Development of the NVH schema format for lexicographic purposes

Marek Medveď, Miloš Jakubíček, Vojtěch Kovář, Tomáš Svoboda

Raslan 2023

sketchengine.eu

Name-Value Hierarchy (NVH)

- lightweight markup language targeted at dictionary development
- user-friendly alternative to XML-encoded plain text formats
- store dictionary entries in a more compact and readable form
- Name-Value Hierarchy introduced in: *Using NVH as a Backbone Format in the Lexonomy Dictionary Editor, Jakubíček et. all, RASLAN 2022*
- currently the backbone format of the Lexonomy system (recent development required changes in NVH)

NVH sample

- consists of a **list of nodes**, where each node has its **name** and **optional value** separated by a colon-space character
- each node can also contain a list of **child nodes**

```
hw: car
    lemma: car
    pos: NOUN
    image: car.jpg
        quality: good
        explicit: no
```

sketchengine.eu

Basic NVH schema node restrictions

```
hw: +
    lemma:
    lempos: ?
    pos:
    freq: ?
    audio: *
    image: 2+
        quality: ?
        explicit: ?
        source: ?
    examples:
        example: 2+
    translation: *
        language:
    affiliation: ?
```

sketchengine.eu

New NVH schema

```
hw: +
    lemma: ?
    lempos: ? ~.*-.
    pos: ?
    freq: ? int
    audio: * audio
    image: 1-5 image
        quality: ? ["good","bad"]
        explicit: ? bool
        source: ? url ~.*pixabey.*
    examples: empty
        example: 2+ ~.{1,50}
    translation: *
        language: ~.{3}
    affiliation: * ["MU (Brno)", "VUT \"Brno\""]
```

sketchengine.eu

Extended definition for number of nodes

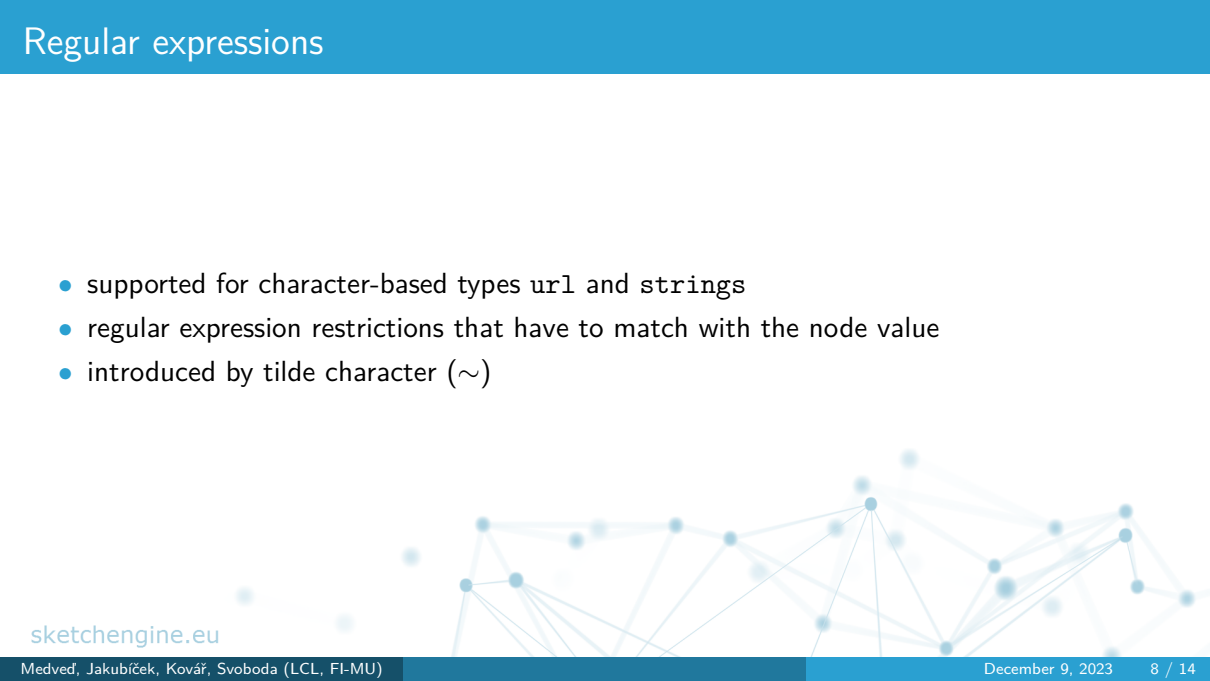
- new schema definition can encode how many nodes with the specific name are required inside the dictionary entry
- new schema definition introduces range "*1-5*" that can set the upper bound as well as the lower bound

Value types

- `audio` type: `.mp3`, `.wav`, `.wma`, ...
- `bool` type: `True`, `False`, `true`, `false`, `Yes`, `No`, `yes`, `no`, `0`, `1`.
- `empty` type does not put any restriction on the value but requires the value to be **empty**. -> used as a wrapper
- `image` type: `.jpeg`, `.jpg`, `.png`, ...
- `int` type
- `list` type
- `string` (default)
- `url` type

Regular expressions

- supported for character-based types `url` and `strings`
- regular expression restrictions that have to match with the node value
- introduced by tilde character ($\sim$)



New NVH schema

```
hw: +
    lemma: ?
    lempos: ? ~.*-.
    pos: ?
    freq: ? int
    audio: * audio
    image: 1-5 image
        quality: ? ["good","bad"]
        explicit: ? bool
        source: ? url ~.*pixabey.*
    examples: empty
        example: 2+ ~.{1,50}
    translation: *
        language: ~.{3}
    affiliation: * ["MU (Brno)", "VUT \"Brno\""]
```

sketchengine.eu

NVH script modifications

- `nvh.py` is adopted to the mentioned modifications
- adoped schema validation, as well as schema generation operations
- new JSON export for Lexonomy: frontend **validates** whether annotators' input **follow the restrictions** defined by the schema **before being stored** in the final dictionary NVH file

JSON export

```
"hw": {"min": 1, "max": Infinity, "type": "string",
       "children": ["lemma", "lempos", "pos", "freq",
                    "audio", "image", "examples",
                    "translation", "affiliation"]},
"lemma": {"max": 1, "type": "string"},
"lempos": {"max": 1, "type": "string", "re": ".*-."},
"pos": { "max": 1, "type": "string"},
"freq": {"max": 1, "type": "int"},
"audio": {"max": Infinity, "type": "audio"},
"image": {"min": 1,"max": 5, "type": "image",
          "children": ["quality", "explicit",
                       "source"]},
"quality": {"max": 1, "type": "list",
            "values": ["good", "bad"]},
"explicit": {"max": 1, "type": "bool"},
"source": {"max": 1, "type": "url", "re": ".*pixabey.*"},
```

ENTRY STRUCTURE

Chose modules you want to use in your dictionary. You can change this later.

AVAILABLE MODULES

- ✓ ENTRY
  - ☐ FLAG
  - ✓ SENSE
    - ✓ SENSE EXAMPLE
      - ☐ EXAMPLE TRANSLATIONS
    - ☐ TRANSLATIONS

FINAL ENTRY STRUCTURE

```
entry:
  headword:
  partOfSpeech:
  sense:
    definition:
    example:
      text:
```

Conclusions

- new NVH schema modifications
- new **type restrictions** together with **regular expressions** -> more control over dictionary entries
- schema strictly **directs the annotators** during the dictionary development
- **prevents mistakes** and unnecessary **post-processing** of inconsistent annotations

sketchengine.eu

Thank you for your attention.

sketchengine.eu