|  |  |  |
| --- | --- | --- |
| Dataset | Settings | Data |
| UW | yes | yes |
| MovieLens | No (aggregators?) | No (random selection) |
| OMOP | No | Randomly generated using simulator |
| WebKB | yes | Yes. Proximity dataset is not available. |

MovieLens: cannot reproduce because “random selection” [cite where], added “Like” and no settings. But same source.

Comparison with RDN-Boost. We would first like to note that the main point of our evaluation is *not* that RDN-Bayes are generally superior to RDN-Boost. These are quite different model classes with different strengths and weaknesses. Ultimately we believe the best system would be a combination, as we outlined in Section xxx. For comparison, in the propositional setting it would be unusual to compare boosted regression trees with Bayesian network classifiers. For us the important point is that the accuracy of our novel method is competitive on benchmark datasets with the state of the art.

Reproducing RDN-Boost results. The main difference between the experiments we report and those in the RDN-Boost paper by Natarajan et al. is that they report accuracy results for one or two target predicates, whereas we report the average accuracy over *all* predicates. As the other reviewer notes, dependency networks are a generative model that jointly models all predicates, so considering all predicates seemed to us to be an even stronger evaluation.

That said, we could of course report the results for the target predicates selected in the RDN-Boost paper separately. However, although we made a lot of effort to reproduce the results in the RDN-Boost paper by Natarajan et al. we were unsuccessful, for reasons we detail now. While we are happy to add comparisons on more datasets, the concern of the reviewer seemed not to be that our n datasets were an insufficient number, but with reproducing the RDN-Boost results exactly. Given the difficulties with reproducibility that we detail below, we added only one more dataset to address the scalability question.

Details on RDN-Boost reproducibility. Our paper reports measurements that were obtained by using the code posted by the RDN-Boost creators, available from http://pages.cs.wisc.edu/~tushar/Boostr/down.html. We followed the instructions at <http://pages.cs.wisc.edu/~tushar/Boostr/tutorial.html> . On some datasets, the results of their posted code are different from what they report in the RDN-Boost paper. We contacted the RDN-Boost creators about this discrepancy but have received no reply.

Reproducibility details for each dataset. The datasets with individual prediction target (as opposed to collective classification) in the RDN-Boost paper are UW, MovieLens, OMOP, and WebKB.

*UW*. We used the data posted at <http://pages.cs.wisc.edu/~tushar/Boostr/datasets/uw.zip>. This includes data and a “background file” with mode declaration. The biggest difference is that we report accuracy results for *all predicates in the dataset,* whereas the Natarajan et al. paper reports only results for the “AdvisedBy” target. We’d be happy to report results for “AdvisedBy” separately.

However, there is still a reproducibility issue for “AdvisedBy”. Running this setup produces the following measurements for the target “AdvisedBy”, which is the target chosen by Natarajan et al. (Section 4.1).

AUC ROC = 0.982

AUC PR = 0.400

CLL = -0.132

[Actual output for folder 0 from RDN-Boost code]

[zqian@cs-oschulte-02 scripts]$ ./run\_uw\_rdn.sh

....

% Computing Area Under Curves.

%Pos=16

%Neg=2385

%LL:-6.574941231859055

%LL:-293.39050475670047

% Running command: java -jar ../auc.jar ../data/uw-cse\_rdn/test0\_advisedby/AUC/aucTemp.txt list 0.0

% WAITING FOR command: java -jar ../auc.jar ../data/uw-cse\_rdn/test0\_advisedby/AUC/aucTemp.txt list 0.0

% DONE WAITING FOR command: java -jar ../auc.jar ../data/uw-cse\_rdn/test0\_advisedby/AUC/aucTemp.txt list 0.0

% F1 = 1.0

% Threshold = 0.44207531332577377

% AUC ROC = 0.975708

% AUC PR = 0.351768

% CLL = -0.122195

% Precision = 0.131579 at threshold = 0.500

% Recall = 0.937500

% F1 = 0.230769

% Total inference time (20 trees): 4.432 seconds.

This is the output for folder 0, the other values are similar. The AUC-ROC is similar to that in Table 1 of Natarajan et al., but the AUC-PR is much lower: 0.36 vs. 0.95. We don’t know why. As mentioned, we received no reply to a query about this from the creator of the RDN Boostr package.

*MovieLens*. Here too the main difference is that we report accuracy results for on all binary predicates (e.g., gender) *in the dataset,* whereas the Natarajan et al. paper reports only results for the Likes target. We’d be happy to report results for Likes separately. However, even so it would be difficult to reproduce the results reported in Natarajan et al, for the following reasons.

1. Natarajan et al. subsampled a set of 100 users and 603 movies randomly. This subset is not posted so we cannot get the exact same data as theirs.   
   [We can make a small dataset like that]
2. They report using four aggregators in the paper. However, the posted RDN-code does not support the use of aggregators.

*OMOP*. As Natarajan et al. state, “we used the OMOP simulator to generate a dataset of 10,000 patients that included records of drugs and diagnoses”. This random dataset is not available.

*WebKB*. The Boostr website posts a version of this dataset at <http://pages.cs.wisc.edu/~tushar/Boostr/datasets/webkb.zip>, for the target “faculty” (not “Student” as discussed in Natarajan et al. Sec. 4.3). Running this setup produces the following measurements

AUC ROC = 1

AUC PR = 1

CLL = -0.04

In other words, perfect classification performance. We looked at the rules to see how this performance is achieved. It’s because URLs are partitioned into Faculty and Student URLs. When classifying a URL as Faculty or not, the facts.txt background file includes the information about the Student status. For instance, to classify

faculty(ahttpwwwcsutexaseduusersxfeng)

the program has access to the background fact that

student(ahttpwwwcsutexaseduusersxfeng)

The rules learned essentially say that if a URL is a student URL, it’s not a faculty URL, and otherwise it’s a Faculty URL. Here is an example of a regression tree learned.

% FOR faculty(A):

% if ( student(A) )

% then return -0.14185106490048832; // std dev = 0.000, 186.000 (wgt'ed) examples reached here. /\* #neg=186 \*/

% else if ( sameperson(A, A) )

% | then return 0.8581489350995111; // std dev = 4.94e-08, 107.000 (wgt'ed) examples reached here. /\* #pos=107 \*/

% | else return -0.1418510649004878; // std dev = 0.000, 17.000 (wgt'ed) examples reached here. /\* #neg=17 \*/

We ran our RDN-Bayes system on the same data and also achieved perfect classification. With all due respect to the RDN-Boost group, this does not seem a very interesting finding. We’d be happy to add classification over *all* predicates in the WebKB dataset as a comparison if that strengthens the paper.

May 4th, 2015

MovieLens (100 users, 80% training, 20% test)

|  |  |  |
| --- | --- | --- |
|  | AUC-PR | CLL |
| RDN\_Boost | 0.59 | -0.65 |
| MLN\_Boost | 0.56 | -1.48 |
| Bayes\_Boost | **1.0** | **-0.43** |