HScraper - A Haskell library to parse, crawl and scrape webpages

Ayush Agarwal, Nishant Gupta, M.Arunothia

Indian Institute of Technology Kanpur

Aim

The aim of this project is to build a Haskell library that enables the user to correctly parse, crawl and scrape web-pages.

Motivation

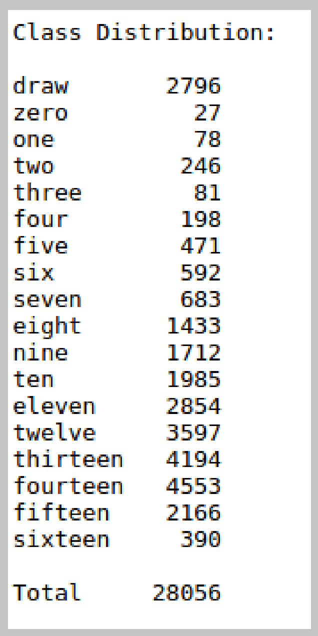
Introduction

Inductive Logic Programming has been defined as the intersection of Machine Learning and Logic Programming. The examples given to the learning system are expressed in a logical programming language such as prolog. Moreover, the conceptes which the learning system develops from the examples are also expressed in the same language. This feature of ILP has been used in this project because it enables us to get attribute values well defined in the language of logic only. [?]

Data Base

- Source
  https://archive.ics.uci.edu/ml/machine-learning-databases/chess/king-rook-vs-king/
- Format
  - White King file (column)
  - White King rank (row)
  - White Rook file
  - White Rook rank
  - Black King file
  - Black King rank
  - optimal depth-of-win for White in 0 to 16 moves,otherwise draw.

[?]



```
Class Distribution:

draw       2796
zero         27
one          78
two         246
three        81
four        198
five        471
six         592
seven       683
eight      1433
nine       1712
ten        1985
eleven     2854
twelve     3597
thirteen   4194
fourteen   4553
fifteen    2166
sixteen     390

Total     28056
```

Figure 1: Distribution of data

Method

- Progol is an implementation of Inductive Logic Programming used in computer science that combines "Inverse Entailment" with "general-to-specific search" through a refinement graph. "Inverse Entailment" is used with mode declarations to derive the most-specific clause within the mode language which entails a given example. This clause is used to guide a refinement-graph search. [?]
- The attributes used in the body mode of the check mate configuration is mentioned in the table and the figure.

Other Trials,Learnings and Future Improvements

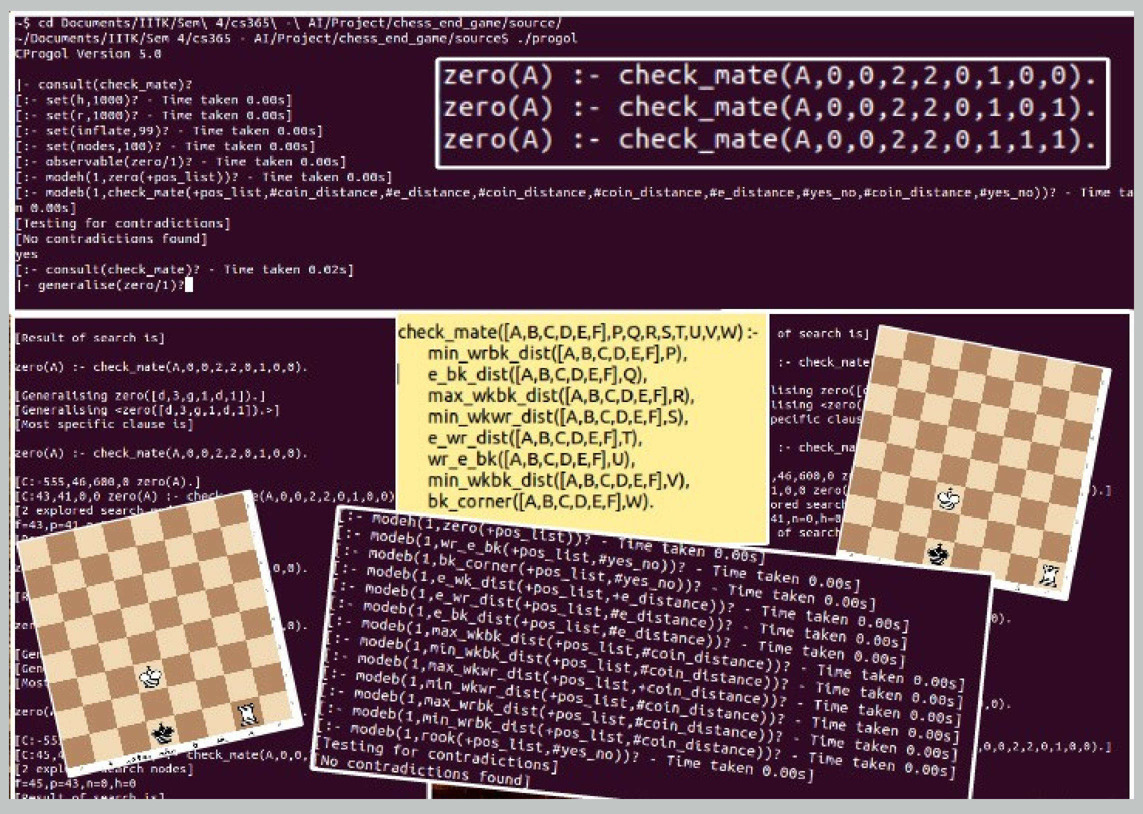
- Tried this method to give a rule for the draw configuration. The number of draw positions being too high could not produce a good result. One of the main reasons for failure is the limit on the number of attributes I am able to define and the lack intuition towards the draw configuration.
- Search heuristics and pruning strategies could be added to make this approach extensible.
- This approach could be clubbed along with the stage-wise categorisation mentioned in the paper *Learning long-term chess strategies from databases* [?].

Results

Attribute	Value
Minimum File/Rank difference between White Rook and Black King	0
Distance from edge for Black King	0
Maximum File/Rank difference between White King and Black king	2
Minimum File/Rank difference between White King and White Rook	2
Distance from edge for White Rook	0
Is White Rook on same edge as Black King?	1
Minimum File/Rank difference between White King and Black King	0/1
Is black King on the corner?	0/1

Table 1: Check Mate (KRK) Rules Learnt

Results: Check Mate (KRK) Rules Learnt



```
~$ cd Documents/IITK/Sem\ 4/cs365\ -\ AI/Project/chess_end_game/source/
~/Documents/IITK/Sem 4/cs365 - AI/Project/chess_end_game/source$ ./progol
CProgol Version 5.0

|- consult(check_mate)?
[:- set(h,1000)? - Time taken 0.00s]
[:- set(r,1000)? - Time taken 0.00s]
[:- set(inflate,99)? - Time taken 0.00s]
[:- set(nodes,100)? - Time taken 0.00s]
[:- observable(zero/1)? - Time taken 0.00s]
[:- modeh(1,zero(+pos_list))? - Time taken 0.00s]
[:- modeb(1,check_mate(+pos_list,#coin_distance,#e_distance,#coin_distance,#coin_distance,#e_distance,#yes_no,#coin_distance,#yes_no))? - Time ta
n 0.00s]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(check_mate)? - Time taken 0.02s]
|- generalise(zero/1)?
```

```
zero(A) :- check_mate(A,0,0,2,2,0,1,0,0).
zero(A) :- check_mate(A,0,0,2,2,0,1,0,1).
zero(A) :- check_mate(A,0,0,2,2,0,1,1,1).
```

```
[Result of search is]

zero(A) :- check_mate(A,0,0,2,2,0,1,0,0).

[Generalising zero([d,3,g,1,d,1]).]
[Generalising <zero([d,3,g,1,d,1]).>]
[Most specific clause is]

zero(A) :- check_mate(A,0,0,2,2,0,1,0,0).

[C:-555,46,600,0 zero(A).]
[C:43,41,0,0 zero(A) :- check_mate(A,0,0,2,2,0,1,0,0)
[2 explored search
f=43,p=41
```

```
check_mate([A,B,C,D,E,F],P,Q,R,S,T,U,V,W) :-
    min_wrbk_dist([A,B,C,D,E,F],P),
    e_bk_dist([A,B,C,D,E,F],Q),
    max_wkbk_dist([A,B,C,D,E,F],R),
    min_wkwr_dist([A,B,C,D,E,F],S),
    e_wr_dist([A,B,C,D,E,F],T),
    wr_e_bk([A,B,C,D,E,F],U),
    min_wkbk_dist([A,B,C,D,E,F],V),
    bk_corner([A,B,C,D,E,F],W).
```

```
[:- modeh(1,zero(+pos_list))? - Time taken 0.00s]
[:- modeb(1,wr_e_bk(+pos_list,#yes_no))? - Time taken 0.00s]
[:- modeb(1,bk_corner(+pos_list,#yes_no))? - Time taken 0.00s]
[:- modeb(1,e_wk_dist(+pos_list,+e_distance))? - Time taken 0.00s]
[:- modeb(1,e_wr_dist(+pos_list,#e_distance))? - Time taken 0.00s]
[:- modeb(1,e_bk_dist(+pos_list,#e_distance))? - Time taken 0.00s]
[:- modeb(1,max_wkbk_dist(+pos_list,#coin_distance))? - Time taken 0.00s]
[:- modeb(1,min_wkbk_dist(+pos_list,#coin_distance))? - Time taken 0.00s]
[:- modeb(1,max_wkwr_dist(+pos_list,+coin_distance))? - Time taken 0.00s]
[:- modeb(1,min_wkwr_dist(+pos_list,#coin_distance))? - Time taken 0.00s]
[:- modeb(1,max_wrbk_dist(+pos_list,#coin_distance))? - Time taken 0.00s]
[:- modeb(1,min_wrbk_dist(+pos_list,#coin_distance))? - Time taken 0.00s]
[:- modeb(1,rook(+pos_list,#yes_no))? - Time taken 0.00s]
[Testing for contradictions]
[No contradictions found]
```

Figure 2: Check mate (krk) configurations solved fig:ref - http://en.lichess.org/editor

Conclusion

- The rules obtained for check-mate condition is very intuitive. It can easily be understood and contemplated by humans. Do refer results column.

References

Acknowledgments

I thank my senior Mr. Ashudeep Singh and Prof. Amitabha Mukherjee for helping me through this project.

Contact Information

- Email: arunothi@iitk.ac.in
- Roll No: 13378