A Customizable WordNet Editor

Andrei-Marius Avram
Research Institute for Artificial Intelligence, Romanian Academy,
University POLITEHNICA of Bucharest
avram.andreimarius@gmail.com

Verginica Barbu Mititelu
Research Institute for Artificial Intelligence, Romanian Academy
vergi@racai.ro

Abstract

This paper presents an open-source wordnet editor that has been developed to ensure further expansion of the Romanian wordnet. It comes with a web interface that offers capabilities in selecting new synsets to be implemented, editing the list of literals and their sense numbers and adding these new synsets to the existing network, by importing from Princeton WordNet (and adjusting, when necessary) all the relations in which the newly created synsets and their literals are involved. The application also comes with an authorization mechanism that ensures control of the new synsets added in novice or lexicographer accounts. Although created to serve the current (more or less specific) needs in the development of the Romanian wordnet, it can be customized to fulfill new requirements from developers, either of the same wordnet or of a different one for which a similar approach is adopted.

Keywords: wordnet editor, Romanian, synset creation, semantic relations, non-lexicalized synsets

1. Introduction

The importance of wordnets as linguistic knowledge repositories and as resources exploitable by language applications has been widely acknowledged and at present the wordnet community is still an active one, where:

- mature wordnets are:
  - exploited with state of the art technologies (Kafe, 2019), and
  - enriched (Dziob et al., 2019), even voluntarily (McCrae et al., 2019),
- projects for building wordnets still continue, either:
  - in the traditional way (Dziob et al., 2019) or
  - adding new dimensions to the lexico-semantic network: audio (Kashyap et al., 2019), visual (Deng et al., 2009) or another modality (Lualdi et al., 2019),
- wordnet development projects are debuting (Sio and da Costa, 2019),
- existing wordnets are converted from their various formats to a common format (Bond and Foster, 2013) in order to ensure their interlinking, as well as linking to other resources (Simov et al., 2019).

Against this effervescent background, we present the work for developing a tool for continuing the enrichment of the Romanian wordnet (RoWN, henceforth) with new synsets and relations. The development of the RoWN started in the BalkaNet project (Tufiș et al., 2004). That was when the implementation principles were established, the necessary tools were developed (Barbu and Tufiș, 2004)

and a core RoWN was created. Its enrichment continued in other projects (Tufiş et al., 2013), but had to stop abruptly when technical problems occurred and no support could be offered to lexicographers. However, enrichment with new relations and new types of information relevant to various applications was possible: further derivational relations between literals were added (Barbu Mititelu, 2013), verbal multiword expressions were annotated with four types (Barbu Mititelu and Mitrofan, 2019) established within the PARSEME initiative (Ramisch et al., 2018).

When developing a wordnet manually, an editor customized to the steps to take in this process is a must and many teams have developed their own (see section 2. below). For RoWN the tools, WNBuilder and WNCorrect (Barbu and Tufiș, 2004), installed locally, ensured a two-phase process, and several scripts complemented them: the synsets to be implemented were selected from Princeton WordNet (Fellbaum, 1998) (PWN, henceforth) with the help of a script, following criteria of interest at different moments in the project evolution, WNBuilder was used by lexicographers to create the synsets (i.e., enumerate the literals and specify their sense number) and their glosses, these synsets were automatically added to the RoWN file together with all the respective relations imported from PWN (thus ensuring the alignment of RoWN with PWN), ensuring their structural correctness at the same time (i.e., all literals must have a sense number, a literal could not occur twice in the same synset irrespective of its sense number, etc.). All semantic errors (i.e., the same literal with the same sense number occurring in at least two different synsets) were automatically identified with the help of a script and they were uploaded in the WNCorrect interface, where lexicographers manually corrected them. The semantic correction phase was iterative until no more errors were found. As mentioned above, the tools are no longer usable, given technical limitations (their dependence on older version of operating systems and of other software).

Continuing the development of the RoWN required a tool that would allow for synchronous development by different lexicographers and that would ensure both the syntactic and the semantic correctness of the new synsets. We present here the tool we have created to serve these needs.

The paper is organized as follows: section 2. briefly presents the wordnet editing tools available, in section 3. we explain the specificities of RoWN that required a new editing tool, which is described in details in section 4. The possibilities for customizing this editor for other wordnets are presented in section 5., before concluding the paper.

2. Other Wordnet Editors

As mentioned above, almost each team developing a wordnet has created a tool to help in this process. Some of them are project-specific, others are designed to serve multiple project, as well as multiple tasks: development, validation, visualization. We review below several such tools that we considered for further enrichment of RoWN.

Hydra (Rizov, 2014) is an open-source system allowing for PWN synsets cloning and their further modification (in any way: adding literals, deleting literals, deleting the whole synset, undoing any action, as well as redoing it), relations import, creation of language-specific synsets and their linking to existing synsets in the wordnet. It allows for simultaneous access and use by multiple users, with modifications becoming accessible to all of them instantly. Hydra can be used in browser, does not require local installation. RoWN can be queried in the 'Hydra for web' tool[1], built on top of Hydra, either in a single view mode or in parallel with other wordnets uploaded in the tool and which RoWN is aligned with at the synset level. Given the logic model behind Hydra (Rizov, 2008), the sense numbering of homograph literals ignores their part of speech, as well as the sense numbers in the original wordnets. Given the almost parallel development of the Bulgarian and Romanian wordnets (Barbu Mititelu et al., 2019), Hydra would have made an adequate solution for continuing the RoWN development, provided that it had offered a solution to the sense numbering system specific to RoWN (see section 3.) and to allowing access to language resources used by lexicographers when implementing new synsets or ensuring their quality.

DEBVisDic (Horák et al., 2006) is another tool for creating and further editing wordnets, as well as for querying them and visualizing the results even in several wordnets at a time, in a synchronous mode

[1] http://dcl.bas.bg/bulnet/

(the autolookup function). It also offers the possibility of multiple users working simultaneously, without interfering with the work of each other. However, some relations have to be manually added, it is quite restrictive in access (noncommercial, nonprofit internal research purposes only), runs only on Mozilla's Firefox, cannot cope with the RoWN sense numbering system. At the beginning of the BalkaNet project, our team used a previous version of this tool, namely VisDic (Horák and Smrz, 2004), but only for visualization of wordnets content, as well as of our in-house Romanian explanatory dictionary that was in XML format, compliant with the requirements of the VisDic tool.

OMW editor (da Costa and Bond, 2015) is web-based, but requires local installation. It can be used for various languages (thus being advertised as a "multilingual editing environment"), by any number of users simultaneously, with their work becoming available for the others immediately. The tool also allows for checks of the work done, i.e. ensuring the structural validity of synsets, with no sense numbers, definitions, etc. missing.

Other wordnet editors were mainly tailored on the wordnet to be developed, on the respective creation process, on the resources used in this process, and many others. Their adaptation to the specificity of other wordnets and to the needs of other wordnets developers are not trivial and also lengthy: see the adaptation of WordnetLoom, created for the development of the Polish wordnet, to the requirements of the Portuguese wordnet development, as reported by Naskret et al. (2018).

As such, we faced the challenge of finding a new solution for continuing the development of our wordnet with new synsets.

3. The Romanian Wordnet

The method for creating RoWN consists in finding the right equivalent(s) of the PWN literals in a set of synsets chosen to be implemented and the transfer of the PWN semantic relations between the implemented synsets. This is the expand method (Rodriguez et al., 1998) in wordnets building. Two principles were always observed when selecting a set of synsets to implement: the Hierarchy Preservation Principle (semantic relations were imported automatically from PWN) and the Conceptual Density Principle (ensuring that no orphan synsets, i.e. lower-level synsets without direct ancestors, are created) (Tufiș et al., 2004).

One of the resources exploited for creating the RoWN was an in-house explanatory dictionary of Romanian. Its senses are organized in nests, which means that the main senses of a lexical entry are identified and for each such sense all its subsenses are defined; even subsenses can have sub-subsenses. The subsenses are clearly semantically related to the respective main sense and the semantic similarity between a subsense and its main sense is more obvious than the semantic similarity between main senses. The same applies to sub-subsenses and subsenses. While main senses are distinguished by integer sense numbers (e.g., literal:*1*, literal:*2*, etc.), subsenses are assigned decimal sense numbers (e.g., literal:*1.1*, literal:*1.2*, literal:*2.1*, literal:*2.2*, and even literal:*1.1.1*, literal:*1.1.2*, etc. for sub-subsenses). This sense numbering system was preserved for the literals included in RoWN (Tufiş et al., 2013), as semantically valuable and relevant information for cases when RoWN is used for calculating semantic distances between words or word senses. Moreover, these sense numbers can also be suffixed with ".x" whenever a subsense of a sense existing in the dictionary is necessary for rendering the PWN equivalent. Whenever for a literal existent in the Romanian dictionary a sense was needed that was not recorded in the dictionary (but lexicographers found it attested in Romanian corpora), the respective literal got the sense number "x" (this time, not a suffix to a sense number, but a sense number itself). The same sense number was used for any sense of a literal nonexistent in the dictionary. We can thus notice the multiple values this "x" has. When two different PWN synsets are difficult to understand as semantically different by our lexicographers (after analysing the glosses, the examples, the sets of synonyms, other lexicographic resources or corpora), they are considered artificially distinct synsets and their equivalent Romanian synsets contain the same literal with the same sense number, this time suffixed with ".c", which is further semantic information in our wordnet, signaling the possibility of semantically clustering together the respective synsets (Tufiş et al., 2013).

Sense numbering starts from 1 for the homonyms belonging to a given part of speech, which is, in

fact, also the case in PWN. As such, it becomes mandatory to specify the part of speech for a combination literal:sense number, as the same combination can apply to two or even more parts of speech, thus being ambiguous.

4. Implementation and Functionalities

This section further presents the decisions that were made during the development of RoWN Editor and the technical details of its implementation. The main aims in its development were portability, so it can easily be deployed on any operating system, and ease of implementation and maintenance, so a reliable and easy to use application was to be developed. We found out that Python was a perfect fit because it is an interpreted, general-purpose programming language that offers a lot of useful packages.

In order to make RoWN Editor a web-based application, we had to choose a web framework. Flask[2] proved to be the best candidate because of its lightweight nature and because it was designed to make getting started quick and easy, with the ability to scale up to complex applications. We also used packages that offer support for wordnets: RoWN API (Dumitrescu et al., 2018) for the RoWN and Natural Language Toolkit (NLTK) (Loper and Bird, 2002) for PWN. The rest of Python dependencies are listed in the `requirements.txt` file and can easily be installed with The Python Package Index (PyPI) via the `pip install -r requirements.txt` command. RoWN Editor has no system dependencies.

From the architectural point of view, the application is composed of three modules: synset selection, synset creation and authorization mechanisms. Each of these modules will be described in the following subsections and a workflow diagram that shows the interactions of these modules is depicted in Figure 1.

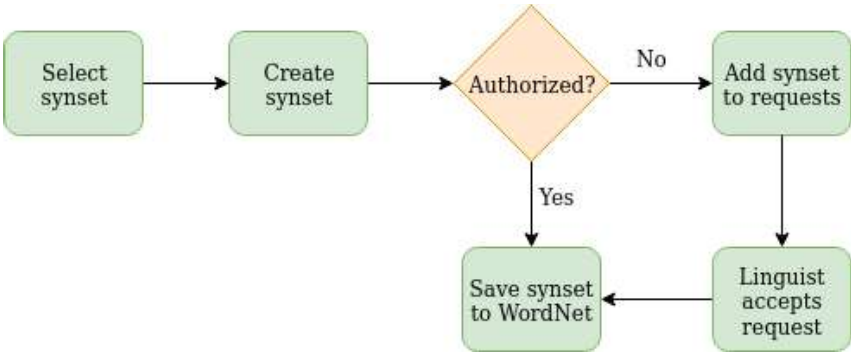

Figure 1: The work flow diagram of RoWN Editor. The user selects a new synset and implements it. The newly created synset will be directly added to the Romanian WordNet if the user is authorized (i.e. (s)he is a lexicographer). Otherwise, the synset will be saved in a list of requested synsets and will not be added until a lexicographer accepts the request.

4.1. Synset Selection

The RoWN Editor comes with an automatic way of suggesting new synsets that can be further added to the structure of wordnet. It does this by exploiting the partial mapping of synset IDs between RoWN and PWN. It selects the nodes (synsets) that exist in PWN, but not in RoWN, and have at least one edge (relationship) with a node that is implemented in RoWN. Figure 2 depicts the role of the synsets from a part of the hypernym tree when the selection algorithm is run. The green nodes represent the common synsets between RoWN and PWN. The red nodes represent the selected synsets - that are implemented in PWN, but not implemented in RoWN, and have at least one relation with one of the common synsets (the green nodes). The white nodes represent the English synsets that are neither implemented in RoWN, nor have a relation with one of the common synsets.

It must be noted that the existence of common edges between the selected synsets and the already implemented synsets in RoWN is a necessary condition, because otherwise it would imply the creation of a disconnected graph, thus disobeying the Conceptual Density Principle (see section 3.). Another

[2]The official documentation of Flask can be found at `https://flask.palletsprojects.com/en/1.1.x/`

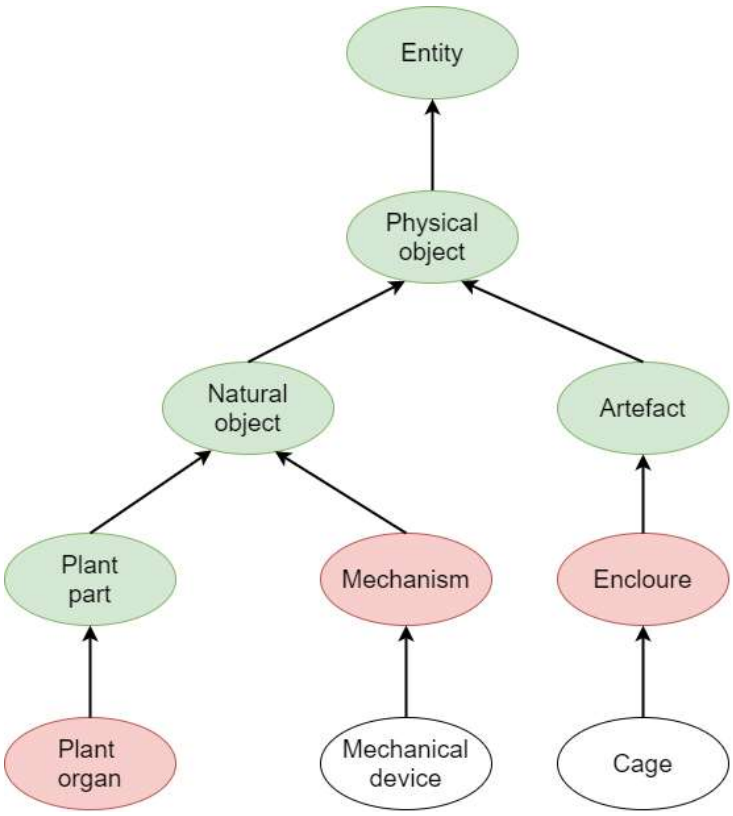

Figure 2: The synset selection algorithm.

important observation is that the algorithm does not return all the possible synsets, but only the first N nodes found, simply because of time constraints. N is a configurable parameter, that can also be specified as "all" to return every possible synset. However, this is not recommended because the selection algorithm might take a long time to find all the synsets. When needed, further restrictions can be imposed on the selected synsets: e.g., synsets belonging to a certain domain or involved in a certain type of relation, etc.

4.2. Synset Creation

Another important feature of the RoWN Editor is the ease of creating new synsets in the web interface. Once a user has selected an unimplemented synset, (s)he will be redirected to a new page where (s)he will have to complete the wordnet entry with the defining fields of a synset (definition, nonlexicalized, stamp, lemmas and lemmas sense number). The user can freely add and remove any number of lemmas, using the buttons "Add lemma" and "Remove lemma", respectively. If the Nonlexicalized box is checked, this option will be disabled and the added lemmas will not be taken into consideration. Once the synset is considered complete, the user can create it by clicking the button "Create synset". RoWN Editor will automatically import its identification number (ID) and all the relations between Romanian synsets from PWN, and, depending on user privileges, it will be either added directly to the wordnet or to a request list. To facilitate the implementation, the top of the page contains the English description of the selected synset. Figure 3 depicts the creation interface for the PWN synset {four-stroke engine:1, four-stroke internal-combustion engine:1}.

4.3. Authorization

The application has two types of users: the novice user and the lexicographer. One of the roles of the latter is to supervise the former by accepting, editing and rejecting the synsets proposed by him/her. RoWN Editor implements this feature by saving the synsets of the novice user in a list of requests, that will be added to the wordnet only after a lexicographer approves it. This mechanism ensures consistency

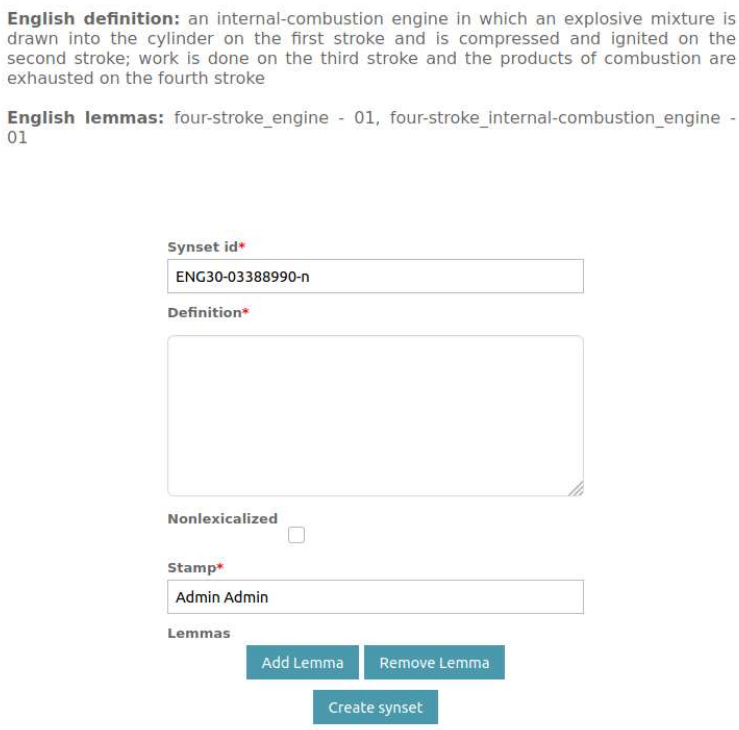

Figure 3: Synset creation interface for the synset with id ENG-30-03388990-n

in the working methodology.

5. Compatibility with Other Wordnets

The RoWN used in the application can be replaced with any wordnet by simply replacing the file `RoWordNet.xml` from the `rowordnet` directory with another wordnet in the XML format. However, the new wordnet must be aligned to PWN so that the synset selection algorithm can work. Also the new wordned XML must observe the following format:

- the root tag must be named `WORDNET`;
- each synset must be inside the `WORDNET` tag and must be marked by the `SYNSET` tag;
- a synset must contain the following tags:
  - `ID` - the identification number of the synset: this is identical to the PWN corresponding synset;
  - `POS` - the part of speech of the literals in the synset;
  - `SYNSONYM` - this is where the literals of the synset are listed. Each literal will be marked by the `LITERAL` tag, and each literal must contain the `SENSE` tag, denoting its sense number;
  - `STAMP` - it contains the name of the creator of the synset; this is an optional tag;
  - `DEF` - the gloss of the synset;
  - `ILR` - an inbound relation of the synset. It must contain the `TYPE` tag that denotes the type of relation with the synset it refers to. This tag can repeat as many times as many relations the synset has with other synsets.

Figure 4 depicts the synset with ID ENG30-00006269 from the RoWN XML. It can be observed that besides the above mentioned tags, it also contains the `DOMAIN`, `SUMO` and `SENTIWN` tags. These were imported into RoWN, but their presence is not manadatory in other wordnets and, thus, they can be omitted.

```
<SYNSET>
  <ID>ENG30-00006269-n</ID>
  <POS>n</POS>
  <SYNONYM>
    <LITERAL>viață<SENSE>1</SENSE></LITERAL>
  </SYNONYM>
  <STAMP>Dan Cristea</STAMP>
  <ILR>ENG30-00004258-n<TYPE>hypernym</TYPE></ILR>
  <ILR>ENG30-07993776-n<TYPE>hyponym</TYPE></ILR>
  <DEF>forme de viață, văzute în mod global; " Nu există viață pe Marte "</DEF>

  <DOMAIN>factotum</DOMAIN>
  <SUMO>Organism<TYPE>=</TYPE></SUMO>
  <SENTIWN>
    <P>0.0</P>
    <N>0.0</N>
    <O>1.0</O>
  </SENTIWN>
</SYNSET>
```

Figure 4: XML format for a synset in the RoWN.

6. Conclusion

This paper introduces the RoWN Editor, an application that facilitates the extension of the RoWN by offering an intuitive graphical interface through which a user can create new synsets. It comes with an algorithm that automatically suggests new synsets by comparing the RoWN with the PWN, and also an authorization mechanism that ensures its consistency by allowing only a lexicographer to add new synsets. Furthermore, the application has been developed for RoWN, but it can be customized to allow editing any other wordnet that respects the presented XML format.

RoWN Editor is a web application developed entirely in Python using Flask. It has no system dependencies, has an open MIT license and can be installed by following the steps at `https://github.com/avramandrei/RoWordNet-Editor`.

References

Barbu, E. and Tufiș, D. (2004). A Methodology and Associated Tools for Building Interlingual Wordnets. In *Proceedings of LREC2004*, pages 1067–1070.

Barbu Mititelu, V. and Mitrofan, M. (2019). Leaving No Stone Unturned When Identifying and Classifying Verbal Multiword Expressions in the Romanian Wordnet. In *Proceedings of Global WordNet Conference*, pages 10–15.

Barbu Mititelu, V., Stoyanova, I., Leseva, S., Mitrofan, M., Dimitrova, T., and Todorova, M. (2019). Hear about Verbal Multiword Expressions in the Bulgarian and the Romanian Wordnets Straight from the Horse's Mouth. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 2–12.

Barbu Mititelu, V. (2013). Increasing the Effectiveness of the Romanian Wordnet in NLP Applications. *Computer Science Journal of Moldova*, 21:320–331.

Bond, F. and Foster, R. (2013). Linking and extending an open multilingual wordnet. In *51st Annual Meeting of the Association for Computational Linguistics: ACL-2013*, pages 1352—-1362.

da Costa, L. M. and Bond, F. (2015). OMWEdit - The Integrated Open Multilingual Wordnet Editing System. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 73–78. ACL and AFNLP.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dumitrescu, S. D., Avram, A. M., Morogan, L., and Toma, S.-A. (2018). RoWordNet–A Python API for the Romanian WordNet. In *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE.

Dziob, A., Piasecki, M., and Rudnicka, E. (2019). plWordNet 4.1 – a Linguistically Motivated, Corpus-based Bilingual Resource. In *Proceedings of the 10th Global WordNet Conference*, pages 353–362.

Fellbaum, C., Ed. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Horák, A. and Smrz, P. (2004). VisDic - Wordnet Browsing and Editing Tool. In *Proceedings of GWC2004*, pages 136–141.

Horák, A., Pala, K., Rambousek, A., and Povoln, M. (2006). Debvisdic–first version of new client-server wordnet browsing and editing tool. In *Proceedings of the Third International Wordnet Conference (GWC-06)*.

Kafe, E. (2019). Fitting Semantic Relations to Word Embeddings. In *Proceedings of the 10th Global WordNet Conference*, pages 228–237.

Kashyap, K., Sarma, S. K., and Sweta, K. (2019). Spoken WordNet. In *Proceedings of the 10th Global WordNet Conference*, pages 260–263.

Loper, E. and Bird, S. (2002). Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Lualdi, C., Hudson, J., Fellbaum, C., and Buchholz, N. (2019). Building ASLNet, a Wordnet for American Sign Language. In *Proceedings of the 10th Global WordNet Conference*, pages 315–322.

McCrae, J. P., Rademaker, A., Bond, F., Rudnicka, E., and Fellbaum, C. (2019). English WordNet 2019 – An Open-Source WordNet for English. In *Proceedings of the 10th Global WordNet Conference*, pages 245–252.

Naskret, T., Dziob, A., Piasecki, M., Saedi, C., and Branco, A. (2018). WordnetLoom – a Multilingual Wordnet Editing System Focused onGraph-based Presentation. In *Proceedings of the 9th Global Wordnet Conference*.

Ramisch, C., Cordeiro, S., Savary, A., Vincze, V., Barbu Mititelu, V., Bhatia, A., Buljan, M., Candito, M., Gantar, P., Giouli, V., Güngör, T., Hawwari, A., Iñurrieta, U., Jolanta Kovalevskaite, and Simon Krek, a. T. L., Liebeskind, C., Monti, J., Parra Escartín, C., QasemiZadeh, B., Ramisch, R., Schneider, N., Stoyanova, I., Vaidya, A., and Walsh, A. (2018). Edition 1.1 of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multi-word Expressions and Constructions*, pages 222–240.

Rizov, B. (2008). Hydra: A Modal Logic Tool for Wordnet Development, Validation and Exploration. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 1523–1528.

Rizov, B. (2014). Hydra: A Software System for Wordnet. In *Proceedings of the Global WordNet Conference*, pages 142–147.

Rodriguez, H., Climent, S., Vossen, P., Bloksma, L., Peters, W., Alonge, A., Bertagna, F., and Roventini, A. (1998). The top-down strategy for building eurowordnet: Vocabulary coverage, base concepts and top ontology. *Computers and the Humanities*, 32(2-3):117–152.

Simov, K., Osenova, P., Laskova, L., Radev, I., and Kancheva, Z. (2019). Aligning the Bulgarian BTB WordNet with the Bulgarian Wikipedia. In *Proceedings of the 10th Global WordNet Conference*, pages 290–297.

Sio, J. U.-S. and da Costa, L. M. (2019). Building the Cantonese Wordnet. In *Proceedings of the 10th Global WordNet Conference*, pages 206–215.

Tufiş, D., Barbu Mititelu, V., Ştefănescu, D., and Ion, R. (2013). The Romanian Wordnet in a Nutshell. *Language Resources and Evaluation*, 47(4):1305–1314.

Tufiş, D., Cristea, D., and Stamou, S. (2004). BalkaNet: Aims, Methods, Results and Perspectives. A General Overview. *Romanian Journal of Information Science and Technology*, 7(1-2):9–43.