Intrinsically Interlingual: The Wn Python Library for Wordnets

Michael Wayne Goodman and Francis Bond
Nanyang Technological University
goodmami@uw.edu, bond@ieee.org

Abstract

This paper introduces Wn, a new Python library for working with wordnets. Unlike previous libraries, Wn is built from the beginning to accommodate multiple wordnets—for multiple languages or multiple versions of the same wordnet—while retaining the ability to query and traverse them independently. It is also able to download and incorporate wordnets published online. These features are made possible through Wn's adoption of standard formats and methods for interoperability, namely the WN-LMF schema (Vossen et al., 2013; Bond et al., 2020) and the Collaborative Interlingual Index (Bond et al., 2016). Wn is open-source, easily available,[1] and well-documented.[2]

1 Introduction

Wordnet is a popular tool for natural language processing, and there are interfaces in many programming languages. For Python alone, there are 99 packages that mention wordnet in the Python Package Index.[3] Many of them provide an interface to a single language, like Romanian (Dumitrascu et al., 2019), or multiple languages from the same project, like Panjwani et al. (2018) for the Indian languages. Of course, there are many interfaces for English, of which the Natural Language Tool Kit's implementation is very widely used (Bird et al., 2009). The NLTK has a very well documented and clear interface to the Princeton WordNet (PWN; Fellbaum, 1998), with several distance metrics also implemented. The interface makes some design decisions that simplify wordnet structure, such as treating the word ↔ sense ↔ synset triad as a lemma ↔ synset dyad.

The Wn library introduced by this paper differs from the existing Python packages in several ways. It is not tied to any particular wordnet and does not immediately include any wordnet data, but instead it can read and use any wordnet published in the WN-LMF format (Vossen et al., 2013; Bond et al., 2020). As a convenience, dozens of such wordnets hosted online are indexed by Wn (see Appendix A) for easy downloading. Wn also differs from the NLTK in that it uses the triadic structure, but in many ways it is deliberately similar to the NLTK's interface in order to help smooth the transition for new users.

In 2015, the NLTK was extended to cover the wordnets in the Open Multilingual Wordnet (OMW 1.0; Bond and Foster, 2013).[4] OMW 1.0 was made to deal with wordnets produced by the **expand** method which take the structure of an existing wordnet, in this case PWN 3.0, and extend it by adding lemmas in a new language to existing synsets (Vossen, 1998). Most wordnets are built in this manner (Bond and Paik, 2012). The main advancement of OMW 1.0 was in the production of multiple wordnets, all in the same format, under open licenses.[5]

This structure where all synsets are shared among all languages has several advantages, such as the straightforward translation of words and the implicit sharing of structure which makes smaller wordnets more useful. It also has immediate disadvantages, the most prominent being that synsets not in PWN cannot be included. Some disadvantages are more subtle: as the OMW structure is the union of the structure of all the wordnets, new paths could become available when another wordnet is added,

[1] https://pypi.org/project/wn
[2] https://wn.readthedocs.io
[3] https://pypi.org/search/?q=wordnet

[4] NLTK was built for teaching, and the first version of the OMW wordnet extension was actually built by students as a programming assignment in a computational linguistics class!

[5] To make its data available to Wn, OMW 1.0 now additionally publishes each wordnet as a WN-LMF file at https://github.com/bond-lab/omw-data. The many wordnets derived from Wiktionary data (Bond and Foster, 2013) are not published but can similarly be converted.

which has ramifications for reproducibility in research. In practice, for OMW 1.0, all structure came from PWN 3.0 and non-English wordnets contributed no relations so this was never an issue, but we are anticipating future developments to OMW which may cause such problems.

To allow for wordnets with different structures and synsets not in PWN, a new version of the OMW (2.0) is under development. It uses the Collaborative Interlingual Index (CILI; Bond et al., 2016) to link synsets. This allows wordnets to define their own synset structure while maintaining interlingual linking through the shared, resource-agnostic index. The software for the Open Multilingual Wordnet 2.0[6] is released as open-source software with the aim of making it easily available for everyone. However, its primary goals were to allow the browsing of the unified resource and to facilitate the validation, addition, and management of new and historical wordnets and CILI entries—not to assist the individual researcher with downloading and using particular wordnets from its collection. As such, the software is not optimized for loading just one or two wordnets, and while it can run locally, it is expected to run as a web service.

In contrast, Wn does fewer checks and assumes that the wordnets are generally well-formed. This assumption should hold if the wordnets come from a source that performs these checks, such as the OMW. Wn allows a user to only load the wordnets they need and to access them distinctly. It is designed for wordnet users running things locally.

This paper describes a new Python interface for modeling wordnet data, including those from OMW 2.0, designed to replace the existing NLTK interface in a researcher's workflow. Wn is the first Python module designed from the beginning to use the Collaborative Interlingual Index to link separate wordnets. We discuss the desiderata for the software further in Section 2. We then briefly discuss the design of the system in Section 3. In Section 4 we give a brief tour of Wn's functionality. Finally we discuss a couple of aspects of why we think Wn is an improvement over previous implementations in Section 5 and conclude in Section 6.

2 Desiderata

The main goals of Wn are as follows:

Resource Independence: Each lexicon loaded into Wn is treated as a distinct resource and may be added and removed without affecting other lexicons.

WN-LMF Compliance: Wordnets in the modern WN-LMF format are fully supported and information is not lost upon loading or exporting wordnets.

Precise Modeling: All information in a wordnet is available to and discoverable by the user through intuitive structures. Notably, word senses have first-class status, just like words and synsets.

Interlingual Queries: Queries may traverse multiple wordnets, or not, depending on what the user specifies.

User Convenience: Data sources and query results are readily available; the user does not need to comprehend the complexity of the software to use it.

3 Design

Here we discuss several aspects of Wn's design, from the low-level database design in Section 3.1 to the user-facing Python data structures in Section 3.2 and the methodology for performing interlingual queries in Section 3.3. To support the distribution of wordnets as individual resources or in collections, formats for packaging wordnets are described in Section 3.4.

3.1 Database Design

From the outset, Wn was designed to handle both monolingual and interlingual queries over a multitude of wordnets. All loaded lexicons are stored in the same database,[7] but the elements are keyed to the lexicon that contributed them. Identifiers that are unique within a single wordnet, such as for synsets, are not necessarily unique when multiple wordnets are present, so their uniqueness is not enforced in the database. Instead, relationships between elements are linked via globally-unique table row identifiers and the original wordnet identifiers are only used for direct lookups within a lexicon. No identifiers are shared across lexicons except for CILI IDs, which are the only way to perform interlingual queries.

[6] https://github.com/globalwordnet/OMW/

[7] In the current implementation, the database engine is SQLite (https://www.sqlite.org/) but this detail should not concern most users as all operations in the public API are abstracted from the underlying infrastructure.

3.2 Class Modeling

The primary entities in WN-LMF wordnets are the lexical entries (i.e., words) and synsets. Word senses are essentially the link between words and synsets, but as they may be assigned metadata, take part in sense relations, and contain examples, they are given status as first-class entities in Wn. Each of these gets a Python class—`Word`, `Sense`, and `Synset`—that models its data and relationships. Figure 1 illustrates these entities and their relationships to each other. In addition, a `Wordnet` class represents a selection of lexicons used to filter queries.

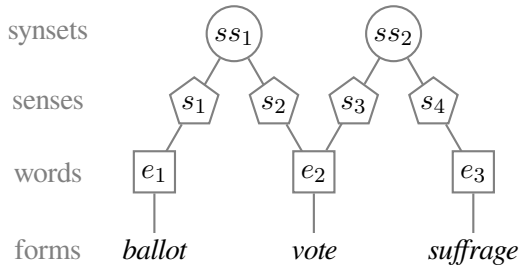

Figure 1: Modeling wordnet entities

All queries to the database are made through instances of these classes, which act as database abstraction layers. The primary queries, for words, senses, or synsets, are made through the `Wordnet` objects. Secondary queries, such as for word forms, synset or sense relations, definitions, examples, etc., are made through the `Word`, `Sense`, or `Synset` objects. These entity objects each contain a reference to the `Wordnet` object that was used to find them. This reference allows for the secondary queries to make use of the same lexicon filters. We give examples of querying the senses in Section 4.3 and Section 4.4.

3.3 Interlingual Queries

All interlingual queries must go through shared ILI links. Figure 2 illustrates how Wn translates a synset ss^f in lexicon f to other lexicons through shared ILI links. Every ILI has a synset in a queried language. If no synset is explicitly given in the lexicon, an implicit, empty synset is used instead.

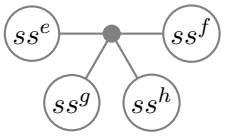

Figure 2: Translating synsets via ILI
The gray node is the ILI link.

Figure 3 illustrates how Wn can find synset relation paths from a synset ss_1^f, even when there are no relations defined in its lexicon, by sharing relations from a second lexicon e. From synset ss_1^f, Wn expands the search to ss_1^e via a shared ILI link. As ss_1^e has a relation to ss_2^e, Wn first traverses it and then attempts to find a corresponding synset in the original lexicon f. Since there is no synset in f for the ILI, it instead returns an inferred (and empty) synset. An inferred synset contains no information except its ILI link and the lexicon filters in force, but this is enough information to allow Wn to search for the next relation. Wn can then cross the ILI to ss_2^e, traverse the relation to ss_3^e, and cross the ILI again to ss_3^f, which is in the target lexicon.

This situation is common in OMW lexicons which only provide words and senses for a subset of PWN's synsets but offer no synset relations of their own. In this process, the synsets that may be the result of the relation traversal, such as ss_1^f and ss_3^f, are called the *target* set, while the synsets that may be used via ILI links for their relations are called the *expand* set.

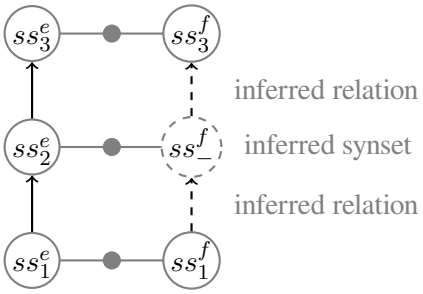

Figure 3: Traversing external relations via ILI

The inferred synsets are only necessary for relation traversal with expand lexicons. Standalone wordnets that do not require an expand lexicon have no use for inferred synsets.

3.4 Packaging Wordnets

As will be shown in Section 4.1, Wn is able to download and add wordnets from the web as well as from local files. Some wordnets, such as the English WordNet, are distributed just as the WN-LMF XML file, while others, such as the Open German WordNet (Siegel and Bond, 2021), include the full text of the license and the canonical citation as accompanying files, and the full OMW is distributed as a collection of multiple wordnets. In order to accommodate these different modes of distribution, we have designed three levels of packaging for wordnets. Each of these three may be distributed uncompressed or compressed with gzip or LZMA com-

pression.

Resource A file containing lexicon data is called a resource. While future versions of Wn may allow multiple formats, such as the JSON or RDF variants of WN-LMF (McCrae et al., 2021), currently only the XML format is supported.

Package A package is a directory containing a resource and optionally metadata files containing the license (`LICENSE`), basic documentation (`README`), or the canonical citation (`citation.bib`). Exactly one resource file is allowed in a package and at most one of each of the metadata files. Other files are allowed, but they will be ignored by Wn. A package directory, when distributed over the web, should be archived as a tarball.

Collection Multiple packages may be distributed in a directory called a collection. Wn will only search for packages in the collection's top directory and not under subdirectories, so a collection is a flat list of packages. In order for Wn to better distinguish between packages and collections, which are both directories, resource files may not appear in the collection without being in a package directory. A collection may optionally contain the same metadata files as packages, where any license, documentation, or citation pertains to the collection itself and not the individual packages. Any other files or directories in a collection will be ignored by Wn. As with packages, collections should be distributed as a tarball.

4 Usage

In this section we give a brief tour of Wn's programming interface.

4.1 Loading Wordnets

Wn was created for wordnets following the WN-LMF schema (Vossen et al., 2013; Bond et al., 2020) as this format requires synsets to declare their association with a CILI ID, if any. Older formats, such as WNDB,[8] are not directly supported, but conversion tools exist.[9]

The Wn project keeps an index of publicly available and open wordnets in the WN-LMF format, such as the English WordNet (McCrae et al., 2020) and OdeNet, the Open German WordNet (Siegel and Bond, 2021).[10] They can be listed with Wn's `wn.projects()` function as shown in Figure 4, which shows an abbreviated list. The full list of the current release is given in Table 1 (Appendix A).

Users can install wordnets from this list with Wn's `wn.download()` function, for instance by specifying the project's identifier and version:

```
>>> import wn
>>> wn.download('ewn:2020')
Download complete (13643357 bytes)
Added ewn:2020 (English WordNet)
```

As a convenience, if the user only specifies the project identifier (e.g., `'ewn'`), Wn will get the latest known version. For wordnets not indexed by Wn, users can provide an explicit URL as the first argument, and the lexicon ID and version will be extracted from the downloaded file. For instance, if OdeNet were not indexed by Wn, we could download it directly by the URL of its resource file:

```
>>> wn.download(url_of_odenet)
Download complete (2001396 bytes)
Added odenet:1.3 (Offenes Deutsches WordNet)
```

We encourage more wordnets to provide a persistent URL for this usage. Wordnets from OMW 1.0 (or in that format) can be automatically converted to the 2.0 (WN-LMF) format and loaded. Since there is a mapping from PWN synsets to CILI, synsets in wordnets that are built from PWN can be automatically linked to CILI as well.

Wordnets can be installed from a local WN-LMF file using the `wn.add()` function:

```
>>> wn.add('wnja.xml')
Added wnja:2.0 (Japanese Wordnet)
```

Wn is robust to small errors in the wordnet file (like different parts of speech in the synset and word or a confidence less than 0 or greater than 1) but will generally warn the user when they occur.

The `wn.lexicons()` function lists all installed lexicons. The objects returned by this function can be inspected to find the name, version, language, license, contact email, and other kinds of metadata of a lexicon:

```
>>> for l in wn.lexicons():
...     print(l.id, l.version, l.label)
...
ewn 2020 English WordNet
wnja 2.0 Japanese Wordnet
odenet 1.3 Offenes Deutsches WordNet
```

[8] https://wordnet.princeton.edu/documentation/wndb5wn

[9] https://github.com/jmccrae/gwn-scala-api

[10] https://github.com/hdaSprachtechnologie/odenet

```
>>> some_projects = wn.projects()[:6]
>>> [(p['id'], p['label'], p['version'], p['language']) for p in some_projects]
[('ewn', 'Open English WordNet', '2019', 'en'),
('ewn', 'Open English WordNet', '2020', 'en'),
('pwn', 'Princeton WordNet', '3.0', 'en'),
('pwn', 'Princeton WordNet', '3.1', 'en'),
('odenet', 'Open German WordNet', '1.3', 'de'),
('omw', 'Open Multilingual Wordnet v1.3', '1.3', 'mul')]
```

Figure 4: Listing indexed projects with Wn; see Appendix A for the full list

4.2 Selecting Lexicons

As mentioned, primary queries go through a `Wordnet` object, so one must be instantiated first. To motivate this step, consider a user who has installed both the English WordNet and the French wordnet WOLF (Sagot and Fišer, 2008). If they search for synsets for the word form *chat*, the `Wordnet` object determines if they receive synsets related to the English verb meaning *to talk*, those related to the French word for a cat, or both.

```
# Instantiation                Results
wn.Wordnet()                 # all
wn.Wordnet(lang='fr')        # only French
wn.Wordnet(lexicon='ewn')    # only EWN
```

Also, since it is possible to load multiple versions of the same wordnet, filtering on the lexicon ID only (`ewn`) only uses the most recently installed version (whether or not it's a newer release). A version specifier on the `lexicon` argument may be necessary to precisely differentiate:

```
wn.Wordnet(lexicon='ewn')       # recent
wn.Wordnet(lexicon='ewn:2020')  # 2020 only
wn.Wordnet(lexicon='ewn:*')     # all EWN
```

A user may wish to search a subset of the installed lexicons at once, such as when they have installed an extension lexicon containing additional words. In this case, the `lexicon` argument may take a space-separated list of lexicon specifiers. Finally, users may choose the lexicons to use for the *expand* set of interlingual queries, as described in Section 3.3, with the `expand` parameter:

```
wn.Wordnet(lexicon='wnja', expand='ewn')
```

If the `expand` parameter is not given, Wn allows any installed lexicon to be used in the expand set, in order to mimic the behavior of the OMW. A user may also specify an empty expand set (`expand=''`) to block ILI traversals when exploring relations.

Once a `Wordnet` object has been instantiated as described above, any queries performed on the object will restrict the search to the matching lexicons.

4.3 Primary Queries

All primary queries have several optional parameters which are used to narrow down the results. The first parameter is for a matching wordform and the second is for part-of-speech. Synsets also have a parameter for selecting by ILI ID. Below, assume `w` is a `Wordnet` object instantiated as above.

```
w.words()             # all words
w.words('犬')         # words w/ form '犬'
w.words(pos='n')      # all nominal words
w.senses()            # all senses
w.synsets()           # all synsets
w.synsets(ili='i1')   # synsets w/ ili 'i1'
```

Here is an example of getting synsets for the Japanese noun 犬 *inu* "dog":

```
>>> ja = wn.Wordnet(lang='ja')
>>> ja.synsets('犬', pos='n')
[Synset('wnja-02084071-n'),
 Synset('wnja-10641755-n')]
```

4.4 Secondary Queries

Secondary queries happen on the objects returned by primary queries. Below, assume `e` is a `Word` object, `s` is a `Sense` object, and `ss` is a `Synset` object. The list of secondary queries below is not exhaustive.

```
e.senses()        # senses for e
e.lemma()         # canonical lemma for e
e.forms()         # all word forms for e
s.word()          # the sense's word
s.synset()        # the sense's synset
s.derivations()   # derivation relation
ss.senses()       # synset's senses
ss.hypernyms()    # synset's hypernyms
ss.definition()   # synset definition
```

In the following example, we find the hypernyms of one synset for 犬 *inu* "dog":

```
>>> inu = ja.synsets('犬', pos='n')[0]
>>> inu.hypernyms()[0]
Synset('wnja-01317541-n')
```

4.5 Shortcut Functions

As a convenience to the user, Wn provides functions for primary queries that do not require them to first instantiate a `Wordnet` object:

```
wn.words()
wn.senses()
wn.synsets()
```

Each of these functions will create a `Wordnet` object when it is called and use it for the query. As such, these functions additionally take the `lang` and `lexicon` parameters which are passed on to the `Wordnet` object.

Additionally, there are shortcut secondary queries to go directly from words to synsets and vice-versa:

```
e.synsets()  # all synsets for e
ss.words()   # all words for ss
ss.lemmas()  # lemmas of all words for ss
```

The following lists the lemmas for the hypernym found in the previous example:

```
>>> hyp = inu.hypernyms()[0]
>>> hyp.lemmas()
['家畜']
```

4.6 Translating via ILI

Words, senses, and synsets can all be translated to some other lexicon via a synset's ILI link. The most natural object to translate is a sense, as it links a specific word to a specific concept, but all translation go through the ILI and thus through a synset. Translations of a synset will return at most one translated synset per target lexicon,[11] but the function returns a list because there may be multiple target lexicons. Translations of a sense return a list of senses in the target lexicon(s) shared by the sense's translated synset. Translations of a word return a mapping of senses to lists of sense translations, and this is because a word may have multiple unrelated concepts so it wouldn't make sense to group them in a flat list. The `translate()` methods below all take a `lang` or `lexicon` parameter to filter the target lexicons.

```
e.translate()   # translate a word
s.translate()   # translate a sense
ss.translate()  # translate a synset
```

Continuing the example from above, here are lemmas of translations for the found hypernym:

```
>>> hyp.translate(lang='en')[0].lemmas()
['domestic animal', 'domesticated animal']
```

5 Discussion

Here we discuss how Wn improves over previous offerings for users and researchers.

[11] Every ILI should have only one synset in a lexicon.

5.1 Query Language Persistence

One common point of confusion with the NLTK's interface is that the default language is English regardless of the operations used previously, and this is confounded by the fact that synsets for all languages in the OMW 1.0 (which the NLTK distributes) use the same PWN set. This problem is illustrated in Figure 5.

```
>>> from nltk.corpus import wordnet
>>> ss1 = wordnet.synsets("door")[0]
>>> ss1.lemma_names()
['door']
>>> ss2 = wordnet.synsets("pintu",
...                       lang="zsm")[0]
>>> ss2.lemma_names()
['door']
>>> ss2.lemma_names(lang="zsm")
['laluan', 'pintu']
```

Figure 5: The NLTK's interface defaults to English language queries.

In Wn, each lexicon has its own synset structure, and the results of primary queries keep a reference to the `Wordnet` object that was used, so the lexicon restrictions of the first query persist for follow-up queries.

5.2 Reproducibility

The OMW, both 1.0 and 2.0, is considered one large, multilingual wordnet, but it is not versioned as a single resource. Individual lexicons may be added or get updated without changing the OMW's version number. Also, changes to the structure of OMW through such updates can affect the results of queries on completely different lexicons, as synset relations are always implicitly shared. This means that a researcher performing an experiment using OMW data cannot guarantee reproducibility unless they can somehow recreate the exact database used. The OMW 2.0 database stores the information about which relations come from which wordnet, but the current OMW web interface does not allow you to filter on this.

Wn, in contrast, versions each individual lexicon and allows queries to specify which lexicons are used in the queries. This allows them to much more precisely state the requirements of their research product and thereby better describe a reproducible experiment.

6 Conclusions

This paper describes Wn: software for accessing wordnets in the global wordnet associations LMF format, linked by the collaborative interlingual index. Wn is built from the beginning to accommodate multiple wordnets while retaining the ability to query and traverse them independently. NLTK is already widely used amongst NLP researchers; we provide an enhanced functionality that goes beyond the current English based mapping.

Wn is open-source and available on GitHub.[12] We strongly encourage everybody to download, use, and, if possible, contribute back to the project. In future work, we intend to add the following capabilities:

(i) unloading wordnets from the database

(ii) exporting wordnets

(iii) modifying wordnet data locally

(iv) supporting information content (Resnik, 1995) and related similarity measures

(v) supporting new features in recent updates to the WN-LMF format (McCrae et al., 2021), such as wordnet dependencies and extensions

(vi) enabling morphological normalization for word lookup, similar to the use of Morphy in Princeton WordNet, but with hooks for external resources in other languages

Acknowledgments

Thanks to Liling Tan for the initial inspiration and early discussions and to three anonymous reviewers, Andrew Devadason, and Merrick Choo for comments on the paper. We would also like to thank the Google Season of Docs (2020), especially Yoyo Wu, for their contributions to the documentation.

References

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Francis Bond, Luis Morgado da Costa, Michael Wayne Goodman, John P. McCrae, and Ahti Lohk. 2020. Some issues with building a multilingual wordnet. In *Proceedings of the 12th Language Resource and Evaluation Conference (LREC 2020)*, pages 3189–3197.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual Wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Sofia, Bulgaria. Association for Computational Linguistics.

Francis Bond and Kyonghee Paik. 2012. A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue. 64–71.

Francis Bond, Piek Vossen, John P. McCrae, and Christiane Fellbaum. 2016. CILI: the Collaborative Interlingual Index. In *Proceedings of the Global WordNet Conference 2016*.

Stefan Dumitrascu, Avram Andrei, Morogran Luciana, and Stefan-Adrian Toma. 2019. Rowordnet –a python api for the romanian wordnet. In *10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*.

Christine Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

John P. McCrae, Michael Wayne Goodman, Francis Bond, Alexandre Rademaker, Ewa Rudnicka, and Luis Morgado da Costa. 2021. The global wordnet formats: Updates for 2020. In *11th International Global Wordnet Conference (GWC2021)*. (this volume).

John Philip McCrae, Alexandre Rademaker, Ewa Rudnicka, and Francis Bond. 2020. English WordNet 2020: Improving and Extending a WordNet for English using an Open-Source Methodology. In *Proceedings of the Multimodal Wordnets Workshop at LREC 2020*, pages 14–19.

Ritesh Panjwani, Diptesh Kanojia, and Pushpak Bhattacharyya. 2018. pyiwn: A python-based api to access indian language wordnets. In *Proceedings of the Global WordNet Conference*, volume 2018.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada*, pages 448–453.

Benoît Sagot and Darja Fišer. 2008. Building a free French wordnet from multilingual resources. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Melanie Siegel and Francis Bond. 2021. Compiling a German wordnet from other resources. In *11th International Global Wordnet Conference (GWC2021)*. (this volume).

Piek Vossen. 1998. Introduction to EuroWordNet. *Computers and the Humanities*, 32(2):73–89.

Piek Vossen, Claudia Soria, and Monica Monachini. 2013. Wordnet-LMF: A standard representation for multilingual wordnets. *LMF Lexical Markup Framework*, pages 51–66.

[12] https://github.com/goodmami/wn

A Indexed Wordnets

Name	ID	Versions	Language
English WordNet	ewn	2020	English [en]
		2019	
Princeton WordNet	pwn	3.1	English [en]
		3.0	
Open German WordNet	odenet	1.3	German [de]
Open Multilingual Wordnet	omw	1.3	multiple [mul]
Albanet	alswn	1.3+omw	Albanian [als]
Arabic WordNet (AWN v2)	arbwn	1.3+omw	Arabic [arb]
BulTreeBank Wordnet (BTB-WN)	bulwn	1.3+omw	Bulgarian [bg]
Chinese Open Wordnet	cmnwn	1.3+omw	Mandarin (Simplified) [zh]
Croatian Wordnet	hrvwn	1.3+omw	Croatian [hr]
DanNet	danwn	1.3+omw	Danish [da]
FinnWordNet	finwn	1.3+omw	Finnish [fi]
Greek Wordnet	ellwn	1.3+omw	Greek [el]
Hebrew Wordnet	hebwn	1.3+omw	Hebrew [he]
IceWordNet	islwn	1.3+omw	Icelandic [is]
Italian Wordnet	iwn	1.3+omw	Italian [it]
Japanese Wordnet	jpnwn	1.3+omw	Japanese [jp]
Lithuanian WordNet	litwn	1.3+omw	Lithuanian [lt]
Multilingual Central Repository	catwn	1.3+omw	Catalan [ca]
Multilingual Central Repository	euswn	1.3+omw	Basque [eu]
Multilingual Central Repository	glgwn	1.3+omw	Galician [gl]
Multilingual Central Repository	spawn	1.3+omw	Spanish [es]
MultiWordNet	itawn	1.3+omw	Italian [it]
Norwegian Wordnet	nobwn	1.3+omw	Norwegian (Bokmål) [nb]
Norwegian Wordnet	nnown	1.3+omw	Norwegian (Nynorsk) [nn]
Open Dutch WordNet	nldwn	1.3+omw	Dutch [nl]
OpenWN-PT	porwn	1.3+omw	Portuguese [pt]
plWordNet	polwn	1.3+omw	Polish [pl]
Romanian Wordnet	ronwn	1.3+omw	Romanian [ro]
Slovak WordNet	slkwn	1.3+omw	Slovak [sk]
sloWNet	slvwn	1.3+omw	Slovenian [sl]
Swedish (SALDO)	swewn	1.3+omw	Swedish [sv]
Thai Wordnet	thawn	1.3+omw	Thai [th]
WOLF (Wordnet Libre du Français)	frawn	1.3+omw	French [fr]
Wordnet Bahasa	indwn	1.3+omw	Indonesian [id]
Wordnet Bahasa	zsmwn	1.3+omw	Malaysian [zsm]

Table 1: A listing of wordnets indexed by Wn; all with `1.3+omw` as a version are included in the Open Multilingual Wordnet and are also available individually.