Magellan User Experience Survey

Clarence Cheung {kcheung6@wisc.edu}
Jin Ruan {jruan3@wisc.edu}

In this report, we present the result of a user experience survey we have done on Magellan. The main study material of the survey is the explanation reports of blocking stage and matching stage from all the teams in CS784 this semester. Due to the large number of teams in class, we sampled 12 out of 24 teams (including our own team) and went through their blocking and matching explanations in order to extract useful information about Magellan user experience. Combining these ideas from other teams and our own experience, we give a summary of feedbacks below.

<u>Summary</u>

1. Exception Handling

Both Ashish's team and Ali's team mentioned better exception handling is necessary to help user understand why something failed. In Ali's team's case, they encountered some generic exceptions like 'Column qualifies to be more than one type'. This error is thrown in attributeutils.py, which contains all the relevant information about which columns are colliding and what are their respective types. And they suggested that this information should be included in the stacktrace that bubbles up.

2. Documentation for Magellan

Several teams suggested some kinds of improvements on the user manual of Magellan. Ashish's team said the documentation for the library needs to be updated since some of the function names used in the manual are not consistent with the ones in the code of the library. Tithy's team mentioned that a Javadoc-style of method comments would be helpful to understand it. And Shuang's team hoped more examples for the functions in the manual.

In our case, we found that the documentation of using trigger is not comprehensive enough. It would be appreciated if more examples of utilizing triggers are provided.

3. Scalability

Some groups ran into the issues of scalability like Ashish's team, Tithy's team etc. It would be painful to use Rule Based Blocker and Black Box Blocker directly on the tables we started out because these blockers don't scale for such large tuple pairs right now.

Also, there is scalability issue in the debugger in the blocking stage. Shuang's team told that it would take a huge amount of time to run debugger based on original tables. Therefore, a candidate-set level debugger would be preferred if it could be provided.

4. Debugger

a) FP & FN

For dealing with False Positive and False Negative using debugger in the matching stage, some teams thought the current visual debugging capabilities of DT and RF approaches are very helpful but there are still many improvements that can be made, such as approaches to visualize the FP&FN after adding some rules on the matchers, extending the current capability to other learning approaches.

Besides, when using the current visual debugger, it would be better if we can use it to access the FP&FN cases in the table directly, i.e. change the wrong label or delete the row in the table.

b) Scalability

There is a scalability issue for the debugger in the blocking stage as we described in the Scalability part.

5. Blocker

The issues and suggestions of blocker focus on Overlap blocker although some of them can also be applied to other blockers. For Overlap blocker, the teams suggested the following points:

- It would be good to add functionality for case sensitive/insensitive match of attributes.
- It would be better if we could give a percentage of the number of tokens that should match for a particular attribute as an input to the blocker instead of only the number of tokens to match.
- Overlap blocker discards tuples with missing values for selected attribute. There should be an option for the user to specify whether tuples with missing values should be discarded or not. Actually, our team faced the same missing-value problem as well but with Attribute-Equivalent blocker. This missing-value handling mechanism should be carefully considered for all the possible related blockers.

6. Feature Selection

Some teams felt confused about selecting features from a large feature space. Neha's team mentioned they wished there were a shrinkage tool to help them with feature selection.

In our case, it is a little bit hard to select disjointed features from the feature table, e.g. selecting feature[3:13], feature[15:17] and feature[20:24]. It would be helpful if there is a function that can support such action in Magellan. Shuang's team also identified this problem and gave a solution.

7. Miscellaneous Topics

- Two teams encountered the problem of handling spaces in the attribute names in the blocking stage. It took them quite a while to figure it out.

- A team requested the intersection operation for combining the output of different blockers.
- Quite a few teams experienced that the testing on J results for SVM are all zeros (0.00%) if multiple learning-based matchers are executed at the same time (in the same ipython cell,) and then printing them individually. Instead, we fix the problem by executing each learning-based matcher in each ipython cell. Shuang's team also identified this problem and gave another solution.

Epilogue

Why this report? After finishing our bonus feedback report, we came up with a question that how well other teams did their projects and whether they experienced the same Magellan issues with us. Then we started doing it right away after we had this idea. However, it was really not fun to read others' report at the very end of the semester. But we hope we can provide a more comprehensive and more useful feedback or a different perspective of Magellan in this way of doing survey.

The text scripts we extracted from 11 teams are provided in the appendix. We used these ideas to compose our report, most of which come from the blocking explanations of the teams. Text in red indicates the topic in the paragraph for the sake of fast reading.

Appendix

1. Ashish Shenoy, ashenoy@cs.wisc.edu; Shruthi R, shruthir@cs.wisc.edu
blocking_explanation.pdf
• Magellan could not handle spaces in the attribute names. For example, when we first tried to run the attribute equivalence blocker on our tables, the Phone Number column had space in it and the blocker failed without giving any indication that it failed due to the space in the column name.
• Better exception handling is necessary to help user understand why something failed.
• It would be a good to have feature if we could run the Rule Based Blocker and the Black Box Blocker directly on the tables we started out with instead of the candidate sets. Right now these blockers don't scale for such large tuple pairs.
• The documentation for the library should be updated. Some of the function names used in the manual are not consistent with the ones in the code of the library.

2. Ali Hitawala, alihitawala@cs.wisc.edu; Shaleen, shaleen@cs.wisc.edu
blocking_strategy.pdf
5.2 FEATURES THAT WE CAN EXTEND
1. Exception Handling - We ran to issues where we see generic exception like 'Column qualifies to be more than one type'. This error, for example, is thrown in attributeutils.py which contains all the relevant information about which columns are colliding and what are their respective type. Ideally, this information should be included in the stacktrace that bubbles up.
3. Overlap Blocker issues - There are 2 points that we wish to highlight in Overlap Blocker implementation.
• Overlap blocker does not functionality for case sensitive/insensitive match of attributes. This is easy to implement and would be a good feature to add since data is almost always noisy specially with respect to case sensitivity.

3. Manish Bansal, mbansal2@wisc.edu; Bijoy Sarkar, bsarkar2@wisc.edu
CS784_Blocking_Report_Final.pdf
Magellan ran into problem when the attribute name in the csv file contained spaces. For example, we got error while blocking because of the field "Release Date". When we fixed it to "ReleaseDate", things worked smoothly. We also noticed that when we submit request to execute heavy blocker/debugger, it took a while before communicating anything to us. This was confusing because we initially mistook that there was some issue with Magellan or its setup.

4. Walter Cai, wcai@cs.wisc.edu; Guangshan Chen, gchen9@wisc.edu
blocking-explanation.pdf
Another challenge was the fact that sorted neighborhood has not yet been implemented. We believe this is a crucial next addition and are considering implementing it as the final stage of this project. A resulting challenge was

understanding the desired syntax for the blackbox blocker constructor although this was quickly understood as well.

5. Rasiga Gowrisankar, rasiga@cs.wisc.edu; Ganesh Kumar Velu Rajendran, velurajendra@wisc.edu
blocking-explanation.pdf
Feature Requests
- Combining the output of different blockers based on the intersection operation in addition to the union operation.

matching.pdf
Feature Requests
- Extending the visual debugging capabilities of the DT and RF approaches to other approaches.

6. Shreya Kamath, shreyakamath@cs.wisc.edu; Mushahid Alam, malam5@wisc.edu
blocking-explanation.pdf
• Overlap blocker discards tuples with missing values for selected attribute, we think there should be an option for the user to specify whether tuples with missing values should be discarded or not. This applies to attribute level blockers as well.

7. Tithy Sahu, tsahu@wisc.edu; Jyotiprakash Mishra, jmishra@cs.wisc.edu
blocking_explanation.pdf
Scalability issues:
Time of execution for a rule-based blocker on features increases with the size of tables, unlike overlap/attr-equiv blockers.

Feedback:
1. A Javadoc-style of method comments would be helpful to know the method description, its parameter list and the meaning of each parameter to be passed.

matching_explanation.pdf
2. How do we visualize the FP/FN obtained in the evaluation summary? This result is different from the one obtained from doing the visual debug process.

8. Qing Li, qing.li@wisc.edu; Fan Ding, fding5@wisc.edu
blocking-explanation.pdf
There is a setup issue in both Windows 7 and Windows 10. After finishing installing Magellan, when import Magellan in Ipython notebook, it will complain about,

9. Kirthanaa Raghuraman, kraghuraman@wisc.edu; Rogers Jeffrey L, leojohn@wisc.edu
Blocking-Explanation.pdf
Feedback on Magellan:
• For the overlap blocker, we faced an issue where we required the number of tokens to overlap be greater than 2 or 3. But some song names had only one token

and they were eliminated if we imposed the above condition while running the overlap blocker. Hence, we felt that it would be better if we could give a percentage of the number of tokens that should match for a particular attribute as an input to the blocker instead of only the number of tokens to match. This would greatly increase the accuracy of the blocker in filtering out the unwanted tuples.

Matching-Report.pdf
Suggestions:
One important feature we realized that Magellan was lacking was that we weren't able to debug the matcher after applying rules. Such a feature would be very helpful since we would know if the rule had actually fired and that the rule is working properly and not causing more errors. It would be really helpful if this feature is also added in Magellan.

10. Neha Godwal, godwal@wisc.edu; Thu Le, thuhale@stat.wisc.edu
Matcher_Report.pdf
V. THE CAVEATS:
• We wish there were a shrinkage tool to help us with feature selection.

11. Shuang Wu, swu56@wisc.edu
BlockingExplanation.pdf
6. feedback: the scalability issue matters, it take a huge amount of time to run debugger based on original tables, candset level debugger would be preferred if it could be provided. more buildin sim functions and global level blocker are preferred.

feedback.pdf
The overall experience for Magellan is awesome. Some feedback here for it:
1. More examples for function utilization in the user manual would be very helpful.
3. I found that some classmates in their reports say they don't know how to select features from a large feature space. Perhaps you could add example like following for advice:
7. 'mg.vis_debug_rf ()' function is very good to figure out FP and FN, but we need another to way to visualize the bugs(FP&FN) after adding some rules on the matchers.
8. you could add a function for the debugging UI to access the FP & FN cases in the table directly if possible. (e.g. the data analyst could change the wrong label or delete the row from the debugging UI directly. this would be extremely efficient for them to debug).