Development of the NVH Schema Format for Lexicographic Purposes

Marek Medveď[1,2], Miloš Jakubíček[1,2], Vojtěch Kovář[1,2], and Tomáš Svoboda[1]

[1] Lexical Computing,
Brno, Czechia
[2] Masaryk University
Brno, Czechia
name.surname@sketchengine.eu

Abstract. A unified e-dictionary entry format is one of the most important things to consider when building a new dictionary. In the Lexonomy tool, where the new NVH lightweight markup language is used to store dictionary data, an NVH schema is assigned to each dictionary, specifying the NVH structure belonging to each dictionary entry. Until now, the schemata used in Lexonomy were quite limited and focused only on the position of a node in the NVH hierarchy and on the arity of its occurrence. In the recent development, we identified a need for a more fine-grained restriction mechanism and, therefore, extended the NVH schema format so that it can also inspect the value of each node according to its type and confirm match according to a predefined regular expression.

Keywords: NVH, XML, Name-Value Hierarchy, Lexonomy, Sketch Engine

1 Introduction

The Name-Value Hierarchy or NVH is a lightweight markup language targeted at dictionary development [1]. It is a user-friendly alternative to XML-encoded plain text formats, currently used in many digital dictionaries, and store dictionary entries in a more compact and readable form.

The NVH language is currently the backbone format of the Lexonomy system [2,3], which is focused on digital dictionary development. Recent developments of the Lexonomy system led to changes that needed to be incorporated into the NVH language, especially the schema that specifies the final structure of the developed dictionary.

In this paper, we will introduce dictionary schema modifications that restrict each node value inside the NVH according to the model needs.

2 Name-Value Hierarchy markup language

Name-Value Hierarchy (NVH) is a plain text-based format similar to XML with much simpler syntax. An NVH file consists of a list of nodes, where each node

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2023, pp. 101–106, 2023.

has its name and optional value separated by a colon-space character. Each node can also contain a list of child nodes prefixed by Python-like indentation (see Figure 1).

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

Fig. 1: NVH sample

3 Schema

A dictionary schema is always required along with the dictionary content in NVH. The purpose of the schema is to avoid nodes that are not important or unwanted for the dictionary in hand and also to automatically check if the dictionary content complies with the predefined dictionary schema.

The previous version of an NVH schema supported essential value restriction that defines the number of required nodes with the specific name (see Figure 2). Using the question mark character ("?") on the position of the value, the schema allows the use of none or one node with the specified name (i.e., lempos). The plus character ("+") requires at least one node with the name inside the dictionary (like hw). Similarly, the number followed by the plus character requires using at least the given number of nodes. The star character ("*") puts no restrictions on the node as it can appear from zero to infinity times (i.e., audio). Finally, the names with no value in the schema must appear exactly once (i.e., lemma).

The expressive ability of the previous schema implementation was very limited, and could not catch most mistakes made by annotators. For example, we could require *lempos* attribute to be present in each dictionary entry. Still, there is no option to set the format of the *lempos* value if we want it to be a combination of the lemma and one character representing the part of speech. Therefore, we expanded the original schema form with a new set of parameters.

In this new NVH schema, we introduce an extended definition for the number of nodes, type of the value, and usage of regular expressions for string values.

3.1 Extended definition for number of nodes

Similarly to the previous version, the new schema definition can encode how many nodes with the specific name are required inside the dictionary entry.

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

Fig. 2: Basic NVH schema node restrictions

Using the *"?"*, +, *, and *2+*, we can restrict the number of nodes as we introduced above. On top of that, the new schema definition introduces range *"1-5"* that can set the upper bound as well as the lower bound, which was not previously available (i.e., image node in Figure 3).

```
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\"", "UK, Praha"]
```

Fig. 3: NVH schema with all supported value restrictions

3.2 Value types

The new schema format introduces the support for typing. There are seven types that have been developed according to Lexonomy usage and should cover the majority of user needs inside the Lexonomy tool:

- `audio` type is a string that restricts the node value an audio file by matching the string with a list of supported audio extensions. In the current version we support these audio extensions: `.3gp`, `.aa` , `.aac`, `.aax` , `.act` , `.aiff` , `.alac` , `.amr` , `.ape` , `.au`, `.awb`, `.dss`, `.dvf`, `.flac`, `.gsm`, `.iklax`, `.ivs`, `.m4a`, `.m4b`, `.m4p`, `.mmf`, `.movpkg`, `.mp3`, `.mpc`, `.msv`, `.nmf`, `.ogg`, `.oga`, `.mogg`, `.opus`, `.ra`, `.rm`, `.raw`, `.rf64`, `.sln`, `.tta`, `.voc`, `.vox`, `.wav`, `.wma`, `.wv`, `.webm`, `.8svx`, `.cda`.
- `bool` type restrict the node value to just Boolean values `True` and `False`. These two values can be expressed by: `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. This type should be used for the nodes that introduce a container, like *examples* node in Figure 3.
- `image` type is the same as audio type except for the list of supported extensions: `.jpeg`, `.jpg`, `.png`, `.gif`, `.bmp`, `.tiff`, `.svg`, `.raw`, `.ico`, `.webp`, `.heic`, `.heif`, `.psd`, `.eps`, `.ai`, `.tga`, `.pdf`.
- `int` type restricts the value only to integer numbers.
- `list` type is not explicitly used inside the NVH schema but is determined according to the list of possible values (like *quality* and *affiliation* in Figure 3).
- `string` is the default type and does not need to be explicitly used in the schema.
- `url` type confirms if the node's value is a URL link.

3.3 Regular expressions

Character-based types `url` and `string` additionally support regular expression restrictions that have to match with the node value. The tilde character (∼) always introduces a regular expression. The format of the regular expression follows the Python regular expression syntax of the `re` module[3] or any other user-specified format that should be provided as a comment in the schema (a comment is any line starting with the # character).

3.4 NVH script modifications

The Python script `nvh.py` is adopted to the above-mentioned modifications. The schema validation, as well as schema generation operations, are modified to account for the new types and regular expressions.

[3] https://docs.python.org/3/library/re.html

```
{
"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": {"children": ["quality", "explicit", "source"],
          "min": 1,"max": 5, "type": "image"},
"quality": {"max": 1, "type": "list", "values": ["good", "bad"]},
"explicit": {"max": 1, "type": "bool"},
"source": {"max": 1, "type": "url", "re": ".*pixabey.*"},
"examples": {"children": ["example"], "min": 1, "max": 1, "type": "empty"},
"example": {"min": 2, "max": Infinity, "type": "string", "re": ".{1,50}"},
"translation": {"children": ["language"], "max": Infinity, "min":0,
                "type": "string"},
"language": {"min": 1, "max": 1, "type": "string", "re": ".{3}"},
"affiliation": {"max": Infinity, "min": 0, "type": "list",
                "values": ["MU (Brno)", "VUT \"Brno\"", "UK, Praha"]}
}
```

Fig. 4: JSON export of the NVH schema from Figure 3

For Lexonomy purposes, we also include a new type of export. The NVH schema can now be exported into the JSON format that can be useful for any tool incorporating the NVH format. Currently, this export is used by the Lexonomy system frontend to validate whether annotators' input follow the restrictions defined by the schema before being stored in the final dictionary NVH file. Figure 3 presents an example of JSON export of the schema from Figure 4.

4 Conclusions

This paper presents new NVH schema modifications that allow dictionary managers to define more precisely how the dictionary entry will be developed. The new type of restrictions together with regular expressions can exactly specify the value of each node. This unique design of the NVH schema strictly directs the annotators during the dictionary development and avoids mistakes and unnecessary post-processing of inconsistent annotations.

References

1. Jakubíček, M., Kovář, V., Měchura, M., Rambousek, A.: Using NVH as a Backbone Format in the Lexonomy Dictionary Editor. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2022. (2022) 55–61

2. Měchura, M.B., et al.: Introducing Lexonomy: an open-source dictionary writing and publishing system. In: Electronic Lexicography in the 21st Century: Lexicography from Scratch. Proceedings of the eLex 2017 conference. (2017) 19–21
3. Rambousek, A., Jakubíček, M., Kosem, I.: New Developments in Lexonomy. Electronic lexicography in the 21st Century (eLex 2021) Post-editing lexicography (2021) 86