

# CREATING CHATBOT WITH FUZZY REGULAR EXPRESSIONS

Natural Language Processing  
Ac. Year 2023/2024

Presented by  
Nurbek Mauletkhan  
Marina Nurgaliyeva

# CONTENT



- 1 Project description
- 2 Methodology
- 3 Theoretical Concepts
- 4 Implementation and Results
- 5 Summary
- 6 References

# PROJECT DESCRIPTION

## Chatbot

---

Restaurant Reservation Assistant Bot  
Answers questions related to restaurant reservations, menu, address and other common questions

## Fuzzy machining algorithm

---

We aimed to improve the chatbot's accuracy and flexibility in understanding user intent, even with minor typographical errors or variations in phrasing.

## Enhancing accessibility

---

We implemented **accessibility features** such as **high-contrast mode**, **large text**, and **voice interaction** to support visually impaired users

# METHODOLOGY

- 1 Programming Language: Python
- 2 Framework: Flask for web application development
- 3 Fuzzy Matching: thefuzz library for implementing fuzzy regular expressions
- 4 Accessibility: Web technologies such as HTML, CSS, and JavaScript for creating an accessible user interface
- 5 Voice Interaction: Web Speech API for voice input and output
- 6 Data Storage: JSON file for storing intents and responses

# What is fuzzy string matching?

It's a technique used to identify two elements of text strings that match partially but not exactly. It involves a rough comparison of two texts and allows you to determine the similarity between two texts.

---

"Helo"  $\approx$  "Hello"

"working hour\$"  $\approx$  "working hours"

## Theoretical Concepts

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

```
Str1 = "Apple Inc."  
Str2 = "apple Inc"  
Distance = levenshtein_ratio_and_distance(Str1,Str2)  
print(Distance)  
Ratio = levenshtein_ratio_and_distance(Str1,Str2,ratio_calc = True)  
print(Ratio)  
  
The strings are 2 edits away  
0.8421052631578947
```

# How to determine the closeness of two strings?

---

Levenshtein distance is a measure that calculates distance of two sequences of words from each other.



# TheFuzz Package

`from thefuzz import fuzz`

TheFuzz is a Python package that uses a combination of fuzzy logic and distance Levenshtein to find the similarity between two strings.

It provides several methods for different types of fuzzy matching:

- Simple Ratio
- Partial Ratio
- Token Sort Ratio
- Token Set Ratio

# PARTIAL RATIO

compares the shortest string to all possible substrings of the longer string. It finds the best possible match, and this is why order can matter here.



```
1 # Order matters with partial ratio
2 # Check the similarity score
3 name = "Kurtis Pykes"
4 full_name = "Kurtis Pykes K D"
5
6 print(f"Partial ratio similarity score: {fuzz.partial_ratio(name, full_name)}")
7
8 # But order will not effect simple ratio if strings do not match
9 print(f"Simple ratiosimilarity score: {fuzz.ratio(name, full_name)}")
10
11 """
12 Partial ratio similarity score: 100
13 Simple ratio similarity score: 86
14 """
```

# SIMPLE RATIO

measures the similarity between two strings by computing the edit distance. It simply calculates how many edits (insertions, deletions, substitutions) are required to make one string the other.



# Chatbot Development

We developed the chatbot using the Flask framework. The chatbot uses a JSON file to store intents and responses. User queries are processed by normalizing the text (lowercasing and removing punctuation) and using fuzzy matching to find the best matching intent.

```
# Importing flask module in the project is mandatory
# An object of Flask class is our WSGI application.
from flask import Flask
# Flask constructor takes the name of
# current module (__name__) as argument.
app = Flask(__name__)
# The route() function of the Flask class is a decorator,
# which tells the application which URL should call
# the associated function.
@app.route('/')
# '/' URL is bound with hello_world() function.
def hello_world():
    return 'Hello World'
# main driver function
if __name__ == '__main__':
    # run() method of Flask class runs the application
    # on the local development server.
    app.run()
```

Example code for the Flask application

# Fuzzy Matching Implementation

The `find_match_token_set_ratio` method in the Chatbot class uses fuzzy matching to compare user inputs with predefined patterns in the intents. This allows the chatbot to understand and respond to queries even when they are not phrased exactly as expected

```
def preprocess_text(self, text):  
    # Normalize text by converting to lowercase and removing punctuation  
    text = text.lower()  
    text = re.sub(r'^\w\s', '', text)  
    return text  
  
def find_match_token_set_ratio(self, user_input, threshold=75):  
    user_input = self.preprocess_text(user_input)  
    best_match = 0  
    match = None  
    for intent in self.intents:  
        for pattern in intent['patterns']:  
            pattern = self.preprocess_text(pattern)  
            similarity = fuzz.token_set_ratio(user_input, pattern)  
            if similarity >= threshold and similarity > best_match:  
                best_match = similarity  
                match = intent['tag']  
    return match
```

Example code for preprocessing text and finding the best match

# Accessibility Features



We implemented several accessibility features

**High-Contrast Mode:**  
Provides better  
visibility for users with  
visual impairments.

**Voice Interaction:**  
Allows users to input  
queries and receive  
spoken responses using  
the Web Speech API.

**Large Text:**  
Increases the font  
size for better  
readability.

# Summary

The goal of creating a chatbot with enhanced natural language understanding and accessibility features was successfully achieved. The fuzzy matching algorithm significantly improved the chatbot's ability to understand varied user inputs. The accessibility features made the chatbot more user-friendly for visually impaired users. Future improvements could include expanding the range of intents and responses and integrating a database for more complex data handling.



# References

- Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing.
- Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing.
- Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python.
- Duckett, J. (2014). HTML and CSS: Design and Build Websites.
- Flanagan, D. (2011). JavaScript: The Definitive Guide.
- W3C. (2018). Web Content Accessibility Guidelines (WCAG) 2.1.
- W3C. (2014). Web Speech API Specification.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms.
- Bray, T. (2014). The JavaScript Object Notation (JSON) Data Interchange Format.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts.
- "Fuzzy String Matching in Python" - DataCamp. Available at:  
`\url{https://www.datacamp.com/tutorial/fuzzy-string-python}`

**Thank you for your attention**