Optimization and Machine Learning

Lecture 2: IP models for regression & knowledge graphs

Sanjeeb Dash
IBM Research
(co-lecturer Parikshit Ram)

JPOC 13 Summer School, June 26-28, 2023
University of Clermont-Ferrand

Lecture 2 Outline

► Rule-based regression

► Knowledge graphs
- Rule-based methods
- Embedding-based methods

► Linear Programming Formulation
- Column Generation Technique

► Results

Regression

- Features: $X_1, \ldots, X_m$
- Data: $\{(x_i; y_i) : i \in 1, 2, \ldots, n\}$ where $x_i \in \mathbb{R}^m$.
- Label $y_i \in \mathbb{R}$ ($\{0, 1\}$ for logistic regression)
- Feature X_j is either numeric or categorical.
- Goal: Find function f such that $y_i \approx f(x_i)$.

Regression contd.

Linear regression: $f \equiv c_1 X_1 + c_2 X_2 + \cdots + c_m X_m$

Rule-based Linear regression: Add new features/variables corresponding to rules

Related column generation results

► **Rule-based Linear regression**: Eckstein, Kagawa, Goldberg ’17, ’19

► **Rule-based Logistic regression**: Wei, Dash, Gao, Günlük ’19

▷ Goal: Given features X_j, find function $f(X) = \beta_0 + \sum_{j=1}^{m} \beta_j X_j$ such

$$P(y_i = 1 \mid x_i) \approx \frac{1}{1 + e^{-f(x_i)}}$$

- Classify new data point x_i with label 1 if $\frac{1}{1+e^{-f(x_i)}} > \frac{1}{2}$, 0 otherwise.

▷ Find $f(X)$ by minimizing the negative log-likelihood on the training data (use ℓ_1 regularization for sparsity)

$$\min_{\beta} \ \frac{1}{n} \sum_{i=1}^{n} \left[\log(1 + e^{(-1)^{y_i} \sum \beta_j x_{ij}}) \right] + \sum_{j=0}^{m} \lambda_j |\beta_j|.$$

Rule-based Logistic regression..

$$\min_{\beta} \quad \frac{1}{n}\sum_{i=1}^{n}\left[\log(1+e^{(-1)^{y_i}\sum \beta_j x_{ij}})\right] + \sum_{j=0}^{t}\lambda_j|\beta_j|.$$

Consider a "missing" feature k ($\beta_k = 0$). To find the effect of increasing β_k, compute partial derivative of ith objective term w.r.t. increasing β_k

$$\frac{(-1)^{y_i}\ a_{ik}\ e^{(-1)^{y_i} f(x_i)}}{1+e^{(-1)^{y_i} f(x_i)}} \Rightarrow$$

Partial derivative w.r.t. increasing β_k =

$$\frac{1}{n}\sum_{i=1}^{n} r_i a_{ik} + \lambda_k.$$

$\hat{y}(x_i)$ is the predicted value for data point x_i, and $r_i = \hat{y}(x_i) - y_i$

Pricing IP

reduced cost of clause incl. complexity penalty

$$\min_{z,a} \pm\frac{1}{N} \sum_{i=1}^{N} r_i a_i + \lambda_0 + \lambda_1 \sum_{j=1}^{d} z_j$$

clause acts as conjunction of features

$$a_i + \sum_{j:x_{ij}=0} z_j \geq 1, \qquad a_i \geq 0, \qquad i: y_i = 0$$

$$a_i + z_j \leq 1, \qquad i: y_i = 1, \qquad j: x_{ij} = 0$$

$$z_j \in \{0,1\}, \qquad j = 1, \ldots, d$$

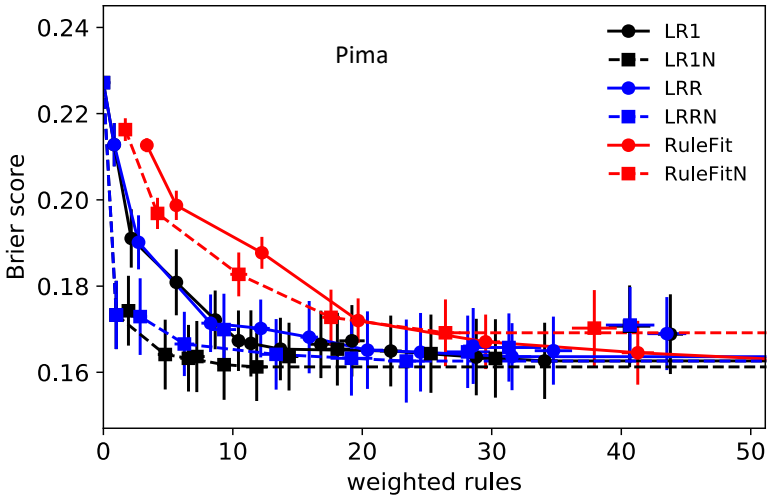
whether to select feature j

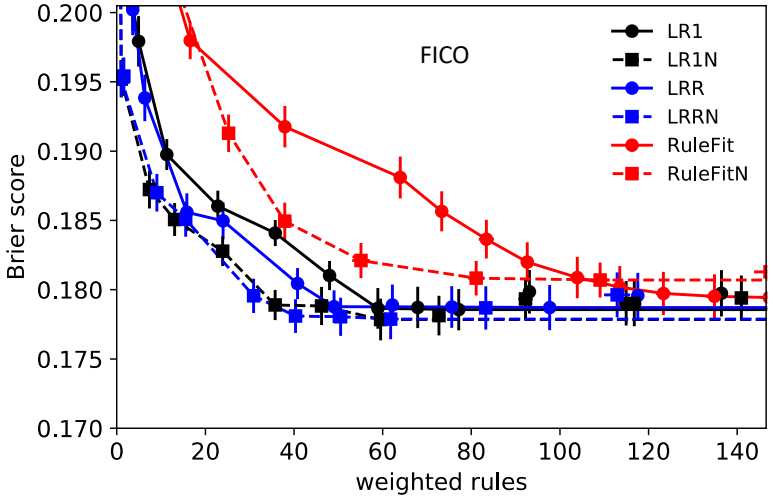
Pricing IP Contd.

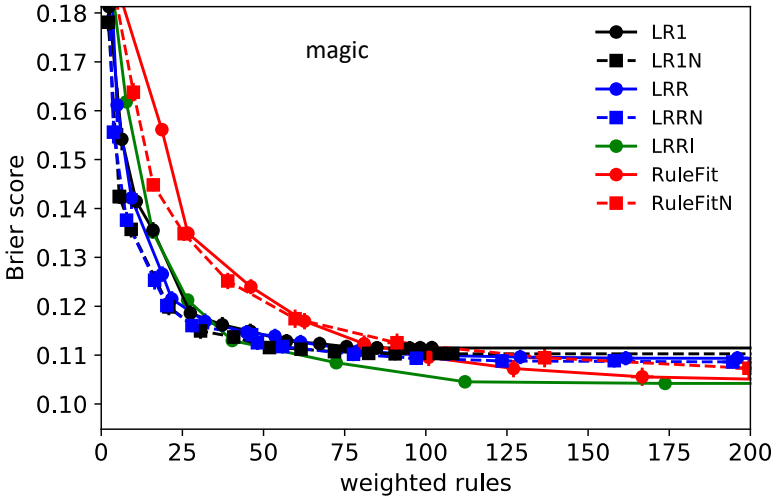
All data is assumed to be binary and in the form $(x_i, y_i) \in \{0,1\}^m \times \{0,1\}$ for logistic regression. Non-binary x is binarized as in lecture 1.
- x_{ij} is the value of x_i for feature j.
- a_i, z_j are binary variables; z_j is 1 iff feature j is chosen to be part of a rule/clause
- a_i is the value given by chosen rule/clause to data point i

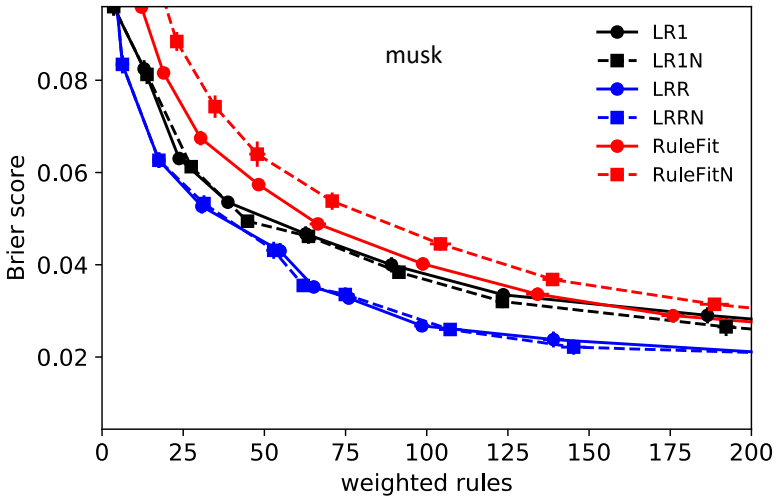
The $\pm$ term in the objective means we solve two optimization problems to get the best clauses for increasing/decreasing β_k from 0.

Results


Pima
LR1
LR1N
LRR
LRRN
RuleFit
RuleFitN
Brier score
weighted rules
0.24
0.22
0.20
0.18
0.16
0
10
20
30
40
50


FICO
LR1
LR1N
LRR
LRRN
RuleFit
RuleFitN
Brier score
weighted rules
0.200
0.195
0.190
0.185
0.180
0.175
0.170
0
20
40
60
80
100
120
140


magic
LR1
LR1N
LRR
LRRN
LRRI
RuleFit
RuleFitN
Brier score
weighted rules
0.18
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0
25
50
75
100
125
150
175
200


musk
LR1
LR1N
LRR
LRRN
RuleFit
RuleFitN
Brier score
weighted rules
0.08
0.06
0.04
0.02
0
25
50
75
100
125
150
175
200

Results

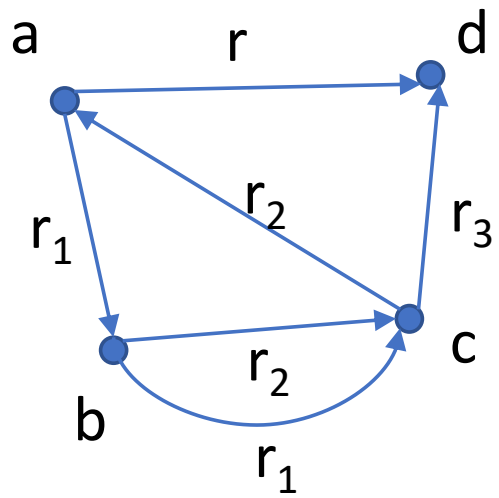
method	LRR	LRRN	RuleFit	RuleFitN	GBM	SVM
logistic regression mean rank	4.1	**3.6**	4.8	**3.6**	5.3	4.0
linear regression mean rank	4.9	**3.0**	4.5	3.5	3.4	5.0

Logistic/Linear Rule Regression with CG (LRR, LRRN) is highly competitive when tuned to maximize performance and uses 2-4 times fewer rules than RuleFit [Friedman & Popescu, 2008]

Knowledge Graph completion

Knowledge Graph (KG): Directed node/edge-labeled multigraph; each edge is a "fact"; edge labels represent binary relations between nodes.

Example: (a, r_1, b) is a fact or $r_1(a, b)$ is true
a, b, c, d could be individuals,
r, r_1, r_2 could *son_of, brother_of, related_to*



Knowledge graphs often have missing (and incorrect) facts.

KG completion problem:
Find missing facts e.g., *(b, brother_of, a), (c, brother_of, a)*

Popular methods: Rule based & Embedding based

Rules

Example: $(X, \text{son\_of}, Y) \wedge (Y, \text{son\_of}, Z) \rightarrow (X, \text{grandson\_of}, Y)$

KG Completion Problem: Answer query $(a, r, ?)$

Standard Approach:

1. Learn rule-based function $f_r(X, Y)$ that gives high scores to likely facts (X, r, Y) where X, Y are nodes in the graph, and r is an edge-label

2. Answer query $(a, r, ?)$ by finding x such that $f_r(a, x)$ has highest score.

3. If the correct answer is b, measure accuracy by average rank/reciprocal rank of b (MR/MRR)

Prior work

Kok, Domingos ’05, Richardson, Domingos ’06 – Markov Logic Networks
Yang, Yang, Cohen ’17 (NeuralLP) – Neuro-symbolic methods
Rochstätel, Riedel ’17 (NTP) – „
Sadeghian, Armandpour, Ding, Wang ’19 (DRUM) – „
Evans, Grefenstette ’18 – Differential ILP
Das et al. ’18 (Minerva) – Reinforcement Learning
Qu et. al. ’21 (RNNLogic) – RNN + Probabilistic methods
Meilicke et. al. ’19 (AnyBURL) – Data mining
Teru, Denis, Hamilton ‘20 (GraIL) – Subgraph reasoning

Advantages: (1) Inductive reasoning is possible.
(2) Interpretable models when few rules are generated.

Drawbacks: (1) Lower levels of accuracy compared to embedding methods
(2) Current methods do not scale

Embedding based methods

Approach: Find $v_a \in \mathbb{R}^k$ for each node a and a mapping $T_r : \mathbb{R}^k \to \mathbb{R}^k$ for each relation r such that the score $||T_r(v_a) - v_b||$ is small for each fact (a, r, b).

Bordes, Usunier, Garcia-Duran, Weston, Yakhnenko '13 (TransE)

Yang, Yih, He, Gao, Deng '15 (DistMult)

Trouillon, Welbl, Riedel, Gaussier, Bouchard '16 (ComplEx)

Dettmers, Pasquale, Pontus, Riedel '18 (ConvE)

Lacroix, Usunier, Obozinski '18 (ComplEx-N3)

Sun, Deng, Nie, Tang '19 (RotatE)

Advantages: (1) Reasonable accuracy
(2) Scalable

Drawbacks: (1) Not effective for inductive reasoning
(2) Model is not interpretable.

Our work

Goals: Develop a scalable, rule-learner returning compact rule sets

- Interpretability is an explicit goal, and we return low-complexity rules

- We trade off complexity versus accuracy

- Scalability is attained by solving linear programming models instead of non-convex models

Our approach

Approach: Learn few (FOL) rules $R_1, \ldots, R_p$ and positive weights $w_1, \ldots, w_p$ where each R_i has the form

$$r_1(X, X_1) \wedge r_2(X_1, X_2) \wedge \cdots \wedge r_l(X_{l-1}, Y) \rightarrow r(X, Y)$$

where $r_1, \ldots, r_l$ are relations in G.

Length of this rule is l; left-hand-side is the clause $C_i : V \times V \rightarrow \{0, 1\}$

The learned prediction/scoring function $f_r : V \times V \rightarrow \mathbb{R}_+$ for r is:

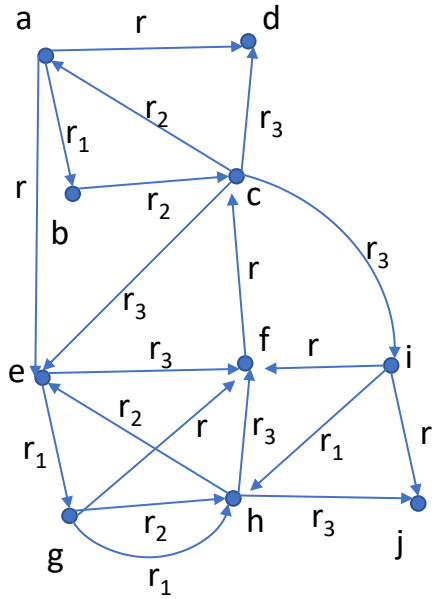
$$f_r(X, Y) = \sum_{i=1}^{p} w_i C_i(X, Y) \;\; \forall X, Y \in V$$

Details

$C_1(X, Y)$

Rule $r_1(X, X_1) \wedge r_2\ (X_1, X_2) \wedge r_3\ (X_2, Y) \rightarrow r(X, Y)$ and associated clause-edge vector

KG:
a-j are entities
r, r_1, r_2, r_3 are relations



edge	$r_1 \wedge r_2 \wedge r_3$	r
(a,d)	1	**1**
(a,e)	1	**1**
(f,c)	0	**1**
(g,f)	1	**1**
(i,f)	1	**1**
(i,j)	0	**1**
(e,f)	**1**	0
(a,i)	**1**	0
(e,j)	**1**	0

a_{i1}

$r_1(a, b) \wedge r_2\ (b, c\ ) \wedge r_3\ (c, d)$ is true and $r(a, d)$ is true

positive instances:
edges in KG = E_r

negative instances:
non-edges (sample)

LP to learn KG rules

Minimize error for weighted collection of rules:

loss on positive instances — loss on negative instances

$$\min_{w,\xi} \sum_{i:y_i=1} \xi_i + \tau \sum_{k \in K} \text{neg}_k w_k$$

cover positives → $\xi_i + \sum_{k \in K} a_{ik} w_k \geq 1, \quad \xi_i \geq 0, \quad (t_i, h_i) \in E_r$

value of scoring fn.

complexity bound → $\sum_{k \in K} c_k w_k \leq C$

select clause *k* or not → $w_k \in [0,1], \qquad k \in K$

Model details

- E_r = set edges labeled by r, and (t_i, h_i)= th edge in E_r
- w_k variable gives weight for rule k; $w_k > 0$ implies rule k is chosen
- a_{ik} is a constant $= C_k(t_i, h_i)$
- c_k is a constant = 1+ rule length
- C is a parameter bounding weighted complexity of chosen rules
- τ is a parameter, neg_k is a constant

Modeling – Use all positive facts for a relation + sample some negative facts for the LP model

Algorithmic issues – Use simple shortest path heuristics to find relational paths, and associated rules – Iterate over different values of tau and complexity

Code available at: https://github.com/IBM/LPRules

Column generation

Step 0 – Fix an initial complexity and tau value

Step 1 – Use simple heuristics to create an initial collection of rules

Step 2 – Set up LP model and solve it

Step 3 – Obtain dual values of LP model

Step 4 – Dual values indicate which facts are “well-covered” and which are not. Heuristically generate new rules that “cover” facts that are not well-covered.

Step 5 – Repeat Steps 2 – 4 till termination criterion

Sizes of datasets

Datasets	# Relations	# Entities	# Train	# Test	# Valid
Kinship	25	104	8544	1074	1068
UMLS	46	135	5216	661	652
FB15k-237	237	14541	272115	20466	17535
WN18RR	11	40943	86835	3134	3034
YAGO3-10	37	123182	1079040	5000	5000

Neuro-symbolic methods take a long time on FB15k-237 and cannot handle YAGO3-10

Experiments (accuracy)

Datasets	ComplEx-N3	AnyBURL	NeuralLP	DRUM	RNNLogic	LPRules
Kinship	0.889	0.626	0.652	0.566	*0.687*	**0.746**
UMLS	0.962	**0.940**	0.750	0.845	0.748	*0.869*
FB15k-237	0.362	0.226	0.222	0.225	†**0.288**	*0.255*
WN18RR	0.469	*0.454*	0.381	0.381	0.451	**0.459**
YAGO3-10	0.574	**0.449**				**0.449**

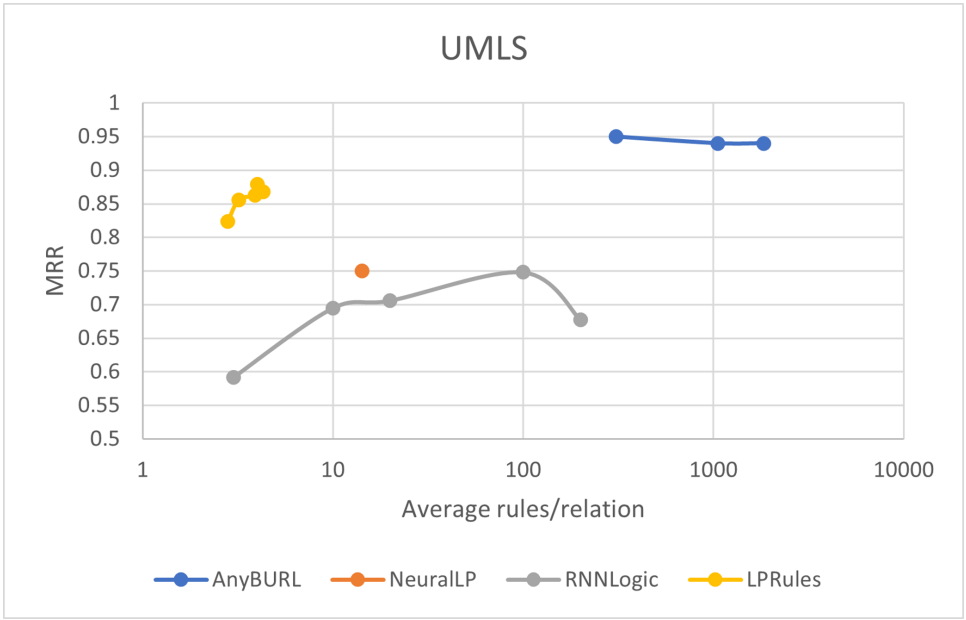
† We could not run RNNLogic on FB15k-237 and report numbers taken from Qu et al. (2021)

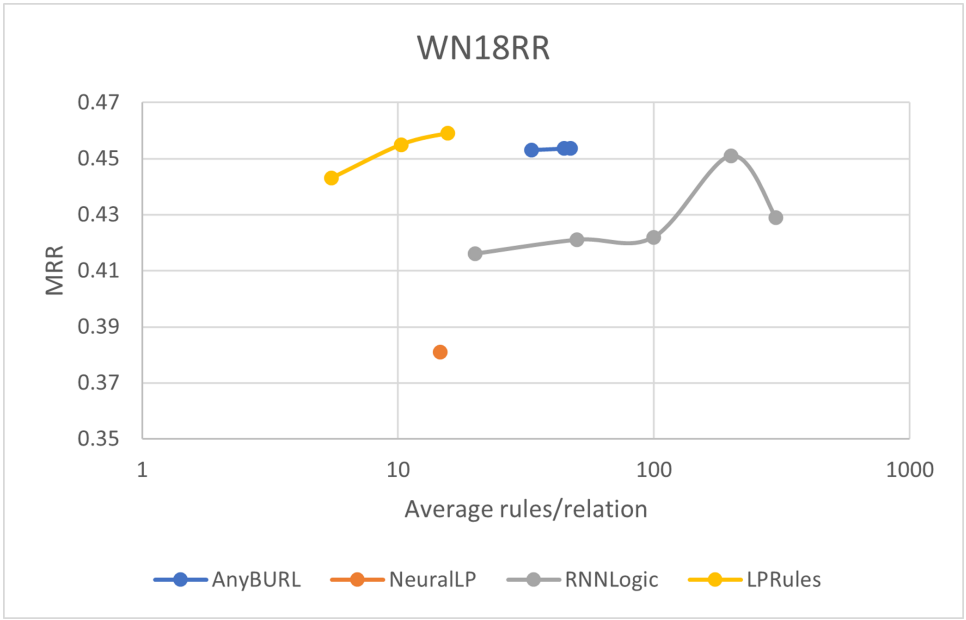
Running time + number of rules

Datasets	ComplEx-N3	AnyBURL	NeuralLP	DRUM	RNNLogic	LPRules
Kinship	0.889	0.626	0.652	0.566	*0.687*	**0.746**
UMLS	0.962	**0.940**	0.750	0.845	0.748	*0.869*
FB15k-237	0.362	0.226	0.222	0.225	†**0.288**	*0.255*
WN18RR	0.469	*0.454*	0.381	0.381	0.451	**0.459**
YAGO3-10	0.574	**0.449**				**0.449**

Avg number of rules per relation and wall clock running time on a 60 core machine

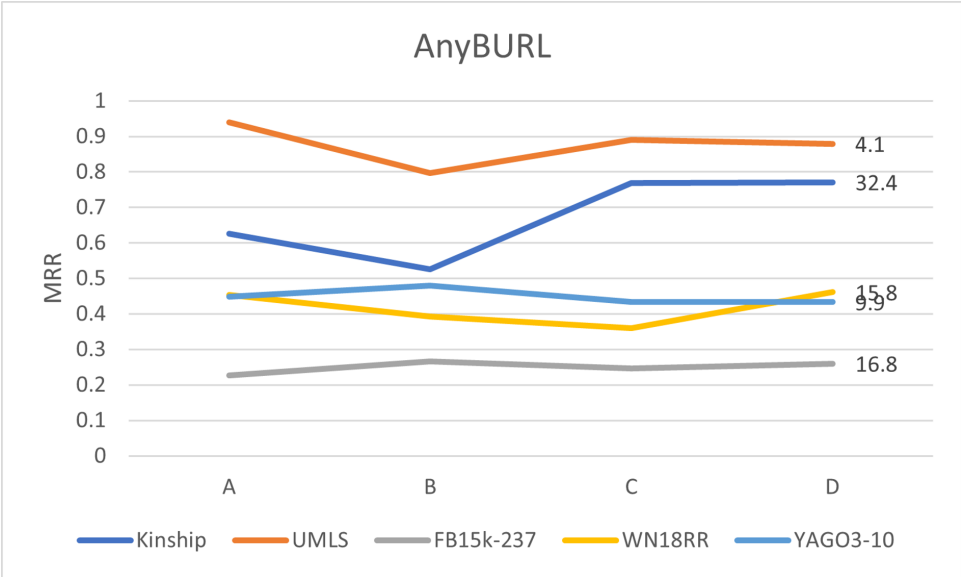
Accuracy versus Complexity tradeoff

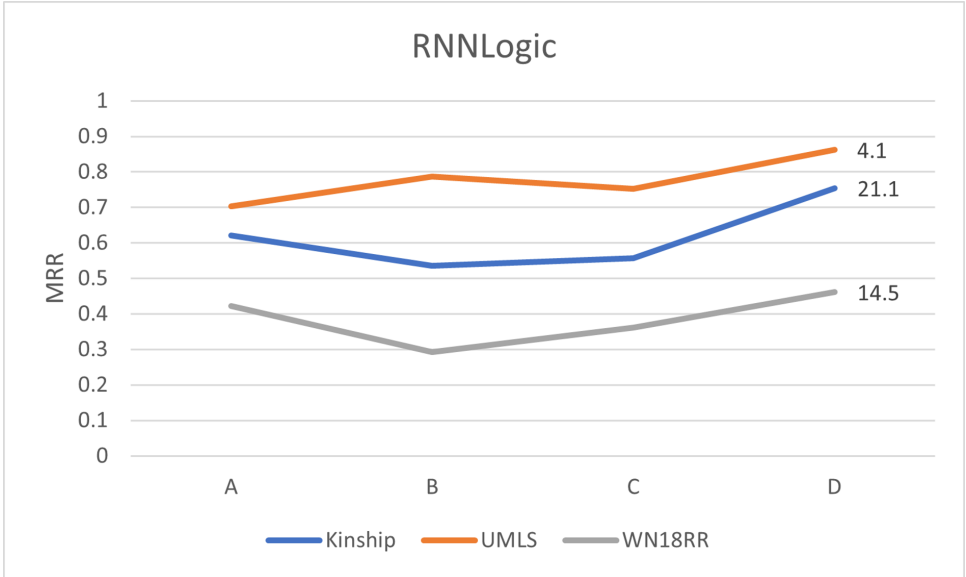




Change in MRR with change in average rules per relation

LPRules + rules from other codes





MRR values using rules generated by AnyBURL and RNNLogic (experiments A-D)

A – Use other rule-based code
B – Take rules and weights and use in our prediction function
C – Recalculate weights using complexity bound
D – Add our rules and recalculate weights

References

1. D. Wei, S. Dash, O. Gunluk, T. Gao. Generalized linear rule models, ICML 2019, PMLR 97:6687-6696. Supplementary material
2. S. Dash, J. Goncalves, Rule induction in knowledge graphs using linear programming, AAAI 2023.