Semantic Web - Project

Group - 11

Khyati Seth (2016050)
Richa Goswami (2016077)
Suyash Singh (2016105)

Issue -1

KeyError: 'http://schema.org/item' #96

Link : https://github.com/digitalbazaar/pyld/issues/96

Issue: pyld generates a keyerror 'http://schema.org/item'

For the code :

```
from pyld import jsonld
x = jsonld.frame(
  {
    "@context": "http://schema.org",
    "@graph": [
      {
        "@type": "BreadcrumbList",
        "itemListElement": [
          {
            "@type": "ListItem",
            "item": {
              "@id": "https://example.com/",
              "@type": "WebPage",
              "url": "https://example.com/"
            }
          }
        ]
      }
    ]
  },
  {}
)
```

The problem seems to be in the frame function which in turn redirects to _remove_embed function with a series of calls.

Upon running the code in Jupyter notebook and tracking the error we made it to the final function call of _remove_embed :

```
~/anaconda3/lib/python3.7/site-packages/pyld/jsonld.py in _remove_embed(self, state, id_)
   3782         else:
   3783             # replace subject with reference
-> 3784             use_array = _is_array(embed['parent'][property])
   3785             JsonLdProcessor.remove_value(
   3786                 embed['parent'], property, subject,

KeyError: 'http://schema.org/item'
```

There may be a problem in the input syntax for the given framing query as both @id and url are not required and if one of them is removed then the code generates a frame as per the requirements.

```
Out[12]: {'@graph': [{'@type': 'http://schema.org/BreadcrumbList',
          'http://schema.org/itemListElement': {'@id': '_:b1',
           '@type': 'http://schema.org/ListItem',
           'http://schema.org/item': {'@id': '_:b2',
            '@type': 'http://schema.org/WebPage',
            'http://schema.org/url': {'@id': 'https://example.com/'}}}},
         {'@id': '_:b1',
          '@type': 'http://schema.org/ListItem',
          'http://schema.org/item': {'@id': '_:b2',
           '@type': 'http://schema.org/WebPage',
           'http://schema.org/url': {'@id': 'https://example.com/'}}},
         {'@id': '_:b2',
          '@type': 'http://schema.org/WebPage',
          'http://schema.org/url': {'@id': 'https://example.com/'}},
         {'@id': 'https://example.com/'}]}
```

This error is not supposed to be a key error.

<u>Checking the Test case:</u>

```
x = jsonld.frame(
  {
   "@context": "http://schema.org",
   "@graph": [
     {
      "@type": "BreadcrumbList",
      "itemListElement": [
        {
         "@type": "ListItem",
         "item": {
          "@id": "https://example.com/",
          "@type": "WebPage",
```

```
        "url": "https://example.com/"
      }
    }
  ]
 }
]
},
{})
```

Expected Output:

```
Out[36]: {'@graph': [{'@type': 'http://schema.org/BreadcrumbList',
          'http://schema.org/itemListElement': {'@id': '_:b1',
           '@type': 'http://schema.org/ListItem',
           'http://schema.org/item': {'@id': 'https://example.com/',
            '@type': 'http://schema.org/WebPage'}}},
         {'@id': '_:b1',
          '@type': 'http://schema.org/ListItem',
          'http://schema.org/item': {'@id': 'https://example.com/',
           '@type': 'http://schema.org/WebPage'}},
         {'@id': 'https://example.com/', '@type': 'http://schema.org/WebPage'}]}
```

<u>Our Approach</u>

The above error of schema.org was not giving an error in the pyld as keyerror, rather it showed as a JsonLd error. The problem in the syntax was that the URL given for context was not returning the correct JSON-LD document.

```
In [13]: x = jsonld.frame(
             {
             "@context": "https://schema.org",
             "@graph": [
               {
                 "@type": "BreadcrumbList",
                 "itemListElement": [
                   {
                     "@type": "ListItem",
                     "item": {
                       "@id": "https://example.com/",
                       "@type": "WebPage",
                       "url": "https://example.com/"
                     }
                   }
                 ]
               }
             ]
           }
           ,
           {}
           )

JsonLdError: ('Could not retrieve a JSON-LD document from the URL.',)
Type: jsonld.LoadDocumentError
Code: loading document failed
Cause: Expecting value: line 1 column 1 (char 0)  File "/home/suyash-singh/anaconda3/lib/python3.7/site-packa
ges/pyld/documentloader/requests.py", line 72, in loader
    'document': response.json()
```

This issue is technically a syntactic issue in JSON-LD however upon running this problem in json-ld playground the output was as follows:

Processing error:

```
jsonld.InvalidUrl: Dereferencing a URL did not result in a valid JSON-LD object. Possible causes are an inaccessible URL perhaps due to a same-origin policy (ensure the server uses CORS if you are using client-side JavaScript), too many redirects, a non-JSON response, or more than one HTTP Link Header was provided for a remote context.
```

Expanded | Compacted | Flattened | Framed | N-Quads | Normalized | Table | Visualized | Signed with RSA | Signed with Bitcoin

```
"@graph": [
  {
    "@id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
```

But the output for framed did work:


```
{
  "@graph": [
    {
      "@id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "http://schema.org/alumniOf": {
        "@type": "http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML",
        "@value": "Example University"
      }
    },
    {
      "@id": "https://example.com/credentials/1872",
      "@type": [
        "https://www.w3.org/2018/credentials#VerifiableCredential",
        "https://json-ld.org/playground/AlumniCredential"
      ],
      "https://www.w3.org/2018/credentials#credentialSubject": {
        "@id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
        "http://schema.org/alumniOf": {
          "@type": "http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML",
          "@value": "Example University"
        }
```


Which means that there was scope of handling Schema.org as context too.

We browsed through the json-ld playground github to check if there was any possible solution but it was all written in js and html, and was difficult to figure out. So we searched for any other alternating way to dereference some other schema.org url which might provide a json-ld object. Upon searching we found a page which had scope of solving this issue.

Vocabulary Definition Files

To assist developers, files containing the definition of the core Schema.org vocabulary and its extensions are available for download in common RDF formats.

Older releases can be found (under data/releases/) at GitHub.

Select the file and format required and click Download. The CSV format downloads are split accross two files: *Types* includes definitions of Types and Enumeration Values, including lists of associated properties; *Properties* contains property definitions.

Note: File *schema* contains the definition of the core vocabulary; *bib* contains only the definitions for the bib.schema.org extension; *all-layers* contains definitions for the core plus all the extensions.

File: schema ▾ Format: JSON-LD ▾

/version/latest/schema.jsonld

Download

If we replaced schema.org with schema.org/version/latest/schema.jsonld then we might have a json ld object that we want to retrieve.
Upon opening the above link we get a json ld object of schema.org schemas.


```
{
  "@context": {
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@graph": [
    {
      "@id": "http://schema.org/downloadUrl",
      "@type": "rdf:Property",
      "http://schema.org/domainIncludes": {
        "@id": "http://schema.org/SoftwareApplication"
      },
      "http://schema.org/rangeIncludes": {
        "@id": "http://schema.org/URL"
      },
      "rdfs:comment": "If the file can be downloaded, URL to download the binary.",
      "rdfs:label": "downloadUrl"
    },
    {
      "@id": "http://schema.org/UserLikes",
      "@type": "rdfs:Class",
      "http://schema.org/supersededBy": {
        "@id": "http://schema.org/InteractionCounter"
      },
      "rdfs:comment": "UserInteraction and its subtypes is an old way of talking about users interacting with pages. It is generally better to use <a
class=\"localLink\" href=\"http://schema.org/Action\">Action</a>-based vocabulary, alongside types such as <a class=\"localLink\"
href=\"http://schema.org/Comment\">Comment</a>.",
      "rdfs:label": "UserLikes",
      "rdfs:subClassOf": {
        "@id": "http://schema.org/UserInteraction"
      }
    },
    {
      "@id": "http://schema.org/CafeOrCoffeeShop",
      "@type": "rdfs:Class",
      "rdfs:comment": "A cafe or coffee shop.",
      "rdfs:label": "CafeOrCoffeeShop",
      "rdfs:subClassOf": {
        "@id": "http://schema.org/FoodEstablishment"
      }
    },
    {
      "@id": "http://schema.org/pagination",
```


However, upon changing the @context as schema.org/version/latest/schema.jsonld the output in both playground and code were coming same but not same as the output expected.

Output code:

```
In [7]:  x = jsonld.frame(
           {
             "@context": "http://schema.org/version/latest/schema.jsonld",
             "@graph": [
               {
                 "@type": "BreadcrumbList",
                 "itemListElement": [
                   {
                     "@type": "ListItem",
                     "item": {
                       "@id": "https://example.com/",
                       "@type": "WebPage",
                       "url": "https://example.com/"
                     }
                   }
                 ]
               }
             ]
           },
           {}
         )

In [8]:  print(x)

         {'@graph': [{'@type': '/BreadcrumbList'}]}
```

Playground output:

```
{
  "@context": "http://schema.org/version/latest/schema.jsonld",
  "@graph": [
    {
      "@type": "BreadcrumbList",
      "itemListElement": [
        {
          "@type": "ListItem",
          "item": {
            "@id": "https://example.com/",
            "@type": "WebPage",
            "url": "https://example.com/"
          }
        }
      ]
    }
  ]
}
```

Expanded | Compacted | Flattened | Framed | N-Quads | Non

```
{
  "@type": "https://json-ld.org/playground/BreadcrumbList"
}
```

Since, this is the only plausible way so we have handled the schema.org issue via replacing in the context where schema.org is coming to "http://schema.org/version/latest/schema.jsonld"
The following changes are made in the code (line 779-781 and 4844-4846) :

```
776
777     # if input is a string, attempt to dereference remote document
778     #schema org error to be handled here
779     if _is_object(input_) and "@context" in input_.keys() and _is_string(input_["@context"]) and (input_["@context"] == "h
780             input_["@context"]= input_["@context"].replace("schema.org", "schema.org/version/latest/schema.jsonld")
781     if _is_string(input_):
782         remote_doc = options['documentLoader'](input_)
783     else:
784         remote_doc = {
785             'contextUrl': None,
786             'documentUrl': None,
787             'document': input_
788         }
789
```

```
4840
4841                # retrieve URL
4842                try:
4843                    # print(url)
4844                    if url == "https://schema.org" or url == "http://schema.org":
4845                        url = "https://schema.org/version/latest/schema.jsonld"
4846                    remote_doc = load_document(url)
4847                    ctx = remote_doc['document']
4848
4849                except Exception as cause:
```

First update is made in the expand method where the context is checked, if there are any schema.org then it'll be handled there. Second update is made in _retrieve_context_urls, in case there are any schema.org urls present in the urls while parsing.

Post these changes the new output on schema.org will be as follows:

```
In [2]:  x = jsonld.frame(
           {
             "@context": "http://schema.org",
             "@graph": [
               {
                 "@type": "BreadcrumbList",
                 "itemListElement": [
                   {
                     "@type": "ListItem",
                     "item": {
                       "@id": "https://example.com/",
                       "@type": "WebPage",
                       "url": "https://example.com/"
                     }
                   }
                 ]
               }
             ]
           },
           {}
         )

In [3]:  print(x)

         {'@graph': [{'@type': '/BreadcrumbList'}]}
```

The output is not exactly as expected output but there is scope of improvement in this solution. However, the problem of schema.org key-error has been handled in this case.
The json-ld document retrieved from the "http://schema.org/version/latest/schema.jsonld" site is as follows:

```
document retrieved from URL: http://schema.org/version/latest/schema.jsonld
{'@context': {'rdf': 'http://www.w3.org/1999/02/22-rdf-syntax-ns#', 'rdfs': 'http://www.w3.org/2000/01/rdf-sc
hema#', 'xsd': 'http://www.w3.org/2001/XMLSchema#'}, '@graph': [{'@id': 'http://schema.org/downloadUrl', '@ty
pe': 'rdf:Property', 'http://schema.org/domainIncludes': {'@id': 'http://schema.org/SoftwareApplication'}, 'h
ttp://schema.org/rangeIncludes': {'@id': 'http://schema.org/URL'}, 'rdfs:comment': 'If the file can be downlo
aded, URL to download the binary.', 'rdfs:label': 'downloadUrl'}, {'@id': 'http://schema.org/UserLikes', '@ty
pe': 'rdfs:Class', 'http://schema.org/supersededBy': {'@id': 'http://schema.org/InteractionCounter'}, 'rdfs:c
omment': 'UserInteraction and its subtypes is an old way of talking about users interacting with pages. It is
generally better to use <a class="localLink" href="http://schema.org/Action">Action</a>-based vocabulary, alo
ngside types such as <a class="localLink" href="http://schema.org/Comment">Comment</a>.', 'rdfs:label': 'User
Likes', 'rdfs:subClassOf': {'@id': 'http://schema.org/UserInteraction'}}, {'@id': 'http://schema.org/CafeOrCo
ffeeShop', '@type': 'rdfs:Class', 'rdfs:comment': 'A cafe or coffee shop.', 'rdfs:label': 'CafeOrCoffeeShop',
'rdfs:subClassOf': {'@id': 'http://schema.org/FoodEstablishment'}}, {'@id': 'http://schema.org/pagination',
'@type': 'rdf:Property', 'http://purl.org/dc/terms/source': {'@id': 'http://www.w3.org/wiki/WebSchemas/Schema
DotOrgSources#source_bibex'}, 'http://schema.org/domainIncludes': [{'@id': 'http://schema.org/Chapter'}, {'@i
d': 'http://schema.org/PublicationVolume'}, {'@id': 'http://schema.org/PublicationIssue'}, {'@id': 'http://sc
hema.org/Article'}], 'http://schema.org/rangeIncludes': {'@id': 'http://schema.org/Text'}, 'http://www.w3.or
g/2002/07/owl#equivalentProperty': {'@id': 'http://purl.org/ontology/bibo/pages'}, 'rdfs:comment': 'Any descr
```

Which means that the document is retrieved but not properly parsed, probably we might need to implement a module which will parse it and expand our json-ld graph as per the expected output so that we get a final framed document as per our expectations.
This issue is hence, partially solved.

Issue -2

JSON Literals not supported via @json KEYWORD #100

Link - [JSON Literals not supported via @json KEYWORD · Issue #100 · digitalbazaar/pyld]

@json keyword is not supported in pyld as opposed to what is given in the official Json-LD documentation
Explanation of Json literals - https://w3c.github.io/json-ld-syntax/#json-literals

JSON-LD has a special feature of handling the cases for graph nodes with type "@json". For such objects, during expansion their value is purely json serializable and hence, the expected output of these types of objects were purely handling such cases as pyld has not considered the case for such Json-literals.
Screenshot of the exception occurring when the test case is entered:

```
from pyld.jsonld import expand

document = expand(
 {
  "@context": {
    "@version": 1.1,
    "e": {"@id": "http://example.com/vocab/json", "@type": "@json"}
  },
  "e":[1,2,3]
 }
)
```

```
---------------------------------------------------------------------------
JsonLdError                               Traceback (most recent call last)
<ipython-input-24-2ff7f0c4ea56> in <module>
      7     "e": {"@id": "http://example.com/vocab/json", "@type": "@json"}
      8   },
----> 9   "e":[1,2,3]
     10 }
     11 )

~/anaconda3/lib/python3.7/site-packages/pyld/jsonld.py in expand(input_, options
    159     :return: the expanded JSON-LD output.
    160     """
--> 161     return JsonLdProcessor().expand(input_, options)
    162
    163
```

```
JsonLdError: ('Invalid JSON-LD syntax; an @context @type value must be an absolute IRI.',)
Type: jsonld.SyntaxError
Code: invalid type mapping
Details: {'context': {'@version': 1.1, 'e': {'@id': 'http://example.com/vocab/json', '@type': '@json'}}}
```

Stack Trace - Exceptions raised

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
Traceback (most recent call last):
  File "/home/khyati/PycharmProjects/swebproject/sweb2.py", line 10, in <module>
    "e": [1,2,3]
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 161, in expand
    return JsonLdProcessor().expand(input_, options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 835, in expand
    expanded = self._expand(active_ctx, None, document, options, False)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2098, in _expand
    active_ctx, element['@context'], options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2833, in _process_context
    self._create_term_definition(rval, ctx, k, defined)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 4464, in _create_term_definition
    {'context': local_ctx}, code='invalid type mapping')
pyld.jsonld.JsonLdError: ('Invalid JSON-LD syntax; an @context @type value must be an absolute IRI.',)
Type: jsonld.SyntaxError
Code: invalid type mapping
Details: {'context': {'@version': 1.1, 'e': {'@id': 'http://example.com/vocab/json', '@type': '@json'}}}

Process finished with exit code 1
```

Our Approach

We started working recursively starting from the last exception that was raised in the stack trace. The last exception that was raised stated that {'context': local_ctx}, code='invalid type mapping'). A context maps a term to an IRI or a map. The pyld.jsonld.JsonLdError was ('Invalid JSON-LD syntax; an @context @type value must be an absolute IRI.',). Here, the @type refers to the type assigned to the JSON object node in context. It is basically a map which links the IRI representing the type of the object to the object. The error showed that @type was not an absolute IRI for which we checked the function _is_absolute_iri(type_)

```
4455            if type_ != '@id' and type_ != '@vocab':
4456                # expand @type to full IRI
4457                type_ = self._expand_iri(
4458                    active_ctx, type_, vocab=True,
4459                    local_ctx=local_ctx, defined=defined)
4460                if not _is_absolute_iri(type_):
4461                    raise JsonLdError(
4462                        'Invalid JSON-LD syntax; an @context @type value must '
4463                        'be an absolute IRI.', 'jsonld.SyntaxError',
4464                        {'context': local_ctx}, code='invalid type mapping')
4465                if type_.startswith('_:'):
4466                    raise JsonLdError(
4467                        'Invalid JSON-LD syntax; an @context @type values '
4468                        'must be an IRI, not a blank node identifier.',
4469                        'jsonld.SyntaxError', {'context': local_ctx},
4470                        code='invalid type mapping')
```

The function _is_absolute_iri(type_) was defined as follows and returned true if the type_ had an absolute IRI. The absolute IRI is the full IRI of the referenced object.

```python
def _is_absolute_iri(v):
    """
    Returns True if the given value is an absolute IRI, False if not.

    :param v: the value to check.

    :return: True if the value is an absolute IRI, False if not.
    """
    return _is_string(v) and ':' in v


def _is_relative_iri(v):
    """
```

_is_absolute_iri()

The error was hence in the type_ variable. For resolving this, the following addition was made in the second debug point (line 4460). This issue reverts us back to the case where type_ variable is being handled. General strategy followed in the case of a normal @type instance is that it requires the value of @type to be an absolute IRI by default, which is further expanded using function _expand_iri function (see line 4353 in the image below for reference).

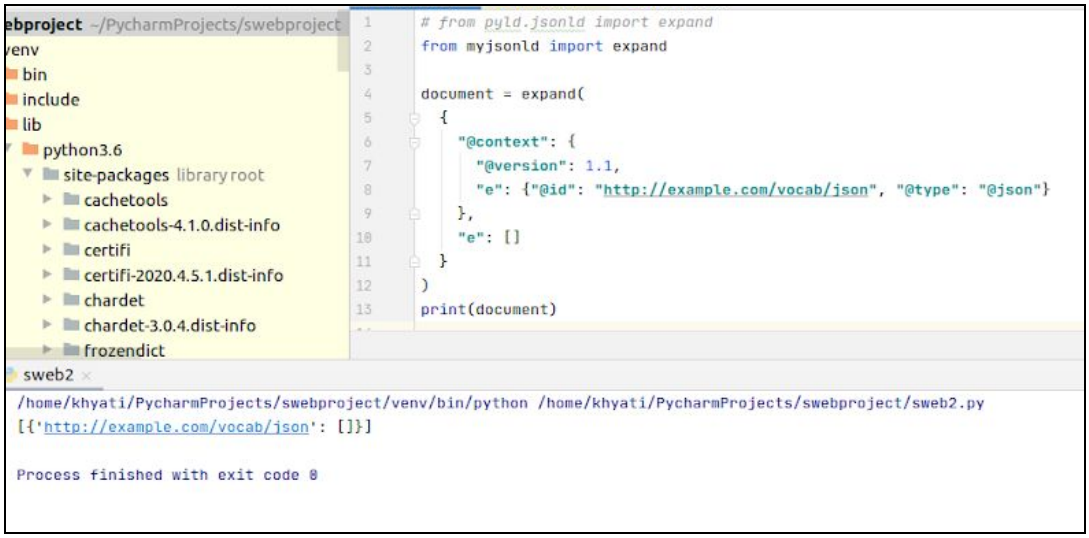
Handling the issue:

This was our first point of debugging this issue i.e. handling the special @json value for type (see line 4527-4532 in the image below) and checking the version of active context by using _processing_mode method. This Update in code resolved the @json issue that we presented in our report.

```python
4524            ## Handling types in there
4525            if type_ != '@id' and type_ != '@vocab':
4526                # Handling the case of @json
4527                if type_ == '@json':
4528                    if self._processing_mode(active_ctx, 1.0):
4529                        raise JsonLdError(
4530                            "JSON-LD pre processing error")
4531                    else:
4532                        print("@Type is @Json")
```

This resolution of error successfully removed the exception and started producing the results. The first test case we tried was for a blank list. The result was as follows:

Test Case-1



Next, we tried running the code for our first test case as described in the proposal. This test case though removed the exception, the result was slightly different than the result that was produced in JSON playground.

Test case -2:

```
from jsonld import expand
document = expand(
  {
   "@context": {
    "@version": 1.1,
    "e": {"@id": "http://example.com/vocab/json", "@type": "@json"}
   },
   "e": [1,2,3]
  }
)
print(document)
```

Output when run in JSON Playground

```
[
  {
    "http://example.com/vocab/json": [
      {
        "@type": "@json",
        "@value": [
          1,
          2,
          3
        ]
      }
    ]
  }
]
```

Output shown in the console

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
[{'http://example.com/vocab/json': [{'@type': '@json', '@value': 1}, {'@type': '@json', '@value': 2}, {'@type': '@json', '@value': 3}]}]

Process finished with exit code 0
```

Another test case that we tried for identifying this issue was a case wherein the @type was @json explicitly. The test case produced a different output too.
This problem arises in all the cases where the node is type @json and has value as a complex json object rather than just one plain list. Due to this, all our advanced cases, that we extracted from the Json-ld page on w3.org , were giving empty objects along with the main value as only the topmost node in the list (56.0 in the example below). This part of the issue was a tough challenge as the error that we were getting was a logical error than a runtime or compile time error. Upon understanding the code of this project we found a spot where we missed to handle the values of special @type as @json.

Test Case-3

```
from jsonld import expand
document = expand( {
 "@context": {
  "@version": 1.1,
  "e": {"@id": "http://example.com/vocab/json", "@type": "@json"}
 },
 "e": [
  56.0,
  {
   "d": True,
   "10": None,
   "1": [ ]
```

```
  }
 ]
})
print(document)
```

Output when run in JSON Playground

```
[
  {
    "http://example.com/vocab/json": [
      {
        "@type": "@json",
        "@value": [
          56,
          {
            "1": [],
            "10": null,
            "d": true
          }
        ]
      }
    ]
  }
]
```

Output shown in the console

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
[{'http://example.com/vocab/json': [{'@type': '@json', '@value': 56.0}, {}]}]

Process finished with exit code 0
```

Upon observing the _expand_object module, we realized that there was no way the module was internally handling these special @json cases. The reason why we got the output of test case 2 and 3 (as in their figures above) was because the _expand_object module was handling the list items as per the algorithm which was created to handle any normal literal in a list. This was our debugging point number 2 where we had to handle the special case of @type as @json.
In order to do so, the following update was made to the code (see image below for reference line 2430: 2432).

```
2421            # handle index container (skip if value is not an object)
2422            elif '@index' in container and _is_object(value):
2423                expanded_value = self._expand_index_map(term_ctx, key, value, '@index', '@graph' in container, options)
2424            elif '@id' in container and _is_object(value):
2425                expanded_value = self._expand_index_map(term_ctx, key, value, '@id', '@graph' in container, options)
2426            elif '@type' in container and _is_object(value):
2427                expanded_value = self._expand_index_map(term_ctx, key, value, '@type', False, options)
2428            else:
2429                # recurse into @list or @set
2430                is_list = (expanded_property == '@list')
2431                # Handling the json serializable objects as values @json type ##New_Update :
2432                if JsonLdProcessor.get_context_value(active_ctx, key, '@type') == '@json':
2433                    # print ("JSON Value")
2434                    # print(value)
2435                    expanded_value = {
2436                        '@type': '@json',
2437                        '@value': value
2438                    }
2439                elif is_list or expanded_property == '@set':
```

After this update, we again tried Test case 3 and the following output was produced on the console which is satisfactory as it matches with the output of json-ld playground :

Our Output

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
@Type is @Json
[{'http://example.com/vocab/json': [{'@type': '@json', '@value': [56.0, {'d': True, '10': None, '1': []}]}]}]

Process finished with exit code 0
```

PlayGround Output:

```
[
  {
    "http://example.com/vocab/json": [
      {
        "@type": "@json",
        "@value": [
          56,
          {
            "1": [],
            "10": null,
            "d": true
          }
        ]
      }
    ]
  }
]
```

Lastly, an even more advanced case was designed and tried with a @json type node which has a json object embedded in another Json object ({“internal_key”:[1,2,3,4]} embedded in the main document). It produced the same result as that expected by running the test case in JSON Playground.

Test Case-4

```
from jsonld import expand
document = expand( {
 "@context": {
  "@version": 1.1,
  "e": {"@id": "http://example.com/vocab/json", "@type": "@json"}
 },
 "e": [
  56.0,
  {
   "d": True,
   "10": None,
   "1": {"internal_key": [1,2,3,4]}
```

```
    }
  ]
})
print(document)
```

Output when run in JSON Playground

```
[
  {
    "http://example.com/vocab/json": [
      {
        "@type": "@json",
        "@value": [
          56,
          {
            "1": {
              "internal_key": [
                1,
                2,
                3,
                4
              ]
            },
            "10": null,
            "d": true
          }
        ]
      }
```

Output shown in the console

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
@Type is @Json
[{'http://example.com/vocab/json': [{'@type': '@json', '@value': [56.0, {'d': True, '10': None, '1': {'internal_key': [1, 2, 3, 4]}}]}]}]

Process finished with exit code 0
```

This issue is hence completely solved.

Issue -3

Issue with jsonld.normalize(). #99

Link - **https://github.com/digitalbazaar/pyld/issues/99**

Issue: Normalize function does not work on many examples which have JsonLD 1.1 spec.
Example for one such case is :

```
from pyld.jsonld import expand

document =  {
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "https://example.com/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": "Example University"
  }
}
jsonld.normalize(document)
```

```
   1137                    'Could not expand input before serialization to '
-> 1138                    'RDF.', 'jsonld.RdfError', cause=cause)
   1139

JsonLdError: ('Could not expand input before serialization to RDF.',)
Type: jsonld.RdfError
Cause: ('Invalid JSON-LD syntax; @context property values must be strings or objects.',)
Type: jsonld.SyntaxError
Code: invalid term definition
```

However it works fine for the previous standards like:

```
from pyld.jsonld import expand

document =  {
    "@context": "http://schema.org",
    "@graph": [
      {
        "@type": "BreadcrumbList",
        "itemListElement": [
          {
            "@type": "ListItem",
            "item": {
              "@id": "https://example.com/",
              "@type": "WebPage",
              "url": "https://example.com/"
            }
          }
        ]
      }
    ]
}
jsonld.normalize(document)
```

```
{'@default': [{'subject': {'type': 'IRI', 'value': 'https://example.com/'},
   'predicate': {'type': 'IRI', 'value': 'http://schema.org/url'},
   'object': {'type': 'IRI', 'value': 'https://example.com/'}},
  {'subject': {'type': 'IRI', 'value': 'https://example.com/'},
   'predicate': {'type': 'IRI',
    'value': 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type'},
   'object': {'type': 'IRI', 'value': 'http://schema.org/WebPage'}},
  {'subject': {'type': 'blank node', 'value': '_:c14n0'},
```

For the given test case:

```
from pyld import jsonld
import requests
document =  {
 "@context": [
   "https://www.w3.org/2018/credentials/v1",
   "https://www.w3.org/2018/credentials/examples/v1"
 ],
 "id": "https://example.com/credentials/1872",
 "type": ["VerifiableCredential", "AlumniCredential"],
```

```
 "issuanceDate": "2010-01-01T19:23:24Z",
 "credentialSubject": {
   "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
   "alumniOf": "Example University"
 }
}
jsonld.normalize(document)
```

Output produced when run on JSON playground:

Expanded | Compacted | Flattened | Framed | N-Quads | Normalized | Table | Visualized | Signed with RSA | Signed with Bitcoin

```
<did:example:ebfeb1f712ebc6f1c276e12ec21> <http://schema.org/alumniOf> "Example University"^^<http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML> .
<https://example.com/credentials/1872> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://json-ld.org/playground/AlumniCredential> .
<https://example.com/credentials/1872> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://www.w3.org/2018/credentials#VerifiableCredential> .
<https://example.com/credentials/1872> <https://www.w3.org/2018/credentials#credentialSubject> <did:example:ebfeb1f712ebc6f1c276e12ec21> .
<https://example.com/credentials/1872> <https://www.w3.org/2018/credentials#issuanceDate> "2010-01-01T19:23:24Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

Output produced on the console:

```
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
Traceback (most recent call last):
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 1134, in to_rdf
    expanded = self.expand(input_, options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 835, in expand
    expanded = self._expand(active_ctx, None, document, options, False)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2098, in _expand
    active_ctx, element['@context'], options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2833, in _process_context
    self._create_term_definition(rval, ctx, k, defined)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 4343, in _create_term_definition
    {'context': local_ctx}, code='invalid term definition')
pyld.jsonld.JsonLdError: ('Invalid JSON-LD syntax; @context property values must be strings or objects.',)
Type: jsonld.SyntaxError
Code: invalid term definition
Details: {'context': {'@version': 1.1, '@protected': True, 'id': '@id', 'type': '@type', 'VerifiableCredential': {'@id': 'https://www.w3.org/2018/credentials#VerifiableCredential', '@context': {'@version': 1.1, '@protect

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 1052, in normalize
    dataset = self.to_rdf(input_, opts)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 1138, in to_rdf
    'RDF.', 'jsonld.RdfError', cause=cause)
pyld.jsonld.JsonLdError: ('Could not expand input before serialization to RDF.',)
Type: jsonld.RdfError
Cause: ('Invalid JSON-LD syntax; @context property values must be strings or objects.',)
Type: jsonld.SyntaxError
Code: invalid term definition
Details: {'context': {'@version': 1.1, '@protected': True, 'id': '@id', 'type': '@type', 'VerifiableCredential': {'@id': 'https://www.w3.org/2018/credentials#VerifiableCredential', '@context': {'@version': 1.1, '@protect
    expanded = self.expand(input_, options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 835, in expand
    expanded = self._expand(active_ctx, None, document, options, False)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2098, in _expand
    active_ctx, element['@context'], options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2833, in _process_context
    self._create_term_definition(rval, ctx, k, defined)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 4343, in _create_term_definition
    {'context': local_ctx}, code='invalid term definition')
```

Working recursively on the stack trace, the last exception, is as follows:

```
4332
4333        # convert short-hand value to object w/@id
4334        _simple_term = False
4335        if _is_string(value):
4336            _simple_term = True
4337            value = {'@id': value}
4338
4339        if not _is_object(value):
4340            raise JsonLdError(
4341                'Invalid JSON-LD syntax; @context property values must be '
4342                'strings or objects.', 'jsonld.SyntaxError',
4343                {'context': local_ctx}, code='invalid term definition')
4344
4345        # create new mapping
4346        mapping = active_ctx['mappings'][term] = {'reverse': False}
4347
4348        # make sure term definition only has expected keywords
4349        valid_keys = ['@container', '@id', '@language', '@reverse', '@type']
4350        if self._processing_mode(active_ctx, 1.1):
JsonLdProcessor › _create_term_definition() › if not _is_object(value)
```

The error displayed is {'context': local_ctx}, code='invalid term definition'). Here, it says that the @context property could only accept values in strings or objects. In order to track this we had to go through the different functions which overlapped with functions like expand, etc.

Now the issue in PYLD is that each function is called multiple times within others. Here, we traced the entire exception stack for the above error but it could not be resolved. This was because the context was referenced in the expansion function, which was then used in normalize function.

```
Type: jsonld.SyntaxError
Code: invalid term definition
Details: {'context': {'@version': 1.1, '@protected': True, 'id': '@id', 'type': '@type', 'VerifiableCredential': {'@id': 'https://www.w3.org/2
    expanded = self.expand(input_, options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 835, in expand
    expanded = self._expand(active_ctx, None, document, options, False)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2098, in _expand
    active_ctx, element['@context'], options)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 2833, in _process_context
    self._create_term_definition(rval, ctx, k, defined)
  File "/home/khyati/PycharmProjects/swebproject/venv/lib/python3.6/site-packages/pyld/jsonld.py", line 4343, in _create_term_definition
    {'context': local_ctx}, code='invalid term definition')
```

```
834         # do expansion
835         expanded = self._expand(active_ctx, None, document, options, False)
836
837         # optimize away @graph with no other properties
838         if (_is_object(expanded) and '@graph' in expanded and
839                 len(expanded) == 1):
840             expanded = expanded['@graph']
841         elif expanded is None:
842             expanded = []
843
844         # normalize to an array
845         return JsonLdProcessor.arrayify(expanded)
```

We moved to the second issue since it also referenced the expansion function, hoping that if we solve the second issue and the subsequent errors in the expansion function, the normalization

issue would become easier to solve. However, even after thoroughly understanding the expansion algorithm, the normalization issue could not be solved. This was also because of the repeated context problem which occurred.

After going through the case of issue 2, we came back to analyze this issue. the issue in this case is because of the context provided in the list are dereferenced to two different jsonld objects which need to be resolved, however the context resolution in the _process_context function is not properly resolving the issue, as the error is pertained in one of the cases from the jsonld document retrieved which was of type boolean.

So, in order to handle this we reiterated and analyzed the _process_context function, as to where we can handle this. In_process_context, the function _create_term_definition is being called which resolves each context, while working on the same, we found that when @context had a value of non object, the _create_term_definition function used to raise an error.

Upon printing and checking for each case, the problem arose that many jsonld object might have a boolean type of value, this is handled by adding a condition to check if the value is boolean then simply return from _create_term_definition function as there is no need to resolve a boolean type value.

Below is the change in code:

```
#Handling issue 3:
if _is_bool(value):
    return


if not _is_object(value):

    raise JsonLdError(
        'Invalid JSON-LD syntax; @context property values must be '
        'strings or objects.', 'jsonld.SyntaxError',
        {'context': local_ctx}, code='invalid term definition')
```

It handles the case when value is boolean and gives proper output. Thus, for the above mentioned test case, the following is the output produced on JSON playground followed by the output that is generated on the console.

Expected Output (Output when run on JSONLD Playground)

Expanded | Compacted | Flattened | Framed | N-Quads | Normalized | Table | Visualized | Signed with RSA | Signed with Bitcoin

```
<did:example:ebfeb1f712ebc6f1c276e12ec21> <http://schema.org/alumniOf> "Example University"^^<http://www.w3.org/1999/02/22-rdf-syntax-
ns#HTML> .
<https://example.com/credentials/1872> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://json-
ld.org/playground/AlumniCredential> .
<https://example.com/credentials/1872> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<https://www.w3.org/2018/credentials#VerifiableCredential> .
<https://example.com/credentials/1872> <https://www.w3.org/2018/credentials#credentialSubject> <did:example:ebfeb1f712ebc6f1c276e12ec21>
.
<https://example.com/credentials/1872> <https://www.w3.org/2018/credentials#issuanceDate> "2010-01-
01T19:23:24Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

Output on the console

```
sweb2
/home/khyati/PycharmProjects/swebproject/venv/bin/python /home/khyati/PycharmProjects/swebproject/sweb2.py
{'@default': [{'subject': {'type': 'IRI', 'value': 'did:example:ebfeb1f712ebc6f1c276e12ec21'}, 'predicate': {'type': 'IRI', 'value': 'http://schema.org/alumniOf'}, 'object': {'type': 'literal', 'datatype':
'http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML', 'value': 'Example University'}}, {'subject': {'type': 'IRI', 'value': 'https://example.com/credentials/1872'}, 'predicate': {'type': 'IRI', 'value': 'http://www.w3
.org/1999/02/22-rdf-syntax-ns#type'}, 'object': {'type': 'IRI', 'value': 'https://www.w3.org/2018/credentials#VerifiableCredential'}}, {'subject': {'type': 'IRI', 'value': 'https://example.com/credentials/1872'},
'predicate': {'type': 'IRI', 'value': 'https://www.w3.org/2018/credentials#credentialSubject'}, 'object': {'type': 'IRI', 'value': 'did:example:ebfeb1f712ebc6f1c276e12ec21'}}, {'subject': {'type': 'IRI', 'value':
'https://example.com/credentials/1872'}, 'predicate': {'type': 'IRI', 'value': 'https://www.w3.org/2018/credentials#issuanceDate'}, 'object': {'type': 'literal', 'datatype': 'http://www.w3
.org/2001/XMLSchema#dateTime', 'value': '2010-01-01T19:23:24Z'}}]}

Process finished with exit code 0
```

This issue is hence, completely solved.

Project - PyLD

Project URL - https://github.com/digitalbazaar/pyld

Issue 1 URL - https://github.com/digitalbazaar/pyld/issues/96

Issue 2 URL - https://github.com/digitalbazaar/pyld/issues/100

Issue 3 URL - https://github.com/digitalbazaar/pyld/issues/99

How to run the demo:

Go in the folder lib/pyld/

Using jupyter notebook run the test_case.ipynb

Or simply open the test_case.py and run it.