NLP: Final Project: Software License Classificator

Due on 10^{th} of May

Teacher: Leon Derczynski

Author: Evgeny Gryaznov

Project Description

Today there exist three primary categories of software licenses:

I. *Open Source.* Those are the licenses which impose either no or little restrictions on a licensee. Examples: MIT, BSD, Mozilla Public License.

II. *Copyleft-like.* Licenses of this type forbid the licensee from rights escalation. For instance, you cannot make a project closed-source if it's under such license. Typical example: GNU GPL.

III. *Proprietary.* A license for which the software's publisher or another person retains intellectual property rights usually copyright of the source code, but sometimes patent rights.

In every category, there are hundreds of licenses, each represented as a text file. It would be great to have some way of automatically determining the type and the name of a given license.
The goal of the project is to develop a program that can identify the category and the name of a license given its text representation. Technical details are the following:

- Programming Language: Python 3.
- Interface: Command-line.
- Input Format: Text file with a license description.
- Output Format: Two strings – license's type and name.

The cool thing about this project is not only that it perfectly fits into course's syllabus, but also has the potential to be implemented via different methods and skills which we learned in this course.
For example, since the described task is an instance of well-known *classification problem*, we can apply different ML techniques to solve it. Finally, the structure of a license file is typically well-formed, thus a number of rule-based methods can be used.

How To Use

In order to solve proposed problems, we developed a Python script called *sappy*. It can automatically infer the name of a license, under which a given Github repository is published. Sappy supports the following arguments:

- `--URL` an URL to a Github repository, which license should be detected.
- `--user` a name of a Github user, whose repository is considered.
- `--repo` a title of a Github repository, which license should be detected.
- `--branch` a branch of a Github repository from which license file should be obtained. Default: 'master'.

For example, in order to detect a license of a repository at `github.com/serverless/serverless` you can execute:

```
$ python3 sappy.py --url https://github.com/serverless/serverless
Output: this repository is probably licensed under MIT license.
```

If you know the username, repository title and branch, you can supply:

```
$ python3 sappy.py --user name --repo title --branch hotfix
```

But you should either specify an URl or username and repository title in order to execute the script.

How It Works

So how does *Sappy* actually works? Basically, it just compares the downloaded license of a repository with prepared templates stored in `license-templates` directory. However, There are three main steps:

1. Parse command-line arguments. We use `argparse` Python module to extract username and repository title.
2. Download license file. We assume that repository contains license in a separate file.
3. Compare downloaded license file with all other license templates. The comparison is done via transforming each license to a TF-IDF vector and then applying cosine similarity.

Also, we stem and lower every document before processing. The accuracy of our program relies on a set of license templates which can be easily enhanced with new documents. After a license name was detected it can be looked up on websites like `tldrlegal.com` in order to get a brief description of such license.

Conclusions

Here we discussed problems that were considered in our project and solutions to them. We wrote a CLI program which can automatically detect a license of a given Github repository. It works by comparing repository's license to prepared license templates.
For comparison process, we choose to transform all documents to a TF-IDF normalized vectors and calculating *cosine similarity* between them. The list of templates can be easily enhanced by new documents, thus increasing the functionality of our program.