for university admission system. *International Journal of Advanced Computer Science and Applications,* 3(10), 17–21.

Nasiboğlu, R., & Akdoğan, A. (2020). Estimation of the secondhand car prices from data extracted via web scraping techniques. *Journal of Modern Technology and Engineering*, 5(2), 157-166.

Palm, R., Winther, O., & Laws, F. (2017). CloudScan - A configuration-free invoice analysis system using recurrent neural networks. *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto, Japan, pp. 406 – 413.

Quinlan, J. (1986). Induction of decision trees. *Machine learning*, 1(1), 81–106.

Schulz, K., & Mihov, S. (2002). Fast string correction with Levenshtein-automata. *International Journal of Document Analysis and Recognition*, 5(1), 67–85.

Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C., Aliyev, K., & Hofmeier, A. (2013). Intellix – end-user trained information extraction for document archiving. *12th International Conference on Document Analysis and Recognition*, pp 101–105.

Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, pp. 629–633.

Wang, Y., Qin, J., & Wang, W. (2017). Efficient Approximate Entity Matching Using Jaro-Winkler Distance. *18th International Conference on Web Information Systems Engineering*, Puschino, Russia, pp 231-239.

Watanabe, T., Tsukada. H., & Isozaki, H. (2009). A succinct n-gram language model. *International Joint Conference on Natural Language Processing (IJCNLP)*, Singapore, pp. 341–344.

Xiaoliang, Z., Jian, W., Hongcan, Y., & Shangzhuo, W. (2009). Research and Application of the improved Algorithm C4.5 on Decision Tree. *International Conference on Test and Measurement (ICTM)*, Hong Kong, 2, 184-187.

Yıldız, İ., & Karadeniz, A. (2019). Enhancement of Breast Cancer Diagnosis Accuracy with Deep Learning. *European Journal of Science and Technology*, (Special Issue), 452-462.

Zelic, F., & Sable, A. (2020). A comprehensive guide to OCR with Tesseract, OpenCV and Python. https://nanonets.com/blog/ocr-with-tesseract/. Accessed 02.06.2020.