GitHub Repository Classifier

Derzeit entstehen in vielen Bereichen neuartige Formen der Unterstützung von Wissensarbeitern durch intelligente und lernfähige Assistenzsysteme. Die Informationsbasis für eine automatisierte Entscheidungsfindung wächst stetig und ermöglicht eine kontinuierliche Verbesserung der Vorhersagen.

So wurden zum Beispiel im Bereich der Softwareentwicklung Empfehlungsdienste (sog. *Recommender Systeme*) von der syntaktischen Autovervollständigung zu intelligenten Empfehlungsdiensten basierend auf natürlichsprachlichen Anforderungsspezifikationen und Software Repository Mining weiterentwickelt. Neue Möglichkeiten ergeben sich insbesondere mit der wachsenden Informationsbasis über Softwareentwicklungsprojekte durch die zunehmende Verbreitung von Social Coding und der Softwareentwicklung auf kollaborativen Entwicklungsplattformen wie GitHub[1].

Empfehlungsdienste sind bereits allgegenwärtig wenn Objekte entsprechend ihrer Relevanz bezüglich einer Suchanfrage sortiert werden sollen. Insbesondere im Bereich der Softwareentwicklung kommt einer effektiven Suche nach Informationen eine hohe Bedeutung zu, verbringen Softwareentwickler doch nachweislich die meiste Zeit mit der Suche nach Informationen.

Ein Softwareentwickler könnte beispielsweise mehr über andere Projekte erfahren wollen, die bestimmte Programmierbibliotheken seiner Wahl einsetzen. Ein Hochschullehrer könnte sich Inspiration von anderen Bildungseinrichtungen holen wie diese Tutorien, Vorlesungen und Übungen inklusive den damit verbundenen Code-Beispielen verwalten. Studierende könnten daran interessiert sein, wie sie im Team arbeiten, ihre Hausaufgaben sowie den damit verbundenen Code versionieren und mit anderen Ansätzen vergleichen können.

Den neuen Möglichkeiten der Anwendung künstlicher Intelligenz und automatischer Empfehlungsdienste im Bereich der Softwareentwicklung widmet sich der InformatiCup 2017 mit der Aufgabe der automatischen Klassifizierung von Repositories auf der kollaborativen Entwicklungsplattform GitHub.

[1] https://github.com

Aufgabe

Implementieren Sie eine automatische Klassifizierung der auf GitHub gehosteten Repositories.

Analysieren Sie dazu zunächst, welche Merkmale der Repositories für eine automatische Klassifizierung relevant sind. Dokumentieren Sie diese Merkmale.

Verwenden Sie schließlich diese relevanten Merkmale, um – zum Beispiel mit Verfahren des maschinellen Lernens – Repositories automatisch in die folgenden sieben Kategorien einzuordnen.

- **DEV**: Repositories für die Entwicklung eines Tools, einer Softwareanwendung, einer App, einer Bibliothek, einer API, oder ähnliche Softwareentwicklungsprojekte
- **HW**: Repositories mit Lösungen und Quelltexten für Hausaufgaben und Übungsblätter
- **EDU**: Repositories mit didaktischen Inhalten und Quelltexten für Vorlesungen und Tutorien
- **DOCS**: Repositories für die Verwaltung und Speicherung von nicht-didaktischen (d.h. nicht EDU) Inhalten und Quelltexten
- **WEB**: Repositories für das Hosting persönlicher Web-Seiten oder Blogs
- **DATA**: Repositories für die Speicherung von Datensätzen
- **OTHER**: Repositories, die sich nicht in die anderen Kategorien einordnen lassen

Beispiele für Repositories dieser Kategorien finden Sie in Anhang A. Die dort aufgeführten Beispiele klassifizierter Repositories sollen dabei nur als grobe Orientierungshilfe dienen. Es ist wichtig, dass Sie Ihre Algorithmen und relevanten Merkmale mit einem größeren (Trainings-)Datensatz entwickeln. Die Dokumentation Ihrer Überlegungen und Vorgehensweise fällt bei der Bewertung Ihrer Lösung wesentlich mehr ins Gewicht als eine Übereinstimmung mit den vorab klassifizierten Repositories in Anhang A.

Orientieren Sie sich bei der Bearbeitung dieser Aufgabe bitte an den folgenden Bearbeitungsschritten, die eine gute Vergleichbarkeit der eingereichten Lösungen sicherstellen sollen.

Datenexploration und Vorhersagemodell

In einem ersten Schritt analysieren und dokumentieren Sie die für die Klassifizierung eines Repositories relevanten Mermale (sog. *Features*). Merkmale von Repositories sind zum Beispiel die README.md, die Dateinamen, die Dateitypen, der Inhalt der Dateien, die Commit-Email-Adressen, die Kommentare in den Issues und Pull-Requests oder die verwendeten Programmiersprachen.

Merkmale sind dann relevant, wenn sie in einem Zusammenhang mit der zu bestimmenden Kategorie eines Repositories stehen und damit nützlich für Ihr Vorhersagemodell sind. Die Auswahl der richtigen Merkmale und eines geeigneten Vorhersagemodells sind entscheidend für die Qualität Ihrer Ergebnisse. Achten Sie bei der Definition der relevanten Merkmale und ihres Vorhersagemodells zum Beispiel auf die Vermeidung von Überanpassung (sog. *Overfitting*).

Erläutern Sie in Ihrer Dokumentation, warum Sie welche Merkmale für die Klassifizierung verwenden und wie Sie Ihr Vorhersagemodell entwickelt haben.

Für Ihr Vorhersagemodell und die Umsetzung entsprechender Klassifikatoren stehen Ihnen alle Arten von Klassifikationsverfahren – insbesondere auch maschinelles Lernen – offen.

Zur Informationsabfrage über die zu klassifizierenden Repositories können Sie die GitHub APIs und weitere Quellen Ihrer Wahl verwenden. Eine kurze Einführung in die Verwendung der GitHub APIs finden Sie in dieser Aufgabenbeschreibung im Abschnitt Getting Started.

Automatische Klassifizierung

Basierend auf den Ergebnissen Ihrer Datenexploration und Ihres Vorhersagemodells sollen auf GitHub gehostete Repositories *automatisch* in die sieben Kategorien wie in dieser Aufgabenbeschreibung definiert eingeordnet werden.

Implementieren Sie dazu eine Software, die das in Abschnitt Eingabeformat spezifizierte Format verarbeitet und die Klassifikation in dem in Abschnitt Ausgabeformat spezifizierten Format ausgibt.

Falls Sie ein maschinelles Lernverfahren verwenden sollte außerdem entweder 1) Ihre Software interaktiv für die Auswahl erforderlicher Trainingsdaten verwendbar und die zu verwendenden Trainingsdaten dokumentiert sein oder 2) die erlernten Modelle unmittelbar in Ihrer Software verfügbar sein.

Validierung

Wenden Sie Ihren Klassifikator auf die in Anhang B aufgeführten GitHub-Repositories an. Die entsprechende Eingabedatei finden Sie auf http://www.informaticup.de. Erstellen Sie eine Wahrheitsmatrix in der Sie für diese Repositories sowohl Ihre intuitive Klassifikation, um was für eine Kategorie es sich handelt, als auch die Ergebnisse Ihres automatischen Klassifikators eintragen.

Berechnen Sie die erzielte *Ausbeute*: den Anteil der, bezüglich Ihrer intuitiven von Hand vorgenommenen Einordnung, korrekt klassifizierten Repositories für die verschiedenen Kategorien. Berechnen Sie die erzielte *Präzision*: den Anteil der korrekt in eine bestimmte Kategorie eingeordneten Repositories an der Gesamtheit der in diese Kategorie eingeordneten Repositories.

Diskutieren Sie die Güte Ihrer erzielten Ergebnisse und argumentieren Sie, warum Ihrer Meinung nach für die Klassifizierung von Repositories eine hohe Ausbeute oder eine hohe Präzision wünschenswert ist.

Integration (optional)

Falls nach der erfolgreichen Implementierung der Grundanforderungen noch Zeit für Erweiterungen bleibt, seien Sie kreativ mit einer alternativen Bedienoberfläche für Ihr Programm. Eine Idee wäre zum Beispiel, eine Web-basierte Applikation zu erstellen, die mittels OAuth Zugriff auf das GitHub-Profil des derzeit eingeloggten Nutzers anfordert und dessen Repositories automatisch klassifiziert. Eine andere Idee wäre die Visualisierung der Entscheidungsfindung Ihres Programms. Seien Sie kreativ!

Eingabeformat

Ihre Implementierung muss aus einer "Plain text"-Datei eine Menge von Repository-URLs wie in Beispieleingabe 1 verarbeiten. Einzelne Zeilen sind durch einen Zeilenumbruch (Newline) getrennt.

Die URLs der zu klassifizierenden Repositories sind im GitHub-Format gegeben:

```
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
```

Beispieleingabe 1: Menge von Repository URLs

Ausgabeformat

Ihre Implementierung muss die Klassifikation der Repositories in eine "Plain text"-Datei wie in Beispielausgabe 1 ausgeben. Jeder Repository-URL folgt, durch ein Leerzeichen (Space) getrennt, das Ergebnis der Klassifizierung in Form des Kürzels einer der sieben Repository-Kategorien wie in dieser Aufgabenbeschreibung definiert.

Einzelne URLs mit ihrem Klassifikationsergebnis sind durch einen Zeilenumbruch (Newline) getrennt.

```
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
```

Beispielausgabe 1: Menge von klassifizierten Repository URLs

Getting Started

GitHub stellt eine große Auswahl an REST APIs zur Erlangung von Repository-Metadaten, Repository-Inhalten, sowie Repository-Aktivität, Issues, Pull requests und Reactions zur Verfügung. Diese REST-APIs können mit beliebigen Programmiersprachen kostenlos genutzt werden. Eine Auswahl von SDKs ist unter https://github.com/octokit zu finden. Falls Ihre Lieblingssprache nicht im Octokit enthalten ist, besteht eine große Chance, ein alternatives SDK auf GitHub zu finden. Für Java-Programmierer bietet sich beispielsweise http://github-api.kohsuke.org an.

Es folgt ein Kommandozeilenbeispiel, das die Grundinformationen zum GitHub-Repository https://github.com/WebpageFX/emoji-cheat-sheet.com auflistet (gekürzt):

```
$ 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
}
```

Da für die Aufgabe nur öffentlich verfügbare Repositories genutzt werden, muss sich nicht unbedingt bei der REST-API authentifiziert werden. Allerdings ist es mit einem persönlichen Zugriffstoken möglich, mehr Anfragen an die GitHub-API pro Stunde zu stellen, als dies anonym der Fall wäre.

Hier ist ein Beispiel, das alle Dateien im Hauptverzeichnis des Master-Branch von https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com mit Authentifizierung auflistet:

```
$ 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",
      ...
}]
```

Es ist auch möglich, sich alle Dateien eines Verzeichnisses rekursiv anzeigen zu lassen, zum Beispiel für das `lib/` Verzeichnis:

```
$ curl -u username:token
https://api.github.com/repos/WebpageFX/emoji-cheat-sheet.com/git/trees/a911de9d
621cc0594492bed38eaa5197226a5ea6?recursive=1
```

Als letztes noch ein Beispiel, um alle Commits im Master Branch inklusive Emails der Committer aufzulisten:

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

Wenn Sie mehrere Datenquellen mit einem API-Aufruf verbinden möchten beziehungsweise eine Aggregation auf GitHub-Serverseite vornehmen wollen, so steht Ihnen auch die neu eingeführte GraphQL API zur Verfügung.

Neben der offiziellen GitHub-API, die den Ist-Zustand von Repositories auf GitHub liefert, existieren mehrere Archive, die sämtliche Events, die in GitHub-Repositories in den letzten Jahren auftraten, in maschinenlesbarer Form zur Verfügung stellen.

Das GHTorrent Projekt ermöglicht es, GitHub Daten ohne Internetzugang abzufragen und verweist auf einige interessante Forschungsarbeiten zum Thema Repository Data Mining.

GitHub Archives ermöglicht den Zugriff auf alle öffentlichen Events, die auf GitHub.com auftraten und stellt diese als Zip-Datei und Google Big Query zur Verfügung. Google Big Query ist für die ersten 1 TB an verarbeiteten Daten pro Monat kostenlos.

Hinweis: Der InformatiCup schreibt nicht die Nutzung einer bestimmten API für die Klassifizierung vor. Es ist Teil der Aufgabenstellung, zu analysieren, welche Daten Sie für eine effiziente und effektive Klassifizierung verwenden möchten.

Außerdem

Dokumentieren Sie bitte 3 Software Repositories, von denen Sie annehmen, dass Ihr Programm im Vergleich zu denen anderer Teams gute Ergebnisse liefert.

Erstellen Sie für Ihre Implementierung bitte eine Bedienungs- und Installationsanleitung. Dokumentieren Sie die von Ihnen getroffenen Entscheidungen bei der Auswahl verwendeter Algorithmen und Datenstrukturen und in der Software-Entwicklung.

Die FAQs zum laufenden Wettbewerb und die Beispielein- und Ausgaben zu dieser Aufgabenbeschreibung finden Sie online auf http://www.informaticup.de.

Anhang A - Beispiele

Beispiele für 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

Beispiele für 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

Beispiele für 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

Beispiele für 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

Beispiele für 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

Beispiele für 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

Anhang 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