Steps for construct TensorFlow-based SSC

- Step 1. Collecting packages on TensorFlow SSC. Commencing with TensorFlow, we initiate the collection of dependents from the PACKAGE-type dependency graph. TensorFlow itself had 3,352 dependents, constituting the first layer. Subsequently, we gather the dependents of each package in the first layer, designating them as the second layer. This iterative process continued, accumulating downstream dependents, reaching up to the sixth layer. In total, we collected 18,642 packages across these layers, with each layer comprising 3,352, 4,754, 4,353, 1,460, 1,261, and 3,462 packages, respectively. It's important to mention that when recording the position of packages within SSC, we mark a package shallower if it is a dependent of packages in different layers. For instance, if a package relies on both first-layer and second-layer packages, we designate it as belonging to the second -layer.
- Step 2. Collecting dependency information between versions of packages. First, we filter out packages that have at least one released version. Out of the collected packages, 9,597 meet this criterion, with the distribution across each layer being 1,710, 2,438, 2,036, 810, 705, and 1,898, respectively. Subsequently, we proceed to gather dependency information for the various versions within each package. GitHub offers support for multiple formats of dependency files across several package ecosystems, such as *requirements.txt, pipfile, pipfile.lock*, and *setup.py* for the PyPI ecosystem. Leveraging the GitHub API, we are able to access the versions for each package and retrieve the dependency files in these four formats. This allows us to obtain comprehensive information about the package's dependencies on the target packages across different versions. After further analysis, these selected packages collectively account for a total of 117,853 versions.
- Step 3. TensorFlow SSC construction. With the collected dependency information, we can build a dependency network of versions. However, our primary focus lies on the versions of packages susceptible to influence by TensorFlow. Due to inconsistent dependency information present among different versions of the same package, a particular version of a package in our current collection may not be linked to TensorFlow within the dependency network. For instance, the repository *lucidrains/enformer-pytorch* depends on *transformers* and indirectly on TensorFlow, but only after version 0.4.0; versions prior to 0.4.0 should not be included in TensorFlow SSC. Hence, we undertake a filtering process to retain only the versions of packages that genuinely belong to the SSC. We continue to utilize TensorFlow as the initial reference point for constructing the SSC between the first-layer packages and various versions of TensorFlow. This is achieved by scrutinizing the dependency information of each version of the first-layer package on the corresponding TensorFlow version. Any packages or versions with incomplete dependency information are systematically removed. Subsequently, we employ the same methodology to establish the SSC connections between the first-layer packages and the second-layer packages. This process is repeated iteratively for the subsequent layers of packages.