E-HowNet

Release v3.0.0

Mu Yang

Jan 26, 2021

OVERVIEW

CHAPTER

ONE

CKIP E-HOWNET TOOLS

1.1 Git

https://github.com/emfomy/ehownet

1.2 PyPI

https://pypi.org/project/ehownet

1.3 Documentation

https://ehownet.readthedocs.io/

1.4 Related Websites

- E-HowNet Website http://ehownet.iis.sinica.edu.tw
- E-HowNet Visualization Demo https://ckip.iis.sinica.edu.tw/service/ehnvis

1.5 Author

- Mu Yang <https://muyang.pro>

1.6 Requirements

- Python 3.6+
- PLY (Python Lex-Yacc) 3.11+
- TreeLib 1.6.0+
- dataclasses 0.8+
- wcwidth 0.2.5+

1.7 Installation

```
pip install -U ehownet
```

One may download the database file manually from https://ckip.iis.sinica.edu.tw/CKIP/ehownet_reg/ .

1.8 License

CHAPTER

TWO

TUTORIALS

2.1 Grammar

This section describes the grammar of the E-HowNet expression.

2.1.1 Tokens

- `TEXT`
  - Any non empty string containing the following characters:
    - Alphabets and Numbers (`A-Za-z0-9`)
    - Unicode Characters (`\x80-\U0010FFFF`)
    - `|`, `#`.
- `NUMBER`
  - e.g. `1`, `0.1`, `1e-4`
- `COINDEX`
  - `x`, `x1`, `x2`, ...
- `x?` (refer to the unmentioned subject entity)

2.1.2 Nodes

Entity

Entities are basic elements in E-HowNet definition.

- *`EhnParseNormalEntity`*

  A normal entity.

  - Syntaxes:
    - `{TEXT}`
    - `{TEXT:FEATURE}`
    - `{TEXT:FEATURE,FEATURE}`
    - `{TEXT:FEATURE,FEATURE,...}`
    - `{TEXT_COINDEX}`

        * {TEXT_COINDEX:FEATURE}
        * {TEXT_COINDEX:FEATURE,FEATURE}
        * {TEXT_COINDEX:FEATURE,FEATURE,...}
      - Description:
        * TEXT is the head (the label of the inherited concept) of this entity.
        * FEATUREs are the additional features to this entity.
        * COINDEX is the anchor of this entity for further reference.
      - Example:

        {human|:kind={other|}}

- *EhnParseFunctionEntity*

  An entity with function as its head.

  - Syntaxes:
    * {FUNCTION}
    * {FUNCTION:FEATURE}
    * {FUNCTION:FEATURE,FEATURE}
    * {FUNCTION:FEATURE,FEATURE,...}
    * {FUNCTION_COINDEX}
    * {FUNCTION_COINDEX:FEATURE}
    * {FUNCTION_COINDEX:FEATURE,FEATURE}
    * {FUNCTION_COINDEX:FEATURE,FEATURE,...}
  - Description:

    Similar to normal entity, but replace the head by a function (FUNCTION) of entity/entities.

- *EhnParseNameEntity*

  A "name".

  - Syntaxes:
    * {"TEXT"}
  - Example:
    * {country|:location={Europe|},quantifier={definite|},
      name={""}}

      The name of this country is .

- *EhnParseNumberEntity*

  A number.

  - Syntaxes:
    * {NUMBER}
  - Example:
    * {month|:sequence={1}}

The sequence of this mouth if **1**.

Reference

References refer to other entities.

- *EhnParseCoindexReference*

  Refers to previous mentioned entity.

  - Syntaxes:
    - `{COINDEX}`
  - Description:
    - Refers to the entity with anchor `_COINDEX`.
    - If `_COINDEX` does not exist, represent that all `{COINDEX}` with the same name are the same placeholder.
  - Example:
    - `{A_x1:r={B:b={x1}}}`

      `{x1}` refers to `{A_x1:...}`.

    - `{vehicle|_x1:predication={fly|:theme={x1}}}`

      `{x1}` refers to `{vehicle|_x1:...}`.

- *EhnParseSubjectReference*

  Refers to unmentioned subject entity.

  - Syntaxes:
    - `{x?}`
  - Example:
    - `r={B:b={x?}}`

      `{x?}` refers to the unmentioned subject entity `S` with the following attribute `{S_x1:r={B:b={x1}}}`.

    - `:predication={|ServeInArmy:agent={x?},aspect={Vgoingon|}}`

      `{x?}` refers to the unmentioned subject entity `S`. For example, the word of .

- *EhnParseTildeReference*

  Refers to the root entity.

  - Syntaxes:
    - `{~}`
  - Example:
    - `{A:r={B:b={~}}}`

      `{~}` refers to `{A:...}`.

    - `{vehicle|_x1:predication={fly|:theme={~}}}`

      `{~}` refers to the root entity `{vehicle|_x1:...}`.

Note: Deprecated since version 0.6.

Placeholder

Placeholders represent any entities under the given restriction.

- *EhnParseRestrictionPlaceholder*

  A restriction placeholder.

  - Syntaxes:
    - `/ENTITY`
    - `/ENTITY_COINDEX`
  - Description:
    - `ENTITY` shows that this node can be replace by any hyponymy/instance(s) of the `ENTITY`.
    - `COINDEX` is the anchor of this restriction for further reference.
  - Example:
    - `{CentrePart(/{place|})}`

      The argument of **CentrePart** must be a hyponymy/instance of **place**l.

- *EhnParseAnyPlaceholder*

  A placeholder without restriction.

  - Syntaxes:
    - `{}`
  - Description:

    Represent a placeholder without any restriction. Only used as the value of a feature.
  - Example:
    - `feature={}`

      `{}` represent that the value of this feature can be any entity.

Feature

Features provides extra information to entities.

- *EhnParseNormalFeature*

  A normal feature.

  - Syntaxes:
    - `TEXT=ENTITY`
    - `TEXT=REFERENCE`
    - `TEXT=RESTRICTION`
    - `TEXT={}`

  - Description:
    * `TEXT` is the head (the name) of the this feature.
    * `ENTITY`/`RESTRICTION` is the value of this feature.
  - Example:
    * `{thing|:qualification={concrete|}}`

      The **qualification** of **thing|** is **concrete|**. Here `qualification={concrete|}` is a normal feature.

- *`EhnParseFunctionFeature`*

  A function feature.

  - Syntaxes:
    * `FUNCTION=ENTITY`
    * `FUNCTION=REFERENCE`
    * `FUNCTION=RESTRICTION`
    * `FUNCTION={}`
  - Description:

    Similar to normal feature, but replace the head by a function (`FUNCTION`) of entity/entities.
  - Example:
    * `{animate|:ability({SelfMove|})={very|}}`

      The **ability** of **SelfMove|** of **animate|** is **very|**. Here `ability({SelfMove|})={very|}` is a function feature.

Function

Functions act on entities.

- `EhnParseFunction`

  A function of entity/entities or restriction.

  - Syntaxes:
    * `TEXT()`
    * `TEXT(RESTRICTION)`
    * `TEXT(ENTITY)`
    * `TEXT(ENTITY,ENTITY)`
    * `TEXT(ENTITY,ENTITY,...)`
  - Description:
    * `TEXT` is the head (the name) of the this function.
    * `ENTITY`s are the arguments of this function; every `ENTITY` can be replaced by a `REFERENCE`.
    * `RESTRICTION` represent that the arguments of this function can be anything under this restriction.

* `TEXT()` represent that the arguments of this function can be any entity/entities.

Note: `TEXT({})` is not valid. Use `TEXT()` instead.

2.1.3 Valid Expressions

A valid expression can be an `ENTITY` or any number of `FEATURE`s joined by `,` s.

- `ENTITY`
- `FEATURE`
- `FEATURE,FEATURE`
- `FEATURE,FEATURE,...`

2.2 Database

This section describes the E-HowNet database.

One may download the database file manually from https://ckip.iis.sinica.edu.tw/CKIP/ehownet_reg/ .

class EhnDb(*, *db_file*)

The E-HowNet database.

See also: *ehn.db.core.EhnDb*

Parameters **db_file** (*str*) – The path to the SQLite3 database file.

tree: treelib.Tree

A TreeLib tree.

text2nid_concept: dict

A dictionary that maps concept label to its node ID.

text2nid_word: dict

A dictionary that maps word label to its node ID.

text2nid_partial: dict

A dictionary that maps any subtext of concept label to its node ID.

For example, both "entity" and "" maps to the node ID of the concept "entityl".

get_nids(*text*, *, *concept=True*, *word=True*, *full_match=False*)

Query node IDs.

Parameters

- **text** (*str*) – the query text.
- **concept** (*boolean*) – returns concept node.
- **word** (*boolean*) – returns word node.
- **full_match** (*boolean*) – returns only the nodes that fully match their label.

Returns A list of node IDs.

get_nodes(*text*, *, *concept=True*, *word=True*, *full_match=False*)

Query *EhnDbNode*.

Parameters

- **text** (*str*) – the query text.
- **concept** (*boolean*) – returns concept node.
- **word** (*boolean*) – returns word node.
- **full_match** (*boolean*) – returns only the nodes that fully match their label.

Returns A list of nodes.

class EhnDbNode

The E-HowNet database node.

See also: *ehn.db.data.EhnDbNode*

nid: int

The node ID.

label: str

The node label.

data: ~ehn.db.data.EhnDbNodeData

The node data.

Note that one may access data attribute directly (e.g. **obj.defn** of this object **obj** returns **obj.data.defn**).

class EhnDbNodeData

The E-HowNet database node data.

See also: *ehn.db.data.EhnDbNodeData*

defn: str

The node definition.

type: ~ehn.db.data.EhnDbNodeType

The node type.

words: List[~ehn.db.data.EhnDbWordData]

The list of attached words.

definite: bool

Whether this node is an instance of is parent node of not.

class EhnDbWordData

The E-HowNet database word data.

See also: *ehn.db.data.EhnDbWordData*

word: str

The word.

sense_no: str

The sense number ID.

class EhnDbNodeType

The enum class of E-HowNet database node type.

See also: *ehn.db.data.EhnDbNodeType*

C = 'C'

The concept type.

W = 'W'

The word type.

2.3 Parser

This section describes the E-HowNet parser.

2.3.1 Python API

```
from ehn.parse import EhnParser

text = '{MusicTool|_x1:predication={own|:possession={|PushingButton:whole={x1}}}}'

parser = EhnParser()
ress = parser(text, debug=False)
for res in ress:
   res.tree().show()
```

Output:

```
[Entity $x1] MusicTool|
└── [Feature] predication
    └── [Entity] own|
        └── [Feature] possession
            └── [Entity] |PushingButton
                └── [Feature] whole
                    └── $x1
```

2.3.2 CLI

One may also use the parser in command line directly.

```
# Usage
ehn-parser <text> [<text> ...]

# Example
ehn-parser \
   "{MusicTool|_x1:predication={own|:possession={|PushingButton:whole={x1}}}}" \
   "{InstitutePlace|:telic={or({experiment|:location={~}},{research|:location={~}})}}
↪" \
   "{festival|:TimePoint={x?},telic={congratulate|:content={year|:qualification={new|}
↪}}}" \
   "TimePoint={},manner={urgent|}" \
   "direction={toward()}"
```

Output:

```
#1
[Entity $x1] MusicTool|
└── [Feature] predication
    └── [Entity] own|
        └── [Feature] possession
            └── [Entity] |PushingButton
                └── [Feature] whole
                    └── [Reference] $x1

#2
```

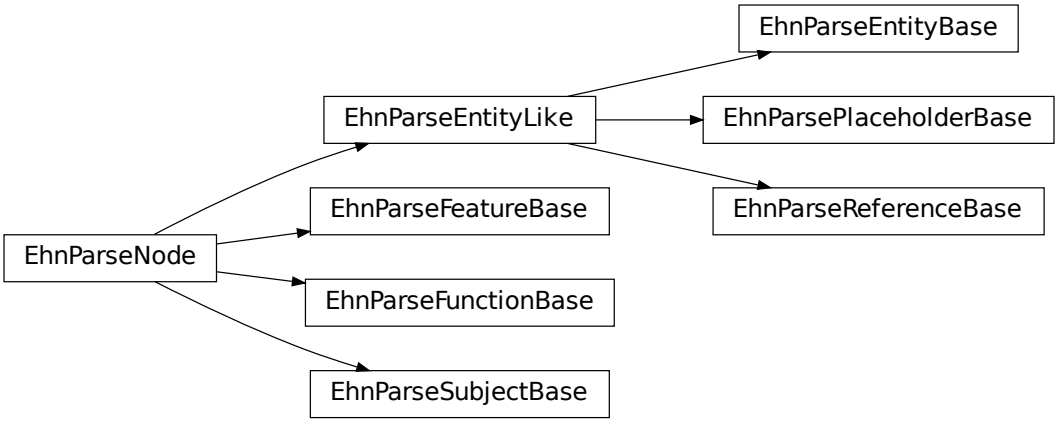
(continues on next page)

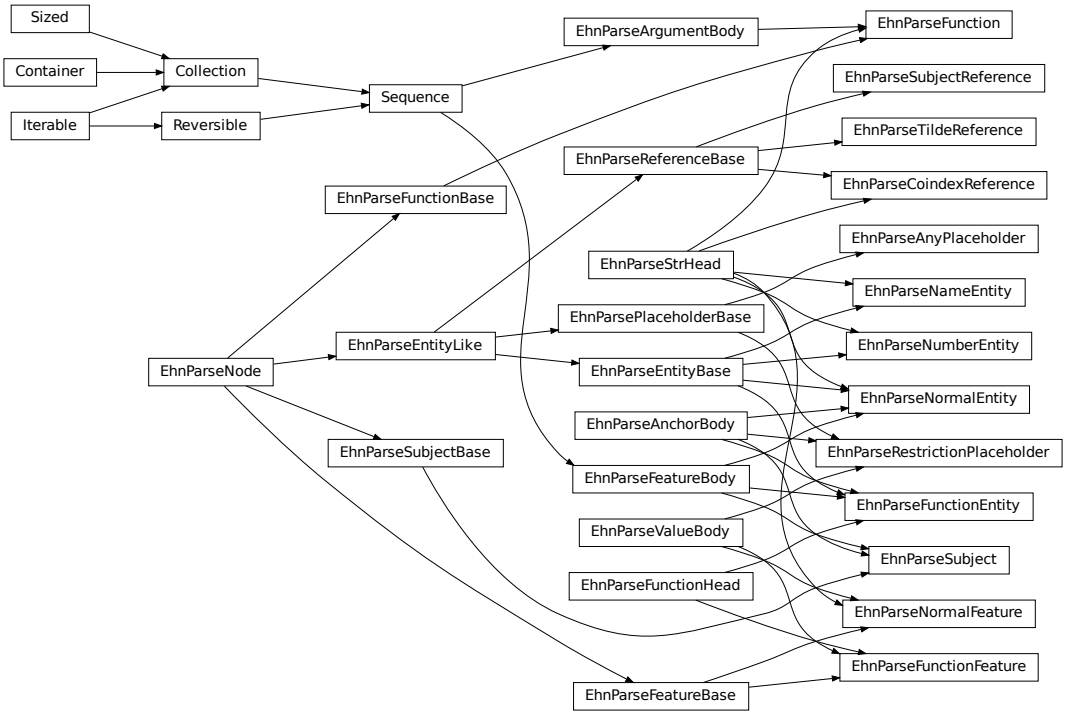
(continued from previous page)

```
[Entity] InstitutePlace|
└── [Feature] telic
    └── [FunctionEntity]
        └── [Function] or
            ├── [Entity] experiment|
            │   └── [Feature] location
            │       └── [TildeReference]
            └── [Entity] research|
                └── [Feature] location
                    └── [TildeReference]

#3
[Entity] festival|
├── [Feature] TimePoint
│   └── [SubjectReference] $x?
└── [Feature] telic
    └── [Entity] congratulate|
        └── [Feature] content
            └── [Entity] year|
                └── [Feature] qualification
                    └── [Entity] new|

#4
[Subject $x?]
├── [Feature] TimePoint
│   └── [Any]
└── [Feature] manner
    └── [Entity] urgent|

#5
[Subject $x?]
└── [Feature] direction
    └── [FunctionEntity]
        └── [Function] toward
            └── [Any]
```

2.4 Parse Nodes

There are five types of nodes in E-HowNet expression — *Entity*, *Reference*, *Placeholder*, *Feature*, *Function*, and *Subject*.


EhnParseEntityBase
EhnParseEntityLike
EhnParsePlaceholderBase
EhnParseReferenceBase
EhnParseFeatureBase
EhnParseNode
EhnParseFunctionBase
EhnParseSubjectBase


Sized
Container
Iterable
Collection
Reversible
Sequence
EhnParseArgumentBody
EhnParseFunction
EhnParseSubjectReference
EhnParseTildeReference
EhnParseReferenceBase
EhnParseCoindexReference
EhnParseFunctionBase
EhnParseAnyPlaceholder
EhnParseStrHead
EhnParseNameEntity
EhnParsePlaceholderBase
EhnParseEntityLike
EhnParseNumberEntity
EhnParseNode
EhnParseEntityBase
EhnParseNormalEntity
EhnParseAnchorBody
EhnParseSubjectBase
EhnParseRestrictionPlaceholder
EhnParseFeatureBody
EhnParseFunctionEntity
EhnParseValueBody
EhnParseSubject
EhnParseFunctionHead
EhnParseNormalFeature
EhnParseFunctionFeature
EhnParseFeatureBase

2.4.1 Node Prototype

class EhnParseNode
The prototype of E-HowNet parsing nodes.

See also: *ehn.parse.node.base.EhnParseNode*

head: str
The head of this node.

get_features(*key=None*)
Get the features (or `[]` if not exists).

If **key** is set, returns only the features with **feature.head==key**.

get_arguments()
Get the arguments (or `[]` if not exists).

get_value()
Get the value (or `None` if not exists).

get_function()
Get the function (or `None` if not exists).

get_anchor()
Get the anchor (or `None` if not exists).

get_coindex()
Get the coindex key (the head of the anchor, the head of *reference* nodes, or `None` otherwise).

children()
Yields all direct child nodes of this node.

descendant()
Yields all descendant nodes (including self) of this node.

dumps()
Converts to text representation.

tree() → *ehn.parse.node.base.EhnParseTree*
Generates a tree representation of this node and all its descendant nodes.

2.4.2 Entity-Like Nodes

Entity

class EhnParseEntityBase
The base class of E-HowNet parsing entity nodes.

See also: *ehn.parse.node.base.EhnParseEntityBase*

Subclasses:

- *EhnParseNormalEntity* A normal entity. Can be an *anchor*.
- *EhnParseFunctionEntity* An entity with *function head*. Can be an *anchor*.
- *EhnParseNameEntity* A name entity.
- *EhnParseNumberEntity* A number entity.

property features
A list of *Features*.

Reference

class EhnParseReferenceBase
The base class of E-HowNet parsing reference nodes.

See also: *ehn.parse.node.base.EhnParseReferenceBase*

Subclasses:

- *EhnParseCoindexReference* An entity refers to an anchor entity.
- *EhnParseSubjectReference* An entity refers to the unmentioned subject entity (*EhnParseSubject* in feature-based expressions.)
- *EhnParseTildeReference* An entity refers to the root entity.

Placeholder

class EhnParsePlaceholderBase
The base class of E-HowNet parsing restriction nodes.

See also: *ehn.parse.node.base.EhnParsePlaceholderBase*

Subclasses:

- *EhnParseRestrictionPlaceholder*. Can be an *anchor*.
- *EhnParseAnyPlaceholder* A placeholder without restriction.

property value
Must be an *Entity* (for *EhnParseRestrictionPlaceholder*) or `None` (for *EhnParseAnyPlaceholder*).

2.4.3 Non-Entity-Like Nodes

Feature

class EhnParseFeatureBase
The base class of E-HowNet parsing feature nodes.

See also: *ehn.parse.node.base.EhnParseFeatureBase*

Subclasses:

- *EhnParseNormalFeature* A normal feature.
- *EhnParseFunctionFeature* An feature with *function head*.

property value
Can be a *Entity-Like Node*.

Function

class EhnParseFunctionBase
The base class of E-HowNet parsing function nodes.

See also: *ehn.parse.node.base.EhnParseFunctionBase*

Subclasses:

- `EhnParseFunction`.

property arguments
A list of *Entity-Like Nodes*.

Subject

class EhnParseSubjectBase
The base class of E-HowNet parsing unmentioned subject nodes. Works similar to entities but is not an entity. Used only in feature-based expressions.

See also: *ehn.parse.node.base.EhnParseSubjectBase*

Subclasses:

- *EhnParseSubject*. Always an *anchor* of `x?`.

property features
A list of *Features*.

2.4.4 Partial Nodes

Function Head

class EhnParseFunctionHead
The base class of nodes with a function as its head.

Note that the attribute **obj.head** of this object **obj** returns **obj.function.head**.

See also: *ehn.parse.node.base.EhnParseFunctionHead*

Subclasses:

- *EhnParseFunctionEntity*
- *EhnParseFunctionFeature*

property function
Must be a *Function*.

Anchor Body

class EhnParseAnchorBody
The base class of anchor nodes.

See also: *ehn.parse.node.base.EhnParseAnchorBody*

Subclasses:

- *EhnParseNormalEntity*
- *EhnParseFunctionEntity*
- *EhnParseRestrictionPlaceholder*
- *EhnParseSubject*

property anchor
The *Anchor*.

Anchor

class EhnParseAnchor
The coindex target.

See also: *ehn.parse.node.base.EhnParseAnchor*

head: str
The coindex of this anchor.

2.5 Graph Builder

This package provides are two type of graphs — the *standard graph* and the *vis.js graph* — for E-HowNet definitions.

2.5.1 Standard Graph

class EhnStandardGraphBuilder
Generates graphs from E-HowNet definitions.

See also: *ehn.graph.standard.EhnStandardGraphBuilder*

__call__(*root*)

Parameters **root** (EhnParseNode) – The root parse node of a E-HowNet definition.

Return type *EhnStandardGraph*

class EhnStandardGraph
See also: *ehn.graph.standard.EhnStandardGraph*

nodes: dict
A dictionary that maps the node ID (a random UUID or a coindex) to a list of parse nodes.

edges: list
A list of triplets **(subject node ID, predicate node ID, object node ID)**. The IDs are the keys in **nodes**.

functions: list
A list of pairs **(function node ID, argument node ID)**. The IDs are the keys in **nodes**. Note that different argument of a single function node will be listed separately.

restrictions: list
A list of pairs **(placeholder node ID, restriction node ID)**. The IDs are the keys in **nodes**.

root_id: int
The ID of the root node in the definition.

2.5.2 Vis Graph

class EhnVisGraphBuilder(*definite_labels=None*)
Generates graphs from E-HowNet definitions for vis.js.

See also: `ehn.graph.standard.EhnVisGraphBuilder`

Parameters **definite_labels** (*set*) – a set of the labels of the definite concepts.

__call__(*root*)

Parameters **root** (`EhnParseNode`) – The root parse node of a E-HowNet definition.

Return type EhnVisGraph

class EhnVisGraph
Please refers vis.js's documentation.

See also: `ehn.graph.standard.EhnVisGraph`

nodes: dict
The nodes.

edges: list
The edges

CHAPTER

THREE

EHN PACKAGE

Subpackages

3.1 ehn.db package

Please refer the tutorial "*Database*".

Submodules

3.1.1 ehn.db.core module

Please refer the tutorial "*Database*".

class ehn.db.core.**EhnDb**(*, *db_file=None*)
Bases: object

E-HowNet Database.

3.1.2 ehn.db.data module

Please refer the tutorial "*Database*".

class ehn.db.data.**EhnDbNodeType**(*value*)
Bases: enum.Enum

E-HowNet Database Node Type.

C = 'C'
concept.

W = 'W'
word.

class ehn.db.data.**EhnDbWordData**(*word: str*, *sense_no: int*)
Bases: object

E-HowNet Database Word Data.

word: str
the word.

sense_no: int
the sense number.

class ehn.db.data.**EhnDbNodeData**(*type: ehn.db.data.EhnDbNodeType*, *defn: Optional[str] = None*, *words: List[ehn.db.data.EhnDbWordData] = <factory>*, *definite: bool = False*)

Bases: object

E-HowNet Database Node Data.

type: *ehn.db.data.EhnDbNodeType*
the node type.

defn: str = None
the definition.

words: List[*ehn.db.data.EhnDbWordData*]
the attached words.

definite: bool = False
whether is an instance of not.

class ehn.db.data.**EhnDbNode**(*tag=None*, *identifier=None*, *expanded=True*, *data=None*)

Bases: treelib.node.Node

E-HowNet Database Node.

data_class
alias of *ehn.db.data.EhnDbNodeData*

3.2 ehn.graph package

Please refer the tutorial "*Graph Builder*".

Submodules

3.2.1 ehn.graph.standard module

Please refer the tutorial "*Graph Builder*".

class ehn.graph.standard.**EhnStandardGraph**(*nodes: dict*, *edges: list*, *functions: list*, *restrictions: list*, *root_id: int*)

Bases: object

The standard E-HowNet graph.

class ehn.graph.standard.**EhnStandardGraphBuilder**

Bases: object

The standard E-HowNet graph builder.

class ehn.graph.standard.**EhnStandardGraphBuilderWorker**(*root*)

Bases: object

The standard E-HowNet graph builder worker.

3.2.2 ehn.graph.vis module

Please refer the tutorial "*Graph Builder*".

class ehn.graph.vis.**EhnVisGraph**(*nodes: dict*, *edges: list*)
Bases: object

The E-HowNet graph for vis.js.

class ehn.graph.vis.**EhnVisGraphBuilder**(*definite_labels=None*)
Bases: object

The E-HowNet graph builder for vis.js.

class ehn.graph.vis.**EhnVisGraphBuilderWorker**(*root*, *label*, *, *definite_labels*)
Bases: object

The E-HowNet graph builder worker for vis.js.

3.3 ehn.parse package

Subpackages

3.3.1 ehn.parse.node package

Please refer the tutorial "*Parse Nodes*".

Submodules

ehn.parse.node.base module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.base.**EhnParseTree**(*tree=None*, *deep=False*, *node_class=None*, *identifier=None*)
Bases: treelib.tree.Tree

show(**args*, *data_property='_tree_label'*, ***kwargs*)
Print the tree structure.

class ehn.parse.node.base.**EhnParseNode**
Bases: object

E-HowNet Parsing: Base Node

class ehn.parse.node.base.**EhnParseEntityLike**
Bases: *ehn.parse.node.base.EhnParseNode*

E-HowNet Parsing: Entity Like Node

class ehn.parse.node.base.**EhnParseEntityBase**
Bases: *ehn.parse.node.base.EhnParseEntityLike*

E-HowNet Parsing: Base Entity Node

class ehn.parse.node.base.**EhnParseReferenceBase**
Bases: *ehn.parse.node.base.EhnParseEntityLike*

E-HowNet Parsing: Base Reference Node

class ehn.parse.node.base.**EhnParsePlaceholderBase**
Bases: *ehn.parse.node.base.EhnParseEntityLike*

E-HowNet Parsing: Base Placeholder Node

class ehn.parse.node.base.**EhnParseFeatureBase**
Bases: *ehn.parse.node.base.EhnParseNode*

E-HowNet Parsing: Base Feature Node

class ehn.parse.node.base.**EhnParseFunctionBase**
Bases: *ehn.parse.node.base.EhnParseNode*

E-HowNet Parsing: Base Function Node

class ehn.parse.node.base.**EhnParseSubjectBase**
Bases: *ehn.parse.node.base.EhnParseNode*

E-HowNet Parsing: Base Subject Node

class ehn.parse.node.base.**EhnParseAnchor**(*head=None*)
Bases: object

E-HowNet Parsing: Node Anchor

class ehn.parse.node.base.**EhnParseStrHead**(*head*)
Bases: object

E-HowNet Parsing: Base Node with String Head

class ehn.parse.node.base.**EhnParseFunctionHead**(*function*)
Bases: object

E-HowNet Parsing: Base Node with Function Head

class ehn.parse.node.base.**EhnParseValueBody**(*value*)
Bases: object

E-HowNet Parsing: Base Node with Value

class ehn.parse.node.base.**EhnParseFeatureBody**(**features*)
Bases: collections.abc.Sequence

E-HowNet Parsing: Base Node with Feature

class ehn.parse.node.base.**EhnParseArgumentBody**(**arguments*)
Bases: collections.abc.Sequence

E-HowNet Parsing: Base Node with Argument

class ehn.parse.node.base.**EhnParseAnchorBody**(**, coindex=None, anchor=None*)
Bases: object

E-HowNet Parsing: Base Node with Anchor

ehn.parse.node.entity module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.entity.**EhnParseNormalEntity**(*head*, **features*, *coindex=None*, *anchor=None*)

Bases: *ehn.parse.node.base.EhnParseEntityBase*, *ehn.parse.node.base.EhnParseStrHead*, *ehn.parse.node.base.EhnParseFeatureBody*, *ehn.parse.node.base.EhnParseAnchorBody*

E-HowNet Parsing: Normal Entity Node

feature_type
alias of *ehn.parse.node.base.EhnParseFeatureBase*

class ehn.parse.node.entity.**EhnParseFunctionEntity**(*function*, **features*, *coindex=None*, *anchor=None*)

Bases: *ehn.parse.node.base.EhnParseEntityBase*, *ehn.parse.node.base.EhnParseFunctionHead*, *ehn.parse.node.base.EhnParseFeatureBody*, *ehn.parse.node.base.EhnParseAnchorBody*

E-HowNet Parsing: Function Entity Node

feature_type
alias of *ehn.parse.node.base.EhnParseFeatureBase*

class ehn.parse.node.entity.**EhnParseNameEntity**(*head*)

Bases: *ehn.parse.node.base.EhnParseEntityBase*, *ehn.parse.node.base.EhnParseStrHead*

E-HowNet Parsing: Name Entity Node

class ehn.parse.node.entity.**EhnParseNumberEntity**(*head*)

Bases: *ehn.parse.node.base.EhnParseEntityBase*, *ehn.parse.node.base.EhnParseStrHead*

E-HowNet Parsing: Number Entity Node

ehn.parse.node.feature module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.feature.**EhnParseNormalFeature**(*head*, *value*)

Bases: *ehn.parse.node.base.EhnParseFeatureBase*, *ehn.parse.node.base.EhnParseStrHead*, *ehn.parse.node.base.EhnParseValueBody*

E-HowNet Parsing: Normal Feature Node

value_type
alias of *ehn.parse.node.base.EhnParseEntityLike*

class ehn.parse.node.feature.**EhnParseFunctionFeature**(*function*, *value*)

Bases: *ehn.parse.node.base.EhnParseFeatureBase*, *ehn.parse.node.base.EhnParseFunctionHead*, *ehn.parse.node.base.EhnParseValueBody*

E-HowNet Parsing: Function Feature Node

value_type
alias of *ehn.parse.node.base.EhnParseEntityLike*

ehn.parse.node.other module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.other.**EhnParseSubject**(**features*)
Bases: *ehn.parse.node.base.EhnParseSubjectBase*, *ehn.parse.node.base.EhnParseFeatureBody*, *ehn.parse.node.base.EhnParseAnchorBody*

E-HowNet Parsing: Subject Node

feature_type
alias of *ehn.parse.node.base.EhnParseFeatureBase*

class ehn.parse.node.other.**EhnParseFunction**(*head*, **arguments*)
Bases: *ehn.parse.node.base.EhnParseFunctionBase*, *ehn.parse.node.base.EhnParseArgumentBody*, *ehn.parse.node.base.EhnParseStrHead*

E-HowNet Parsing: Function Node

argument_type
alias of *ehn.parse.node.base.EhnParseEntityLike*

ehn.parse.node.placeholder module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.placeholder.**EhnParseRestrictionPlaceholder**(*value*, ***, *coindex=None*, *anchor=None*)
Bases: *ehn.parse.node.base.EhnParsePlaceholderBase*, *ehn.parse.node.base.EhnParseValueBody*, *ehn.parse.node.base.EhnParseAnchorBody*

E-HowNet Parsing: Restriction Placeholder Node

value_type
alias of *ehn.parse.node.base.EhnParseEntityBase*

class ehn.parse.node.placeholder.**EhnParseAnyPlaceholder**
Bases: *ehn.parse.node.base.EhnParsePlaceholderBase*

E-HowNet Parsing: Any Placeholder Node

ehn.parse.node.reference module

Please refer the tutorial "*Parse Nodes*".

class ehn.parse.node.reference.**EhnParseCoindexReference**(*head*)
Bases: *ehn.parse.node.base.EhnParseReferenceBase*, *ehn.parse.node.base.EhnParseStrHead*

E-HowNet Parsing: Coindex Reference Node

class ehn.parse.node.reference.**EhnParseSubjectReference**
Bases: *ehn.parse.node.base.EhnParseReferenceBase*

E-HowNet Parsing: Subject Reference Node

class ehn.parse.node.reference.**EhnParseTildeReference**
Bases: *ehn.parse.node.base.EhnParseReferenceBase*

E-HowNet Parsing: Tilde Reference Node

Deprecated since version 0.6.

Submodules

3.3.2 ehn.parse.parser module

Please refer the tutorial "*Parser*".

exception ehn.parse.parser.**EhnSyntaxError**(**args*, *pos=None*)
Bases: SyntaxError

E-HowNet Syntax Error.

show_pos(*text*)
Show error position.

Parameters **text** (*str*) – original input text

class ehn.parse.parser.**EhnLexer**(***kwargs*)
Bases: ehn.parse.parser._EhnLexer

E-HowNet Lexer.

__call__(*data*)
Run tokenization.

class ehn.parse.parser.**EhnParser**(*lexer=None*, ***kwargs*)
Bases: ehn.parse.parser._EhnParser

E-HowNet Parser.

__call__(*data*)
Run parsing.

CHAPTER

FOUR

INDEX

CHAPTER

FIVE

MODULE INDEX

PYTHON MODULE INDEX

e

INDEX

Symbols

A

C

D

E

F

G

H

L

M

N

R

S

T

V

W