Chess Tutor – semantic chess analyzer

**Motivacija :**

Usled trenutne epidemiološke situacije i globalnog karantina, igra Šah je 2020. godine 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 koji analiziraju odigrane partije šaha i pružaju feedback 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.
* Ako posmatramo alat za analiziranje partija trenutno najpopularnije online platforme za igranje šaha (chess.com) možemo lako primetiti da alat nije prilagođen za početnike. Prilikom analize, alat svakom odigranom potezu dodeljuje jednu od predefinisanih labela (blunder, mistake, inaccuracy, good move, best move) koja je ocena kvaliteta datog poteza i to je sav feedback koji pruža korisniku. Za iskusnije igrače je ovo sasvim u redu, jer oni mogu relativno brzo da sami shvate šta je kod datog poteza dobro/loše nakon što im se skrene pažnja, ali početnicima samo nastaju nova pitanja “Šta je uopšte kod ovog poteza dobro/loše?”.
* Ideja je napraviti ekspertski sistem koji analizira odigrane poteze u šahu i daje više uvida oko toga šta je dobro ili loše kod nekog konkretnog poteza.
* Igra šah je izuzetno kompleksna igra, sa više od 10^123 mogućih poteza, ipak tokom godina eksperti su uspeli da izdvoje neke uobičajene taktike i strategije. Strategije se odnose na dugoročne ciljeve koje igrač želi da postigne, dok su taktike šabloni koji ostvaruju neki kratkoročni cilj. Dalje se mogu i strategije i taktike podeliti na podoblike, obično na napadačke, odbrambene ili pozicione.

**---------------------------------------------------------------------------------------------**

* **NAPOMENA :** Smatram da bi bilo previše obimno za mene da pokušam da implementiram sve vrste strategija i taktika, zato bi voleo da se fokusiram samo na napadačke taktike kao i odbrambene taktike kojima se mogu izbeći napadačke. Time će moj sistem moći da prepozna definisane taktike i dati feedback da li je ta taktika upotrebljiva u datoj poziciji i zbog čega jeste/nije.

Trenutne taktike koje sam razmatrao:

* + Napadačke - “Remove defender”, “Fork”, “Skewer”, “Pin”...
  + Odbrambene - “Shield”, “Escape with check”, “Move to defend”, “Counter Attack”...
  + Takođe su potrebna jednostavnija pravila/taktike kao na primer “Should Capture” na kojima će se zasnivati gore navedene taktike.

----------------------------------------------------------------------------------------------

**Metodologija rada:**

* **Input** – Sistem će kao input primati šahovske pozicije date u FEN notaciji ili šahovske partije date u PNG notaciji.
* **Output** – Sistem će kao output pregledno prikazati pronađene taktike u poziciji, dati mišljenje da li je pronađena taktika korisna u datoj poziciji i objasniti zašto jeste/nije.
* **Baza znanja** – Ideja je da baza znanja bude semantički graf. Čvorovi grafa su: Square, Piece, Tactic... dok grane predstavljaju razne relacije među njima. **Osnovne relacije** koje će svaka pozicija imatu su:
  + Piece – [isOn] -> Square
  + Piece – [canMoveOn] -> Square
  + Piece – [isAttacking] -> Piece
  + Piece –[isDeffending] ->Piece
  + Piece – [isXraying] -> Piece
  + ...

Dalje sa dodavanjem taktika putem pravila pojaviće se relacije koje su šablona:

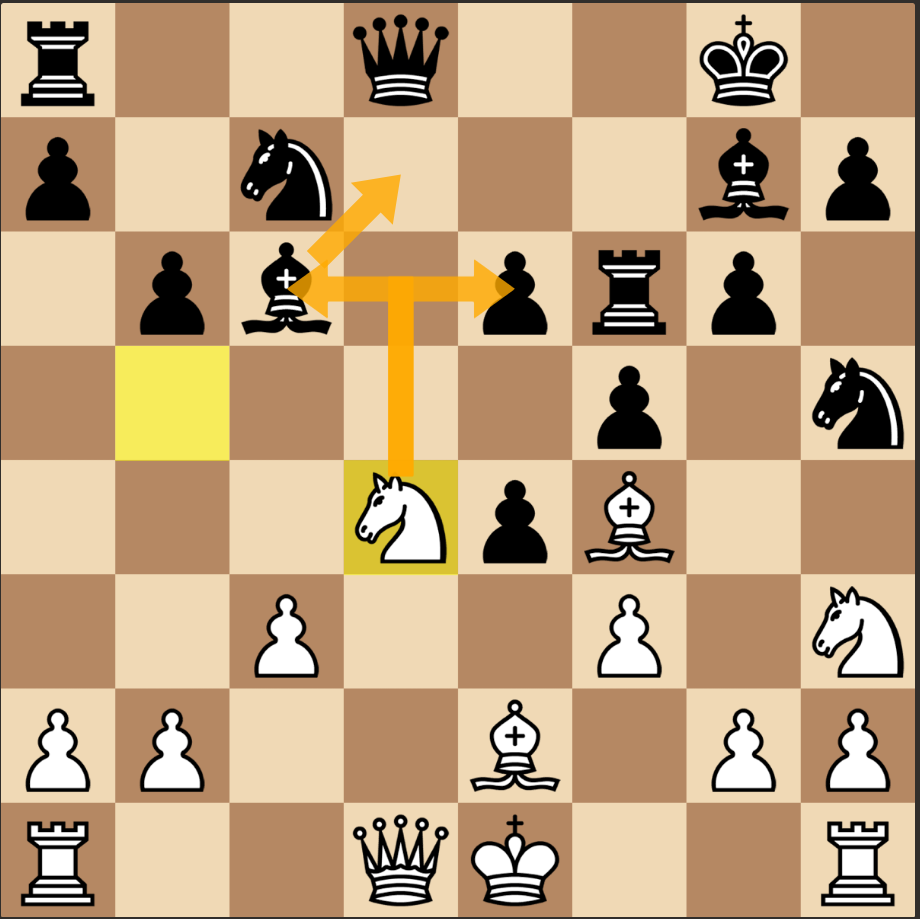
* + - Piece – [hasTactic] -> Tactic - [on] -> Piece
    - ...

Tri sloja pravila:

* Sloj pravila koji će kreirati Piece i Square čvorove i osnovne relacije među njima.
* Sloj pravila koji će na osnovu osnovnih relacija kreirati Tactic čvorove i hasTactic/on relacije.
* Sloj pravila koji će na osnovu svih Tactic čvorova rezonovati koje taktike su dobre/loše i vratiti feedback zbog čega.

**Reprezentativni primer:**

Za reprezentativni primer posmatraćemo taktiku “Fork”. To je taktika kada dovedete vašu figuru u poziciju da istovremeno napada 2 ili više protivničke figure što često dovodi do “besplatnog” osvajanja jedne. Ovu taktiku ćemo posmatrati na sledećem primeru:



Nakon prvog sloja pravila bitne činjenice za ovaj primer koje će se generisati će biti

* “WhiteKnight - [isAttacking] -> BlackPawn”
* “WhiteKnight - [isAttacking] -> BlackBishop”
* “BlackBishop - [canMoveOn]-> d7”

Na osnovu ovih činjenica “WhiteKnight - [isAttacking] -> BlackRook” i “WhiteKnight - [isAttacking] -> BlackBishop” aktiviraće se pravilo koje će generisati Tactic:Fork čvor I relacije [hasTactic] i [on]

* **“WhiteKnight - [hasTactic] -> Tactic:Fork - [on] - BlackPawn”**

**- [on] -BlackBishop”**

Ova nova činjenica I činjenica iz prvog koraka “BlackBishop - [canMoveOn]-> d7” aktiviraće pravilo koje će generisati odbrambenu taktiku **Tactic:Move to defend**

* **“BlackBishop - [hasTactic] -> Tactic:MoveToDefend - [on] -> BlackPawn”**

Na osnovu ove dve činjenice poslednji sloj će doći do zaključka I vratiti korisniku **“The fork is not a good tactic in this position because the black bishop can move to defend the black pawn”**.