Proiect inferenta predicativa prin rationament inainte

Cuprins

1. Descrierea problemei considerate

Proiectul abordat se concentreaza pe aplicarea inferentei predicative prin rationament inainte pentru a demonstra doua teoreme cu relevanta matematica. In contextul inteligentei artificiale, acest tip de rationament este folosit pentru a deriva concluzii logice plecand de la un set de premise initiale.

Problemele selectate pentru demonstratie sunt:

1) Dezvoltarea binomiala a expresiei “(a + b) ^ n” utilizand Teorema binomiala a lui Newton.
2) Crearea unui sistem de recomandari bazat pe istoricul de achizitii al unui utilizator pentru un magazin online.

2. Aspecte teoretice privind algoritmul

Inferenta predicativa prin rationament inainte presupune utilizarea unui set de reguli logice pentru a deriva concluzii din premise. Algoritmul urmeaza pasii urmatori:

1) Se identifica premisele de intrare.
2) Se verifica regulile aplicabile fiecarei premise.
3) Se genereaza concluzii intermediare pana cand se ajunge la scopul final sau la o solutie satisfacatoare.

Teorema Binomiala a lui Newton: Aceasta afirma ca orice expresie de forma "(a + b) ^ n" poate fi dezvoltata ca o suma ponderata de termeni, unde coeficientii binomiali sunt calculati folosind formula:

C(n, k) = n! / [k!(n-k)!]

Sistemul de Recomandari: Un sistem de recomandari care utilizeaza istoricul utilizatorului pentru a anticipa interesele viitoare, bazandu-se pe principii precum:

- Categoriile preferate de utilizator.
- Gama de preturi a achizitiilor anterioare.
- Similaritatea dintre produsele achizitionate si cele disponibile.

3. Modalitatea de rezolvare

Pentru teorema binomiala:

- Implementarea formulei generale pentru dezvoltarea binomiala folosind Python.
- Calcularea coeficientilor binomiali pentru fiecare termen.
- Generarea termenilor expansiunii, afisarea rezultatelor intermediare si a rezultatului final.

Pentru sistemul de recomandari:

- Crearea de utilizatori si adaugarea istoricului de cumparare.
- Salvarea si citirea datelor despre utilizatori si produse din fisiere JSON.
- Analizarea categoriilor preferate si a intervalului de preturi.
- Selectarea produselor relevante utilizand o metoda de filtrare bazata pe reguli simple.

Reguli folosite pentru recomandare:

```
def matches_category(user, product):
  return product["category"] in user["categories"]

def matches_price_range(user, product):
  return user["price_range"][0] <= product["price"] <= user["price_range"][1]

def not_purchased_before(user, product):
  return product["name"] not in user["purchase_history"]
```

4. Parti semnificative din codul sursa

Teorema binomiala:

Aceasta este premisa si formula generala a functiei **newton_binomial** ce va fi apelata pentru dezvoltarea binomului lui Newton:

```
def newton_binomial(a, b, n):

  steps = []



  steps.append(f"Premisa: Formula generală pentru Binomul lui Newton este:")

  steps.append(f"(a + b)^n = ∑(k=0 până la n) [C(n, k) * a^(n-k) * b^k]\n")
```

Secventa de cod de mai jos acopera cazul de baza (in care valoarea puterii (n) este 1), in care codul prezinta formula simplificata “(a+b) ^ 1” si returneaza rezultatul expanded si numeric:

```
if n == 1:

  steps.append("Caz de bază: Pentru n=1, formula devine:")

  steps.append("(a + b)^1 = a + b")

  expanded_result = f"{a} + {b}"

  numeric_result = float(a) + float(b)

  return steps, expanded_result, numeric_result
```

Calculul termenilor binomiali pentru celelalte cazuri posibile (n > 1)

```
expanded_terms = []
numeric_terms = []

for k in range(n + 1):
  coef = binomial_coefficient(n, k)
  steps.append(f"Pasul {k + 1}: Calculăm termenul pentru k={k}.")
  steps.append(f"Coeficientul binomial C({n}, {k}) = {coef}")

  expanded_term = f"{coef} * {a}^{n - k} * {b}^{k}"
  expanded_terms.append(expanded_term)
  steps.append(f"Termenul expandat este: {expanded_term}")

  numeric_term = coef * (float(a) ** (n - k)) * (float(b) ** k)
  numeric_terms.append(numeric_term)
  steps.append(f"Valoarea numerică este: {numeric_term}")


  steps.append(f"Adăugăm termenul în expansiune.\n")
```

In secventa de cod prezentata mai sus, apelul functiei **binomial_coefficient** are ca scop calculul coeficientului binomial C(n,k), pentru a putea determina coeficientii in expansiunea binomialului.

```
def binomial_coefficient(n, k):
  return math.comb(n, k)
```

Restul codului prezentat se ocupa de formatarea valorilor obtinute in coef pentru a putea fi afisate in forma expanded si numerica, urmand a fi adaugati in expansiune si repetand procesul pentru toate expansiunile necesare, dimensiune data de variabila n.

```
expanded_result = " + ".join(expanded_terms)
numeric_result = sum(numeric_terms)
steps.append("Toți termenii au fost calculați. Expansiunea este:")
steps.append(f"{expanded_result}")
steps.append(f"\nIar valoarea numerică este: {numeric_result}")
```

Aceasta ultima secventa de cod extrasa din functia **newton_binomial** se ocupa de gruparea rezultatelor pentru a putea calcula rezultatul numeric final si pentru a putea afisa rezultatul simbolic complet.

Functia **main**

```
def main():
  print("Demonstrație: Binomul lui Newton prin raționament înainte\n")
  a = float(input("Introduceți valoarea lui a: "))
  b = float(input("Introduceți valoarea lui b: "))
  n = int(input("Introduceți valoarea lui n: "))
```

```
if(n < 0):
    raise Exception("valoarea lui n trebuie sa fie mai mare sau egala cu 0.")


steps, expanded_result, numeric_result = newton_binomial(a, b, n)


print("\nPașii demonstrației:")
for step in steps:
    print(step)
```

Functia main are ca scop obtinerea input-ului de la utilizator pentru valorile a,b si n, ca ulterior sa putem apela functia **newton_binomial** pentru a obtine rezultatele dorite folosindu-ne de valorile oferite de utilizator.

Cod relevant pentru sistemul de recomandari:

```
import json
def recommend():
  users = load_data(USERS_DB)
  products = load_data(PRODUCTS_DB)


  user_id = input("Reccommend for user(user_id): ")
  user = next((u for u in users if u["id"] == user_id), None)
```

```
if not user:
    print("The user id was not found")
    return

recommendations = generate_recommendations(user, products)
print("Recommendations:", recommendations)
```

```
def generate_recommendations(user, products, max_recommendations=3):
    recommendations = []

    for product in products:
        if (
            matches_category(user, product) and
            matches_price_range(user, product) and
            not_purchased_before(user, product)
        ):
            recommendations.append(product["name"])

    if len(recommendations) > max_recommendations:
        recommendations = random.sample(recommendations, max_recommendations)

    return recommendations
```

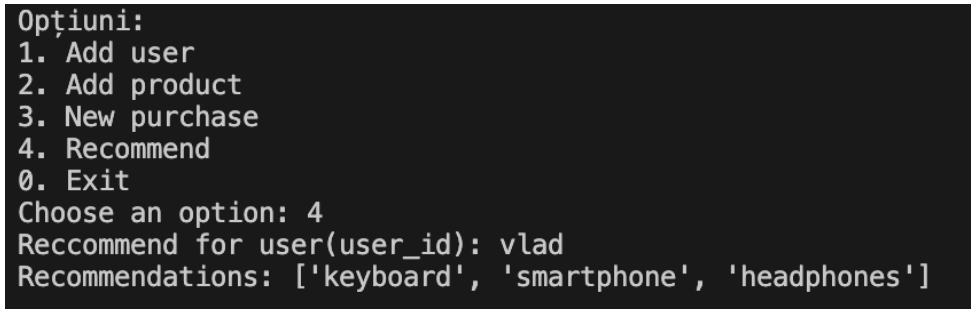
Explicatie*:* Algoritmul selecteaza produse care corespund categoriilor si intervalului de preturi ale utilizatorului.

5. Rezultatele obtinute

Pentru sistemul de recomandari:

- Input: Utilizatorul Vlad(putem vizualiza datele userului folosind functia 5.History)

- Output: Recomandari: tastatura, telefon, casti.



Pentru Teorema Binomiala:

- Input: a = 3, b = 4, n = 3
- Output: 1 * 3.0^3 * 4.0^0 + 3 * 3.0^2 * 4.0^1 + 3 * 3.0^1 * 4.0^2 + 1 * 3.0^0 * 4.0^3

```
Demonstrație: Binomul lui Newton prin raționament înainte

Introduceți valoarea lui a: 3
Introduceți valoarea lui b: 4
Introduceți valoarea lui n: 3

Pașii demonstrației:
Premisa: Formula generală pentru Binomul lui Newton este:
(a + b)^n = ∑(k=0 până la n) [C(n, k) * a^(n-k) * b^k]

Calculăm dezvoltarea pentru n=3. Vom determina fiecare termen al sumei pas cu pas:
Pasul 1: Calculăm termenul pentru k=0.
Coeficientul binomial C(3, 0) = 1
Termenul expandat este: 1 * 3.0^3 * 4.0^0
Valoarea numerică este: 27.0
Adăugăm termenul în expansiune.

Pasul 2: Calculăm termenul pentru k=1.
Coeficientul binomial C(3, 1) = 3
Termenul expandat este: 3 * 3.0^2 * 4.0^1
Valoarea numerică este: 108.0
Adăugăm termenul în expansiune.

Pasul 3: Calculăm termenul pentru k=2.
Coeficientul binomial C(3, 2) = 3
Termenul expandat este: 3 * 3.0^1 * 4.0^2
Valoarea numerică este: 144.0
Adăugăm termenul în expansiune.

Pasul 4: Calculăm termenul pentru k=3.
Coeficientul binomial C(3, 3) = 1
Termenul expandat este: 1 * 3.0^0 * 4.0^3
Valoarea numerică este: 64.0
Adăugăm termenul în expansiune.

Toți termenii au fost calculați. Expansiunea este:
1 * 3.0^3 * 4.0^0 + 3 * 3.0^2 * 4.0^1 + 3 * 3.0^1 * 4.0^2 + 1 * 3.0^0 * 4.0^3

Iar valoarea numerică este: 343.0
```

6. Concluzii

Proiectul demonstreaza aplicabilitatea inferentei predicative in rezolvarea unor probleme practice si teoretice. Implementarea algoritmilor a condus la rezultate corecte si relevante, iar documentatia detaliaza abordarea completa.

Limitari:

- Algoritmul de recomandari nu include tehnici de machine nlearning, ci este bazat pe reguli simple.
- Complexitatea expansiunii binomiale creste exponential in functie de n, ce limiteaza utilizarea implementarii in cazul valorilor mari.

7. Bibliografie

https://www.geeksforgeeks.org/forward-chaining-and-backward-chaining-inference-in-rule-based-systems/

https://www.geeksforgeeks.org/forward-chaining-in-ai-with-fol-proof/

https://www.appliedaicourse.com/blog/forward-chaining-and-backward-chaining-in-ai/

Materiale curs Inteligenta Artificiala 2024.

8. Lista Contributiilor

- **Lupu Eduard:** Implementarea teoremei binomiale si redactarea teoriei matematice.
- **Apostol Vlad:** Dezvoltarea sistemului de recomandari si testarea functionalitatii acestuia.