Project proposal

Student: Michael de Jong (11246618)
Supervisor: Dr. Maarten Marx

April 10, 2020

1 Search engine for large collections of Jupyter Notebooks

There are over 6 million Jupyter notebooks on github and not a good way to search trough them to be able to find their desired part of the notebooks. The main goal of the project is to make a search engine that is aware of the Jupyter notebook format so that it improve the search. In the notebook format, there are multiple types of information in different cells. Thus the search engine has to search the different cell types and in graphs. In Grappiolo et al. 2019 they implemented the "Semantic Snake Charmer" or SSC, which is a domain knowledge-based search engine for Jupyter Notebooks. They composed the SSC out of three modules: (1) a human-machine cooperative module to identify internal documentation which contains the most relevant domain knowledge, (2) a natural language processing module capable of transforming relevant documentation into several semantic graph types, (3) a reinforcement learning based search engine which learns, given user feedback, the best mapping between input queries and semantic graph type to rely on. The question that arises is if it is necessary to be aware of the notebook format or not. Is there a significant difference to make a search engine aware of the notebook format?

To implement an experiment to deduce if this is the case there is a need for two search engines. One that takes the notebook format into account and one that does not. The two different search engine will then be compared to each other. The first thing that is needed is the Jupyter notebook data. The data that will be used originates from Rule 2018 which is a collection of about 1.25 million notebooks from github at the time. With that data a test set for both search engines will be implemented in order to be able to test relevance on different queries.

Both search engines will require different ways of indexing (tokenization) into an inverted index. Only one of them gets the index specialised to work with the notebook format and can thus take advantage of separating the different notebook cells. Both search engines will have the same ranking method, query classifier and interface to be able to evaluate both search engines equally. The search algorithm that will be used for the implementation is Elastic Search.

For the ranking of the search engine a neural method is used. The neural method is an evolutionary strategy (ES-Rank) that uses different scoring metrics is described in Ibrahim and Landa-Silva 2018. The input needed for this method is the training set of query-document pairs (or feature vectors). ES-Rank works on a population of size two. The first is the current solution (parent) and the second is the candidate solution (child). The child is a result of a mutation of the parent. If the offspring is as good as the parent or better, it replaces the parent for the next generation, otherwise the offspring is forgotten. A solution or "chromosome" is a vector of weights all together representing the ranking function being evolved. This outputs a linear ranking function that then is used for the ranking of the results.

The query classifier needs to associate different queries with a certain category of notebook that is needed. A technique that will be used for this task is called query enrichment. This technique is described in

Shen et al. 2006. It takes a short query and then maps it to intermediate objects. Based on the collected intermediate objects, the query is then mapped to target categories that can be used to improve performance.

The interface of the search engines will be similar the old version of google, that is with the notebook that contains the information, a link to the notebook and information snippets from the file. The first result has the highest ranking and the second the second highest ranking and so on.

To evaluate both search engines several evaluation metrics will be used, namely precision, recall, F1, mean average precision and discounted cumulative gain. To be able to use these metrics they require the use of the test set and the queries. The expectation is that the search engine that is aware of the notebook format scores higher on the metrics then the search engine that is unaware of the format.

2 Plan

The plan consists of the following milestones with the approximate time they will take:

1. Implement a large test set, 3 days
2. Implement queries and qrels, 3 days
3. Implement a notebook parser and indexer with the format in mind, 3 days
4. Implement a notebook parser and indexer without the format in mind, 3 days
5. Implement a query classifier, 4 days
6. Train the query classifier, 1 day
7. Implement a the scoring methods, 2 days
8. Implement the evolutionary ranking method, 4 days
9. Train the evolutionary ranking method with the format in mind, 2 days
10. Train the evolutionary ranking method without the format in mind, 2 days
11. Implement a Smart interface which needs:
    - query interface, 1 day
    - SERP, 2 day
    - snippets, 1 day
12. Midterm Presentation preparation and execution, 2 days
13. Evaluate the search engine with the different scoring functions. 5 days
14. Writing thesis, 10 days
15. Reading up on various topics needed, 4 days
16. other unforeseen tasks, 2 days

Not all have to be done in order, but writing the thesis is done last. The contingency plan is to use less data when needed or to implement less advanced query and ranking classifiers. Then it is still possible to anwer the research question.

References

[Gra+19] Corrado Grappiolo et al. "The Semantic Snake Charmer Search Engine: A Tool to Facilitate Data Science in High-tech Industry Domains". In: Mar. 2019, pp. 355–359. DOI: 10.1145/3295750.3298915.

[ILS18] Osman Ali Sadek Ibrahim and Dario Landa-Silva. "An evolutionary strategy with machine learning for learning to rank in information retrieval". In: *Soft Computing* 22.10 (2018), pp. 3171–3185.

[Rul18] Aurélien; Hollan James D Rule Adam; Tabard. "Exploration and Explanation in Computational Notebooks." In: *UC San Diego Library Digital Collections*. 2018. DOI: 10.6075/J0JW8C39.

[She+06] Dou Shen et al. "Query enrichment for web-query classification". In: *ACM Transactions on Information Systems (TOIS)* 24.3 (2006), pp. 320–352.