# Replication Report of Watchman

Xiaoning Chang

State Key Lab of Computer Science
Institute of Software, Chinese Academy of Sciences
University of the Chinese Academy of Sciences
changxiaoning17@otcaix.iscas.ac.cn

**Introduction** PyPI [1] has indexed millions of libraries to allow developers to automatically download and install dependencies of their projects based on the specified version constraints. The automation combines a server-side central repository and a client-side library installer to manage library dependencies [2]. The version constraint mechanism for a required library allows developers to restrict the dependencies to a set of compatible versions and enables automatic library evolution. Despite the convenience brought by the automation, such version constraints can easily cause dependency conflicts, resulting in build failures [3]. The root cause is that the installed version of a library violates certain version constraints on the library.

When a library updates its version constraints on other libraries, all of it downstream projects could be affected [4]. With the complex dependencies across Python projects and the frequent updates of the libraries on PyPI, the dependency resolution in the world of Python is far from being easy. The ICSE 2020 paper entitled "Watchman: Monitoring Dependency Conflicts for Python Library Ecosystem" presented an effective technique to address this problem. It performs a holistic analysis from the perspective of the entire PyPI ecosystem and continuously monitors dependency conflict issues (DC issues) caused by library updates. In this report, we aim to replicate the experiment results of the paper using the artifact provided by the authors.

The report is organized as follows:

1. Verifying the empirical study dataset
2. Verifying the functional correctness of Watchman
3. Verifying the experiment results reported in Section 5.1
4. Verifying the experiment results reported in Section 5.2
5. Conclusion

## 1. Verifying the empirical study dataset

The 235 DC issues collected in the empirical study (Section 3) are the basis of the work and any noises in the dataset could pose a threat to the validity of the study results. To verify the dataset, we read and analyzed the issues in the provided file empirical_study_dataset.xlsx. The process involved three postgraduates with software engineering background. We labeled the issues as *validated* if their category and corresponding fixing strategies were consistent with our understanding. We performed the labeling independently and discussed to resolve differences.

After cross-validating the 235 DC issues, we confirm that the categorization by Watchman's authors is reasonable and the fixing strategies are also correctly summarized.

## 2. Verifying the functional correctness of Watchman

Watchman is built based on the Browser/Server mode. To check whether it is correctly implemented, we visited its website http://www.watchman-pypi.com/ to interact with the tool. The authors also provided a video demo at https://youtu.be/3EOz9g3Bw70, which is helpful.

As requested by Watchman's authors, for each given Python project released on PyPI, we checked the FDG built by Watchman, which simulates the process of installing dependencies and detecting DC issues for the project. Specifically, for each project, we entered its name and version number in the "**DIAGNOSIS**" page of Watchman's website. Then, we pressed "**Graph**", "**Save**", and "**Start**" buttons to display the project's FDG, save the FDG in a text file, and detect its DC issues (if any). We used the following subjects in this experiment.

(1) First, we tried three example projects with three types of (potential) DC issues, whose name and version number are provided by Watchman's authors in *README.md*:

a. *Pattern A*: moto 1.3.14

b. *Type1*: ldapdomaindump 0.9.1

c. *Type2*: bcdata 0.3.5

(2) Second, we also randomly selected 20 Python projects with the known (potential) DC issues reported by Watchman, which have been confirmed by developers (see Section 5.2 of the paper). Table 1 lists these projects.

Table 1. Information of the 20 randomly selected Python projects

| Project | Version | Issue Type | Project | Version | Issue Type |
|---|---|---|---|---|---|
| toolium | 1.2.5 | Pattern_A | scrapy-redis-bloomfilter-block-cluster | 1.0.1 | Pattern_A |
| InstaPy | 0.6.7 | Pattern_A | indy-node | 1.12.2.dev1183 | Pattern_A |
| pshtt | 0.6.6 | Pattern_A | runcible | 0.0.5 | Pattern_A |
| cert-issuer | 0.0.11 | Pattern_A | polyaxon-client | 0.5.6 | Pattern_A |
| manuscripts | 0.2.20 | Pattern_A | polyaxon-cli | 0.5.6 | Pattern_A |
| PyInquirer | 1.0.3 | Type1 | certstream- | 1.10 | Type1 |
| fiona | 1.8.13 | Type1 | redis-py-cluster | 1.3.6 | Type1 |
| blizzpy | 1.0.3 | Type2 | angr | 8.19.7.25 | Type1 |
| aws-infrastructure-sdk | 2.1.0 | Type2 | azkaban | 0.9.9 | Type2 |
| mythril | 0.9.2 | Type2 | arxiv-submission-core | 0.7.2rc9 | Type2 |

We confirm that the FDGs and diagnosis information generated by Watchman are consistent with the issue reports confirmed by developers on GitHub.

## 3. Verifying the experiment results reported in Section 5.1

We also tried to replicate the experiment results reported in the Section 5.1 of the paper. We performed the replication based on the artifact Replaying evolution history.zip, which is released by the authors at http://www.watchman-pypi.com/history. With the help of the scripts and dataset, we could play back the evolution history of the 16,421 releases of the 2,067 projects.

### 3.1 Experiment setup

The experiment was conducted on a Windows Server (2019 Standard) machine with two Intel Xeon E5 2640v2 @2.00GHz CPUs, 32GB RAM, and 1TB SSD. As recommended by the authors of Watchman, we installed python-3.7.4-amd64.

### 3.2 Replication steps

(1) *Dataset validation*. We first checked the metadata repository of all the library versions on PyPI from 6 November, 2002 to 31 December, 2020, which is available in the archive file pypi_validity_evaluationSQL.zip.

(2) *Replaying the library evolution history*. Then, we ran the scripts provided in Watchman_Artifacts.zip and checked the results.

### 3.2.1 Dataset validation

Verifying the metadata repository is the first and most important step in this replication experiment. To validate the data, we manually checked the following files, which record the dependency relationships between different libraries and the detailed information of all libraries released on PyPI:

a. pypi_validity_evaluationSQL\pypi_info.sql

b. pypi_validity_evaluationSQL\pypi_info_version_all.sql

The authors provided the data sheet format files, which can be directly imported into our database management tool. We randomly sampled ten Python projects and manually downloaded their dependency management files (i.e., requirement.txt or setup.py) and checked whether our collected data were consistent with the records in the provided data sheets.

To validate the historical fixing records of DC issues in the collected projects, we manually checked the file pypi_validity_evaluationSQL\evaluation_info.sql, which contains the records of the DC issues in the collected projects. The fixing records were mined by the Watchman's authors from the projects' issue tracking systems. We randomly sampled ten records and checked whether the date of each sampled DC issue that has been fixed on GitHub was consistent with the records in the data sheet provided by the authors.

We confirm the validity of the provided dataset. We believe that this dataset is useful for future research.

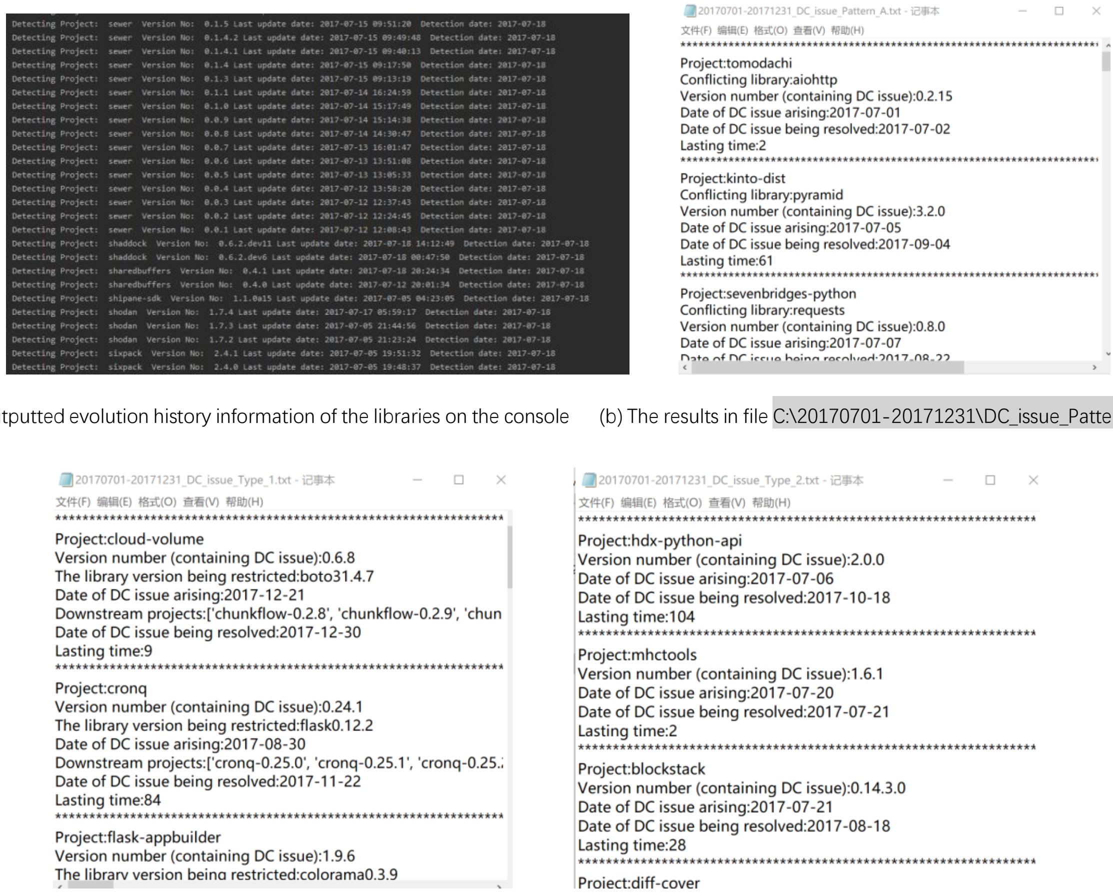
### 3.2.2 Replaying the library evolution history

According to *INSTALL.md*, the authors divided the whole time period into five sub-periods. We randomly selected two periods, namely *Period2* (1 July, 2017 to 31 December, 2017) and

*Period3* (1 January, 2018 to 30 June, 2018), and executed the corresponding scripts in windows console mode for replication:

a. Watchman_Artifacts\validity_evaluation_period2.py
b. Watchman_Artifacts\validity_evaluation_period3.py

When executing the scripts, we can see the outputted evolution history of the libraries in the console and the outputted result files in the directory C:\. Taking *Period2* as an example, the results are shown in Figure 1.



```
Detecting Project:  sewer  Version No:  0.1.5 Last update date: 2017-07-15 09:51:20  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.4.2 Last update date: 2017-07-15 09:49:48  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.4.1 Last update date: 2017-07-15 09:40:13  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.4 Last update date: 2017-07-15 09:17:50  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.3 Last update date: 2017-07-15 09:13:19  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.1 Last update date: 2017-07-14 16:24:59  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.1.0 Last update date: 2017-07-14 15:17:49  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.9 Last update date: 2017-07-14 15:14:38  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.8 Last update date: 2017-07-14 14:30:47  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.7 Last update date: 2017-07-13 16:01:47  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.6 Last update date: 2017-07-13 13:51:08  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.5 Last update date: 2017-07-13 13:05:33  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.4 Last update date: 2017-07-12 13:58:20  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.3 Last update date: 2017-07-12 12:37:43  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.2 Last update date: 2017-07-12 12:24:45  Detection date: 2017-07-18
Detecting Project:  sewer  Version No:  0.0.1 Last update date: 2017-07-12 12:08:43  Detection date: 2017-07-18
Detecting Project:  shaddock  Version No:  0.6.2.dev11 Last update date: 2017-07-18 14:12:49  Detection date: 2017-07-18
Detecting Project:  shaddock  Version No:  0.6.2.dev6 Last update date: 2017-07-18 00:47:50  Detection date: 2017-07-18
Detecting Project:  sharedbuffers  Version No:  0.4.1 Last update date: 2017-07-18 20:24:34  Detection date: 2017-07-18
Detecting Project:  sharedbuffers  Version No:  0.4.0 Last update date: 2017-07-12 20:01:34  Detection date: 2017-07-18
Detecting Project:  shipane-sdk  Version No:  1.1.0a15 Last update date: 2017-07-05 04:23:05  Detection date: 2017-07-18
Detecting Project:  shodan  Version No:  1.7.4 Last update date: 2017-07-17 05:59:17  Detection date: 2017-07-18
Detecting Project:  shodan  Version No:  1.7.3 Last update date: 2017-07-05 21:44:56  Detection date: 2017-07-18
Detecting Project:  shodan  Version No:  1.7.2 Last update date: 2017-07-05 21:23:24  Detection date: 2017-07-18
Detecting Project:  sixpack  Version No:  2.4.1 Last update date: 2017-07-05 19:51:32  Detection date: 2017-07-18
Detecting Project:  sixpack  Version No:  2.4.0 Last update date: 2017-07-05 19:48:37  Detection date: 2017-07-18
```

(a) The outputted evolution history information of the libraries on the console

20170701-20171231_DC_issue_Pattern_A.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
*******************************************************************
Project:tomodachi
Conflicting library:aiohttp
Version number (containing DC issue):0.2.15
Date of DC issue arising:2017-07-01
Date of DC issue being resolved:2017-07-02
Lasting time:2
*******************************************************************
Project:kinto-dist
Conflicting library:pyramid
Version number (containing DC issue):3.2.0
Date of DC issue arising:2017-07-05
Date of DC issue being resolved:2017-09-04
Lasting time:61
*******************************************************************
Project:sevenbridges-python
Conflicting library:requests
Version number (containing DC issue):0.8.0
Date of DC issue arising:2017-07-07
Date of DC issue being resolved:2017-08-22
```

(b) The results in file C:\20170701-20171231\DC_issue_Pattern_A.txt

20170701-20171231_DC_issue_Type_1.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
*******************************************************************
Project:cloud-volume
Version number (containing DC issue):0.6.8
The library version being restricted:boto31.4.7
Date of DC issue arising:2017-12-21
Downstream projects:['chunkflow-0.2.8', 'chunkflow-0.2.9', 'chun
Date of DC issue being resolved:2017-12-30
Lasting time:9
*******************************************************************
Project:cronq
Version number (containing DC issue):0.24.1
The library version being restricted:flask0.12.2
Date of DC issue arising:2017-08-30
Downstream projects:['cronq-0.25.0', 'cronq-0.25.1', 'cronq-0.25.
Date of DC issue being resolved:2017-11-22
Lasting time:84
*******************************************************************
Project:flask-appbuilder
Version number (containing DC issue):1.9.6
The library version being restricted:colorama0.3.9
```

20170701-20171231_DC_issue_Type_2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
*******************************************************************
Project:hdx-python-api
Version number (containing DC issue):2.0.0
Date of DC issue arising:2017-07-06
Date of DC issue being resolved:2017-10-18
Lasting time:104
*******************************************************************
Project:mhctools
Version number (containing DC issue):1.6.1
Date of DC issue arising:2017-07-20
Date of DC issue being resolved:2017-07-21
Lasting time:2
*******************************************************************
Project:blockstack
Version number (containing DC issue):0.14.3.0
Date of DC issue arising:2017-07-21
Date of DC issue being resolved:2017-08-18
Lasting time:28
*******************************************************************
Project:diff-cover
```

(c) The results in file C:\20170701-20171231\DC_issue_Type_1.txt

(d) The results in file C:\20170701-20171231\DC_issue_Type_2.txt

Figure 1. The outputted results of replaying the library evolution history

By manually checking the values of metrics **resolving ratio** and **lasting time**, we obtained the same results as those reported in the Table 3 of the paper. Executing the script of each time period took about 3.5 days.

## 4. Verifying the experiment results reported in Section 5.2

For this replication experiment, we manually checked the following data:

(1) The daily library update information on PyPI captured by Watchman during two time periods (from 1 July, 2019 to 10 August, 2019, and 1 December, 2019 to 31 December, 2019), and the corresponding downstream projects affected by the library updates identified by Watchman.

(2) Diagnosis information and the status of the 279 real issues reported by Watchman to the open-source projects.

## 4.1 Daily library update information

The daily library update information on PyPI captured by Watchman during two time periods (1 July, 2019 to 10 August, 2019 and 1 December, 2019 to 31 December, 2019) are provided on Watchman's website ("UPDATES" page). The authors organized the millions of records of library updates and their corresponding affected downstream projects into an **online searchable table**.

After selecting one date on the "UPDATES" page, we can see all the updated library information. Furthermore, if we select one updated library, all the affected downstream projects will be listed. We sampled ten listed library updates and verified their PyPI records. Besides, we manually downloaded their downstream projects' *requirement.txt* files and checked whether these downstream projects could be affected.

We confirm the validity of the daily library update information captured by Watchman.

## 4.2 Diagnosis information and the status of the reported 279 real issues

For this replication experiment, we manually checked the diagnosis information and the status of the 279 real issues reported by Watchman to the open-source projects. As the status of the issues has changed since the paper submission, we relabeled the issues in the following two tables, where the labels are consistent with the definition in the Watchman paper.

Status 1: The issue has already been fixed using the solution suggested by Watchman authors.

Status 2: The issue has already been fixed using other solutions.

Status 3: The issue has been confirmed and is under fixing using the solution suggested by Watchman authors.

Status 4: The issue has been confirmed and is under fixing using other solutions.

Status 5: The issue is still pending.

Table 2. The 117 issues reported by Watchman from 1 July, 2019 to 10 August, 2019

(These issues were mentioned in the submitted version of the Watchman paper)

| **Manifestation** | **Issue reports** |
|---|---|
| Pattern A.a | Issue#1, aucome; Issue#4, pymacaron; Issue#110, crypto; Issue#1, OrcaSong; Issue#2, pypmml-spark;<br>Issue#138, toolium; Issue#26, GatewayFramework; Issue#1, ScrapyRedisBloomFilterBlockCluster;<br>Issue#3, unblock_youku_gateway; Issue#2, auto_crawler_ptt_beauty_image; Issue#56, Airbnb-data;<br>Issue#2, beauty_image; Issue#1389, Indy-node; Issue#2, Runcible; Issue#95, identification; Issue#96, identification;<br>Issue#212, openpose-plus; Issue#356,Archery; Issue#325, bocadillo; Issue#21, crema;<br>Issue#4, what-digit-you-write; Issue#9, webinfo-crawler; Issue#5, whats-bot; Issue#35, zarp;<br>Issue#4, open-helpdesk; Issue#5, languagecrunch;<br>Issue#103, account-creator; Issue#688, dxr; Issue#39, derrick;<br>Issue#16, Historical-Prices; Issue#2, api-indotel; Issue#9, jawfish;<br>Issue#21, ltiauthenticator;<br>Issue#16, kindred; Issue#545, djangopackages; Issue#13, Generator-GUI;<br>Issue#18526, erpnext; Issue#4778, InstaPy; Issue#198, service-fabric-cli;<br>Issue#1, scrapy-qtwebkit; Issue#3, tabular; Issue#65, armory-bot;<br>Issue#146, django-elasticsearch-dsl-drf; Issue#145, cert-issuer;<br>Issue#2048, cadasta-platform;<br>Issue#1, aoc-mgz-db; Issue#1, twitterbots; Issue#2, gremlin;<br>Issue#17, AWSBucketDump; Issue#130, swapi; Issue#279, explorer;<br>Issue#34 footmark; Issue#3, driver-acs; Issue#56, driver-napi; Issue#11, simulator; Issue#9, Friends-Finder; Issue#1, chatbot-template; Issue#122, adminset;<br>Issue#45, Wallpaper; Issue#28, cryptography; |
| Pattern A.b | Issue#243, bakerydemo; Issue#70, Osmedeus;<br>Issue#4, pytools;<br>Issue#101, aldryn-search; |
| Type 1 | Issue#182, django-dynamic-preferences;<br>Issue#20, ldapdomaindump; Issue#326, redis-py-cluster;<br>Issue#3, GloboNetworkAPI-client-python; Issue#986, faker;<br>Issue#717, newspaper; Issue#120, mixer; Issue#953, django-compressor;<br>Issue#75, PyInquirer; Issue#26, certstream-python; |
| Type 2 | Issue#8, AutoCrawler; Issue#31, BBScan; Issue#2077, freqtrade; Issue#492, pywb;<br>Issue#8, ct-exposer; Issue#71, EagleEyeAC; Issue#1179, mithril; Issue#1, frida-util;<br>Issue#295, sherlock; Issue#36, trains; Issue#4, SecurityManageFramwork;<br>Issue#5, Machine-Learning-with-Python; Issue#34, xbox-smartglass-rest-python-urwid;<br>Issue#167, tldextract; Issue#98, bless; Issue#70, arxiv-submission-core;<br>Issue#2729, plaso; Issue#569, kalliope; Issue#298, glastopf;<br>Issue#17, oauth-dropins;<br>Issue#303, automatic-ripping-machine; Issue#27, ChannelBreakoutBot;<br>Issue#183, messytables; Issue#9, kuberdock-platform; Issue#42, python-weixin;<br>Issue#146, Photon; Issue#911, pyspider; Issue#7, fan; Issue#126, historical;<br>Issue#49, stephanie-va; Issue#979, subliminal; Issue#56, WPSeku;<br>Issue#3, zhihu-crawler-people; Issue#38, Pendingwisp-network-topology;<br>Issue#647, marathon-lb; Issue#962, hangoutsbot; Issue#41, GyoiThon;<br>Issue#120, automation-tools; Issue#4, app-engine-start-vm; Issue#10, ahmia-index;<br>Issue#25, NoDB;Issue#9, Konan; Issue#181, JBPOS |

| **Pattern A.a:** 60 | **Pattern A.b:** 4 | **Type1:** 10 | **Type2:** 43 |
|---|---|---|---|
| Status 1: 33 | Status 1: 2 | Status 1: 1 | Status 1: 13 |
| Status 2: 3 | Status 2: 1 | Status 2: 0 | Status 2: 0 |
| Status 3: 6 | Status 3: 0 | Status 3: 4 | Status 3: 6 |
| Status 4:3 | Status 4: 0 | Status 4: 0 | Status 4: 1 |
| Status 5: 15 | Status 5: 1 | Status 5: 5 | Status 5: 23 |

Table 3. The 162 issues reported by Watchman from 1 December, 2019 to 31 December, 2019

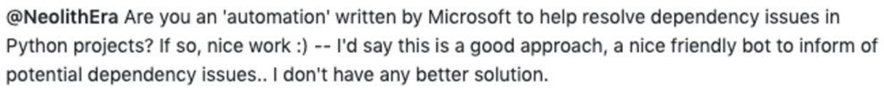
(These issues were detected after the paper submission)

| Manifestation | Issue reports |
|---|---|
| Pattern A.a | Issue#12, afg; Issue#3, django-html-emailer; Issue#1, Generator; Issue#438, geokey; Issue#2, imgsync;<br>Issue#1, iprange-python; Issue#16, manheim-c7n-tools; Issue#245, program-y;<br>Issue#246, program-y; Issue#10, portia2code; Issue#1, scrapbag;<br>Issue#131, scvelo; Issue#18, mockerena; Issue#457, nornir;<br>Issue#4, pyexcel-export; Issue#115, pyGenealogical-Tools;<br>Issue#253, reana-cluster; Issue#18, smap_io; Issue#19, Spartacus;<br>Issue#39, ssmash;<br>Issue#5170, cloud-custodian; Issue#3, pyclics-clustering;<br>Issue#7, textX-languageserver; Issue#153, django-glitter; Issue#30, django-glitter-events;<br>Issue#28, django-glitter-news; Issue#2622, moto;<br>Issue#2, TMO4CT;<br>Issue#114, pyGenealogical-Tools; Issue#14, twitter_markov; Issue#28, tweetynet; Issue#12, baiji-serialization;<br>Issue#64, taar-lite; Issue#764, sockeye;<br>Issue#5, globomap-driver-acs; Issue#3, aclpwn.py;<br>Issue#2, arbok; Issue#1, dedis-cluster; Issue#212, django-cumulus; Issue#3, django-fperms-iscore;<br>Issue#3, django-es-utils; Issue#41, django-prices-openexchangerates; Issue#115, eGO; Issue#1, flask-mongokat;<br>Issue#1, HelpDeskBot; Issue#160, mlpiper; Issue#2, musco-tf; Issue#4, Operation-Pluto;<br>Issue#6, protean-elasticsearch; Issue#1, c7n; Issue#48, kik-python; Issue#276, mapbox-sdk-py; Issue#11, pymacaron;<br>Issue#272, python-bitshares; Issue#2, scPopCorn;<br>Issue#5, py-redis-pubsub-dict; Issue#24, rdfframework;<br>Issue#30, raredecay; Issue#31, raredecay; Issue#32, raredecay; Issue#66, Scriptax; Issue#2, StandardLibrary;<br>Issue#2, Scriptax-Runtime; Issue#4, gmssl; Issue#5, gmssl;<br>Issue#6, gmssl; Issue#7, SurfinPy; Issue#18, target-datadotworld; Issue#2, django-television;<br>Issue#3, django-television; Issue#26, tralXroute; Issue#1760, superdesk-core; Issue#4, musco-pytorch; |
| Pattern A.b | Issue#136, grimoirelab-manuscripts; Issue#36, polyaxon-client;<br>Issue#39, polyaxon-cli; Issue#40, polyaxon-cli;<br>Issue#54, polyaxon-schemas; Issue#271, transifex-client;<br>Issue#272, transifex-client; Issue#55, polyaxon-schemas;<br>Issue#377, money-to-prisoners-common; Issue#378, money-to-prisoners-common; Issue#208, pshtt;<br>Issue#1, dork; Issue#4, CommonMark-py-Extensions;<br>Issue#1, pymacaron-dynamodb; Issue#1, pymacaron-async;<br>Issue#1, Scriptax-Jupyter-Kernel; Issue#1, pymacaron-google-datastore; Issue#2, Scriptax-Jupyter-Kernel;<br>Issue#2, 1a23-telemetry; |
| Type 1 | Issue#986, faker; Issue#26, certstream-python; Issue#69, click-pathlib;<br>Issue#182, django-dynamic-preferences;<br>Issue#20, ldapdomaindump; Issue#326, redis-py-cluster; Issue#3, GloboNetworkAPI-client-python;<br>Issue#75, PyInquirer; Issue#1062, dash; Issue#1092, faker; Issue#2, chaosplatform-grpc; Issue#1902, angr;<br>Issue#1063, dash; Issue#1064, dash; Issue#1065, dash; Issue#850, Fiona;<br>Issue#717, newspaper; Issue#120, mixer; Issue#953, django-compressor;<br>Issue#3, chiki-base; Issue#1, Commandtax; Issue#68, dcplib; |
| Type 2 | Issue#8, AutoCrawler; Issue#31, BBScan; Issue#2077, freqtrade; Issue#298, glastopf;<br>Issue#492, pywb; Issue#8, ct-exposer; Issue#71, EagleEyeAC; Issue#1179, mithril;<br>Issue#1, frida-util; Issue#295, sherlock; Issue#36, trains; Issue#4, SecurityManageFramwork;<br>Issue#5, Machine-Learning-with-Python; Issue#181, JBOPS;<br>Issue#34, xbox-smartglass-rest-python-urwid; Issue#127, allofplos;<br>Issue#4, api-mocker-generator; Issue#772, appscale-tools; Issue#26, Arachnid;<br>Issue#9, AsyncLine; Issue#3, augur; Issue#68, bcdata; Issue#148, aws-adfs;<br>Issue#454, awslimitchecker; Issue#2, bigga;<br>Issue#167, tldextract; Issue#98, bless; Issue#70, arxiv-submission-core; Issue#2729, plaso; Issue#9, Konan;<br>Issue#569, kalliope; Issue#4, ansible-runner-http; Issue#1042, python; Issue#7, argo-models;<br>Issue#1, AwsInfrastructureSdk; Issue#2, aws2; Issue#83, awscfncli; Issue#42, azkaban; Issue#1, BlizzPy;<br>Issue#2, aceql-http-client-python; Issue#7655, certbot; Issue#7, apispec-oneofschema;<br>Issue#184, tldextract; Issue#310, bioblend;<br>Issue#25, NoDB; Issue#17, oauth-dropins;<br>Issue#303, automatic-ripping-machine; Issue#27, ChannelBreakoutBot; Issue#183, messytables;<br>Issue#9, kuberdock-platform; Issue#42, python-weixin; Issue#146, Photon; Issue#911, pyspider;<br>Issue#7, fan; Issue#126, historical; Issue#49, stephanie-va; Issue#979, subliminal; Issue#56, WPSeku;<br>Issue#3, zhihu-crawler-people; Issue#38, Pendingwisp-network-topology; Issue#647, marathon-lb;<br>Issue#962, hangoutsbot; Issue#41, GyoiThon; Issue#120, automation-tools; Issue#4, app-engine-start-vm;<br>Issue#10, ahmia-index; Issue#118, agavepy; Issue#2, agavedb; Issue#32, AGFusion; Issue#232, pyensembl;<br>Issue#7, ambient_api; Issue#7, ambient_aprs; Issue#45, requests-unixsocket; Issue#26, appscale-agents;<br>Issue#1, artem; Issue#1, async_bowl; Issue#78, atomicpuppy; Issue#1, AwsEcrMigration; Issue#6, autoSubTakeover;<br>Issue#3, autochomsky; Issue#1, aws-assume; Issue#37, awscli-login; Issue#12, awsmfa; Issue#5, awswl;<br>Issue#1, awsutils-s3; Issue#10, basecampy3; Issue#1, axonbot_slack; Issue#11, basecampy3; Issue#12, basecampy3;<br>Issue#1, basalunit; Issue#1040, python-zeep; Issue#18, betdaq; Issue#3, biomaj2galaxy; Issue#466, python-binance;<br>Issue#1, binance-db; Issue#6, python-biopipe; Issue#467, python-binance; Issue#2, binance-db;<br>Issue#7, python-biopipe; Issue#8, python-biopipe; Issue#179, bookstore; |

| **Pattern A.a:** 73 | **Pattern A.b:** 19 | **Type1:** 22 | **Type2:** 101 |
|---|---|---|---|
| Status 1: 20 | Status 1: 8 | Status 1: 4 | Status 1: 25 |
| Status 2: 8 | Status 2: 0 | Status 2: 0 | Status 2: 0 |
| Status 3: 4 | Status 3: 3 | Status 3: 12 | Status 3: 19 |
| Status 4: 2 | Status 4: 0 | Status 4: 0 | Status 4: 2 |
| Status 5: 39 | Status 5: 8 | Status 5: 6 | Status 5: 55 |

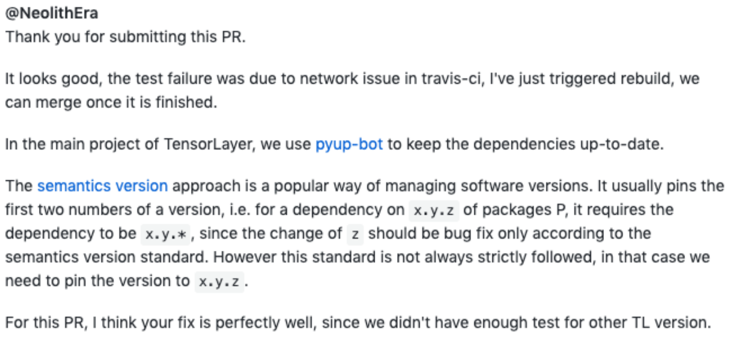
Based on the above statistics, for the first time period, i.e., from 1 July, 2019 to 10 August, 2019, the overall confirmation rate by developers is 73/117 = 62.3% and the corresponding fixing rate is (53/73) = 72.6%, which are higher than those reported in the submitted version of the paper. For the second time period, i.e., from 1 December, 2019 to 31 December, 2019, the overall confirmation rate is 107/162 = 66.0% and the fixing rate is 65/107=60.5%.

By reading the issue reports, we found some comments from developers, which show the usefulness and effectiveness of Watchman, such as:



**@NeolithEra** Are you an 'automation' written by Microsoft to help resolve dependency issues in Python projects? If so, nice work :) -- I'd say this is a good approach, a nice friendly bot to inform of potential dependency issues.. I don't have any better solution.

https://github.com/webrecorder/pywb/pull/494



**@NeolithEra**
Thank you for submitting this PR.

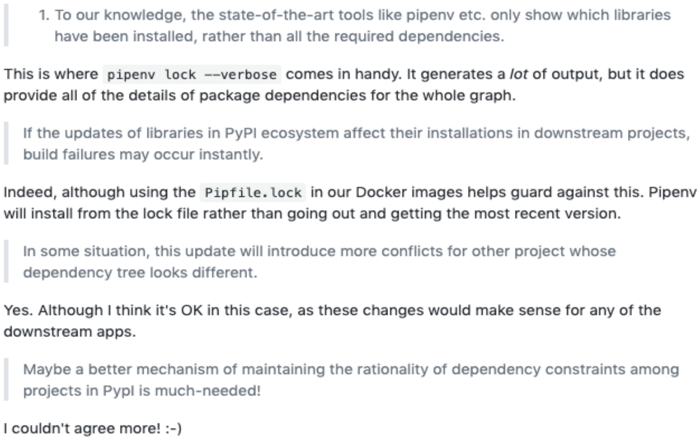
It looks good, the test failure was due to network issue in travis-ci, I've just triggered rebuild, we can merge once it is finished.

In the main project of TensorLayer, we use pyup-bot to keep the dependencies up-to-date.

The semantics version approach is a popular way of managing software versions. It usually pins the first two numbers of a version, i.e. for a dependency on `x.y.z` of packages P, it requires the dependency to be `x.y.*`, since the change of `z` should be bug fix only according to the semantics version standard. However this standard is not always strictly followed, in that case we need to pin the version to `x.y.z`.

For this PR, I think your fix is perfectly well, since we didn't have enough test for other TL version.

https://github.com/tensorlayer/openpose-plus/pull/213



> 1. To our knowledge, the state-of-the-art tools like pipenv etc. only show which libraries have been installed, rather than all the required dependencies.

This is where `pipenv lock --verbose` comes in handy. It generates a *lot* of output, but it does provide all of the details of package dependencies for the whole graph.

> If the updates of libraries in PyPI ecosystem affect their installations in downstream projects, build failures may occur instantly.

Indeed, although using the `Pipfile.lock` in our Docker images helps guard against this. Pipenv will install from the lock file rather than going out and getting the most recent version.

> In some situation, this update will introduce more conflicts for other project whose dependency tree looks different.

Yes. Although I think it's OK in this case, as these changes would make sense for any of the downstream apps.

> Maybe a better mechanism of maintaining the rationality of dependency constraints among projects in Pypl is much-needed!

I couldn't agree more! :-)

https://github.com/arXiv/arxiv-submission-core/pull/71

## 5. Conclusion

We successfully replicated the experiment results presented in the Watchman paper. We tried Watchman's online service and found that the tool can generate correct FDGs as well as diagnosis information for DC issues. We also validated the released dataset. We believe that the dataset can benefit future research on DC issues in the Python ecosystem.

## References

[1] PyPI. https://pypi.org/.

[2] Alexandre Decan, Tom Mens, and Maelick Claes. 2016. On the topology of package dependency networks: A comparison of three programming language ecosystems. In Proceedings of the 10th European Conference on Software Architecture Workshops. ACM, 21.

[3] Dependency specification for Python. https://www.python.org/dev/peps/pep-0508/.

[4] Pietro Abate and Roberto Di Cosmo. 2011. Predicting upgrade failures using dependency analysis. In 27th IEEE International Conference on Data Engineering Workshops. IEEE, 145–150.