შეხვედრა 7: შესავალი ბუნებრივი ენის დამუშავებაში (NLP) – ვასწავლოთ ბოტს წაკითხვა

აქამდე ჩვენი ბოტი მუშაობდა ძალიან მარტივი პრინციპის გათვალისწინებით: ის ელოდა კონკრეტულ სიტყვას ან ფრაზას; შემდეგ ეძებდა ამ სიტყვას/ფრაზას ლექსიკონში; თუ იპოვიდა მას, შესაბამის პასუხს აბრუნებდა. დღეს კი ვისწავლით, როგორ გავხადოთ ჩვენი ბოტი კიდევ უფრო „ჭკვიანი“ და ვასწავლოთ მას ადამიანური ენის „გაგება“.

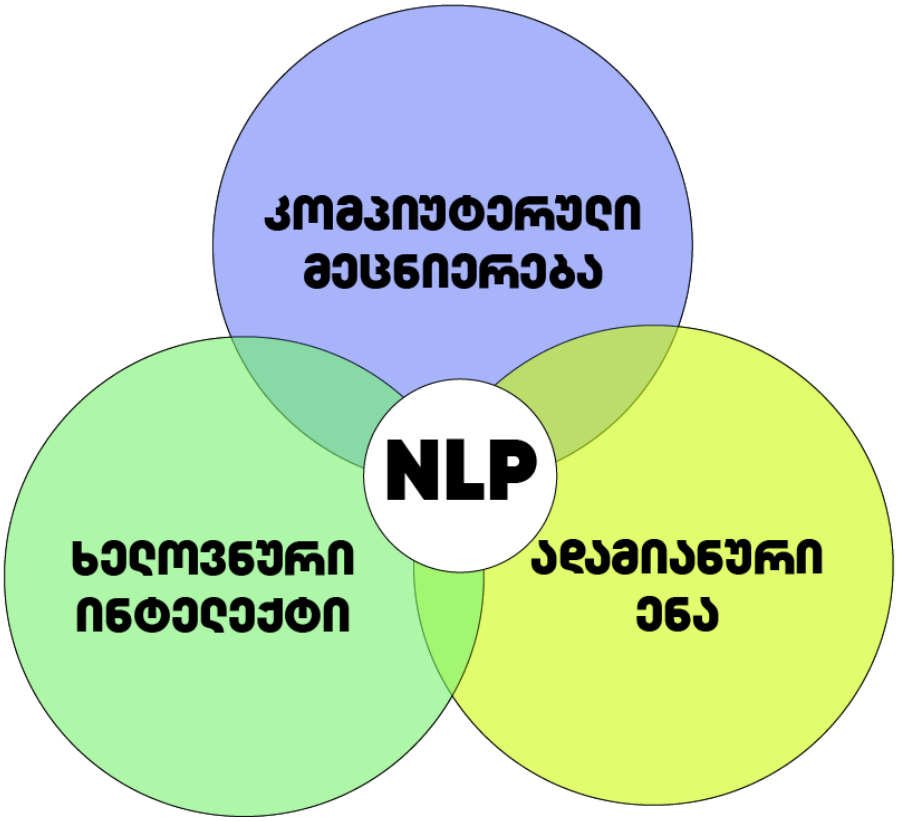
ბუნებრივი ენის დამუშავება (NLP) არის ხელოვნური ინტელექტის ის სფერო, რომელიც კომპიუტერს ასწავლის, როგორ წაიკითხოს გაიგოს, გააანალიზოს და ისაუბროს ადამიანურ ენაზე. ამ შეხვედრის ბოლოს შენ შეძლებ, დაწერო პროგრამა, რომელიც მომხმარებლის შეტყობინებას დაამუშავებს. ამოშლის **უსარგებლო** სიმბოლოებსა და სიტყვებს. მოამზადებს მას ისე, რომ ბოტისათვის „გასაგები“ იყოს. ამგვარად, შენი ბოტი კიდევ უფრო მოქნილი და ჭკვიანი გახდება.

1. როგორ ესმის კომპიუტერს ადამიანური ენა?

ადამიანები ერთმანეთს ბუნებრივი ენის გამოყენებით ვესაუბრებით. ეს ენა სავსეა კონტექსტით, ორაზროვნებით, იდიომებითა და სინონიმებით. კომპიუტერისთვის ეს ძალიან რთული გასაგებია, რადგან ის მხოლოდ მკაცრ წესებსა და ლოგიკას ექვემდებარება. კომპიუტერი, თავისი არსით, გამომთვლელი მანქანაა. მანქანა რომელიც ყველაფერს ციფრებად აღიქვამს და ამ ციფრებს კონკრეტული წესების შესაბამისად ამუშავებს. როდესაც მას ვეუბნებით „ჩვენ ვსაუბრობთ“, ის ხედავს არა სიტყვებს, არამედ სიმბოლოების ერთობლიობას. ამიტომ, ადამიანური ენის რთული სტრუქტურის გასაგებად საჭიროა, ადამიანურმა ენამ ისეთი სახე მიღოს, რომელიც კომპიუტერისთვის იქნება გასაგები. ეს იგივეა, რაც აუხსნა პატარა ბავშვს რთული კონცეფცია მარტივი სიტყვების და ანალოგიების გამოყენებით.

1.1. NLP - (Natural Language Processing)

NLP (ბუნებრივი ენის დამუშავება) არის ხელოვნური ინტელექტის ის სფერო, რომელიც კომპიუტერს ასწავლის, როგორ წაიკითხოს, გაიგოს, გააანალიზოს და როგორ ისაუბროს ადამიანისავის გასაგებ ენაზე. NLP-ის საშუალებით შეგვიძლია, ჩვენს ბოტს ვასწავლოთ, რომ წინადადებები „გამარჯობა“, „სალამი“ და „მოგესალმებით“ ერთსა და **იმავეს** ნიშნავს. ეს მას საშუალებას მისცემს, ერთი და იგივე პასუხი გასცეს მისალმებას, მიუხედავად იმისა, თუ რა სიტყვას გამოიყენებს მომხმარებელი.



1.2. ძირითადი გამოწვევა: ადამიანური ენა სავსეა ორაზროვნებითა და კონტექსტით

NLP-ისთვის ერთ-ერთი უდიდესი გამოწვევა ადამიანური ენის მრავალფეროვნებაა.

ანალოგია: წარმოიდგინე, რომ მეგობარს ეუბნები: „ისეთი რამ მოხდა, კინაღამ გული გამისკდა“. შენთვის და ნებისმიერი სხვა ადამიანისთვის ცხადია, რომ ეს ნიშნავს „ძალიან შემეშინდა“ ან „შოკში ვიყავი“.

კომპიუტერმა კი, რომელსაც ენის ნიუანსები არ ესმის, შესაძლოა ეს წინადადება პირდაპირი მნიშვნელობით გაიგოს და იფიქროს, რომ შენ რეალური, სიცოცხლისთვის საშიში სამედიცინო პრობლემა შეგექმნა. სწორედ NLP (ენის ბუნებრივი დამუშავება) ეხმარება კომპიუტერს, გაარჩიოს ასეთი ფრაზეოლოგიზმები და მიხვდეს, რომ „გულის გასკდომა“ ამ შემთხვევაში არა ფიზიკური, არამედ ემოციური მდგომარეობის აღმნიშვნელია.

2. ტექსტის მომზადება ანალიზისთვის

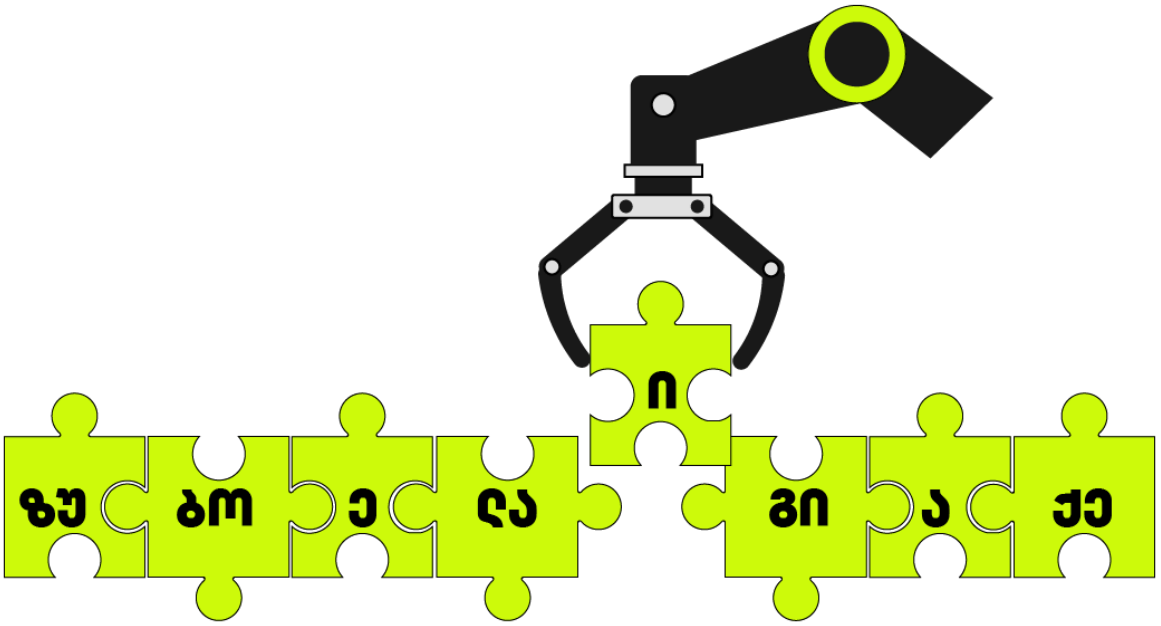
სანამ კომპიუტერი ტექსტს გააანალიზებს, ის უნდა მოვამზადოთ. ეს ჰგავს ნამცხვრის გამოცხობამდე ინგრედიენტების მომზადებას - ფქვილის გაცრას ან კარაქის დარბილებას.

2.1. ტოკენიზაცია: ტექსტის დაშლა სიტყვებად (ტოკენებად)

ტექსტის მომზადებაში ზედმეტი ნაწილების მოცილება და ტოკენიზაცია იგულისხმება. ტოკენიზაცია არის ტექსტის დაშლა ცალკეულ სიტყვებად ან სიმბოლოებად. მარტივად ამის გაკეთება split() მეთოდის გამოყენებით არის შესაძლებელი.

ანალოგია: ტექსტი ჰგავს აწყობილ ფაზლს, სადაც თითოეული ნაწილი (ტოკენი) არის ცალკეული სიტყვა ან სიმბოლო. ტოკენიზაცია ამ ფაზლს ისევ ცალკეულ ნაწილებად ყოფს, რათა კომპიუტერმა თითოეული ნაწილის ცალ-ცალკე დამუშავება შეძლოს.

```
sentence = "როგორ ხარ?"
tokens = sentence.split()
print(tokens) # შედეგი: ['როგორ', 'ხარ?']
```



ინტერაქტიული სავარჯიშო 1:

დაშალე წინადადება - „კოდირება ძალიან საინტერესოა!“ ტოკენებად `split()` მეთოდის გამოყენებით.

```
# მოცემული წინადადება:

my_sentence = "კოდირება ძალიან საინტერესოა!"



# გამოიყენე split() მეთოდი წინადადების სიტყვებად დასაშლელად.
```

```
# და შეინახე ცვლადში `tokens`.


# აქ დაწერე შენი კოდი:



# დაბეჭდე შედეგი

print(tokens)
```

2.2. უსარგებლო სიტყვების (Stop Words) და სასვენი ნიშნების მოშორება

ზოგიერთ სიტყვას (როგორიცაა „და“, „რომ“) არ აქვს დიდი მნიშვნელობა, რადგან ისინი არ ცვლიან წინადადების შინაარსს. ასეთ სიტყვებს Stop Words ჰქვია. ასევე, ზოგჯერ ბოტისთვის სასვენი ნიშნებიც არ არის საჭირო, ამიტომ მათი **წაშლა/გამოუყენებლობა** აუმჯობესებს ანალიზის ხარისხს.

ანალოგია: წარმოიდგინე, რომ ეძებ წიგნის მთავარ იდეას. შენ ყურადღებას **არ აქცევ** ისეთ სიტყვებს, როგორიცაა „და“ ან „არის“ **და** ეძებ **მხოლოდ** საკვანძო სიტყვებს. ეს არის ზუსტად ის, რასაც აკეთებს პროგრამა .

ინტერაქტიული სავარჯიშო 2:

`string` ბიბლიოთეკა შეიცავს ყველა სასვენ ნიშანს `string.punctuation` ცვლადში. დაბეჭდე ეს ცვლადი, რათა დაინახო, რა სიმბოლოებს **ამოშლის** ჩვენი პროგრამა.

```
# string ბიბლიოთეკის შემოტანა

import string


# დაბეჭდე string.punctuation ცვლადის მნიშვნელობა


# აქ დაწერე შენი კოდი:
```

3. სიტყვების ფუძეების პოვნა - ვასწავლოთ ბოტს სინონიმების ამოკითხვა

NLP-ის ერთ-ერთი ყველაზე რთული ამოცანა სიტყვების სხვადასხვა ფორმების ერთსა და იმავე ძირთან დაკავშირებაა. ეს **მოგვაგონებს სიტუაციას,** როდესაც ბავშვს ვასწავლით, რომ წყალი, ყინული და ორთქლი ერთი და იგივე **ნივთიერებაა წარმოდგენილი** სხვადასხვა ფორმით. ასევე, კომპიუტერმა უნდა გაიგოს, რომ "მივდივარ", "წავედი" და "წასვლა" ერთი და იგივე მოქმედების სხვადასხვა ფორმებია.

3.1. ლემატიზაცია: რატომ არის "წავიდა", "მიდის", "წასვლა" ერთი და იგივე ძირის მქონე

წარმოიდგინე, რომ მომხმარებელი გწერს: „მომენატრა“. შენი ბოტისთვის გასაღები კი არის „მონატრება“. იმისთვის, რომ ბოტმა გაიგოს, რომ ეს ორივე სიტყვა ერთსა და იმავე განზრახვას უკავშირდება, ჩვენ უნდა მოვძებნოთ მათი ფუძე. ამას **სტემინგი (Stemming)** ან **ლემატიზაცია (Lemmatization)** ჰქვია.

- **სტემინგი:** უბრალოდ აჭრის სიტყვას ბოლოდან ზედმეტ ნაწილებს (მაგ., „წასვლა“ -> „წასვ“, „წავიდა“ -> „წავ“).
- **ლემატიზაცია:** უფრო ჭკვიანი პროცესია, რომელიც სიტყვას ლექსიკონის ფორმაში აბრუნებს (მაგ., „წავიდა“ -> „წასვლა“).

ეს ტექნიკა ჩვენს ბოტს ეხმარება, გაიგოს, რომ „წავედი“, „მივდივარ“, „წავალ“ ერთსა და იმავე ძირს უკავშირდება.

დავალება 7: ტექსტის დამუშავება

აღწერა: დაწერე ფუნქცია clean_text(text), რომელიც მიიღებს წინადადებას, მოაშორებს სასვენ ნიშნებს (,, ., !, ?), დააბრუნებს გასუფთავებულ ტექსტს.

დავალება 7.1: შეცდომის პოვნა და გასწორება

მოცემულ კოდში დაშვებულია შეცდომა. შენი ამოცანაა იპოვო და გაასწორო ის, რათა პროგრამამ პირობის შესაბამისად იმუშაოს.

```
import string

def clean_text(text):
    for char in string.punctuation:
        text text.replace(char, '')
    return text

user input = "გამარჯობა! როგორ ხარ?"
cleaned clean_text(user_input)
print(cleaned.lower)
```

დავალება 7.2: კოდის დასრულება

მოცემულ პროგრამულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
import string
```

```
def clean_text(text):
    for char in string.punctuation:
        text = text.replace(char, '')

user_input = "გამარჯობა! როგორ ხარ?"
cleaned_text = clean_text(user_input)
print(cleaned_text)
```

დავალება 7.3: კოდის დაწერა ნულიდან
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შექმნის ფუნქციას clean_text(text), ტექსტიდან ამოშლის სასვენ ნიშნებს, დააბრუნებს შემცირებული ზომის ტექსტს.

```
# დაწერე შენი კოდი აქ:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შეგიძლია გადაამოწმო შენი ნამუშევარი

import string

# ფუნქცია, რომელიც ასუფთავებს ტექსტს
def clean_text(text):
    # ყველა სასვენი ნიშნის მოშორება
    for char in string.punctuation:
        text = text.replace(char, '')
    # ტექსტის დაპატარავება და დაბრუნება
    return text.lower()

# საცდელი ტექსტი
user_input = "გამარჯობა! როგორ ხარ?"

# ფუნქციის გამოძახება
cleaned_text = clean_text(user_input)
```

```
# შედეგის დაბეჭდვა
print(cleaned_text)
```