Proiect SmarTest - Inteligenta Artificiala

- Membrii echipei:

  Matcovici Georgiana
  Gheonea Naomi-Denisa
  Stoleriu Denis-Matei
  Tugui Tudor

- Solutii tehnice posibile:

1. Generarea întrebărilor - Naomi

Dataset și preprocesare

Ca sursă de date folosim un dataset format din exerciții și întrebări din examenele anterioare, laboratoare și fișe de curs. Înainte de a fi utilizat pentru generarea automată, datasetul este prelucrat pentru consistență:

- Curățarea textului: eliminarea semnelor de punctuație, a caracterelor speciale, a numerelor și a cuvintelor de legătură (conjuncții, articole, prepoziții).

- Normalizarea: transformarea tuturor cuvintelor în litere mici pentru a evita potriviri sensibile la majuscule.

- Tokenizarea și lematizarea: împărțirea întrebărilor în termeni de bază (forme canonice) pentru o analiză mai precisă a vocabularului.

- Etichetarea (adnotarea) întrebărilor: fiecare întrebare este clasificată manual într-una dintre categoriile principale:

  - Search problem identification – selectarea celei mai potrivite strategii pentru o problemă dată (ex: n-queens, graph coloring, Hanoi, knight's tour)

  - Game Theory (normal form) – întrebări despre echilibrul Nash pur, strategii dominante, matricea de payoff

- Constraint Satisfaction (CSP) – atribuirea variabilelor rămase folosind Backtracking, Forward Checking, MRV sau AC-3
- Adversarial Search (Minimax) – valoarea rădăcinii și numărul de noduri evaluate cu Minimax și Alpha-Beta pruning
- Întrebări teoretice generale – definiții, diferențe, caracteristici ale algoritmilor sau conceptelor AI

Aceste etichete vor fi folosite pentru antrenarea modelelor de clasificare și generare automată.

Determinarea tipului de întrebare (Regex)

În prima fază, întrebările noi sunt analizate cu expresii regulate (regex) pentru a determina tipul lor.

```
PATTERN_TIP_INTREBARE_REGEX = {
  "definitie": [r"ce\s+este", r"defin\w+", r"reprezintă", r"explică"],
  "diferenta": [r"diferen[ţț]a\s+dintre", r"deosebirea\s+dintre"],
  "caracteristici": [r"caracteristicile", r"avantajele"],
  "problema": [r"calculează", r"determină", r"rezolvă"]
}
```

Această clasificare preliminară ajută la selectarea șabloanelor logice și a tipului de întrebări care trebuie generate pentru fiecare concept.

Ontologia AI

Pentru a ghida generarea întrebărilor și a selecta conceptele relevante, folosim o ontologie AI, structurată ca un fișier JSON care conține:

- concepte și algoritmi principali din curs;
- definiții și caracteristici;
- relații între concepte (ex: „is_a”, „related_to”);

- tipul întrebării asociate (teorie, calcul, problemă specifică).

Exemplu simplificat de intrări în ontologie:


```
{
  "backtracking": {
    "definitie": "Algoritm care explorează toate soluțiile posibile sistematic, revenind asupra alegerilor când întâlnește o constrângere.",
    "tip": "CSP",
    "relatii": { "is_a": "algoritm de căutare", "related_to": ["forward_checking", "MRV"] }
  },
  "minimax": {
    "definitie": "Algoritm pentru jocuri cu doi jucători care maximizează șansa de câștig propriu și minimizează șansa adversarului.",
    "tip": "Adversarial Search",
    "relatii": { "is_a": "algoritm de joc", "related_to": ["alpha_beta_pruning"] }
  }
}
```


Această ontologie permite:

- Selectarea automată a conceptelor atunci când utilizatorul alege un capitol;
- Generarea întrebărilor relevante pentru conceptele selectate;
- Validarea termenilor și a relațiilor în întrebările generate.

Generarea automată a întrebărilor (Seq2Seq Neural Network)

Pentru a evita redactarea manuală a șabloanelor și pentru a permite variații naturale, folosim o rețea neuronală de tip sequence-to-sequence, antrenată pe datasetul normalizat și adnotat.

- Modele posibile: T5, GPT-2 local sau LSTM encoder-decoder
- Dataset de antrenament: perechi `(concept, întrebare)` pentru fiecare tip de întrebare

Exemple:

Search problem identification:

Input: "n-queens"

Output: "Dintre metodele BFS, DFS și Backtracking, care este cea mai potrivită pentru rezolvarea problemei n-queens cu n=8?"

Game Theory:

Input: "Nash equilibrium"

Output: "Pentru jocul dat în forma normală, există vreun echilibru Nash pur?"

CSP:

Input: "Backtracking + MRV"

Output: "Care va fi asignarea finală a variabilelor rămase dacă utilizăm algoritmul Backtracking cu optimizarea MRV?"

Adversarial Search:

Input: "Minimax + Alpha-Beta"

Output: "Pentru arborele de joc dat, care va fi valoarea rădăcinii și câte noduri frunză vor fi evaluate aplicând Minimax cu Alpha-Beta pruning?"

Întrebări teoretice generale:

Input: "Definiția algoritmului A*"

Output: "Ce este algoritmul A* și care sunt caracteristicile principale ale acestuia?"

Fluxul complet de generare

1. Selectarea capitolului sau temei de către utilizator.
2. Extracția conceptelor relevante din ontologia AI.
3. Generarea întrebărilor pentru fiecare concept folosind modelul neuronal.
4. Filtrarea și validarea:
    - Eliminarea întrebărilor duplicate sau prea asemănătoare (folosind embeddings semantice, ex: Sentence-BERT)
    - Verificarea lungimii și coerenței gramaticale
    - Clasificarea finală pe tip (teorie vs calcul) și pe subtip (Search, Game Theory, CSP, Minimax)
5. Exportul întrebărilor: în interfața grafică sau PDF, gata pentru test.

Flux logic:

1. Adnotarea manuală creează „datasetul curat” →
2. Ontologia oferă cunoștințe despre concepte →
3. Neural network folosește dataset + ontologie pentru a genera întrebări corecte și relevante.

2. Generarea răspunsului corect - Matei

Inițial iau ca input întrebarea și cursurile, normalizez atât întrebarea cât și cursurile (transform caracterele în litere mici, elimin semnele de punctuație și conjuncțiile) pentru a avea potriviri cât mai precise.
Următorul pas este determinarea tipului de întrebare folosind Pattern-uri (regex) - (caut anumite cuvinte cheie din întrebare) :

- încerc să identific toate posibilitățile din întrebare pentru un răspuns cât mai precis.

De exemplu:

```
PATTERN_TIP_INTREBARE_REGEX = {
    "definitie": [
        r"ce\s+este",
        r"defin\w+",
        r"reprezintă",
        r"explică",
        r"explicați",
    ],
    "diferenta": [r"diferen[țt]a\s+dintre", r"deosebirea\s+dintre"],
    "caracteristici": [r"caracteristicile", r"avantajele", r"proprietățile"],
    "problema": [r"calculează", r"determină", r"rezolvă"],
}
```

Folosesc expresii regulate (regex) deoarece permit potrivirea mai multor forme gramaticale ale aceluiași cuvânt (ex: defin\w+ identifică „definiția”, „definește”, „definită”, „definirea”).

Astfel, sistemul poate recunoaște mai ușor tipul întrebării chiar dacă formularea diferă.

Întrebarea poate fi de teorie sau de rezolvare a unor probleme. O întrebare se încadreaza în categoria ***Teorie*** dacă întrebarea conține o definiție, diferență sau caracteristici. În caz contrar este de ***Calcul***.

După ce identific tipul de întrebare, voi identifica termenii, respectiv expresia pentru care va trebui să generez răspunsul (elimin cuvintele care determină tipul de întrebare, iar cei rămași vor fi termenii pentru care voi genera un răspuns). Apoi determin cursurile în care se regăsesc.

Ultimul pas este generarea răspunsului corect. Vom reutiliza încă o dată expresii regulate separate pentru răspuns în funcție de tipul de întrebare, extrăgând din cursuri fraza unde se regăsesc termenii până la următorul semn de sfârșit de punctuație.

De exemplu:

```
PATTERN_RASPUNS = {
   "definitie": [
      r"{termen}\s+este\s+(.*?)(?:\.|\n)",
      r"{termen}\s+reprezintă\s+(.*?)(?:\.|\n)",
      r"prin\s+{termen}\s+se\s+înțelege\s+(.*?)(?:\.|\n)"
   ],
```

```
        "diferenta": [
            r"diferența\s+dintre\s+{a}\s+și\s+{b}\s+este\s+(.*?)(?:\.|\n)",
            r"{a}\s+se\s+deosebește\s+de\s+{b}\s+prin\s+(.*?)(?:\.|\n)"
        ],
        "caracteristici": [
            r"caracteristicile\s+{termen}\s+sunt\s+(.*?)(?:\.|\n)",
            r"{termen}\s+se\s+caracterizează\s+prin\s+(.*?)(?:\.|\n)"
        ]
    }
```

Pattern-urile vor fi declarate într-un fișier Python separat.

3. Introducerea și evaluarea răspunsurilor

Aplicatia permite introducerea unui raspuns prin intermediul unei interfete grafice, evalueaza raspunsul procentual (0-100%) si afiseaza raspunsul corect.

Putem adapta ideea de la MediTest, vom avea o ontologie pentru Inteligenta Artificiala in loc de una medicala, care include concepte, algoritmi, termeni relatii, folosind fisiere json.

```
{
  "concept_name": {
    "definition": ->text explicativ
    "key_words": ->lista de termeni importanti
    "relationships": {
      "is_a": ->categorie mai generala
      "related_to": ->alte concepte
    },
    "caracteristici": …
  }
}
```

Exemplu:

```
"iterative_deepening_search": {
    "definitie": "Algoritm care combina avantajele BFS si DFS. Exploreaza starile pana la o adancime limitata, crescand limita treptat pentru a evita buclele infinite si pentru a gasi solutia optima.",
```

```
  "cuvinte_cheie": [
    "adancime", "limita", "crestere", "bfs", "dfs", "iterativ"
  ],
  "relatii": {
    "is_a": "algoritm de cautare",
    "related_to": ["depth_first_search", "breadth_first_search"]
  },
  "caracteristici": [
    "evita buclele infinite",
    "gaseste solutia optima",
    "foloseste cautare limitata",
    "reia explorarea cu adancime mai mare"
  ]
}
```


Normalizam raspunsul primit (litere mici, fara semne de punctuatie), identificam ce concepte din ontologia noastra apar in raspuns. Aplicatia genereaza raspunsul corect la intrebare, si calculeaza scorul raspunsului primit bazandu-se pe

- similaritatea textului (cosine similarity) intre acesta si raspunsul corect
- numarul de concepte din raspunsul corect acoperite de raspunsul primit

Pentru diferite tipuri de intrebari:

- ➔ ,,Pentru problema identificată dintr-o listă de minim 4 (n-queens, generalised Hanoi, graph coloring, knight's tour) și o instanță sau un set de instanțe, care este cea mai potrivită strategie de rezolvare, dintre cele menţionate la curs?'' avem stocate in baza de cunostinta strategiile clasice de rezolvare pentru fiecare problema + eventual strategii partial corecte si putem genera un scor al raspunsului: Ex: n-queens-corect:backtracking(100%), partial corect:DFS(50%)
- ➔ ,,Pentru jocul dat în forma normală (în matricea atașată), există echilibru Nash pur?"-aici aplicatia calculeaza toate echilibrele Nash pure si verifica daca raspunsul este corect sau partial corect
- ➔ ,,Care va fi asignarea variabilelor rămase date fiind variabilele, domeniile, constrângerile și asignarea parțială, dacă am utiliza algoritmul Backtracking cu optimizarea (FC, MRV sau AC-3) pentru continuarea rezolvării?" aplicatia aplica algoritmul si compara rezultatul cu raspunsul primit
- ➔ ,,Pentru arborele dat, care va fi valoarea din rădăcină și câte noduri frunze vor fi vizitate în cazul aplicării strategiei MinMax cu optimizarea Alpha-Beta?'' aplicatia contruieste arborele, aplica algoritmul, compara rezultatele cu raspunsul primit

Deci pentru evaluarea raspunsurilor care necesita aplicarea unor algoritmi, aplicatia aplica algoritmul si compara rezultatele cu raspunsul primit, iar pentru evaluarea

unor intrebari teoretice, de rationamentm se poate folosi o ontologie de concepte, termeni relatii.