GitHub Repository Classifier

IT is changing fast; new programming languages, frameworks and problem-solving approaches appear almost every day. As an IT professional, no matter whether you are a student, scientist or software engineer, it is crucial to keep your knowledge and skills up to date.

A common way of learning more about new programming languages and technologies and improving existing skills is provided by social coding platforms like GitHub[1]. GitHub is home to many Open Source communities and commercial software developers worldwide with more than 15 million users, having created more than 36 million repositories.

With more than 36 million repositories hosted, finding the ones of interest and relevance for a specific person with a specific need can be challenging. For example:

- a software engineer might want to learn more about repositories that actively develop using her chosen framework.
- a professor might want to get inspired on how other organizations host tutorials, lectures, educational information and related code.
- a student might be interested in storing his homework and related code and discuss with other students how they approached an assignment.

Fortunately, today's knowledge workers are more and more supported by intelligent and adaptable digital assistants. Digital assistants as used in software engineering have evolved from simple syntax completion to intelligent recommender systems that base their decisions on data retrieved via software repository mining from the ever increasing data on platforms like GitHub. If repositories were automatically classified and tagged based on the type of content and contribution activity, this information could be used to provide far better search results.

This year's InformatiCup challenge is dedicated to the new possibilities that arise based on the application of artificial intelligence and repository mining on the social coding platform GitHub.

[1] https://github.com

Challenge

Implement an automatic classification/labelling of repositories hosted on GitHub.

First, analyze which features are relevant to perform an automatic classification (labelling) and document those.

Using methods like machine learning - apply the relevant features to label repositories according to the following 7 categories.

- **DEV** – a repository primarily used for development of a tool, component, application, app, or API
- **HW** repo – a repository primarily used for homework, assignments and other course-related work and code
- **EDU** – a repository primarily used to host tutorials, lectures, educational information and code related to teaching
- **DOCS** – a repository primarily used for tracking and storage of non-educational documents
- **WEB** - a repository primarily used to host static personal websites or blogs
- **DATA** - a repository primarily used to store data sets
- **OTHER** – use this category only if there is no strong correlation to any other repository category, for example, empty repositories

You can find examples for repositories belonging to these categories in Appendix A. The examples mentioned there should be only used as a first rough idea - your algorithms and relevant features should be trained with a larger data set. The documentation of your design thoughts and training approach will influence the final ranking of your solution way more than a high similarity of your classification results as compared to the examples in Appendix A.

When working on this challenge, please try to structure your documentation according to the following steps to ensure we can compare solutions.

Data exploration and prediction model

First, analyze and document the properties (features) needed to classify (label) repositories according to the seven categories. Examples sources of information for repositories would be the content of README.md, the file names within the repository, file contents, commit email addresses, comments within issues and pull requests or the used programming languages.

Features are relevant if there is a correlation between the repository category that should be determined and hence are useful for the prediction model. The selection of the right features and

appropriate prediction model are crucial for the the quality of your results. While selecting your features and prediction models, please try to avoid overfitting.

Explain within your documentation, why you decided for the features you have used and how you developed your prediction model.

For your prediction model and the implementation of your labelling algorithms you can use any algorithm, machine learning in particular.

To retrieve information about the repositories to be classified, you can use GitHub's APIs and any other sources of your choice. You can find a short introduction on how to use the GitHub APIs in the Getting Started section.

Automated Classification

Based on the results of your data exploration and prediction model, repositories hosted on GitHub.com should be automatically classified into the seven categories discussed within this challenge.

To accomplish this, implement an application that processes data formatted as explained in section input format and output classification results in the format specified in the output format section.

If you are using a machine learning approach your software should either

1) prompt for the training data to use (and have documentation where to point it)
2) directly include the learned models

Validation

Apply your classifier on the repositories included in Appendix B. You can find this file on https://github.com/InformatiCup/InformatiCup2017 as well. Create a boolean matrix where you compare the results where you compare the results of your classifier and your intuitive classification.

Compute the yield per category- the amount of repositories intuitively placed within a category in the set of repositories that got placed in the same category by your classifier.

Compute the *precision* per category- the amount of repositories per category where the results determined by your automatic classifier matched your intuitive classification.

Discuss the quality of your results and argue whether, according to your opinion, a higher yield or a higher precision is more important for automated repository classification.

Extensions (optional)

If you should have any time left after successfully implementing the basic requirements of this challenge, please use your imagination to impress us with an extension to the original idea. One idea would be to create a web based application that is asking the currently logged in user via OAuth to grant access to her profile and automatically classify his repositories. A different extension idea would be to visualize the decision process of your algorithm. Be creative!

Input Format

Your application should process a plain text file with a set of repository URLs as shown in example input 1. Individual lines are separated with a new line. The URLS of the repositories to be classified are specified in the GitHub format:

```
https://github.com/<owner>/<repository name>
```

```
https://github.com/briantemple/homeworkr
https://github.com/spez/RottenTomatoes
https://github.com/DataScienceSpecialization/courses
https://github.com/bundestag/gesetze
https://github.com/BloombergMedia/whatiscode
https://github.com/ericfischer/housing-inventory
https://github.com/jonico/other
```

Example input 1 1: Set of Repository URLs

Output Format

Your application should write the classification results in an output plain text file as shown in example output 1. Each repository URL from the input format should be followed by one space character, followed by the repository category abbreviation as described in this challenge. Individual lines are separated by a newline. The order of repository URLs in the output file should match the order in the input file.

```
https://github.com/briantemple/homeworkr DEV
https://github.com/spez/RottenTomatoes HW
https://github.com/DataScienceSpecialization/courses EDU
https://github.com/bundestag/gesetze DOCS
https://github.com/BloombergMedia/whatiscode WEB
https://github.com/ericfischer/housing-inventory DATA
https://github.com/jonico/other OTHER
```

Example output 1: Set of classified Repository URLs

Getting Started

GitHub provides a large selection of REST APIs to retrieve repository metadata, repository content, as well as repository activities, issues, pull requests and reactions. These REST APIs can be used for free by any programming language. You can find a selection of SDKs under https://github.com/octokit. If your favorite programming language is not included in Octokit, chances are very high there is an alternative SDK available on GitHub. For Java-Programmers you could for instance use http://github-api.kohsuke.org.

Let's have a look at a command line example (shortened), that is listing the basic information of GitHub repository https://github.com/WebpageFX/emoji-cheat-sheet.com:

```
$ curl https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com

{
      "id": 2592600,
      "name": "emoji-cheat-sheet.com",
      "full_name": "WebpageFX/emoji-cheat-sheet.com",
      ...
      "html_url": "https://github.com/WebpageFX/emoji-cheat-sheet.com",
      "description": "A one pager for emojis on Campfire and GitHub",
      "fork": false,
      "Url":"https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com",
      ...
      "size": 19203,
      "stargazers_count": 4108,
      "watchers_count": 4108,
      "language": "HTML",
      "has_issues": true,
      "has_downloads": true,
      "has_wiki": false,
      "has_pages": false,
      "forks_count": 1156,
      "mirror_url": null,
      "open_issues_count": 41,
      "forks": 1156,
      "open_issues": 41,
      "watchers": 4108,
      ...
      "network_count": 1156,
      "subscribers_count": 206
}
```

As only public repositories are used as part of this challenge, you do not necessarily have to authenticate your REST API requests. However, by using a personal access token, you will be able to make way more requests per hour as GitHub applies API rate limiting for anonymous requests.

Here is one example with authentication that is listing all files in the main directory of the master branch of https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com:

```
$ curl -u githubusername:token
https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com/contents/
```

```
[{
      "name": ".gitignore",
      ...
      }
}, {
      "name": "Gemfile",
      ...
      }
}, {
      "name": "Gemfile.lock",
      ...
}, {
      "name": "LICENSE",
      ...
}, {
      "name": "README.md",
      ...
}, {
      "name": "Rakefile",
      ...
}, {
      "name": "config.yaml.sample",
      ...
}, {
      "name": "lib",
      ...
      "type": "dir",
      "sha": "a911de9d621cc0594492bed38eaa5197226a5ea6"
      ...
}, {
      "name": "public",
      ...
}]
```

It is also possible to recursively list all files within a directory, for instance the `lib/` directory:

```
$ curl -u username:token
https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com/git/trees/a911de
9d621cc0594492bed38eaa5197226a5ea6?recursive=1
```

Finally, one example on how to list all commits within the master branch including the committers' email addresses:

```
$ curl -u username:token
https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com/commits
```

If you like to combine multiple data sources within one API call or like to aggregate data on GitHub's server side, you can also use the newly introduced GraphQL API.

Besides from the official GitHub API which is showing the current state of a GitHub repository, there are multiple archives which contain all public events that ever happened on GitHub in a machine readable format.

The GHTorrent project enables you to access GitHub data without Internet access and is linking to some very interesting research work in the field of Repository Data Mining.

The GitHub Archive contains all public events that have happened on GitHub.com since late 2011 and provide results via zip files and Google Big Query. Google Big Query is free for the first processed 1 TB of data per month.

Hint: Our InformatiCup challenge does not prescribe the use of any particular API. It is part of the challenge to analyze which data sources to use to come up with an efficient classification.

Furthermore

Please document three repositories where you assume that your application will yield better results as compared to the results of other teams.

Your application should come with an install and user manual. Please also document the decisions you made selecting your features, algorithms, data structures and software development tools and practices.

The FAQs for our challenge as well as the example input and output files, appendices and the actual task description can be found online at https://github.com/InformatiCup/InformatiCup2017

Appendix A - Examples

Examples for DEV-Repositories

- https://github.com/briantemple/homeworkr
- https://github.com/spring-projects/spring-boot
- https://github.com/facebook/react
- https://github.com/nodegit/nodegit
- https://github.com/scipy/scipy

Examples for HW-Repositories

- https://github.com/spez/RottenTomatoes
- https://github.com/m2mtech/calculator-2015
- https://github.com/bcaffo/751and2
- https://github.com/HPI-SWA-Teaching/SWT16-Project-08
- https://github.com/uwhpsc-2016/example-python-homework

Examples for EDU-Repositories

- https://github.com/DataScienceSpecialization/courses
- https://github.com/githubteacher/intro-november-2015
- https://github.com/alex/what-happens-when
- https://github.com/MostlyAdequate/mostly-adequate-guide
- https://github.com/AllThingsSmitty/jquery-tips-everyone-should-know

Examples for DOCS-Repositories

- https://github.com/CMSgov/HealthCare.gov-Styleguide
- https://github.com/github/maturity-model
- https://github.com/raspberrypi/documentation
- https://github.com/bundestag/gesetze
- https://github.com/fsr-itse/docs

Examples for WEB-Repositories

- https://github.com/BloombergMedia/whatiscode
- https://github.com/JaceRobinson8/jacerobinson8.github.io
- https://github.com/rubymonstas-zurich/rubymonstas-zurich.github.io
- https://github.com/ianli/elbowpatched-boilerplate
- https://github.com/whoisjuan/whoisjuan.github.io

Examples for DATA-Repositories

- https://github.com/ericfischer/housing-inventory
- https://github.com/GSA/data
- https://github.com/OpenExoplanetCatalogue/open_exoplanet_catalogue
- https://github.com/Hipo/university-domains-list
- https://github.com/minimaxir/big-list-of-naughty-strings

Appendix B - Repositories

- https://github.com/ga-chicago/wdi5-homework
- https://github.com/Aggregates/MI_HW2
- https://github.com/datasciencelabs/2016/
- https://github.com/githubteacher/intro-november-2015
- https://github.com/atom/atom
- https://github.com/jmcglone/jmcglone.github.io
- https://github.com/hpi-swt2-exercise/java-tdd-challenge
- https://github.com/alphagov/performanceplatform-documentation
- https://github.com/harvesthq/how-to-walkabout
- https://github.com/vhf/free-programming-books
- https://github.com/d3/d3
- https://github.com/carlosmn/CoMa-II
- https://github.com/git/git-scm.com
- https://github.com/PowerDNS/pdns
- https://github.com/cmrberry/cs6300-git-practice
- https://github.com/Sefaria/Sefaria-Project
- https://github.com/mongodb/docs
- https://github.com/sindresorhus/eslint-config-xo
- https://github.com/e-books/backbone.en.douceur
- https://github.com/erikflowers/weather-icons
- https://github.com/tensorflow/tensorflow
- https://github.com/cs231n/cs231n.github.io
- https://github.com/m2mtech/smashtag-2015
- https://github.com/openaddresses/openaddresses
- https://github.com/benbalter/congressional-districts
- https://github.com/Chicago/food-inspections-evaluation
- https://github.com/OpenInstitute/OpenDuka
- https://github.com/torvalds/linux
- https://github.com/bhuga/bhuga.net
- https://github.com/macloo/just_enough_code
- https://github.com/hughperkins/howto-jenkins-ssl