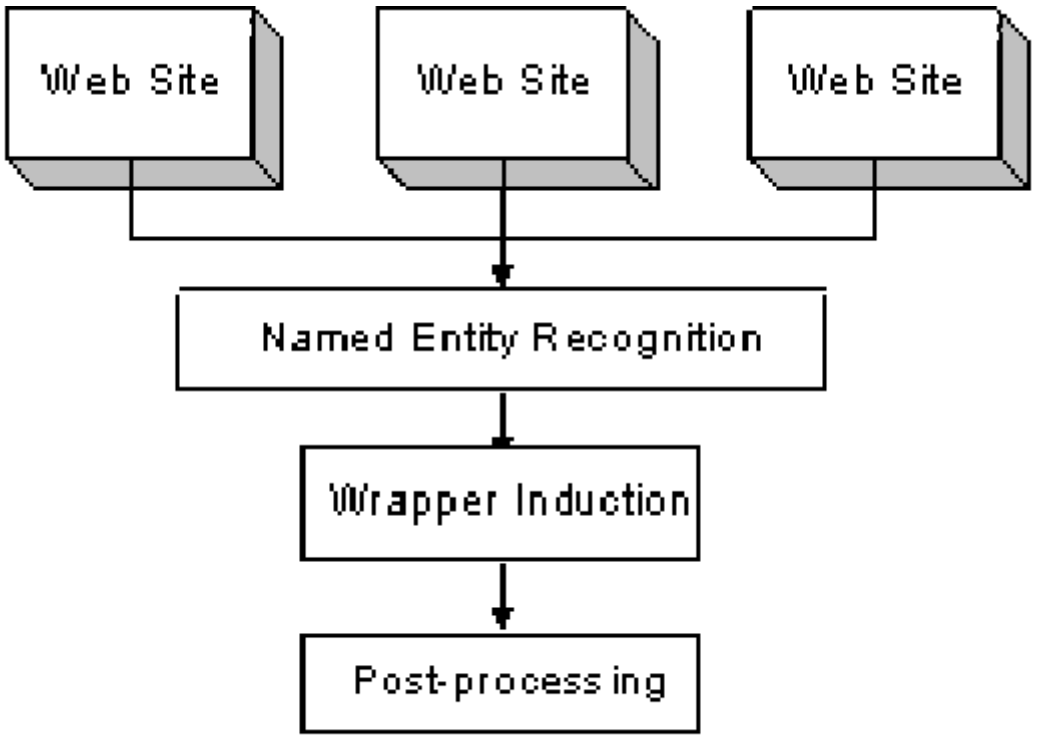
JUNAID GIRKAR | 60004190057 | BE COMPS A2 | WEB INTELLIGENCE

EXPERIMENT - 3

AIM: Implement a wrapper induction technique to gather data from the web

THEORY:
Wrapper induction (or query induction) is a subfield of wrapper generation, which itself belongs to the broader field of information extraction (IE). In IE, wrappers transform unstructured input into structured output formats, and a wrapper generation systems describes the transformation rules involved in such transformations. Wrapper induction is a solution to wrapper generation where transformation rules are learned from examples and counterexamples (inductive learning). The induced wrapper subsequently is applied to unseen input documents to collect further label relations of interest.
To ease annotation of examples by the user, the learning framework is often implemented within a visual annotation environment, where the user selects and deselects elements visually.



JUNAID GIRKAR | 60004190057 | BE COMPS A2 | WEB INTELLIGENCE

CODE & OUTPUT:

```
import requests
from swi import Extractor

extractor = Extractor()
```

```
url1 = "https://pypi.org/project/pip/20.2.1/"
response = requests.get(url1)
train_set_1 = {
    "name":"pip 19.2.1",
    "maintainers":["acsbidoul", "xafer"],
    "description" : "The PyPA recommended tool for installing Python packages."
}
```

```
extractor.add_train_page(response.content, train_set_1)

extractor.predict(response.content)
```

```
{'name': 'pip 19.2.1',
 'maintainers': ['acsbidoul',
  'dstufft',
  'pf_moore',
  'pradyunsg',
  'uranusjr',
  'xafer'],
 'description': 'The PyPA recommended tool for installing Python packages.'}
```

```
url1 = "https://pypi.org/project/lxml/"
response2 = requests.get(url1)
extractor.predict(response2.content)
```

```
{'name': 'lxml 4.9.2',
```

```
 'maintainers': ['faassen', 'scoder', 'zope.wineggbuilder'],
 'description': 'Powerful and Pythonic XML processing library combining
libxml2/libxslt with the ElementTree API.'}
```

```
url2 = "https://pypi.org/project/django/"
response2 = requests.get(url2)
extractor.predict(response2.content)
```

```
{'name': 'Django 4.2',
 'maintainers': ['felixx', 'nessita', 'ubernostrum'],
 'description': 'A high-level Python web framework that encourages rapid
development and clean, pragmatic design.'}
```

CONCLUSION: Due to the manual labeling effort, it is hard to extract data from a large number of sites as each site has its own templates and requires separate manual labeling for wrapper learning. Wrapper maintenance is also a major issue because whenever a site changes the wrappers built for the site become obsolete. Due to these shortcomings, researchers have studied automated wrapper generation using unsupervised pattern mining. Automated extraction is possible because most Web data objects follow fixed templates. Discovering such templates or patterns enables the system to perform extraction automatically.