WEB DATA SCRAPING TECHNIQUE AND PREPARATION FOR COMPARISON TECHNIQUES BETWEEN DIFFERENT DOCUMENTS

Ylber Januzaj, Artan Luma, Azir Aliu, Besnik Selimi, Bujar Raufi

Faculty of Contemporary Sciences and Technologies
South East European University
e-mails: {yj16535, a.luma, azir.aliu, b.selimi, b.raufi}@seeu.edu.mk
North Macedonia

Abstract: Studying the right field has great importance in human life and perspective. Therefore, today's demands have increased significantly for skilled people, and prepared in complex areas, and a consolidation between market demands and university curricula is needed. This paper examines Data mining techniques which are used in order to create an automated model which makes comparisons between market demands and university curricula. First we present how to identify the elements that are used to create the webpage. Then we present the form of extracting information from these websites, and storing them in our database. Saving of data will be made in the specific format that will be assigned to our model. Finally, we present the comparisons techniques that are used in order to compare two different datasets.

Key words: Technology, Data mining, Job market, University curricula, Web scraping.

1. INTRODUCTION

In order to achieve the implementation of our model we identified the tools that will be used. Initially, we identified sources that provide information on job offers in the field of technology. The initial idea was the information that would be used to deal with job requirements of our country and the region, but based on the small number of information provided by the websites in Kosovo and Macedonia, we provided these information from European websites as the information capacity has been higher.

Based on the information that were scraped from various websites, we achieved to scrap only information about title of job offer, which does not result in a proper analysis. In order to have a more accurate analysis, we need to use websites that provide a large number of job offers in the field of technology [1, 2, 3]. The information that we need is also the job descriptions that are provided by the websites. Once the source information is identified, we start applying web scraping

techniques that will help us to extract the information automatically. Since the store of data will be completely unstructured, once the information is extracted, we use text cleaning techniques, which will be used to avoid spaces and special characters that are unusable during our analysis. Once the text cleaning is over, our corpus is considered ready for the application of machine learning techniques.

The same procedure will be applied to the websites of universities that have published their syllabuses, where web scraping techniques we will be able to extract the information that is included in syllabus of the technology programs. After both the information corps are available, the same techniques will be applied to the syllabus section in order to be able to compare the text of job offers and programs offered by universities of the region in Kosovo, Macedonia and Albania. We used Python to apply these techniques, as a powerful language and appropriate for machine learning. Next, we talk about sources information that have been used in order to implement our automated model.

2. SOURCE OF INFORMATION WITH JOB OFFERS

As it is mentioned in previous, the target of job offers were the websites of our country and region. But, based on the small capacity of the information we scraped from our websites in our country, we used European websites as they provide more information [4]. In the following we present the situation when we extract the title of job offers, and we compare later with the situation when we extract the description of job offers. In Figure 1, we express this type of website is inappropriate in our case, as it does not provide useful information.

Fig. 1. Unusable information

We can extract information about the job offer title, as far as the offer description is concerned, and there are no such information in this case. Also, there are situations when websites do not allow us to apply web scraping techniques and we cannot get the information from that web site. Next in Figure 2, we present a case where no web-scraping techniques are not allowed to be applied to a web site.

```
'downloader/response_bytes': 18184,
'downloader/response_count': 2,
'downloader/response_status_count/200': 2,
'finish_reason': 'finished',
'finish_time': datetime.datetime(2019, 3, 22, 13, 47, 29, 93968),
'log_count/DEBUG': 3,
'log_count/INFO': 7,
'memusage/max': 49500160,
'memusage/startup': 49500160,
'response_received_count': 2,
'scheduler/dequeued': 1,
'scheduler/dequeued/memory': 1,
'scheduler/enqueued': 1,
'scheduler/enqueued/memory': 1,
'start_time': datetime.datetime(2019, 3, 22, 13, 47, 28, 738470)}
2019-03-22 14:47:29 [scrapy.core.engine] INFO: Spider closed (finished)
```

Fig. 2. Unsuccessful scraping process

As we can see in a specific web page it is impossible to apply web scraping techniques due to the form of web site design. Based on the facts when some of the websites provide us with some information, and some of them with enough information do not allow us to apply web scraping techniques, we have been forced to look into the European website as well.

What is relevant to identifying information that will be extracted through web scraping techniques is that the code content that was used to construct the web site should be analyzed [8,9,12]. To be more accurately explained, it is necessary to analyze whether the same template was used for all the links that are on this website, since the system will automatically visit all the links in order to extract the necessary information. At the beginning, the algorithm that is designed to extract the information is based on the tags defined in the web content. So if we have differences in the template model that is applied to the links within the webpage, then we will not be able to extract all the information from the webpage automatically. Next, we present the situation when a website has enough and useful information. In Figure 3 it is shown that the specific website possesses enough information to implement our model.

Fig. 3. Useful information from website

It is precisely the information of title, company, and description of job offer that will be used in our situation. It is also important to note that this web site also has its own customizable template, since the same template is used for all links within it [5, 7]. This construction of this web site enables and facilitates the automated extraction of information.

2.1. Scraping process

Next, we present a sketch of how web scraping techniques will be applied to specific web sites. First the initial link will be identified, then the identification of the tags on which the website was created will be also identified. Once the tags are identified on the primary page, they also should be identified on the secondary page by identifying the title, company, and the description of job offer. In addition to each tag, you should also identify the links that contain the number of each website, since the website has a considerable amount of information. We express in Figure 4 the process of extracting information from the website goes through several steps.

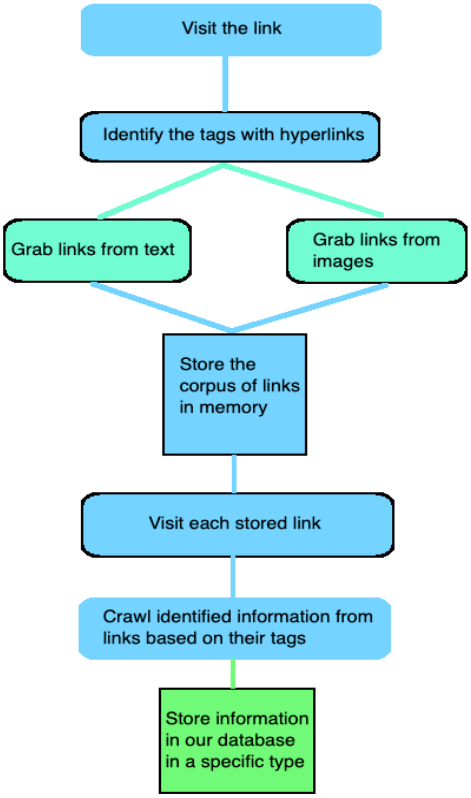

Fig. 4. Scraping process

First, we use the primary link that will be visited by the system in an automatic way. Once the link has been successfully launched, it is the identification of the text and the photos that contain the hyperlinks to which we are redirected within the site where we can get information on job offers [6, 11]. Once the links are extracted from the text and the images, they will be stored in memory so they will be visited later. Once the links are stored in memory, they will be visited, and specific information will be extracted from the content of that link. The content that is extracted from the website will be stored in the database as a specific information that will be used later from automated model. The ability to save information is diverse, but the Json format is a format that is more adapted to our latest approach to applying machine learning techniques. In the following, we present examples from identifying web pages, to continue later with algorithm that was used to extract information automatically. In Figure 5 we show the identified tags that provides hyperlinks.

Fig. 5. Identified tags on website

This link is stored in the inside of the text and inside the image and we need to retrieve, store it in memory so that we can later visit this link. Once the link is saved and clicked, the system will be redirected to the website, which includes a lot of information. Below we present the case for identifying other tags. These tags provide information about title, company and description of job offer. In Figure 6 we see how the identification of the elements used to define the title, publisher, and job descriptions is used.

Based on the web design template, within the "job-header" tag the element that was used to define the job offer title is tag "h1", while the tag used to define the company of job offer is tag "h2 ". While for the description of job offer is used tag "h3" as well as other elements such as "p" and "ul". Once we identified each of these tags, we should also identify the last element of the links within which other websites are located. This part is known as "pagination" and allows the system to access each page automatically, and at the moment it reaches the last page, the system interrupt the visiting link process, ensuring that we do not visit the same site more than one .

Fig. 6. Identified elements on website

Once all the elements are identified, we are ready to begin writing the algorithm in the Python language in order to execute and extract the information.

2.2. Algorithm for scraping process

As we mentioned earlier, the language that was used to execute the scraping process is Python. Next in Figure 7 we present the algorithm divided into several parts in order to give explanations for each part and finally to reach the test phase.

Fig. 7. Import library

As we can see, the initial part is the part of importing a library that is installed through the "pip" package, which installs and manages packages and applications written through the "Python" language. Next in Figure 8, we present the second part of the algorithm.

Fig. 8. Class definition

In this section we can see the part of the class definition which in our case is referred to as "*eurotechSpider*", continuing within the code with the name of the project that will be executed via scrapy. Also in this part of the code is the primary link which is used to visit the website in order to continue with the extraction of information. In Figure 9 it is shown that the definition of "parse" is going to be the output of the links that are defined.

Fig. 9. Parse definition

First the "searchList" element looks for all "a" elements that result in links. Once we have identified all the links that are inside the webpage, they will be stored in memory and will be visited each link that you can extract from the text or from the image. In Figure 10 we have the definition of "vacancy" which will be checked within the "job-header" element, where it is the content of the description of the job offer.

```
15
16          vacancy = response.css('.job-header')
17          if vacancy is not None and len(vacancy) != 0:
18              description = '\n'.join(vacancy.css('li::text').extract())
19
```

Fig. 10. Information that will be extracted

If the aforementioned element is not "none", then all parts of the website that contain the element "li" will be taken and the text will be included. In the present case, the text and the unlisted lists are used to describe the requirements of the published job offer.

The information we could get from the website was different, but in our case we needed only information about job offer. Other information has not been used in a way that does not adversely affect the system's performance during the execution of the algorithm. We express in Figure 11, at the bottom of the code we order the system to display the data that we extract.

```
20          yield {
21              'title': vacancy.css('h1::text').extract_first(),
22              'Company / Place': vacancy.css('h2::text').extract(),
23              'description': description,
24          }
25
26
```

Fig. 11. Other types of information that will be extracted

In this case we identified that the job offer title was defined with the element "*h1*" and also other text with this element will be extract. After extracting information from element "*h1*", we have extracting information from element "*h2*" which in this case they are information about company and its place. And the last part that is considered as the most important part is the description of job offer, and now it will be invoked to be printed. All of the information that will be extracted will be stored in "*json line*" format as a format that is appropriate for applying machine learning techniques. Below we present the algorithm in order to execute and extract the appropriate information. In Figure 12 we present the algorithm that will be used to extract information from the specific web site.

```python
import scrapy


class eurotechSpider(scrapy.Spider):
    name = 'name of project'
    start_urls = [
        'primary link of the website'
    ]

    def parse(self, response):
        for link in response.css('.searchList a'):
            print(link.extract())
            url = link.css('a::attr(href)').extract_first()
            yield response.follow(url, self.parse)

        vacancy = response.css('.job-header')
        if vacancy is not None and len(vacancy) != 0:
            description = '\n'.join(vacancy.css('li::text').extract())

            yield {
                'title': vacancy.css('h1::text').extract_first(),
                'Company / Place': vacancy.css('h2::text').extract(),
                'description': description,
            }
```

Fig. 12. Algorithm for scraping process

Since the algorithm is ready, it can be executed via the command "*scrapy crawl name of the project*". The change that needs to be made is the difference regarding to the format of data that will be stored. In Figure 13, we present the changes that are made in order store records in "*jsonlines*" format.

```
BOT_NAME = 'crawl'

SPIDER_MODULES = ['crawl.spiders']
NEWSPIDER_MODULE = 'crawl.spiders'

FEED_FORMAT = 'jsonlines'
FEED_URI = 'result.json'
FEED_ENCODING = 'utf-8'
```

Fig. 13. The format of extracted information

Also part of this file is the file name that will be stored after executing the code in the Python language, as well as encoding as "utf-8" as an encoding commonly used in this format. Once all these changes have been completed, the algorithm will be executed and after the execution of the algorithm, the files will be kept in an unorganized form. The successful extraction message is shown in Figure 14, where we can say that after the execution of the code, we extracted at least 2000 job offers.

```
'finish_reason': 'finished',
'finish_time': datetime.datetime(2019, 3, 23, 14, 29, 0, 279737),
'item_scraped_count':2264,
'log_count/DEBUG': 532,
'log_count/ERROR': 3,
'log_count/INFO': 8,
'memusage/max': 49393664,
'memusage/startup': 49393664,
'request_depth_max': 1,
'response_received_count':2266,
'scheduler/dequeued':2267,
'scheduler/dequeued/memory':2267,
'scheduler/enqueued':2267,
'scheduler/enqueued/memory':2267,
'start_time': datetime.datetime(2019, 3, 23, 14, 28, 42, 496925)}
2019-03-23 15:29:00 [scrapy.core.engine] INFO: Spider closed (finished)
```

Fig. 14. Successful scraping process

These data have been extracted automatically and stored in our system. After this process, our data are ready to be processed, by removing space and special characters in order to get the most out of the machine learning techniques.

2.3. Text processing

Since the information have been downloaded from specific websites, the same procedure will be applied to the websites of universities that have published their syllabus programs in the field of technology. Knowing the nature of how web scraping works, there is a need to clean up the text that is being collected [10,15]. The reason why it should be processed is that the files are collected from web pages, and there are also downloaded from other data that are known as special characters [13,14]. Next in Figure 15, we show that our web pages still do not appear to be applied to text parsing.

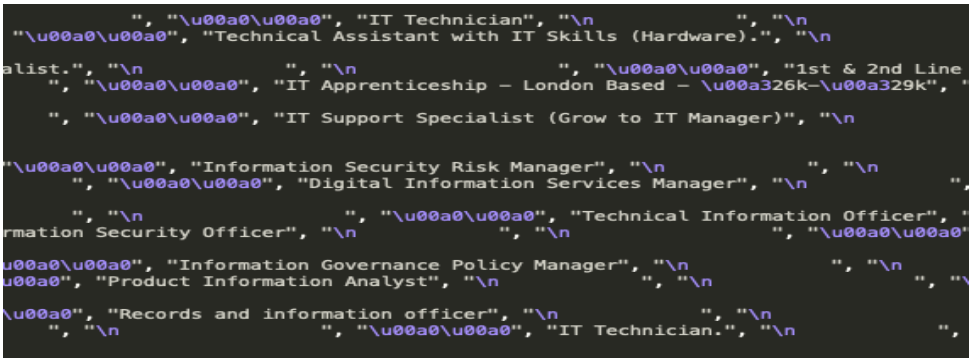

Fig. 15. Unprocessed text

After applying the "*strip ()*" and "*replace ()*" functions, the system removes spaces between words and rows, also special characters are removed based on these functions. In the following we present the process of avoiding spaces and special characters from our document. After applying the text cleansing techniques as shown in Figure 16, the text is considered ready for future techniques.

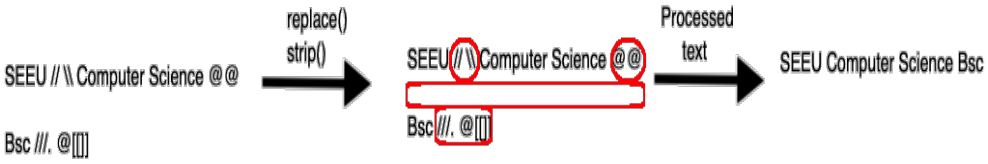

Fig. 16. Text processing sketch

3.TF-IDF TECHNIQUES

The technique used to turn the word into a vector and to count each of them as soon as it is written in the report of all the words in the corpus is known as TF-IDF. This technique is a highly used as a powerful technique that ensures that all of our

corpus texts will be returned to a format that can be used for machine learning techniques.

There are several techniques that enable us to convert vector vectors, such as *word2vec*, *term-frequency*, etc., and all of them assure us that the text that converts to the vector is ready to apply the machine learning techniques .

As mentioned earlier, in terms of the main purpose of vector conversion, the TF-IDF has meant that each word is used as a whole, dividing it into the total number of words, and in this form, gives us a statistic of each word. Next, we present the formula for calculating TF-IDF according to mathematical calculations.

$$(\mathrm{TF}) = \frac{\text{count word in document}}{\text{total number of words in document}} \quad (1)$$

As it is shown in 1, Term Frequency (TF) represents the number of each word divided by the total number of words that the corpus contains. Hence, the value of which we ultimately earn the value of each word, the greater the value that is earned, the greater the volume of that word used in that document will be the greater. However, we may already have cases where a word that is less relevant appears more often, and it cannot be considered that the word is more relevant. The concrete case is with the words: *"the"*, *"this"*, *"a"*, *"an"*, *"on"*, *"in"*, known as stop words, and the ones used in more than one document cannot conclude what they are more relevant than other words. Therefore, in order to regulate such issues, it is required to be normalized, and this normalization is done by dividing the number of those words with the total number of words in the document. In this way, the value of the words presented in our corpus will be reduced.

The function that is needed to divide the number of words with the total of words that is inherent to the corpus also applies to the logarithmic function which calculates the logarithm of the values obtained from the total calculation. The formula that is used for normalization of our records is as follows.

$$\mathrm{IDF} = 1 + \mathrm{LOG}\frac{\text{count word in document}}{\text{total number of words in document}} \quad (2)$$

As we can see in 2, normalization is achieved by calculating the logarithm of the gained values and adding value 1 to avoid specific cases that do not reach negative values due to the number of words we have in the corpus. But in this case, the value of the above mentioned stop words has been normalized.

After computing with this formula we get values as vectors that represent the input of each word based on the score of the TF-IDF that is finally reached. The bigger the value of TF-IDF the greater is the weight of the words and the smaller the value of the TF-IDF the smaller is the weight of that word.

In both of the above formulas we computed TF (Term Frequency) and IDF (Inverse Document Frequency) calculations, and to find the final TF-IDF estimation at any time we calculate the product between TF and IDF.

$$\mathrm{TF} - \mathrm{IDF} = \mathrm{TF} * \mathrm{IDF} \quad (3)$$

In 3 we present an illustration of a simple example that implements the operation of the other calculations that assure us that we obtain accurate TF-IDF values of a document. We use a corpus with a small number of words in order to see which words will be given greater weight by the system.

In order to test the algorithm of the TF-IDF we use a corpus of 5 words where most of them have to do with another. The words used for illustration are:

- *"This testing process is expensive"*
- *"Testing an algorithm gives huge support"*
- *"This software is very complicated"*
- *"The learning process is becoming more complex"*
- *"This software is very secured"*

Based on frequently used words, in this case, the weighted word would be "this", "the", "is", "a" because they are repeated more often. But based on algorithm offered by TF-IDF, these words should be smaller because, as mentioned above, these are known as stop words. Below we present the result that was derived after the calculation of TF-IDF on our list of keywords. In Figure 17, we see the results that were generated after the execution of the algorithm which made the calculation of the TF-IDF of the words that we used.

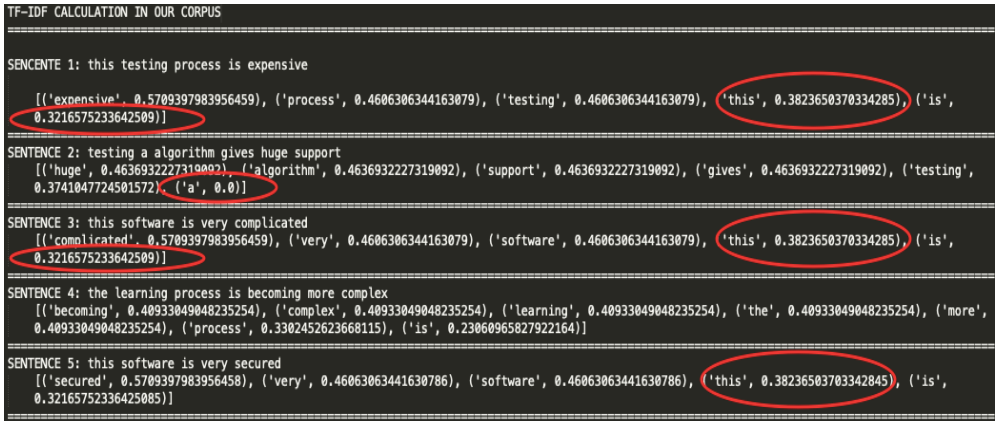

Fig. 17. Tf-idf calculation

Most widely used words are the *"this"*, *"the"*, *"a"*, but higher importance has been given to other words. In the first case, the words *"expensive"*, *"process" and "testing"* were more weighted than the words *"this"* and *"is"* which have a weight of 0.38 and 0.32.

Also in the second sentence we have the word "algorithm", "support" and "a", and as can be seen in Figure 17. As we can see in second sentence, the words

"algorithm" and "support" are more weighted than "a" that has a weight of 0. So in all cases, stop words are given less weight than the other words. This will give you even better results when you have a higher number of words.

3.1. Jaccard similarity

Information which are converted into numeric data and in the vector are ready to be applied to the comparison procedures since these will be the information of job offers and study programs provided by universities in the field of technology.

Jaccard similarity is one of the techniques used for comparing two different datasets. Measuring the level of the similarity between two different datasets is calculated by comparing the two data sets. The level of similarity between the datasets by Jaccard similarity is 0 if there is no similarity, up to 100 if the similarities are identical. So the greater the Jaccard similarity coefficient, the greater the similarity between the two datasets. Next we present the formula that computed the Jaccard similarity calculation to continue with its commentary.

$$\text{Jaccard}(\text{D1}, \text{D2}) = \frac{\text{D1} \cap \text{D2}}{\text{D1} \cup \text{D2}} \quad (4)$$

So, as can be seen in 4, calculating the Jaccard coefficient is calculated by dividing the intersection D1 and D2 and union of D1 and D2. If you want to sum up more than one of the above formulas Jaccard the coefficient we will have.

$$\frac{\text{D1} \cap \text{D2}}{|\text{D1}| + |\text{D2}| - \text{D1} \cap \text{D2}} \quad (5)$$

So we have 5 that represents the division between the intersection of D1 and D2, and the subtraction between D1 and D2 and the intersection between D1 and D2.

To illustrate Jaccard similarity we will use two vector resulting from two different documents.

- **Document 1: {1,3,5,6,7}.**
- **Document 2: {0,2,4,5,6,7,8,9}.**

So, as we can see, there are two documents with different strings that after applying Jaccard similarity it will show the similarity between them.

In 6 we have calculation of intersection between Document 1 and Document 2, which will be the characters that are in Document 1 and Document 2.

$$(Document\ 1\ \cap \text{Document 2}) = (\{1,3,5,6,7\} \cap \{0,2,4,5,6,7,8,9\}) \quad (6)$$

After intersection of Document 1 and Document 2, in 7 we get the following value:

$$(Document\ 1\ \cap \text{Document 2}) = (5,6,7) \quad (7)$$

Based on the calculation of the intersection between Document 1 and Document 2, there are 3 characters that are in the first document but are also in the second

document. Since we have the number of characters that are in the same two documents, in 8 we calculate the union of Document 1 and Document 2.

$$(Document\ 1\ \cup \text{Document } 2) = (\{1,3,5,6,7\} \cup \{0,2,4,5,6,7,8,9\}) \quad (8)$$

After union of Document 1 and Document 2, in 9 we gain this value:

$$(Document\ 1\ \cup \text{Document } 2) = (0,1,2,3,4,5,6,7,8,9) \quad (9)$$

Based on the union calculation between Document 1 and Document 2, there are 14 characters that are in both documents but not duplicated two times for the same character. Since we have both values, they are ready to apply to the formula which is shown in 10.

$$Jaccard(Document1, Document2) = \frac{3}{|5|+|8|-3} = 0{,}3 * 100 = 30\% \quad (10)$$

So from the calculation of Jaccard similarity between D1 and D2 with different words, it results that these documents have similarity at 30% level.

In some cases, depending on the calculation of the similarity between the two datasets, it is also necessary to calculate the distance between two datasets or documents which in their entirety include different words.

Below we present the case of illustration of Jaccard similarity in Python language after the transformation of the agitators. Figure 18 shows the algorithm that computed the Jaccard similarity calculation between two number ranges.

```
1  from math import*
2
3  def Jaccard(d1,d2):
4
5   intersection_of_numbers = len(set.intersection(*[set(d1), set(d2)]))
6   union_of_numbers = len(set.union(*[set(d1), set(d2)]))
7   return intersection_of_numbers/float(union_of_numbers)
8
9  print Jaccard([1,3,5,6,7],[0,2,4,5,6,7,8,9]) * 100
```

Fig. 18. Jaccard similarity algorithm

Initially, the variables d1 and d2 are declared as two different documents, then the *intersection_of_numbers* is declared as intersection between the two documents. Once stated *intersection_of_numbers*, *union_of_numbers* are declared among the above mentioned documents. After calculating the intersection and union, calculating the Jaccard similarity function is the division between the two values. The value gained is 30% which can be referred to as the similarity between documents that cover the range of different numbers.

3.2. Jaccard distance

In order to calculate Jaccard distance, we use the preliminary formula used to find Jaccard similarity. Jaccard distance as well-known as Jaccard dissimilarity is calculated by this formula:

$$Jaccard\ distance(D1, D2) = 1 - (\text{D1}, \text{D2}) \quad (11)$$

In 11 we express the calculation of Jaccard distance subtraction of 1 by Jaccard similarity. The formula will continue in this form:

$$\frac{|D1 \cup D2| - |D1 \cap D2|}{D1 \cup D2} \quad (12)$$

So as we can see in 12, Jaccard distance is the division between the D1 and D2 union and the intersection of D1 and D2 and the union of D1 and D2. Below we present the illustration of the same two previous documents.

$$Jaccard(Document1, Document2) = 1 - \frac{3}{|5|+|8|-3} = (1 - 0{,}3) * 100 = 70\% \quad (13)$$

Based on the calculation that is based on Jaccard distance, in 13 it can be seen that Jaccard similarity between D1 and D2 is 30%, while the distance between both documents is 70%.

Next in Figure 19, we present the Jaccard distance illustration even in Python language and the algorithm used to calculate it.

```
#Jaccard Distance calculation
from math import*

def Jaccard_distance(d1,d2):

 intersection_of_numbers = len(set.intersection(*[set(d1), set(d2)]))
 union_of_numbers = len(set.union(*[set(d1), set(d2)]))
 return intersection_of_numbers/float(union_of_numbers)

print (1 - (Jaccard_distance([1,3,5,6,7],[0,2,4,5,6,7,8,9]))) * 100
```

Fig. 19. Jaccard distance algorithm

As mentioned above, the formula that computed the distance between two documents with Jaccard distance is the same as Jaccard similarity, but is subtraction of 1 with the result obtained by Jaccard similarity.

4. CONCLUSION

Comparison between two different datasets is shown during our paper. This is an important analysis while we apply these techniques in order to compare datasets that hold information about market demand and university curricula. During our paper we expressed how data are extracted from specific websites. This was a necessary step, while the information for job offers and study programs were extracted from websites. After we extracted the information, we expressed the procedures of processing the data in order to apply machine learning techniques.

During our research we also illustrated how the words are weighted based on their frequency in the document. As it is shown, a normalization procedure is needed in order to give importance to the words that are not highly repeated, and to give a lower weight to the stop words that are highly repeated.

Finally we expressed how two different datasets will be compared applying TF-IDF and Jaccard similarity techniques, and we can conclude that these comparison

techniques will give us an accurate result about market demands and university curricula adjustment.

REFERENCES

[1] M. Agaoglu, "Predicting Instructor Performance Using Data Mining Techniques in Higher Education". In *IEEE Access*, May 2016.

[2] T. Xie, Q. Zheng, W. Zhang, H, Qu, "Modeling and Predicting the Active Video – Viewing Time in a Large – Scale E – Learning System". In *IEEE Access*, June 2017.

[3] A. M. Njeru, M. S. Omar, S. Yi, "IoTs for Capturing and Mastering Massive Data Online Learning Courses". In *IEEE Computer Society*, ICIS, Wuhan, China, May 2017.

[4] R. Heartfield, G. Loukas, D. Gan, "You are probably not the weakest link: Towards Practical Prediction of Susceptibility to Semantic Social Engineering Attacks". In *IEEE Access*, October 2016.

[5] E. J. Fortuny, D. Martens, "Active Learning – Based Pedagogical Rule Extraction". In *IEEE Transaction on Neural Network and Learning Systems*, Vol. 26, No. 11, November 2015.

[6] A. Mukhopadhyay, S. Bandyopadhyay, "A Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part I". In *IEEE Transaction on Evolutionary Computation*, Vol. 18, No. 1, February 2014.

[7] S. Malgaonkar, S. Soral, Sh. Sumeet, T. Parekhji, "Study on Big Data Analytics Research Domain". In *International Conference on Reliability, Infocom Technologies and Optimization ICRITO*,Noida, India, September 2016.

[8]K. P. Anicic, B. Divjak, K. Arbanas, "Preparint ICT Graduates for Real – World Challenges: Results of a Meta – Analysis". In *IEEE Transactions on Education*, Vol 60, No. 3, August 2017.

[9] A. Haskova, D. V. Merode, "Professional Training in Embedded Systems and its Promotion". In *IEEE Transacions on Education*, 2016.

[10] S.C. Smith, W. K. Al-Assadi, J. Di, "Integrating Asynchronous Digital Design into the Computer Engineering Curriculum". In *IEEE Transactions on Education*, Vol. 53, No. 3, August 2010.

[11] B. G. Member, V. S. Sheng, K. Y. Tay, W. Romano, Sh. Li, "Incremental Support Vector Learning for Ordinal Regression". In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 7, July 2015.

[12] J. Li, T. Zhang, W. Luo, J. Yang, X. T. Yuan, J. Zhang, "Sparseness Analysis in the Pretraining of Deep Neural Networks". In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 6, June 2017.

[13] Y. Qian, F. Li, J. Liang, B. Liu, Ch. Dang, "Space Structure and Clustering of Categorical Data". In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 27, No. 10, October 2016.

[14] Y. Xiao, B. Liu, Zh. Hao, "A Maximum Margin Approach for Semisupervised Ordinal Regression Clustering". In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 27, No. 5, May 2016.

[15] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, Sh. Li, "Incremental Support Vector Learning for Ordinal Regression". In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 7, July 2015.

Information about the authors:

Ylber Januzaj - has graduated the Faculty of Contemporary Sciences and Technologies in South East European University. He is PhD Candidate in South East European University, in e-Technologies program, specialization in Data Mining. Currently he lecturer in University of Mitrovica "Isa Boletini", in Kosovo.

Artan Luma – has graduated in Faculty of Contemporary Sciences and Technologies in South East European University in Tetovo. He holds a PhD diploma in Computer Sciences from 2010. He is Associate Professor in South East European University.

Azir Aliu - has graduated in Faculty of Mathematics and Natural Sciences- Department Informatics University "St. Cyril and Methodious", Skopje. He holds a PhD diploma in Informatics from 2011. He is Associate Professor in South East European University.

Besnik Selimi – has graduated in Joseph Fourier University (Grenoble 1), Grenoble, France. He holds a PhD diploma in Computer Sciences from 2009. He is Associate Professor in South East European University.

Bujar Raufi - has graduated in Faculty: French Faculty of Electrical Engineering Technical University of Sofia, Sofia, Bulgaria. He holds a PhD diploma in Computer Sciences from 2011. He is Associate Professor in South East European University.

Manuscript received on 14 April 2019