Example of Fuzzy-based Search Mechanism in Python

Miona Stojičević*, Predrag Vukašinović*, Vojkan Nikolic*, Branko Markoski**, Velibor Premcevski**

*University of Criminal Investigation and Police Studies, Cara Dušana 196,11080 Zemun, Republic of Serbia
** University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Republic of Serbia
miona.stojicevic@gmail.com, pvukasinovic5@gmail.com, vojkan.nikolic@kpu.edu.rs,
markoni@uns.ac.rs, velci.aspire@gmail.com

Abstract: The example that will be described in this paper is about to present a concrete use of fuzzy logic. It shows possibilities and certain advantages of fuzzy logic. The basic idea is to compare the fuzzy and the ordinary search, thereby pointing out the basic differences in both the inference and the data processing itself. In particular, several files of different extensions with 30 cities from the territory of the Republic of Serbia were given. These cities will first be loaded and then examined to see if there is a match to the string entered by user. By examining, we will see if it is necessary to enter the correct name, as well as how the fuzzy search works, in this case by using the Python fuzzywuzzy library.

Key words: fuzzy, regex, linguistics, search, cities

I. INTRODUCTION

Artificial intelligence has evolved in two basic directions, namely the study of natural intelligence and the attainment of intelligent behavior by applying different approaches, that is, by not applying natural systems. When it comes to the application of artificial intelligence, expert systems, robotics and neural networks are among the many fields. Perhaps not so well known in the field of artificial intelligence applications, and quite important and useful and that this work will further address, are the fuzzy systems. Fuzzy logic, occurs in the 1920s, as a theoretical, "possible" solution. From then until today, fuzzy has come a long way and reached the very center of artificial intelligence, and some things have become almost unthinkable without such solutions. "Fuzzy-based control systems are used in many consumer electronic devices to fully adapt to changes in the environment. Fuzzy logic concepts and techniques have also been used profitably in linguistics, behavioral sciences, disease diagnostics and even market analysis."[5]

II. FUZZY LOGIC

Lotfi Zadeh was the first to define fuzzy logic in 1965. and explained it as "an ambitious effort to close the gap between mathematics and the human intuitive way of thinking, speech and interaction with the world."[1] Fuzzy logic is a form of artificial intelligence, i.e. type of logical solution with multiple values, where they can be any real number from 0 to 1. In other words, fuzzy deals with partial accuracy. The value of accuracy is in the domain of total accuracy and complete inaccuracies. Fuzzy sets rests on mathematical principles, but understanding the stage of the meetings, they rely on the understanding of classical, discreet sets. An element's affiliation with a classic, discreet set is final and determined entirely, i.e. element, either belongs or not (either has a value of 1 or 0). Fuzzy sets have not seen as affiliation, but in the degree of belonging. This degree is derived from the interval [0, 1], namely belonging to the function that mirrors every universal set element in this interval of real numbers. In this way, the system becomes significantly adaptive. However, it is important to make a difference between the fuzzy sets and the probability. Although at first glance fuzzy logic resembles a probability, differences are significant. It is likely to examine the accuracy of accidental events, while the fuzzy logic has to deal with inconclusive and inadequtions. A good example of such a comparison would be a "tall man" statement. Namely, that would be the case for the fuzzy logic, while the probability would be applied for recurrence and symbolized by incidental process. We can conclude that the fuzzy logic is dealing with the subjectivity of man, whereas the

probability is more and more objective in statistics and science.

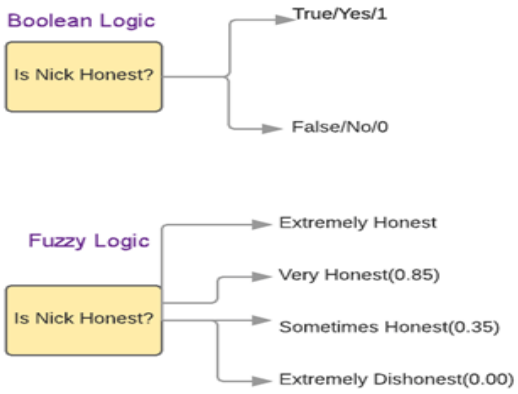

Figure 1 Fuzzy and Boolean logic compared

III. APPLICATION CONCEPT

The primary idea of this example is to compare the fuzzy and regular search. It has been given several files of different extensions (*.txt*, *.csv*, *.docx*, *.xslx*) with 30 cities from the area of the Republic of Serbia. It should be mentioned that the availability of Python libraries plays incredibly big role when it comes to development of intelligent systems such as this one. This application was developed in Python v3.8 and pip installer v19.3, whilst versions from libraries such as: Tkinter, FuzzyWuzzy, docx, xlrd are: 8.6, 0.17.0, 0.8.10, 1.2.0, in order. Based on the use of these libraries, the cities will be loaded first, and then questioned whether there is a match with the entered string. We will be using Python library FuzzyWuzzy for fuzzy search. The basic purpose of this library is to work with the strings, which is also the most common example of fuzzy. Therefore, this example relied precisely on this library.

IV. GUI BASE

In order to achieve the desired goal, which is certainly an adequate input for the user, it had to contain the GUI creation. In the example of our application, the Tkinter library was used to create a graphical interface in the first place, after which it was given the appropriate width and height. When an application runs, the main window is started. Enabling adequate interactivity was possible through the Tkinter function of Askopenfilename. It opens a window for selecting the file from which the search will be done. The window opens by pressing the "Fuzzy pretraga" (eng. Fuzzy search) or the "Obicna pretraga" (eng. Regular search) button.

V. FUNCTIONALITY

In this chapter, we will describe the features that were used to make searches of this application completely usable:

- get_matches (query, limit) - collects the accuracy match of the input with the strings from the files. The base of the function is fuzzywuzzy module process and it's function extract(). Parameters:
- query - forwarded by the function pass_value () in GUI button, or the last saved value for the query that this function will process.
- limit - the number of results that will be shown. On graphical interface user is able to specify the number on an input field next to 'Unesite broj traženih vrednosti' (eng. 'Enter the desired number of results').
- Function city() called to get the array of cities.
- city () - checks the file extension. So if the selected file ends, let's say, *.docx*, it will open it, and it will put the list of cities in an array, that will then be returned (and used weather in regex or fuzzy search).
- re.search(query) – regex search function. Query is passed with pass_value() function in GUI button.

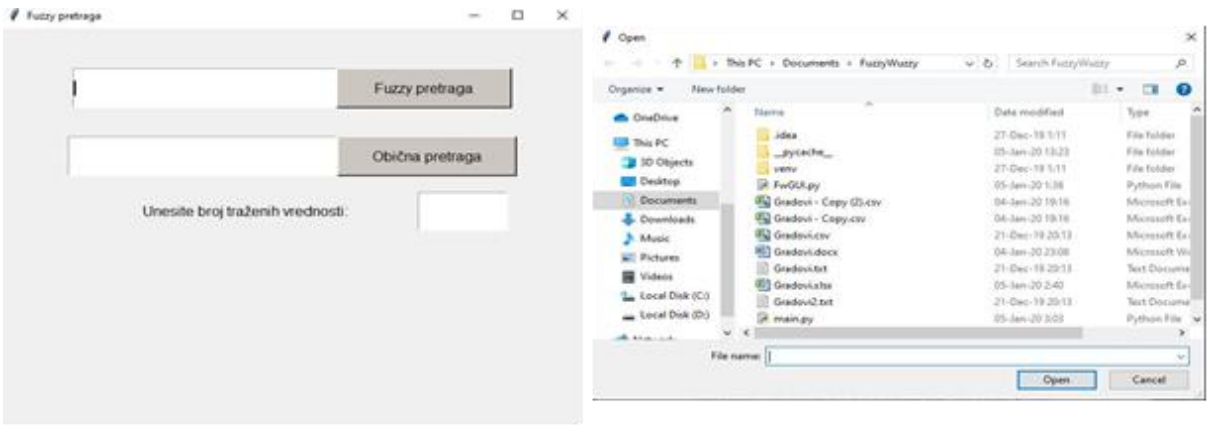

Figure 2 GUI and dialog frame

VI. SEARCH RESULTS AND COMPARISON

We came to the final testing and comparison of searches. On the one hand, we have fuzzy search for which it was necessary to provide several parameters as described above. On the other hand, a "regular" search, by regex. We will do an example of comparing two searches over a .docx file. We'll do a fuzzy search first. As we can see in Figure 3, the number entered specifies the number of results that are displayed. This means that the fuzzy function goes through the whole sequence, finds matches, and then sorts the results by largest. So by choosing 4 results we actually get the 4 best results, shown with their match percentage. The most results that can be displayed depend on the amount of data in the file itself, in our case 30.

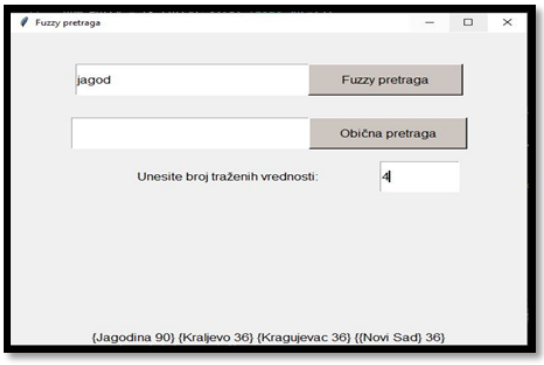

Figure 3 Fuzzy search

Also, it is important to note that the search fuzzy first of all looks to see if there is an entire string match. If it doesn't find the whole, it breaks it down into smaller units and/or characters. For this reason, as well as the length of the names of cities, as a result, we get some cities where the text is not contained, in our example **Kraljevo**, **Kragujevac**, **Novi Sad**. Thus, **Kraljevo** contains **j**, **a**, **o**, **Kragujevac** contains **a**, **j**, while **Novi Sad** contains **o** and **a**. **Jagodina** has the largest match because it contains a complete string in its name.

Next, we move on to a regular search. We will enter the same text as for the fuzzy search and select the same .docx file. As the image itself shows the result is **None** the exact match was not found at all, even though there is a **Jagodina** in our file that contains the same string as the entered text. However, as **Jagodina** has a capital letter **J** and the entry passes a low caps **j**, the search throws out that there are no results. We would get the same case when typing a space, as the first character in our entry. The spacing may be followed by the exact name of the city, however, it is ignored by the fact that there is no city with the initial character being the spacing. The search will only work after typing the required entry with the correct record, in this case in capital letters.

This search is not perfect by itself, as it throws out the coordinates of the string we are looking for in the file (the file is viewed as a coordinate system). Nevertheless, although it can be modified, we have left the original form, to show all the vulnerabilities of ordinary comparisons.

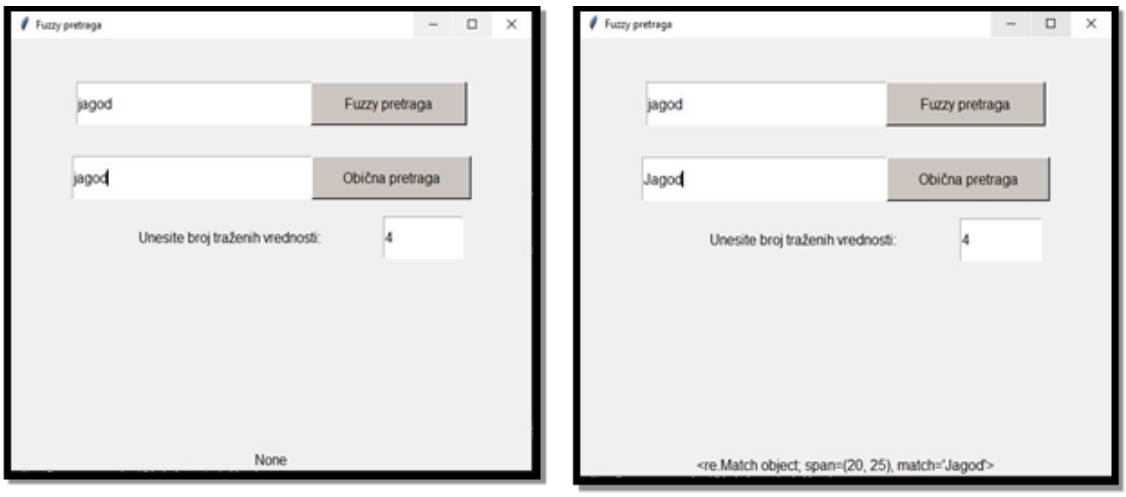

Figure 4 Regex search with and without first capital letter

We got the same results with other types of files that can be opened (described above).

The figure 5 shows the most significant results regarding the differences between these searches. In addition to the aforementioned differences when it comes to lowercase and uppercase letters and spacing, one shortcoming can be observed in the search fuzzy. Namely, fuzzy search (as well as the regex search) does not recognize the Latin letters **š**, **č**, **ć**, **ž**, **đ**. The problem itself lies with Python, so the application itself should be specifically tuned for this purpose. However, in addition to this shortcoming, the fuzzy search found a result at least when it comes to larger strings, such as in Figure 5 example 4. There the only negative result is in the case of **Ša**, because by not recognizing the letter **š**, only the letter **a** remains, which is contained in almost every given city.

	A	B	C	D
1	Redni broj	Upit	Fuzzy pretraga	Obicna pretraga
2	1	"Beograd"	[Beograd 100] [Bor 60] [Sombor 46]	Match(2,9)='Beograd'
3		"Be"	[Beograd 90] [Beocin 90] [Bor 67]	Match(2,5)='Beo'
4		"ogr"	[Beograd 90] [Bor 67] [Sombor 60]	Match(4,7)='ogr'
5				
6	2	"arandjelovac"	[Arandjelovac 100] [Vranje 75] [Kragujevac 64]	none
7		"ar"	[Novi Pazar 90] [Pozarevac 90] [Zajecar 90]	Match(128,130)='ar'
8		"ovac"	Arandjelovac 90] [Leskovac 90] [Vladimirovac 90	Match(10,14)='ovac'
9				
10				
11	3	"Uzice"	[Uzice 100] [Loznica 50] [Subotica 46]	Match(290,295)='Uzice'
12		"U"	[Krusevac 90] [Subotica 90] [Uzice 90]	Match(290,291)='U'
13		"ce"	[Novi Becej 90] [Pancevo 90] [Uzice 90]	Match(153,155)='ce'
14				
15				
16	4	"Šabac"	[Šabac 90] [Požarevac 46] [Vršac 44]	none
17		"Ša"	[Arandjelovac 90] [Beograd 45] [Jagodina 45]	none
18		"aba"	[Šabac 86] [Arandjelovac 60] [Novi Pazar 60]	Match(287,290)='aba'
19				
20				
21	5	" sad"	[Novi Sad 90] [Arandjelovac 60] [Beograd 60]	none

Figure 5 Different results comparing fuzzy and regex

If we dwell briefly on this result, we will notice that the fuzzy for cities with the same degree of match is sorted in the order in the file. The same case occurred in example 2, the case of **ovac**. So, the fuzzy itself sorts the best results, but the problem can arise when matches occur. We then need to know the location of our desired result in the file or request a larger display volume.

VII. CONCLUSION

Created application whose main task is to compare searches, where the first search works on the principle of fuzzy logic, while the second does not have this possibility, is another in a series of indicators that makes fuzzy logic very popular and increasingly present among computer systems. This artificial intelligence concept is an indispensable part of all modern technology, and this time it justified all expectations by delivering the required results with a fairly simple implementation of this technology. A search using fuzzy logic will give the user significantly better results than a search that does not have this capability in itself because of the relevance of the results.

Acknowledgement

This work was partially supported by the Serbian Ministry of Education and Sciences (Grant No: 171039).

REFERENCES:

[1] Djam, X. Y., G. M. Wajiga, Y. H. Kimbi, and N. V. Blamah. "A Fuzzy Expert System for the Management of Malaria." International Journal of Pure & Applied Sciences & Technology 5, no. 2 (2011).

[2] Bede and Barnabas. Mathematics of Fuzzy Sets and Fuzzy Logic. Studies in Fuzziness and Soft Computing: Springer, 2013.

[4] Chen MY, Chen BT. Online fuzzy time series analysis based on entropy discretizationand a
Fast Fourier Transform. Appl Soft Comput.2014;14:156–166.

[5] F, Zhang L, Zhang D, et al. A novel forecasting method based on multi-order fuzzy time
seriesand technical analysis. Inf Sci.2016;367-368:41–57

[6] Gautam SS, Singh SR. A refined method of forecasting based on high-order intuitionistic
fuzzytime series data. Prog Artif Intell.2018;7(4):339–350.

[7] Egrioglu E, Yolcu U, Bas E. Intuitionistic high-order fuzzy time series forecasting method
based onpi-sigma artificial neural networks trained by artificial bee colony. Granular
Comput.2018:1–6.DOI:10.1007/s41066-018-00143-5

[8] Pulkkinen, Pietari, and Hannu Koivisto. "Fuzzy classifier identification using decision tree
and multiobjective evolutionary algorithms." International Journal of Approximate Reasoning
48, no. 2 (2008): 526-543.

[9] MATIAŠKO, Karol, Ján BOHÁCIK, Vitaly LEVASHENKO, and Štefan KOVALÍK.
"Learning fuzzy rules from fuzzy decision trees." Journal of Information, Control and
Management Systems 4, no. 2 (2006).

[10] N. A. Menad, Z. Noureddine, A. Hemmati-Sarapardeh, and S. Shamshirband, "Modeling
temperature-based oil-water relative permeability by integrating advanced intelligent models
with grey wolf optimization: application to thermal enhanced oil recovery processes," *Fuel*, vol.
242, pp. 649–663, 2019.

[11] M. Dehghani, H. Riahi-Madvar, F. Hooshyaripor et al., "Prediction of hydropower
generation using Grey wolf optimization adaptive neuro-fuzzy inference system," *Energies*, vol.
12, pp. 1–20, 2019.

[12] H. Riahi-Madvar, M. Dehghani, A. Seifi et al., "Comparative analysis of soft computing
techniques RBF, MLP, and ANFIS with MLR and MNLR for predicting grade-control scour
hole geometry," *Engineering Applications of Computational Fluid Mechanics*, vol. 13, no. 1, pp.
529–550, 2019.

[13] M. Koopialipoor, A. Fahimifar, E. N. Ghaleini, M. Momenzadeh, and D. J. Armaghani,
"Development of a new hybrid ANN for solving a geotechnical problem related to tunnel boring
machine performance," *Engineering with Computers*, vol. 36, no. 1, pp. 345–357, 2019