Contents

1 Introduction **1**

2 Data Definitions **1**

3 Scripts **3**
3.1 JMdictToJson 4
3.1.1 Usage 4
3.1.2 Options 4
3.1.3 Output 4
3.2 RandomWordsToJSON 5
3.2.1 Usage 5
3.2.2 Options 5
3.2.3 Output 5
3.3 JMdictToTries 5
3.3.1 Usage 5
3.4 QuerySpeedtest 5
3.4.1 Usage 5
3.4.2 Output 5

1 Introduction

This document will serve as the recordings of my work with parsing the incredibly expansive Japanese to English dictionary, JMDICT. The purpose of this parser is to a) generate a JSON version of the file to be used with other projects, namely my Yosh!eru project. I am writing this document both for myself and for demonstration purposes. The dictionary is defined by a grammar, which I detail below.

2 Data Definitions

- JMDict (entry*)
- entry (ent_seq, k_ele*, r_ele+, sense+)
  - ent_seq - Entry sequence number
  - k_ele - kanji elements
  - r_ele - reading elements
  - sense - sense element
- ent_seq (#PCDATA)
  - Unique numeric sequence number for each entry
- k_ele (keb, ke_inf*, ke_pri*)
  - The defining element of each entry. In its absence, the reading element is such.
  - Majority of entries have a single kanji element associated with a word in Japanese
  - Where there are multiple kanji elements within an entry, they will be orthographical variants (alternate spellings mostly)
  - Common ”mis-spellings” may be included
  - Synonyms are not included
- keb (#PCDATA)
  - This element will contain a word or short phrase in Japanese which is written using at least one non-kana character.
  - Valid characters are kanji, kana, related characters such as chouon and kurikaeshi, etc.
- ke_inf (#PCDATA)
  - This is a coded information field related specifically to the orthography of the keb
  - Will typically indicate some unusual aspect, such as okurigana irregularity.
- ke_pri (#PCDATA)
  - Along with re_pri field, provided to record information about the relative priority of the entry
  - Consist of codes indicating the word appears in various references which can be taken as an indication of frequency of usage.

  - Current values in this field are:
    * news1/2: appears in the "wordfreq" file compiled by Alexandre Girardi from the Mainichi Shimbun
    * ichi1/2: appears in the "Ichimango goi bunruishuu", Senmon Kyouiku Publishing, Tokyo, 1998
    * spec1/2: a small number of words use this marker when they are detected as being common, but are not included in other lists.
    * gai1/2: common loanwords, based on the wordfreq file.
    * nfxx: this is an indicator of frequency-of-use ranking in the wordfreq file. "xx" is the number of the set of 500 words in which the entry can be found, with "01" assigned to the first 500, "02" to the second, and so on.

- r_ele (reb, re_nokanji?, re_restr*, re_inf*, re_pri*)
  - The reading element typically contains the valid readings of the word(s) in the kanji element using modern kanadzukai
  - Where there are multiple reading elements, they will typically be alternative readings of the kanji element
  - In the absence of kanji, these elements will define the entry
- reb (#PCDATA)
  - This is restricted to kana and related characters such as chouon and kurikaeshi.
  - Kana usage will be consistent between keb and reb elements (if one contains katakana, both will)
- re_nokanji (#PCDATA)
  - This element, which will usually have a null value, indicates that the reb, while associated with the keb, cannot be regarded as a true reading of the kanji.
  - It is typically used for words such as foreign place names, gairaigo which can be in kanji or katakana, etc.
- re_restr (#PCDATA)
  - This element is used to indicate when the reading only applies to a subset of the keb elements in the entry.
  - In its absence, all readings apply to all kanji elements.
  - The contents of this element must exactly match those of one of the keb elements.
- re_inf (#PCDATA)
  - General coded information pertaining to the specific reading.
  - Typically it will be used to indicate some unusual aspect of the reading.
- re_pri (#PCDATA)
  - See the comment on ke_pri
- sense (stagk*, stagr*, pos*, xref*, ant*, field*, misc*, s_inf*, lsource*, dial*, gloss*)
  - The sense element will record the translational equivalent of the Japanese word, plus other related information.
  - Where there are several distinctly different meanings of the word, multiple sense elements will be employed.
- stagk (#PCDATA)
- stagr (#PCDATA)
  - These elements, if present, indicate that the sense is restricted to the lexeme represented by the keb and/or reb.
- xref (#PCDATA)
  - This element is used to indicate a cross-reference to another entry with a similar or related meaning or sense.
  - The content of this element is typically a keb or reb element in another entry.

  - In some cases a keb will be followed by a reb and/or a sense number to provide a precise target for the cross-reference.
    * Where this happens, a JIS "centre-dot" (0x2126) is placed between the components of the cross-reference. The target keb or reb must not contain a centre-dot.

- ant (#PCDATA)
  - This element is used to indicate another entry which is an antonym of the current entry/sense.
  - The content of this element must exactly match that of a keb or reb element in another entry.

- pos (#PCDATA)
  - Part-of-speech information about the entry/sense.
  - Should use appropriate entity codes.
  - In general where there are multiple senses in an entry, the part-of-speech of an earlier sense will apply to later senses unless there is a new part-of-speech indicated.

- field (#PCDATA)
  - Information about the field of application of the entry/sense.
  - When absent, general application is implied.
  - Entity coding for specific fields of application.

- misc (#PCDATA)
  - This element is used for other relevant information about the entry/sense.
  - As with part-of-speech, information will usually apply to several senses.

- lsource (#PCDATA)
  - This element records the information about the source language(s) of a loan-word/gairaigo.
  - If the source language is other than English, the language is indicated by the xml:lang attribute.
  - The element value (if any) is the source word or phrase.
    * lsource xml:lang - Defines the language(s) from which a loanword is drawn
    * lsource ls_type - Indicates whether the lsource elemnt fully or partially descrives the source word of phrase of the loadword. If absent, it will have the implied value of "full". Otherwise it will contain "part".
    * lsource ls_wasei - Indicates that the Japanese word has been constructed from words in the source language

- gloss (#PCDATA | pri)*
  - Within each sense will be one or more "glosses", i.e. target-language words or phrases which are equivalents to the Japanese word.
  - This element would normally be present, however it may be omitted in entries which are purely for a cross-reference.
    * gloss xml:lang - defines the target language of the gloss
    * gloss g_type - Specifies that the gloss is of a particular type, e.g. "lit" (literal), "fig" (figurative), "expl" (explanation)

- pri (#PCDATA)
  - These elements highlight particular target-language words which are strongly associated with the Japanese word.
  - The purpose is to establish a set of target-language words which can effectively be used as head-words in a reverse target-language/Japanese relationship.

- s_inf (#PCDATA)
  - The sense-information elements provided for additional information to be recorded about a sense.
  - Typical usage would be to indicate such things as level of currency of a sense, the regional variations, etc.

3 Scripts

All scripts can be accessed from the main 'JMdictUtils.py' script, or by running them directly, as detailed below.

3.1 JMdictToJson

3.1.1 Usage

Outputs the JMdict to a JSON file The script can be ran with the following command:

```
python3 -m JMdictToJSON.JMdictToJSON [--option]
```

3.1.2 Options

The following options are available for use with the script.

- indent
    1. By supplying –indent=#, will cause each nested level in the JSON file to have # number of indents inserted at the front of the line
- low memory
    - With –low-memory, tells the parser to keep a low memory profile. The output is written in chunks (currently every 10,000 entries), and entries are not kept in memory.

3.1.3 Output

Current output is hardcoded to fit the needs of my own projects. Revisions will be made to make output format customizable. The following two examples show the format of the output. Note that since certain fields are optional for any given entry, some fields are omitted.

- indent=0


```
{ent_seq:"1004660",k_ele:[{keb:"この外",ke_pri:["spec1"]}],r_ele:[{reb:"このほか",re_pri:
["spec1"]}],sense:[{pos:["conj"],misc:["uk"],gloss:["besides", "moreover","in addition"]}]}
```


- indent=2


```
{
  ent_seq: "1004660",
  k_ele: [
    {
      keb: "この外",
      ke_pri: [
        "spec1"
      ]
    }
  ],
  r_ele: [
    {
      reb: "このほか",
      re_pri: [
        "spec1"
      ]
    }
  ],
  sense: [
    {
      pos: [
        "conj"
      ],
      misc: [
        "uk"
      ],
      gloss: [
        "besides",
        "moreover",
        "in addition"
      ]
    }
  ]
}
```

3.2 RandomWordsToJSON

3.2.1 Usage

Outputs a random selection of words to a JSON file for analytical purposes The script can be ran with the following command:

```
python3 -m RandomWordsToJSON.RandomWordsToJSON [options...]
```

3.2.2 Options

- –min-word-length(-m)=# Minimum length for word to include. Uses first kanji (hiragana if no kanji exists) of element to get word length

3.2.3 Output

Currently writes the random words to a JSON file structured the same as the one generated by JMdict-ToJSON with indent=2

3.3 JMdictToTries

3.3.1 Usage

Generates two tries from JMdict for word search speed analysis. One trie uses the kanji and hiragana of each entry as 'keys', the other uses the entry sequence.

Currently doesn't output any file/has no usage outside other scripts

3.4 QuerySpeedtest

3.4.1 Usage

Uses RandomWords to run lookup queries in order to test retrieval speeds of various methods.

Currently tests against 1) basic Python dictionary and 2) trie data structure

```
python3 -m QuerySpeedtest.QuerySpeedtest
```

3.4.2 Output

Results of the test are written to console