Chess Tutor

Motivacija :

Igra Šah je u poslednjih nekoliko godina dostigla popularnost kakvu nije nikad imala u svojoj dugoj istoriji od vise od 1500 godina. Samim tim broj početnika igre je drastično porastao. Ovo dovodi do potrebe za novijim alatima pomoću kojih početnici mogu da uče i usavršavaju svoju igru.

Pregled problema:

- Trenutni programi(Chess Botovi) protiv kojih igrači mogu da igraju i vežbaju su obično zasnovani na takozvanim Chess Engine-ima. Chess Engine je program koji igra šah, najčešće implementiran algoritmima mašinskog učenja. Ovo programu pruža neverovatnu efikasnost I preciznost prilikom izbora najboljeg poteza ali ljudima daje slab ili nikakav uvid u proces donešenja odluka za dati potez.
- Istorijski najpopularniji Chess Engine je takozvani "Stockfish". Stockfish je open-source program i iz tog razloga je najčešći engine koji se koristi za razvijanje chess botova. Ovi botovi često imaju opciju da igračima pre poteza kažu koji su najbolji potezi u datoj poziciji ali retko pružaju razloge zašto su ti potezi dobri. Samim tim početnici koji koriste ove botove da bi vežbali imaju malu korist, jer samo kopiraju najbolje poteze bez razumevanja I to ne doprinosi mnogo njihovom napretku.
- Ideja je napraviti chess bota koji je zasnovan na Stockfishu protiv kojeg korisnik može da igra šah. Između Stockfisha i igrača će se nalaziti ekspertski sistem koji će dobiti preporuke Stockfisha za najbolje poteze i probati da im da objašnjenja zbog čega su oni korisni u kontekstu date šahovske pozicije.

Metodologija rada:

- **Arhitektura Sistema:**

  Sistem se sastoji iz 3 celine:

  - **Front aplikacija** – interaktivna šahovska tabla pomoću koje korisnik igra i dobija feedback od ostatka sistema
  - **Stockfish servis** (python-flask) - "protivnik" igrača, daje svoj potez koji će da odigra kao i 3 najbolja poteza za igrača
  - **Ekspertski sistem** (Spring/drools) - posrednik između igrača i Stockfisha, rezonuje preporuke 3 najbolja poteza za igrača i rezultat rezonovanja šalje igraču kao feedback u vidu poruka za svaki preporučeni potez

- **Ekspertski sistem**
    - **Inputi** – Prilikom početka partije, u radnoj memoriji će biti formiran model šahovske table, sa 64 polja i 32 figure postavljene u početnoj poziciji. Nakon toga, posle svakog igračevog poteza, igračov potez će ući u sistem, kao i odgovor od stockfisha (potez stockfisha + 3 najbolje sledeća poteza za igrača)
    - **Output** – Sistem će kao output vratiti igraču 3 najbolja poteza prema stockfishu, uz komentare zašto su potezi dobri.

- **Baza znanja**
    - **Činjenice**:
        - Model Table – predstavljaju trenutno stanje šahovske table:
        - Square(Piece piece) - jedno polje šahovske table, sadrži figuru koja se nalazi na njoj, u slučaju praznog polja za figuru je postavljena vrednost **null.**
        - Piece – abstraktna klasa koja predstavlja šahovske figure. Njene podklase su: Pawn, Knight, Bishop, Rook, Queen, King.
        - Osnovne Relacije – kako bih mogli da dođemo do kompleksnijih zaključaka, moramo da imamo činjenice koje modeluju osnovne odnose između šahovskih figura. Ove relacije se formiraju na osnovu pozicija figura na tabli. Osnovne Relacije:
            - CanMoveOn (Piece piece, Square square) - predstavlja činjenicu na koja polja data figura može da se kreće.
            - IsAttacking (Piece attackingPiece, Piece attackedPiece) - predstavlja činjenicu koja figura napada koju figuru.
            - IsDeffending(piece deffenderPiece, Piece deffendedPiece) - predstavlja činjenicu koja figura brani koju figuru.
        - Kompleksne relacije – formiraju se rezonovanjem nad osnovnim relacijama I kao takve već mogu biti output iz sistema.
            - IsChecking(Piece attackingPiece, King king) - predstavlja činjenicu koja figura pravi šah kojem kralju.
            - CanCapture(Piece attackingPiece, Piece attackedPiece, String message) - predstavlja činjenicu koja figura može “bezbedno” da pojede koju figuru uz poruku zašto je jedenje figure bezbedno.
            - …
        - Move(String fromSquare, String toSquare) - predstavlja činjenicu odigranog/potencijalnog poteza

- **Pravila:**
  - Pravila za kretanje figura – Ova pravila definišu za svaki tip figure kako može da se kreće po šahovskoj tabli. Rezultat ovih pravile je kreiranje osnovnih relacija u sistemu koja se logički dodaju u radnu memoriju.
    - Primer pravila kretanje konja gore desno:

```
rule "Knight Moves UP RIGHT"

    when
        $s : Square($piece : piece, $squareText : squareText, $startRank : rank, $startFile : file, $boardNumS : boardNum)
        $k : Knight(this.equals($piece))
        $moveSquare : Square(rank == $startRank + 2, file == $startFile + 1,  $squareText2 : squareText, boardNum == $boardNumS, piece == null)
    then
        insertLogical(new CanMoveOn($k, $moveSquare, CanMoveOnType.KNIGHT,$boardNumS));

end
```

  - Pravila za kreiranje kompleksnih relacija – Ova pravila definišu kada se kreiraju kompleksnije relacije na šahovskoj tabli.
    - Primer pravila za kreiranje CanCapture relacije:

```
rule "Is Attacking A Higher Value Piece"
salience -99
    when
        $p1 : Piece($p1Value : value)
        $p2 : Piece(value > $p1Value)
        IsAttacking($p1, $p2;)
    then
        insertLogical(new CanCapture($p1, $p2, $p2.getPieceName() + " is a more valuable piece then " + $p1.getPieceName()));
end


rule "Is Attacking An Undefended piece"
salience -99
    when
        IsAttacking($p1, $p2;)
        not IsDefending($p3, $p2;)
    then
        insertLogical(new CanCapture($p1, $p2, $p2.getPieceName() + " is not defended"));
end
```

- Pravila za odigravanje poteza – Pravila pomoću kojih se ažurira stanje table nakon odigranih poteza od strane igrača ili stockfisha. Kako su relacije insertovane logički automatski će se obrisati iz memorije ako nakon poteza uslovi za njihovo postojanje više ne važe.

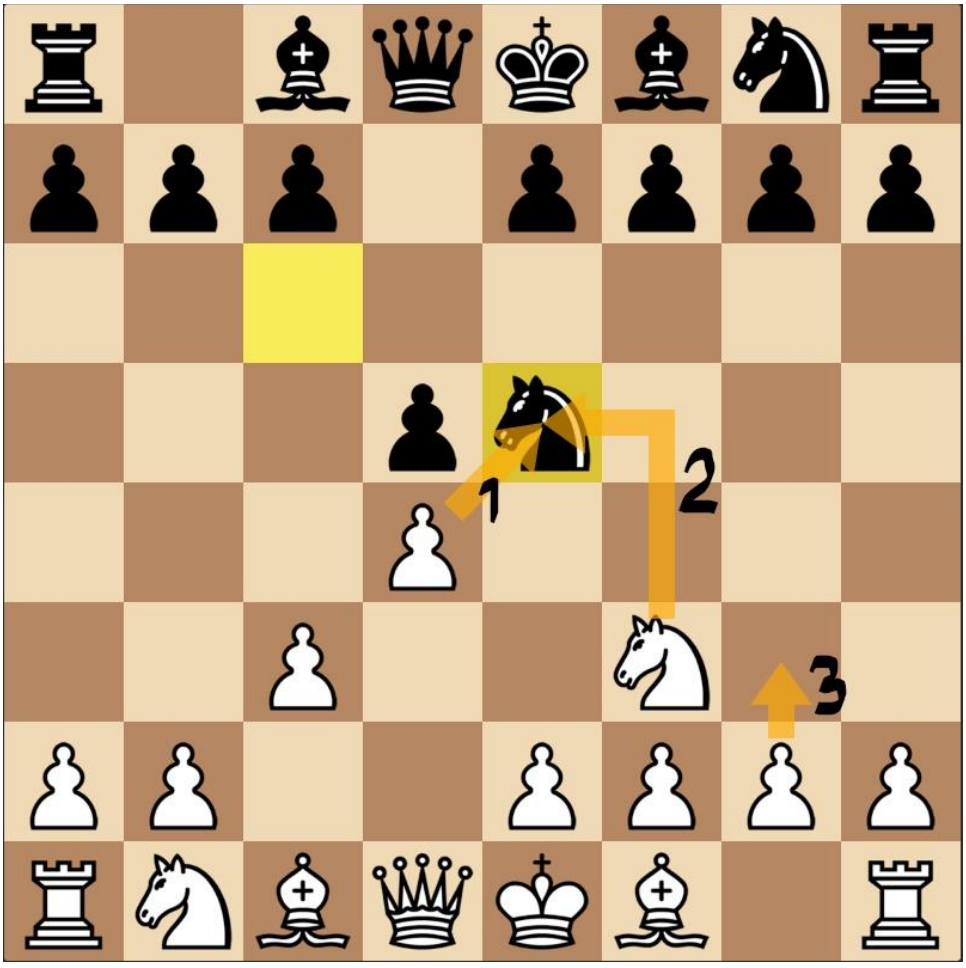
```
rule "Make Capture"
    lock-on-active
    when
        $b : Board($boardNum : boardNum)
        $sm : StockMove($fromSquare : fromSquare, $toSquare : toSquare, $san : san)
        $fromS : Square(squareText.equals($fromSquare), $piece1 : piece, boardNum == $boardNum)
        $p1 : Piece(this.equals($piece1))
        $toS : Square(squareText.equals($toSquare), $piece2 : piece, boardNum == $boardNum)
        $p2 : Piece(this.equals($piece2))
    then
         modify($toS){setPiece($p1)}
         modify($fromS){setPiece(null)}
         retract($p2)
end

rule "Make Move"
    lock-on-active
    when
        $b : Board($boardNum : boardNum)
        $sm : StockMove($fromSquare : fromSquare, $toSquare : toSquare, $san : san)
        $fromS : Square(squareText.equals($fromSquare), $piece1 : piece, boardNum == $boardNum)
        $p1 : Piece(this.equals($piece1))
        $toS : Square(squareText.equals($toSquare), piece == null, boardNum == $boardNum)
    then
        modify($toS){setPiece($p1)}
        modify($fromS){setPiece(null)}
end
```

- Queries – Nakon izvršenih pravila quer-ji će se pozivati iz java koda da se prikupe sve potrebne informacije za pružanje feedbacka igraču
    - Primer : findCanCapture

```
query findCanCapture
  $cc : CanCapture($piece1, $piece2, $message;)
  $fromSquare : Square(piece == $piece1)
  $toSquare : Square(piece == $piece2)
end
```

Reprezentativni primer:

Za reprezentativni primer pogledaćemo sledeću šahovsku poziciju.



Stockfish je napravio potez sa crnim konjem i dobili smo 3 predloga za poteze.

Pravila će se okidati sledećim redosledom:

1. "Make Move" pravilo - ažuriraće položaj crnog konja na tabli. Ovo će pokrenuti pravila za kreiranje osnovnih relacija I obrisati logički insertovane relacije koje više ne važe (bitna je IsDeffending za crnog konja koja je postojala pre ovog poteza)

2. Od ovih, ključna pravila će biti:

    2.1 "Knight Attack UP LEFT" - kreiraće činjenicu IsAttacking(beli konj, crni konj)

    2.2 "White Pawn Attack Right" - kreiraće činjenicu IsAttacking(beli pijun, crni konj)

3. Nove činjenice će okinuti pravila za kreiranje kompleksne relacije:

    3.1 "Is Attacking An Undefended piece" - kreiraće činjenicu CanCapture(beli konj, crni konj) sa porukom :"White Knight can capture the Black Knight because the Black Knight is not defended"

3.2 "Is Attacking A Higher Value Piece" - kreiraće činjenicu CanCapture(beli pijun, crni konj) sa porukom “White Pawn can capture the Black Knight because the Black Knight is worth more than a pawn”

4. Nakon završetka svih pravila, Spring service će pozvati querije da prikupi potrebne informacije za feedback igraču I spojiti ih sa preporučenim potezima od stokfisha. Igrač će klikom na gore označene strelice moći da vidi poruke:

Strelica 1: “White Pawn can capture the Black Knight because the Black Knight is worth more than a pawn”

Strelica 2: “White Knight can capture the Black Knight because the Black Knight is not defended”

Strelica 3: “I don’t know” - Ipak je šah izuzetno kompleksna igra sa 10^40 mogućih pozicija i ova poruka će verovatno biti najčešća, pogotovu u kasnijim fazama igre.