Vol.2018-CH-118 No.6
2018/8/18

古典中国語 UD コーパスの IPFS を用いた表現の試み

守岡 知彦[1]

概要：Universal Dependencies (UD) を用いた古典中国語の係り受けコーパスの開発において、語の認定、どの語からどの語に係り受け関係を設定するか、品詞や係り受けの種類をどうするかなどさまざまな項目を検討する必要があり、こうした任意の要素が試行錯誤でき、さまざまな選択肢を比較できることが望ましい。また、複数人で共同作業する場合、各人が独立に試行錯誤できることが望ましいと考えられ、それらの試行錯誤の結果を簡単に共有できることが望ましいと考えられる。このような観点に基づき、内容ベースのアドレッシングを採用したデータモデルである IPLD とこれに基づく分散型ファイルシステムである IPFS を用いたデータ表現の試みについて述べる。

1. はじめに

京都大学人文科学研究所共同研究班「東アジア古典文献コーパスの実証研究」では 2017 年度より古典中国語（漢文）に対する Universal Dependencies (UD)[1] の適用可能性について議論し、UD に基づく係り受けコーパスの記述に関する研究を行っている。[2] その結果、UD を用いて古典中国語の係り受けコーパスを書くことには一定の妥当性があり、本格的にコーパス作成を進めて行くことになった。

しかし、そのためには編集用ツールやデータ管理の仕組み、そして、実際にコーパス開発を行って行く上で、形態素コーパスの場合 [3] と同様、さまざまな試行錯誤を行う必要があると考えられ、用例を検討し試行錯誤の結果生まれた揺れを把握・制御するための手法が求められるといえる。特に、複数人で共同作業する場合、各人が独立に試行錯誤できることが望ましいと考えられ、それらの試行錯誤の結果を簡単に共有できることが望ましいと考えられる。

形態素コーパスはそこで扱う構造は比較的単純であるが、係り受けコーパスではより複雑な構造を扱う必要があるといえる。形態素コーパスで編集対象となるのは（少なくともデータ上は）文の区切り（何を形態素と認定するか）と品詞だけであったが、UD を用いた古典中国語の係り受けコーパスの開発において、語の認定や品詞の付け方に加えて、どの語を root にしどの語を節や句における head と考えるか、そして、どの語からどの語にどのような依存関係を設定するかを決める必要があり、また、UD では一部の依存関係を決めるためにはある語の並びが節と句のどちらかなのかを決める必要がある。古典中国語において節と句の区別はそれほど明白ではないので、ある程度恣意的に決めざるを得ないと考えられるが、ある程度の量の例文を比較検討し UD のガイドラインや他の言語での運用となるべく齟齬を来さないような書き方を見つける必要があるといえる。

古典中国語は扱う時代が長く、また、古い時代の文章の引用が多いため、どの時代の文法を基準に考えるかが難しい場合があるが、このことは慣用される言い回しを 1 語と看做すかどうかの選択が難しいケースを引き起こし得ることを意味している。即ち、古い時代の文法を基準に考えれば複数の語の間の依存関係のグラフと看做す方が自然だが、後代では 1 語と看做した方が自然というようなケースである。また、そうした場合以外にも、長単位で書くか短単位で書くかというような問題もある。大きな単位での依存関係は単純で書きやすく、短い単位で依存関係を書こうとすると複雑で書きにくいものとなりやすいものとなるため、こうした単位の問題は作業効率の問題にも影響する場合があるといえる。我々の場合、短単位と長単位の対象データを持っていないため、必然的に短単位で書かざるを得ないといえるが、それゆえに、慣用される言い回しを適切に把握する仕組みを用意することは重要であるといえる。

UD では CoNLL-U 形式が標準のデータ形式として用いられているが、この形式では語に付与された行番号 (ID) によって依存関係を記述するため、複数の文で同じ依存構造をその部分構造として持っている場合でも（偶然番号が一致しない限り）同じ記述にならないという問題がある。そこで、なるべく ID に依存しないデータ構造を内部的に採ることが望ましいといえる。

このような観点に基づき、内容に基づくアドレッシング

[1] 京都大学人文科学研究所
Institute for Research in Humanities, Kyoto University

を採用したデータモデルである IPLD とこれに基づく分散型ファイルシステムである IPFS を用いたデータ表現の試みについて述べる。

2. IPFS

IPFS (InterPlanetary File System) [4] [5] は Protocol Labs 社が中心となって開発しているオープンソースの P2P 型分散型ファイルシステムである。IPFS の開発は Git 等で版管理された科学的データを高速に転送するシステムの構築を目的として Juan Benet によって始められたが、後に、分散化され永続化された Web として構想されるようになった。

IPFS では Git と同様にオブジェクトをそのハッシュ値によって参照するという内容に基づくアドレッシング (content-addressing) を用いている。通常の Web では URL (IRI) という資源の場所に基づくアドレッシング (location-addressing) になっており、識別子の指す先が変化する可能性があり、また、オブジェクトに変化がなくてもその識別子が変化してしまうと参照できないという性質を持っている。このことは学術資源の長期保存においては良い性質とはいえず、このため、DOI のような永続的識別子の利用が注目されているといえる。この観点から IPFS を見ると、IPFS では DOI のような永続的識別子を任意のオブジェクトに対して機械的に生成する仕組みと見ることができる。IPFS ではオブジェクトのハッシュ値に基づいて内容 ID (CID; Content IDentifier) を生成する仕組みがプロトコルによって定義されており、いつどこで誰が行っても対象とするオブジェクトが同一のバイト列を持っていれば同じ CID が生成されるため、DOI や DNS のように機関を認証して権限を委譲する必要はなく、結果的に、非常に小さな粒度のデータから非常に大きなデータセットの集合といったスケーラビリティーで永続的な識別子を生成することができる。

IPFS は

アプリケーション	
IPNS	名前解決
IPLD	Merkle DAG によるデータモデル
libp2p	ネットワークやルーチング、データ転送

という 4 層で構成されており、下位の 3 層及び IPFS が提供する幾つかのアプリケーション[*1]に関して、JavaScript と Go 言語による実装が提供されている。[*2]

2.1 IPLD

IPLD (Inter Planetary Linked Data) [6] [7] は IPFS のデータモデルを提供する層で、Merkle DAG と呼ばれる暗号化されたハッシュに基づく有向非巡回グラフ (Directed Acyclic Graph; DAG) と、Merkle paths と呼ばれる Merkle DAG を名前付きリンクをたどってトラバースしたものを UNIX 風の階層的ファイル名に対応させる仕組みと、IPLD の正規化されたデータ形式 (IPLD Canonical Format) 等の仕様により、JSON や XML 等の外部形式を IPLD の世界に格納したり IPLD の世界のオブジェクトを指定した外部形式で出力することができる。

IPLD は IPFS のデータモデルであるので、IPFS 上のさまざまなアプリケーションは全て IPLD の DAG として記述されたデータを持っているといえるが、ここでは IPFS の Go 実装 (go-ipfs) の ipfs コマンドの ipfs dag {get|put} や IPFS HTTP API の /api/v0/dag/{get|put} で読み書きできる IPLD dag-cbor 形式を用いるものとする。

IPLD のオブジェクト（DAG のノードとなるもの）は JSON のオブジェクトと同様な key と value の対の集合である。

例えば、{ ”name”: ”Paul Marie Ghislain Otlet” } は

`zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo`

という CID を持つ IPLD のオブジェクトの JSON 表現である。

IPLD に JSON のオブジェクトを格納する場合、IPFS のようにバイト列そのものが格納される訳ではなく、正規化された CBOR (Concise Binary Object Representation) [8] 形式に変換されて格納される。

例えば、go-ipfs の ipfs コマンドを用いて、

```
% echo '{ ”a”: 1, ”b”: 2, ”c”: 3 }' | ipfs dag put
```

を実行すると、

`zdpuAmNMQJdQumv32j5DDUGBzk92aZmXcg2x78Tqrj3pPV2Ze`

という CID を得るが、

```
% echo '{ ”c”: 3, ”a”: 1, ”b”: 2 }' | ipfs dag put
```

や

```
% echo '{”a”:1,”b”:2,”c”:3}' | ipfs dag put
```

でも

`zdpuAmNMQJdQumv32j5DDUGBzk92aZmXcg2x78Tqrj3pPV2Ze`

という CID になり、空白・改行の有無・個数や key-value 対の順番に関らず同じオブジェクトとして扱われることが判る。

また、現状、go-ipfs の dag サブコマンドでは JSON 以外の形式がサポートされていないが、原理的には、XML や YAML, RDF といった形式であっても、それが同一の key-value 対の集合を表現しているならば同一のオブジェクトとして扱われることになっている。

IPLD では他のオブジェクトへのリンクは key が `"/"` で value が CID である link-object と呼ばれる特殊なオブジェクトで表現される。例えば、

```
{ "/":
```

[*1] 分散ファイルシステムとしての IPFS もこのアプリケーション層に位置するものと看做すことができる。
[*2] go-ipfs と js-ipfs はこのプロトコルスタックの実装をパッケージにまとめたもののようである。

```
    "zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo"
  }
```

は link-object の JSON 表現である。また、

```
{ "name":  "Henri La Fontaine",
  "collaborator":
    { "/":
      "zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo"
    }
}
```

を JSON 表現として持つ IPLD オブジェクト

`zdpuAoDrX6sbMNE5N3i7YTrVuEidCL41EoSryCr5y8UhrR4PC`

のようにある key の value に他のオブジェクトへのリンクやリンクの配列を入れることができ、これにより DAG を構成することができる。

Merkle-paths の規定により、リンクを持つオブジェクトはリンクを値として持つ key を用いて、CID/key のような派生的 CID によってリンク先を参照することができる。

例えば、

```
% ipfs dag get \
zdpuAoDrX6sbMNE5N3i7YTrVuEidCL41EoSryCr5y8UhrR4PC/collaborator
```

は {"name":"Paul Marie Ghislain Otlet"} という出力を返す。

もし、リンク先にも同様に名前付きのリンクがある場合、CID/a/b/c/... のように階層的にリンクをたどることができる。

3. 古典中国語 UD の階層化

UD は語と語の間の依存関係によって構文構造を記述する。例えば、「孟子見梁惠王」の依存構造は図 1 のようになる。ここで「梁惠王」は全体として人名という一つのまとまりを構成しており、句構造文法で記述するならばこれを一つの名詞句としてブロック化しその下に語をぶら下げることができるが、UD ではあくまで語と語の依存関係だけで記述しなければならず、部分的な構造をブロック化することはできない。この場合に許される選択肢は「梁惠王」を一つの語と看做すか「梁」「惠」「王」に分解してその間の関係を記述するかだけである。このことは作業者によって語の認定基準が異なった場合、揺れが生じてしまうことを意味している。

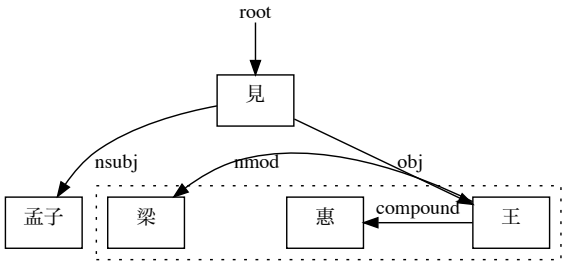

図 1 「孟子見梁惠王」の依存構造

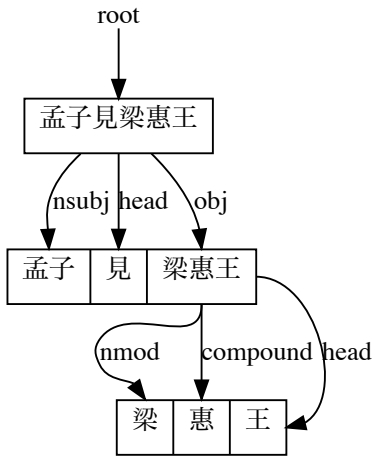

図 2 「孟子見梁惠王」の依存構造の階層化

複雑に入り組んだ文の依存構造を記述する場合、大きな単位（ブロック）での依存関係を記述する方が語と語の依存関係を一度に全て記述するよりも簡単なことが多いが、もし、ブロックをまず語と看做してブロック間（あるいはブロックと語）の依存関係を記述し、その後にブロック内部の依存関係を記述するという段階的な依存構造記述が可能であれば作業がやりやすくなるかも知れない。

こうしたことを考慮し、図 2 のように、部分構造のヘッドとなる部分（部分構造を文と看做した場合に root になる部分）とそこから依存関係で指されている部分が連続している場合、それをブロック化したデータ構造を考え、これを『(UD の）依存構造の階層化』と呼ぶことにする。

4. IPLD での表現

UD の依存構造は DAG で記述可能であるので、語と語との依存関係で記述する本来の UD の構造をそのまま IPLD で表現することは可能であるが、データ管理の容易さや依存関係の交差を許さない方がコーパスとして利用しやすいことを鑑みて、3 節で述べた階層化された依存構造を記述の対象とすることにする。

4.1 形態素オブジェクト

UD には形態素という単位はないが、古典中国語形態素 LOD [9] との連携を考慮して、語とは別に形態素オブジェクトを設けた。これは

form 形態素の表現形
lemma 形態素の正規形

word-class　古典中国語形態素コーパス/LOD における４階層品詞

の３つのメンバーからなる。

例えば、「孟子見梁惠王」における動詞「見」の形態素オブジェクトの JSON 表現は

```
{ "form": "見", "lemma": "見",
  "word-class": "v, 動詞, 行為, 動作"
}
```

のように書くことができ、その CID は

zdpuAnkw4XGfT74YPrjzp4h35YY42tbwZEZsJSLHFJz9gQEW8

となる。

4.2　依存語オブジェクト

UD の依存構造のグラフにおける語（の出現）を表現するものとして依存語オブジェクトを設けた。これは、

deprel　この依存語を含む依存構造オブジェクトのヘッドに対する依存関係。UD における依存関係タグを記載する。但し、この依存語オブジェクトがこれを含む依存構造オブジェクトのヘッドである場合、"head" をその値とする

form　語の表現形

lemma　語の正規形

misc　CoNLL-U 形式における MISC. 古典中国語では "SpaceAfter=No" を入れる

morpheme　形態素オブジェクトへのリンク

ud-pos　UD における品詞タグ

という６つのメンバーからなる。

例えば、「孟子見梁惠王」における動詞「見」の依存語オブジェクトの JSON 表現は

```
{ "deprel": "head", "form": "見", "lemma": "見",
  "misc": "SpaceAfter=No",
  "morpheme": {
    "/":
     "zdpuAnkw4XGfT74YPrjzp4h35YY42tbwZEZsJSLHFJz9gQEW8"
  },
  "ud-pos": "VERB"
}
```

となり、その CID は

zdpuB1yaveZMdbL6a2sbG2SLpcJEz163bW1Jc9a5LNUf1yzGB

となる。

4.3　依存関係オブジェクト

UD の依存構造のグラフにおける文もしくは文中でブロックをなす連続する部分構造を表現するものとして依存関係オブジェクトを設けた。これは、

deprel　この依存関係オブジェクトを含む依存構造オブジェクトのヘッドに対する依存関係。UD における依存関係タグを記載する。但し、この依存構造オブジェクトがこれを含む依存構造オブジェクトのヘッドである場合、"head" をその値とする

form　依存関係オブジェクトの表現形

subnode　この依存関係オブジェクトに含まれる依存語オブジェクトもしくは依存関係オブジェクトへのリンクの配列

という３つのメンバーからなる。

例えば、テキスト「孟子見梁惠王」における「梁惠王」を示す依存関係オブジェクトの JSON 表現は

```
{ "deprel": "obj", "form": "梁惠王"
  "subnode": [
    {"/":
     "zdpuB3aTJpzL73U793sk3qcrdGhJxUji8oAJgajo9JTXF6WKR"
    },
    {"/":
     "zdpuAuEKH1aZhzckESS9DLhGqKVY98ixbrRwEbxbidKwaxvzB"
    },
    {"/":
     "zdpuAoYdCR9vNFvGPbuwc5THnvTjUUrqKE41TqrpMWxnYZhxz"
    } ]
}
```

となり、その CID は

zdpuApLfWuX3i25YW7PvnnqfkeQsXt91FptiJBg6bnU99ZxBi

となる。

また、ここでの文全体を示す依存関係オブジェクトの JSON 表現は

```
{ "deprel": "root", "form": "孟子見梁惠王"
  "subnode": [
    {"/":
     "zdpuArZJwe4685p99EYdu9enX5UP7tk5uJYCSpyhQoNcKPvj4"
    },
    {"/":
     "zdpuB1yaveZMdbL6a2sbG2SLpcJEz163bW1Jc9a5LNUf1yzGB"
    },
    {"/":
     "zdpuApLfWuX3i25YW7PvnnqfkeQsXt91FptiJBg6bnU99ZxBi"
    } ]
}
```

となり、その CID は

zdpuB359bo2P6AtizJpVyzAo4vWVcdxpsqmW4aYX2z4XHoj23

となる。

4.4　テキストオブジェクト

コーパスの項目となるテキストを表現するオブジェクトを文を示すオブジェクトとは別に設けた。これはコーパスの項目が必ずしも１文とは限らず複数の文として分析される場合があるためである。このテキストオブジェクトは

form　テキストオブジェクトの表現形

subnode　このテキストオブジェクトに含まれる依存語オブジェクトもしくは依存関係オブジェクトへのリンクの配列

という２つのメンバーからなる。

例えば、テキスト「孟子見梁惠王」を示すテキストオブジェクトの JSON 表現は

```
{ "form":  "孟子見梁惠王"
  "subnode":  [
   {"/":
    "zdpuB359bo2P6AtizJpVyzAo4vWVcdxpsqmW4aYX2z4XHoj23"
   } ]
}
```

となり、その CID は

`zdpuAvTZQnL8vhSqewxTuaekpwqtm3H32SvnZT9Zquiegebsu`

となる。

5. EgT との連携

IPFS/IPLD ではデータの内容（データのハッシュ値）に基づく ID (CID) によってオブジェクトを管理しているため、データの内容を 1 バイトでも書き換えると別の ID になってしまう。言い替えれば、IPLD のオブジェクトは不変なオブジェクト (immutable object) であり、オブジェクトの編集とは編集前のオブジェクトを編集後のオブジェクトに置き換える行為であるといえる。そして、編集前のオブジェクトと編集後のオブジェクトの CID は別のものであるので、両者を関係づける仕組みが必要であるといえる。

また、ものが存在する場所で示される (location-addressed) もの（例えば、Pulleyblank の “Outline of Classical Chinese Grammar” [10] の 1 番目の例文や URL で示される Web 頁、あるいは、Git リポジトリの指定したバージョンや指定したブランチの最新版など）やその性質で示されるもの（例えば、人の年齢や今日の湿度など）といった形で対象物を指示したい場合もある。

あるいは、あるオブジェクト A が別のオブジェクト B へのリンクを持つ場合、オブジェクト B からオブジェクト A への逆リンクを記録したいとする。もし、この逆リンクをオブジェクト B に記述するとオブジェクト B の CID が変化してしまい、オブジェクト A から参照しているものとは別物になってしまう。そこで、オブジェクト A からのリンクを新しいオブジェクト B に置き換えると今度はオブジェクト A の CID も変化してしまう。それ故に、DAG 構造しか記述できない訳であるが、現実問題として、逆リンクの情報を記述するとしたらそれはオブジェクト B の外側で行うしかないといえる。

IPFS には IPNS という名前解決のための仕組みが存在する。これは公開鍵暗号技術を用いて IPFS ネットワークのノードに Peer ID というものを割り当て、電子署名技術を用いて Peer ID に IPFS の CID を対応づけるものである。よって、これは編集され得るサイトの最新版を配信するような用途には使えるものの、IPLD のさまざまなオブジェクトをその場所や性質で指示するための仕組みとしてはそのままでは使えない。また、Peer ID に紐づけられるという性質から特定のノードを利用可能な利用者しか登録できず、分権化 (decentralize) という観点では問題である。[*3]

きちんと分権化された仕組みに基づいてこうした問題を解決することが重要であるといえるが、当面のアドホックな解決法として既存の Web 技術を併用することで location-addressing の実現や逆リンクや付加情報の管理を行うことを試みた。これは具体的には IPLD のオブジェクトに対応する Concord [11] オブジェクトを作成し、EgT [12] (CHISE-wiki) を用いて参照するという方法である。

5.1 形態素オブジェクト

IPLD の形態素オブジェクトに対しては、古典中国語形態素 LOD [9] における形態素オブジェクトと同じ `morpheme@zh-classical` ジャンルの Concord オブジェクトを生成する。

この ID はオブジェクトを新たに作る場合には IPLD の CID を用い、既存の形態素オブジェクトが存在する場合にはそのオブジェクトを用いる。

また、ID 素性 `=ipld` に CID を格納する。

また、`<-morpheme` に後述する UD 依存構造オブジェクトへの逆リンク情報を格納する。

5.2 UD 依存構造オブジェクト

IPLD の依存語オブジェクト、依存関係オブジェクト、テキストオブジェクトに対しては、`ud@zh-classical` ジャンルの Concord オブジェクトを生成する（これらを総称して UD 依存構造オブジェクトと呼ぶことにする）（図 3, 4）。

UD 依存構造オブジェクトはその ID として IPLD の CID を用い、ID 素性 `=ipld` にも CID を格納する。

また、IPLD オブジェクトの各メンバーの情報を Concord の素性対としてそのまま格納する。Concord オブジェクトの素性名は、原則として、IPLD オブジェクトと同じ名前を用いるが、メンバーが他のオブジェクトへのリンクである場合、関係素性を示す接頭辞 `->` を付ける。例えば、IPLD オブジェクトのメンバー `subnode` は Concord オブジェクトでは関係素性 `->subnode` となり、同様に、メンバー `morpheme` は関係素性 `->morpheme` となる。Concord では関係素性の値の要素となる各オブジェクトの逆関係素性に逆リンクの情報が自動生成されるため、これを利用して用例情報を集積することができる。

また、メンバー `form` の値を名前とする見出しオブジェクトへのリンクを関係素性 `->entry@ud` に格納する（見出しオブジェクトが存在しない場合には新たに作成する）。この見出しオブジェクトは古典中国語形態素 LOD と共通であるので、CHISE-wiki の文字の情報や形態素の見出しの情報から UD の情報へ飛ぶこともできる（図 5）。

[*3] きちんとやるためには、分散台帳を実現する必要があると思われる。

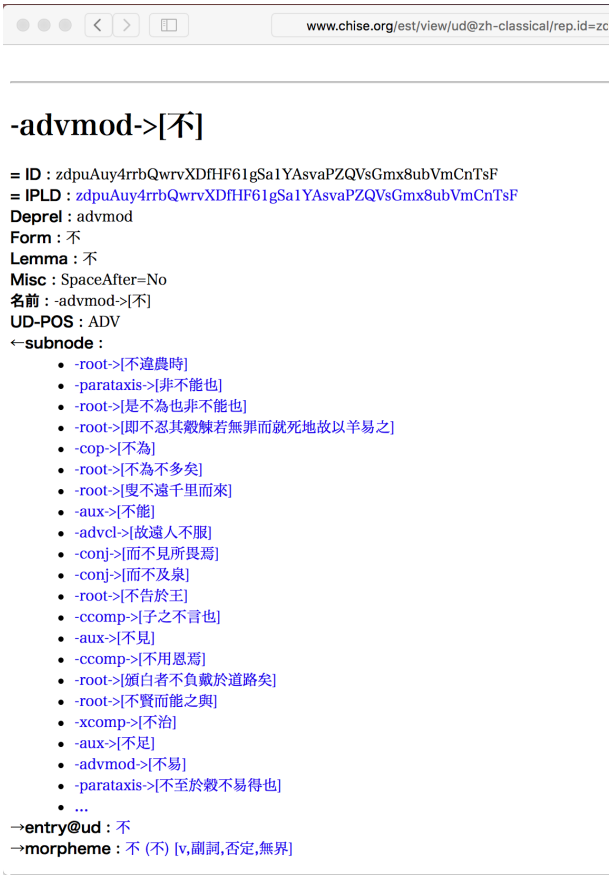

図 3　依存語オブジェクトに対応する EST 頁の例

図 4　依存関係オブジェクトに対応する EST 頁の例

図 5　見出しオブジェクトの EST 頁の例

テキストオブジェクトに対応する UD 依存構造オブジェクトには ID 素性 `=Pulleyblank` を付け、その値に Pulleyblank の “Outline of Classical Chinese Grammar” [10] の例文番号を入れる。これにより、`http://www.chise.org/est/view/ud@zh-classical/rep.Pulleyblank=303` のようにこの例文番号に対応する UD の依存構造の情報を参照することができる（図 6）。

図 6　テキストオブジェクトに対応する EgT 頁の例

6. おわりに

古典中国語（漢文）を対象とした Universal Dependencies (UD) に基づく係り受けコーパスの IPLD を用いたデータ表現の試みについて述べた。

古典中国語の UD コーパスの開発においては、語の認定をどうするか、どの語からどの語に係り受け関係を設定するか、品詞や係り受けの種類をどうするかなどさまざまな項目を検討する必要があり、こうした任意の要素が試行錯誤でき、さまざまな選択肢を比較できることが望ましい。

語の認定基準の揺れへの対処、あるいは、複雑に入り組んだ文の依存構造の把握や大きな単位から小さな単位への段階的な記述の実現のためには、さまざまな単位の部分構造を適切に管理する仕組みがあることが重要であるといえ、ここでは『(UD の) 依存構造の階層化』を提案し、その手法に基づいてデータ表現を行った。但し、この手法は依存関係の交差が存在する場合には適用できず別の手法を用いる必要があるといえる。

一方、JSON や XML 等で記述されるような構造を持ったデータをインターネット上で共有する場合、(RDF でモデル化されているような形で) 各要素に適切に ID を付与することが重要であるが、永続的な ID (IRI) を適切に付与・維持することは必ずしも容易なことではない。また、データを編集した場合、IRI はそのデータの場所に対して付けられた ID であるので、(特別な工夫を行わない限り) どのバージョンを指しているかが判らなくなってしまう。これに対し、IPFS/IPLD はデータの内容によって ID を生成するため、ある特定のバージョンのデータに対する永続的な ID (CID) が容易に生成できる。いわば DOI のような不変なオブジェクトに対する永続的識別子を極めて簡単に発行することができる仕組みだと看做すことができる。

その一方で、現状、IPLD は名前解決の仕組みをそれ自体では持っていないため、実際に運用するためにはなんらかの工夫が必要である。ここでは、CHISE-wiki や古典中国語形態素 LOD 等で用いている EgT を用い、IPLD の世界と IRI ベースの Linked Data の世界をつなぐことによって解決を計った。しかしながら、従来の Web に依存しない IPFS の世界だけで簡潔可能な手法が望ましいといえ、分権的な Linked Data の現実的な運用法について今後も検討したい。

参考文献

[1] Universal Dependencies contributors: Universal Dependencies, `http://universaldependencies.org/`.
[2] 安岡孝一，Wittern, C.，守岡知彦，池田　巧，山崎直樹，二階堂善弘，鈴木慎吾，師　茂樹：古典中国語 Universal Dependencies への挑戦，情処研報，Vol. 2018-CH-116, No. 20, pp. 1–8 (2018).
[3] 安岡孝一，Wittern, C.，守岡知彦，池田　巧，山崎直樹，二階堂善弘，鈴木慎吾，師　茂樹：古典中国語（漢文）の形態素解析とその応用，情報処理学会論文誌，Vol. 59, No. 2, pp. 323–331 (2018).
[4] Benet, J.: IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3), *arXiv preprint arXiv:1407.3561* (2014).
[5] Protocol Labs: IPFS is the Distributed Web, `https://ipfs.io/`.
[6] Protocol Labs: IPLD, `https://ipld.io/`.
[7] Dias, D.: IPLD—The “thin-waist” merkle dag format, `https://github.com/ipld/specs/blob/master/IPLD.md`.
[8] Bormann, C. and Hoffman, P.: *Concise Binary Object Representation (CBOR)*, Internet Engineering Task Force (IETF) (2013). RFC 7049.
[9] 守岡知彦：古典中国語形態素コーパスの Linked Data 化の試み，じんもんこん 2013 論文集，情報処理学会シンポジウムシリーズ，Vol. 2013, No. 4, 情報処理学会，情報処理学会，pp. 187–194 (2013).
[10] Pulleyblank, E. G.: *Outline of Classical Chinese Grammar*, University of British Columbia Press (1995).
[11] 守岡知彦：Concord: プロトタイプ方式のオブジェクト指向データベースの試み，Linux Conference 抄録集，Vol. 4 (2006).
[12] 守岡知彦：Wiki 的手法に基づく構造化データの編集について，人文科学とコンピュータシンポジウム論文集—人文工学の可能性〜異分野融合による「実質化」の方法〜，情報処理学会シンポジウムシリーズ，Vol. 2010, No. 15, 情報処理学会，情報処理学会，pp. 33–40 (2010).