
unihan-tabular Documentation

Release 0.7.4

Tony Narlock

May 15, 2017

Contents

1	Features	3
2	Usage	5
3	Structure	7
3.1	API	7
3.2	Command Line Interface	10
3.3	History	10
	Python Module Index	13

unihan-tabular - tool to build **UNIHAN** into tabular-friendly formats like python, JSON, CSV and YAML. Part of the *cihai* project.

UNIHAN's data is dispersed across multiple files in the format of:

U+3400	kCantonese	jau1
U+3400	kDefinition	(same as U+4E18) hillock or mound
U+3400	kMandarin	qiū
U+3401	kCantonese	tim2
U+3401	kDefinition	to lick; to taste, a mat, bamboo bark
U+3401	kHanyuPinyin	10019.020:tiàn
U+3401	kMandarin	tiàn

\$ unihan-tabular will download Unihan.zip and build all files into a single tabular friendly format.

CSV (default), \$ unihan-tabular:

```
char,ucn,kCantonese,kDefinition,kHanyuPinyin,kMandarin
,U+3400,jau1,(same as U+4E18 ) hillock or mound,,qiū
,U+3401,tim2,"to lick; to taste, a mat, bamboo bark",10019.020:tiàn,tiàn
```

JSON,\$ unihan-tabular -F json:

```
[
  {
    "char": "",
    "ucn": "U+3400",
    "kCantonese": "jau1",
    "kDefinition": "(same as U+4E18 ) hillock or mound",
    "kHanyuPinyin": null,
    "kMandarin": "qiū"
  },
  {
    "char": "",
    "ucn": "U+3401",
    "kCantonese": "tim2",
    "kDefinition": "to lick; to taste, a mat, bamboo bark",
    "kHanyuPinyin": "10019.020:tiàn",
    "kMandarin": "tiàn"
  }
]
```

YAML \$ unihan-tabular -F yaml:

```
- char:
  kCantonese: jau1
  kDefinition: (same as U+4E18 ) hillock or mound
  kHanyuPinyin: null
  kMandarin: qiū
  ucn: U+3400
- char:
  kCantonese: tim2
  kDefinition: to lick; to taste, a mat, bamboo bark
  kHanyuPinyin: 10019.020:tiàn
  kMandarin: tiàn
  ucn: U+3401
```


CHAPTER 1

Features

- automatically downloads UNIHAN from the internet
- export to JSON, CSV and YAML (requires [pyyaml](#)) via `-F`
- configurable to export specific fields via `-f`
- accounts for encoding conflicts due to the Unicode-heavy content
- designed as a technical proof for future CJK (Chinese, Japanese, Korean) datasets
- core component and dependency of [cikai](#), a CJK library
- [data package](#) support
- supports python 2.7, >= 3.5 and pypy

If you encounter a problem or have a question, please [create an issue](#).

CHAPTER 2

Usage

`unihan-tabular` supports command line arguments. See [unihan-tabular CLI arguments](#) for information on how you can specify custom columns, files, download URL's and output destinations.

To download and build your own UNIHAN export:

```
$ pip install unihan-tabular
```

To output CSV, the default format:

```
$ unihan-tabular
```

To output JSON:

```
$ unihan-tabular -F json
```

To output YAML:

```
$ pip install pyyaml
$ unihan-tabular -F yaml
```

To only output the `kDefinition` field in a csv:

```
$ unihan-tabular -f kDefinition
```

To output multiple fields, separate with spaces:

```
$ unihan-tabular -f kCantonese kDefinition
```

To output to a custom file:

```
$ unihan-tabular --destination ./exported.csv
```

To output to a custom file (templated file extension):

```
$ unihan-tabular --destination ./exported.{ext}
```

See [unihan-tabular CLI arguments](#) for advanced usage examples.

CHAPTER 3

Structure

```
# output w/ JSON
{XDG data dir}/unihan_tabular/unihan.json

# output w/ CSV
{XDG data dir}/unihan_tabular/unihan.csv

# output w/ yaml (requires pyyaml)
{XDG data dir}/unihan_tabular/unihan.yaml

# script to download + build a SDF csv of unihan.
unihan_tabular/process.py

# unit tests to verify behavior / consistency of builder
tests/*

# python 2/3 compatibility module
unihan_tabular/_compat.py

# utility / helper functions
unihan_tabular/util.py
```

API

Build Unihan into tabular friendly format and export it.

```
unihan_tabular.process.ALLOWED_EXPORT_TYPES = [u'json', u'csv', u'yaml']
```

Allowed export types

```
unihan_tabular.process.DESTINATION_DIR = u'/home/docs/.local/share/unihan_tabular'
```

Filepath to output built CSV file to.

```
unihan_tabular.process.INDEX_FIELDS = [u'ucn', u'char']
```

Default index fields for unihan csv's. You probably want these.

```
unihan_tabular.process.UNIHAN_FIELDS = [u'kAccountingNumeric', u'kBigFive', u'kCCCII', u'kCNS1986', u'kCNS
```

Default Unihan fields

```
unihan_tabular.process.UNIHAN_FILES = [u'Unihan_RadicalStrokeCounts.txt', u'Unihan_NumericValues.txt', u'Unihan_SyllableLabels.txt']
```

Default Unihan Files

```
unihan_tabular.process.UNIHAN_URL = u'http://www.unicode.org/Public/UNIDATA/Unihan.zip'
```

URI of Unihan.zip data.

```
unihan_tabular.process.UNIHAN_ZIP_PATH = u'home/docs/.cache/unihan_tabular/downloads/Unihan.zip'
```

Filepath to download Zip file.

`unihan_tabular.process.WORK_DIR = u'/home/docs/.cache/unihan_tabular/downloads'`
Directory to use for processing intermittent files.

```
unihan_tabular.process.download(url, dest, urlretrieve_fn=<function urlretrieve>, re-
                               porthook=None)
```

Download a file to a destination.

Parameters

- **url** (*str*) – URL to download from.
- **dest** (*str*) – file path where download is to be saved.
- **urlretrieve_fn** (*function*) – function to download file
- **reporhook** (*function*) – Function to write progress bar to stdout buffer.

Returns destination where file downloaded to.

Return type `str`

`unihan_tabular.process.extract_zip(zip_path, dest_dir)`
Extract zip file. Return `zipfile.ZipFile` instance.

Parameters

- **zip_path** (*str*) – filepath to extract.
- **dest_dir** (*str*) – (optional) directory to extract to.

Returns The extracted zip.

Return type `zipfile.ZipFile`

`unihan_tabular.process.files_exist` (*path, files*)
Return True if all files exist in specified path.

`unihan_tabular.process.filter_manifest` (*files*)
Return filtered UNIHAN_MANIFEST from list of file names.

```
unihan_tabular.process.get_fields(d)
```

Return list of fields from dict of {filename: ['field', 'field1']}.

```
unihan_tabular.process.get_parser()
```

Return `argparse.ArgumentParser` instance for CLI.

Returns argument parser for CLI use.

Return type `argparse.ArgumentParser`

```
unihan_tabular.process.has_valid_zip(zip_path)
    Return True if valid zip exists.
```

Parameters `zip_path` (*str*) – absolute path to zip

Returns True if valid zip exists at path

Return type `bool`

`uniHan_tabular.process.in_fields(c, fields)`
Return True if string is in the default fields.

`uniHan_tabular.process.listify(data, fields)`
Convert tabularized data to a CSV-friendly list.

Parameters `data` (`list`) – List of dicts

Params `fields` keys/columns, e.g. ['kDictionary']

`uniHan_tabular.process.load_data(files)`
Extract zip and process information into CSV's.

Parameters `files` (`list`) –

Return type `str`

Returns string of combined data from files

`uniHan_tabular.process.normalize(raw_data, fields)`
Return normalized data from a UNIHAN data files.

Parameters

- **raw_data** (`str`) – combined text files from UNIHAN
- **fields** (`list`) – list of columns to pull

Returns list of uniHan character information

Return type `list`

`uniHan_tabular.process.not_junk(line)`
Return False on newlines and C-style comments.

`uniHan_tabular.process.zip_has_files(files, zip_file)`
Return True if zip has the files inside.

Parameters

- **files** (`list`) – list of files inside zip
- **zip_file** (`zipfile.ZipFile`) – zip file to look inside.

Returns True if files inside of :py:meth: 'zipfile.ZipFile.namelist()'.

Return type `bool`

Utility and helper methods for script.

util

`uniHan_tabular.util.ucn_to_unicode(ucn)`
Return a python unicode value from a UCN.

Converts a Unicode Universal Character Number (e.g. “U+4E00” or “4E00”) to Python unicode (u‘u4e00’)

`uniHan_tabular.util.ucnstring_to_python(ucn_string)`
Return string with Unicode UCN (e.g. “U+4E00”) to native Python Unicode (u‘u4e00’).

`uniHan_tabular.util.ucnstring_to_unicode(ucn_string)`
Return ucnstring as Unicode.

Test helpers functions for downloading and processing UniHan data.

Command Line Interface

```
usage: unihan-tabular [-h] [-s SOURCE] [-z ZIP_PATH] [-d DESTINATION]
                    [-w WORK_DIR] [-F {json,csv,yaml}]
                    [-f [FIELDS [FIELDS ...]]]
                    [-i [INPUT_FILES [INPUT_FILES ...]]]
```

Named Arguments

-s, --source	URL or path of zipfile. Default: http://www.unicode.org/Public/UNIDATA/Unihan.zip
-z, --zip-path	Path the zipfile is downloaded to. Default: <code>/home/docs/.cache/unihan-tabular/downloads/Unihan.zip</code>
-d, --destination	Output of .csv. Default: <code>/home/docs/.local/share/unihan-tabular/unihan.{json,csv,yaml}</code>
-w, --work-dir	Default: <code>/home/docs/.cache/unihan-tabular/downloads</code>
-F, --format	Possible choices: json, csv, yaml Default: csv
-f, --fields	Default: <code>[u'kAccountingNumeric', u'kBigFive', u'kCCCI', u'kCNS1986', u'kCNS1992', u'kCangjie', u'kCantonese', u'kCheungBauer', u'kCheungBauerIndex', u'kCihaiT', u'kCompatibilityVariant', u'kCowles', u'kDaeJaweon', u'kDefinition', u'kEACC', u'kFenn', u'kFennIndex', u'kFourCornerCode', u'kFrequency', u'kGB0', u'kGB1', u'kGB3', u'kGB5', u'kGB7', u'kGB8', u'kGSR', u'kGradeLevel', u'kHDZRadBreak', u'kHKGlyph', u'kHKSCS', u'kHanYu', u'kHangul', u'kHanyuPinlu', u'kHanyuPinyin', u'kIBMJapan', u'kIICore', u'kIRGDaeJaweon', u'kIRGDaiKanwaZiten', u'kIRGHanyuDaZidian', u'kIRGKangXi', u'kIRG_GSource', u'kIRG_HSource', u'kIRG_JSource', u'kIRG_KPSource', u'kIRG_KSource', u'kIRG_MSource', u'kIRG_TSource', u'kIRG_USource', u'kIRG_VSource', u'kJIS0213', u'kJapaneseKun', u'kJapaneseOn', u'kJis0', u'kJis1', u'kKPS0', u'kKPS1', u'kKSC0', u'kKSC1', u'kKangXi', u'kKarlgrén', u'kKorean', u'kLau', u'kMainlandTelegraph', u'kMandarin', u'kMatthews', u'kMeyerWempe', u'kMorohashi', u'kNelson', u'kOtherNumeric', u'kPhonetic', u'kPrimaryNumeric', u'kPseudoGB1', u'kRSAdobeJapan1_6', u'kRSJapanese', u'kRSKanWa', u'kRSKangXi', u'kRSKorean', u'kRSUnicode', u'kSBGY', u'kSemanticVariant', u'kSimplifiedVariant', u'kSpecializedSemanticVariant', u'kTaiwanTelegraph', u'kTang', u'kTotalStrokes', u'kTraditionalVariant', u'kVietnamese', u'kXHC1983', u'kXerox', u'kZVariant']</code>
-i, --input-files	Default: <code>[u'Unihan_RadicalStrokeCounts.txt', u'Unihan_NumericValues.txt', u'Unihan_Variants.txt', u'Unihan_DictionaryIndices.txt', u'Unihan_DictionaryLikeData.txt', u'Unihan_OtherMappings.txt', u'Unihan_Readings.txt', u'Unihan_IRGSources.txt']</code> , files inside zip to pull data from.

History

- : Allow for local / file system sources for Unihan.zip

- : Only extract zip if unextracted
- : Update package classifiers
- : Add back datapackage
- : Fix python 2 CSV output
- : Default to CSV output
- : Support for custom destination output, including replacing template variable `{ext}`
- : Support for XDG directory specification
- : Move unicodesv module to dependency package
- : Move `__about__.py` to module level
- : Fix python package import
- : Fix readme bug on pypi
- : Support for exporting in YAML and JSON
- : Return data as list
- : More internal factoring and simplification
- : Drop python 3.3 and 3.4 support
- : Only use UnicodeWriter in Python 2, fixes issue with python would encode *b* in front of values
- : Drop datapackages in favor of a universal JSON, YAML and CSV export.
- : Rename from `cihaidata_unihan` to `unihan_tabular`
- : Rename `scripts/` to `cihaidata_unihan/`
- : Enable invoking tool via `$ cihaidata_unihan`
- : Switch license BSD -> MIT
- : Lint code, remove unused imports
- : Improve test coverage
- : Get CLI documentation up again
- : Convert full test suite to pytest functions and fixtures
- : Convert to `pytest assert` statements
- : Major internal refactor and simplification
- : Add dev dependencies for isort, vulture and flake8
- : Lock base dependencies
- : Add support for pypy (why not)
- : Update travis to test up to python 3.6
- : Update links on README to use https
- : Update travis to use coverall
- : Update sphinx theme to alabaster with new logo.
- : Update requirements to use `requirements/` folder for base, testing and doc dependencies.
- : Modernize package metadata to use `__about__.py`

- : Add Makefile to main project
- : Modernize *Makefile* in docs
- : Rebooted

u

- `unihan_tabular`, 7
- `unihan_tabular.process`, 7
- `unihan_tabular.test`, 9
- `unihan_tabular.util`, 9

A

ALLOWED_EXPORT_TYPES (in module unihan_tabular.process), 7

D

DESTINATION_DIR (in module unihan_tabular.process), 7

download() (in module unihan_tabular.process), 8

E

extract_zip() (in module unihan_tabular.process), 8

F

files_exist() (in module unihan_tabular.process), 8

filter_manifest() (in module unihan_tabular.process), 8

G

get_fields() (in module unihan_tabular.process), 8

get_parser() (in module unihan_tabular.process), 8

H

has_valid_zip() (in module unihan_tabular.process), 8

I

in_fields() (in module unihan_tabular.process), 9

INDEX_FIELDS (in module unihan_tabular.process), 7

L

listify() (in module unihan_tabular.process), 9

load_data() (in module unihan_tabular.process), 9

N

normalize() (in module unihan_tabular.process), 9

not_junk() (in module unihan_tabular.process), 9

U

ucn_to_unicode() (in module unihan_tabular.util), 9

ucnstring_to_python() (in module unihan_tabular.util), 9

ucnstring_to_unicode() (in module unihan_tabular.util), 9

UNIHAN_FIELDS (in module unihan_tabular.process), 7

UNIHAN_FILES (in module unihan_tabular.process), 8

unihan_tabular (module), 7

unihan_tabular.process (module), 7

unihan_tabular.test (module), 9

unihan_tabular.util (module), 9

UNIHAN_URL (in module unihan_tabular.process), 8

UNIHAN_ZIP_PATH (in module unihan_tabular.process), 8

W

WORK_DIR (in module unihan_tabular.process), 8

Z

zip_has_files() (in module unihan_tabular.process), 9