Linguistic LOD for Interoperable Morphological Description

Michael Rosner, Maxim Ionov
University of Malta, University of Cologne
mike.rosner@um.edu.mt, mionov@uni-koeln.de

Abstract
Interoperability is frequently cited as one important rationale underlying the use of LLOD representations and is generally regarded as highly desirable. However, the concept is generally taken for granted, and rarely analysed or exemplified. In this paper we attempt to remedy these shortcomings by concentrating on morphology, distinguishing three different kinds of interoperability which are relevant to that field. We providing practical implementations making extensive use of the vocabulary offered by Ontololex Morph.

Keywords: Morphological Resources, Interoperability, Linked Linguistic Open Data

1. Introduction

In general, interoperability is a characteristic of a product or system that seamlessly works with another product or system. For example, when you plug in your toaster to a wall socket, the two systems are (i) the toaster plug, and (ii) the power network that provides the toaster with electricity. The plug is interoperable in the sense that it works with any compatible socket. The advantage of this interoperability is that you can use the toaster anywhere. Conversely, we experience the problem of non-interoperable plugs when we go abroad. Typically, the solution is to use an appropriate *adaptor*, but for the well-travelled this means the additional burden of carrying an array of adaptors in order to guarantee world-wide functionality of your device. The moral here is clear: interoperability of the device implies a maximum degree of independence from the context of use.

Turning to the use of Linked (Open) Data representations, interoperability is frequently cited as the main underlying rationale. It is claimed that due to the use of open standards, such representations facilitate the use of resources in a variety of different contexts with zero or minimal adaptation, either in the resource itself or in the context that uses it. This is in contrast to hand-crafted representations which may require arbitrary adaptation to the resource to be used successfully.

Although these general considerations apply widely and are often mentioned, they are rarely analysed, exemplified, or specifically applied to the narrower scope of Linguistic LOD (LLOD). In this paper we propose to remedy this lacuna.

Some preliminary considerations reveal that when it comes to LRs, there exist other kinds of interoperability. Below we consider three ways of dividing up the landscape: *task* interoperability, *language* interoperability, and *domain interoperability*. In the following sections we take a closer look at each of these types, assess how well they can be addressed using LLOD and where it falls short.

1.1. Task Interoperability

The basic idea underlying task interoperability is that the same machinery is applied without modification to different tasks. There are essentially two kinds of LRs: data-oriented and process-oriented. Data-oriented resources express static facts e.g. an annotated text corpus, or a lexicon containing facts about the words of a language, whilst process-oriented ones (often referred to as tools), such as parsers, translators and chatbots, the focus is on the achievement of tasks or on explicit behaviours which are of interest to us. Importantly, the two are connected: process-oriented resources make use of data-oriented ones. Thus, a chatbot might make use of a lexicon to identify the current topic and determine its important characteristics.

In this paper we focus on the interoperability of data-oriented LRs. As our working example, we choose the lexicon because a lexicon is not only a clear example of a data resource but also one which is crucial for practically every NLP task. Examples are parsing; sentiment analysis; translation. Each of these tasks use a lexicon to associate information with words - but for each task the information is different. Thus for parsing, it concerns part-of-speech information; for sentiment analysis, sentiment values; and for translation, a translational equivalent in another language.

Given this diversity of information types associated with words, there is a tendency to create diverse representations, that for the sake of efficiency, require specialised access and processing procedures. This is a perfect scenario for developing a series of specialised lexicons, one for each task. SentiWordnet (Baccianella et al., 2010) for example, contains opinion information on terms extracted from WordNet, providing a database of term/sentiment information for English. Bilingual lexicons are crucial to the operation of translation systems such as Apertium (Forcada et al., 2011), and DBnary (Sérasset, 2015), a multilingual lexicon based on Wiktionary. The variety of formalisms to represent such data can lead to a lack of compatibility. We explore ways to address this problem using LLOD.

1.2. Linguistic Interoperability

Linguistic interoperability refers to a language processing system that is language agnostic in the sense that it operates correctly with any natural language.

A clear example is the Unicode[1] text encoding standard designed to support the representation of text written in all of the world's major writing systems. Prior to its introduction, different, sometimes proprietary encodings for individual language and language groups were used. This kind of non-interoperability yields, for example, text documents in language L1 which work fine on text processing systems W1 and W2, whilst for language L2 they only work for W2. In contrast, any system that is Unicode compliant can operate with text in any language.

Another example is furnished by Universal Dependencies (Nivre et al., 2017) (UD), a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD trees are an interoperable representation for which language-independent tools can be developed that operate on the syntactic structures of different languages.

1.3. Domain Interoperability

By extension to linguistic interoperability, domain interoperability means that a system or platform is "domain agnostic": it operates correctly and without adaptation in different domains. For this to be possible, the system must have some advance knowledge of the abstract structure of the domain, so that it can unpick the parts that need to be processed. The notion of domain is extremely general, but within language processing communities it refers to a subject area, theme or topic, typically associated with a characteristic vocabulary of words. Examples of such domains are finance, biomedicine, justice. More formally, we can regard this as something close to an *ontology*, i.e. a set of related concepts together with an associated set of terms that are used for naming them. Indeed, for all the above examples, and many others, we find such ontologies: e.g. FIBO (Bennett, 2013) (finance); Hu (2006) (biomedicine); Engers et al. (2008) (justice).

[1]https://home.unicode.org/

1.4. Related Work

There are many approaches and initiatives that aim to increase the interoperability of LRs. Probably one of the most famous ones is Universal Dependencies, already mentioned in Section 1.2. Its success led to many other related projects. One of them is Unimorph (Batsuren et al., 2022), an initiative aimed at creating a unified framework for morphological data across languages. It seeks to provide a standardised format for encoding morphological information, such as inflections, derivations, and other morphological processes, across different languages. Unlike LLOD, the project relies on lists of wordforms combined with the list of grammatical categories[2]. Such a simple representation is very appealing and, at first sight, provides interoperability, since the format is very easy to read and write. On the other hand, any operation requires the creation of a custom solution, or the use of ad-hoc tools created specifically for this. At the same time, simplifying the annotation standard to a flat list does not work for all languages, which is evident from the fact that the format becomes more complex over time when inconsistencies arise. And the more complex the format becomes in order to increase language interoperability, the less straightforward it becomes to parse the dumps, and the lower is the overall interoperability.

On the other spectrum of interoperability and human readability there is another related technology: `xfst` (Beesley and Karttunen, 2003), `foma` (Hulden, 2009) and other Finite-State Transducer (FST) frameworks, powerful computational tools used for modelling and analysing natural language phenomena. These frameworks provide scripting languages that allow users to create transducers that can encode a wide range of linguistic phenomena, including morphological analysis, phonological rules, and syntax. FSTs provide functionalities for composing, intersecting and manipulating these transducers, allowing researchers to model complex linguistic processes in a formal and computationally tractable manner.

Transducers provide task interoperability (they are bidirectional) and, given that there are rules for all the languages of interest, language interoperability. They can generally can be adapted to any domain. However, they operate on strings, so there is almost no way to enrich the dataset with additional information while staying within the formalism. For example, when dealing with homonyms of the same syntactic category (e.g. *bank*), there are is no principled way to encode the particular sense of the word as would be the case when using LLOD or other semantic representations.

[2]The latest version uses a more complex format than a flat list in order to be able to deal with complex cases where a flat list leads to ambiguous parses.

In addition, different FST frameworks have slightly different formats and languages, which makes them non-interoperable with each other. To combat this, Chiarcos et al. (2022) shows that OntoLex-Morph can be used to encode FSTs and that it can function as an interchange format to convert between them.

1.5. Structure of the paper

We have introduced and described three kinds of interoperability. The remaining sections illustrate how all three can be achieved for LRs using an approach based on LLOD. Our examples centre around morphology resources and associated tools for morphological processing and for this reason we rely heavily on the extension of the OntoLex vocabulary for representations of morphology, OntoLex-Morph. Section 2 provides some background on morphology. Section 3 gives an overview of the OntoLex vocabulary and its extension for morphological descriptions. Sections 4–7 explore types of interoperability created by this vocabulary using two examples, and the final sections give an overview of related work and some conclusions.

2. Morphology: facts versus processes

Morphology is the study of forms. Within linguistics, it denotes the study of linguistic forms, i.e. words, word parts, and their relationship to other words. A morphological description of a particular language assigns a structure to all the valid words and provides the rationale for grouping words e.g. into paradigms like verb conjugations or lexical entries that share the same sense. Alongside this purely structural information is the association of word forms with grammatically relevant information concerning e.g. part-of-speech or agreement features like number, gender and person.

From the perspective of NLP (in contrast to that of theoretical linguistics), the morphological description of a language is a data-oriented resource, as identified above. We should be careful to notice that although such a description *assigns* a morphological structure to a valid wordform, it does not tell us how to actually *discover* that structure. The same principle holds in the reverse direction (i.e. for generation of wordforms) which is to say that the morphological description contains enough information to assign one or more wordforms to a valid (but possibly underspecified) morphological structure, but it does not tell you how to go about computing it.

Here then, are two concrete examples of task interoperability: the language description is the static, data-oriented resource, whilst morphological analysis and generation are each oriented toward distinct processes. The basic idea is that the morphological knowledge should be able to interoperate between the two computational tasks, i.e.

$$\begin{tikzcd}
\text{Generation} \\
\text{Morph Knowledge} \arrow[u] \arrow[d] \\
\text{Analysis}
\end{tikzcd}$$

Here the up and down arrows denote distinct computational processes that respectively produce (i) a morphological generator and (ii) a morphological analyser for a given language. These are specified by two programs (one for analysis; another for generation) that each need to make certain assumptions about the format of the underlying morphological knowledge. Our claim is that OntoLex-Morph, a vocabulary designed for representing morphological information as LLOD, is a good choice of representation because we can specify both processes using an appropriately configured SPARQL query that embodies the structural assumptions that are made explicit in OntoLex-Morph.

3. OntoLex-Morph

OntoLex-lemon (McCrae et al., 2017) is the *de facto* standard for publishing lexical resources in RDF, compliant with established web standards. The model revolves around the concept of a `LexicalEntry` — a lexeme or a dictionary entry. It must have at least one (word)form (`canonicalForm`) and can have a number of other forms, as well as lexical senses, which can be linked to either lexical concepts or entities in an ontology (Fig. 1). Basic morphological information such as part of speech and grammatical categories can be provided for lexical entries and forms using elements of any suitable vocabulary, such as LexInfo.[3]

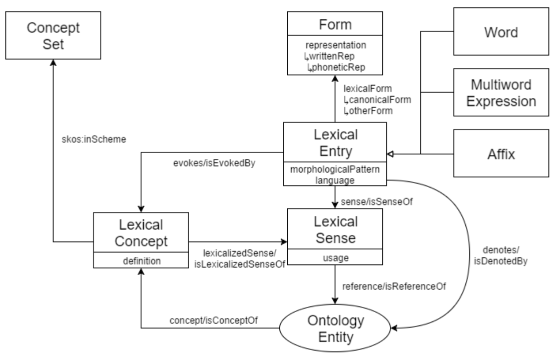

Figure 1: OntoLex-Lemon core model

Although there is a place for including basic morphological information in the core model, the standard does not give a clear way to represent paradigmatic relationships between lexical entries and forms (inflectional morphology) or derivational relationships

[3] `https://lexinfo.net/`.

between lexical entries. To close this gap and establish a standard way to represent this, an extension to the core module, OntoLex-Morph, is being developed.[4]

The model (Fig. 2) consists of three parts: derivation (left), inflection (right), and information on how to generate new forms, both for inflection and derivation (top). The central part of the module is the class `Morph`, which corresponds to a specific realisation of a morpheme. It is a subclass of `LexicalEntry`, which might be a bit counterintuitive at first, but this allows for modelling resources which have morphs as entries of their own.

The representation of rules for generating new forms (inflection) in the model works as follows: (i) A lexical entry can be a part of an inflectional paradigm. (ii) For each paradigm, there can be a number of rules, each of them having information on how to produce a form and grammatical meaning that should be assigned to this form; (iii) The formalism to encode a rule is not strictly set, but the one described in the guidelines is a (POSIX-compatible) regular expression.

For generating new lexical entries (derivation), we need to specify (i) word formation relations specifying what pairs of lexical entries should form a particular relation and potentially having additional information related to it, and (ii) derivation rules that specify how the parts of the words should be attached to each other.

4. Morphological Knowledge: An Illustrative Example

To illustrate how OntoLex-Morph facilitates different types of interoperability, we will use a toy dataset[5] that models a part of a regular verb paradigm of an Italian verb *parlare*. The morphological knowledge we need to represent is summarised in the table below:

person/number	present
1SG	*parl**o***
2SG	*parl**i***
3SG	*parl**a***
1PL	*parl**iamo***
2PL	*parl**ate***
3PL	*parl**ono***

Table 1: A fragment of conjugation of *parlare*

We model the lexical entry and its canonical form in the following way:

```
:parlare a ontolex:LexicalEntry ;
        lexinfo:partOfSpeech :lexinfo:verb ;
        morph:morphologicalPattern :v-are_paradigm;
        ontolex:canonicalForm :parlare_form ;
        morph:baseForm :parlare_form .

:parlare_form a ontolex:Form ;
            ontolex:writtenRep "parlare"@ita .
```

Properties `baseForm` and `morphologicalPattern` specify a base form that is used to create inflected forms and the type of conjugation (i.e. a set of rules that can be applied to the form), respectively.

For each cell in the paradigm, we need to provide an affix and a rule that describes how to create a corresponding form:

```
:suff_o_1sg a ontolex:Affix ;
            rdfs:label "-o"@ita ;
            morph:grammaticalMeaning [
              lexinfo:number lexinfo:singular ;
              lexinfo:person lexinfo:firstPerson ;
            ] .

:v-are_ind_1sg a morph:InflectionRule ;
               morph:paradigm :v-are_paradigm ;
               morph:involves :suff_o_1sg ;
               morph:replacement [
                 a morph:Replacement ;
                 morph:source "are$" ;
                 morph:target "o" ;
               ] .
```

5. Task Interoperability

Using the example described above,[6] we can now examine the state of task interoperability in the OntoLex-Morph vocabulary.

5.1. Generation

As described in Section 3, part of OntoLex-Morph was designed to allow the representation of generation rules for both inflection and derivation. In this way, it is possible to store lexical entries with their dictionary forms in the dataset, along with instructions on how to generate the rest rather than having a complete set of pre-generated forms. The example above provides the data necessary to generate the forms for the case of inflection.

The generation process can be built on top of native RDF technologies (i.e. SPARQL). This provides a general level of interoperability, as these technologies follow open standards, so the implementation does not depend on proprietary tools or particular products that might be discontinued in the future. In real-life applications, of course, the implementation should contain at least a wrapper around RDF technologies, but for the purposes of this paper, the raw output of SPARQL queries is enough.

To show the generation capabilities, we created a SPARQL SELECT query that, when applied to the data described in the previous section, outputs

[4] https://www.w3.org/community/ontolex/wiki/Morphology.

[5] Throughout the paper we have, for the sake of clarity and brevity, adopted the simplest possible examples.

[6] The data is available at https://github.com/max-ionov/ldl-2024-morph-interoperability/blob/main/italian.ttl.

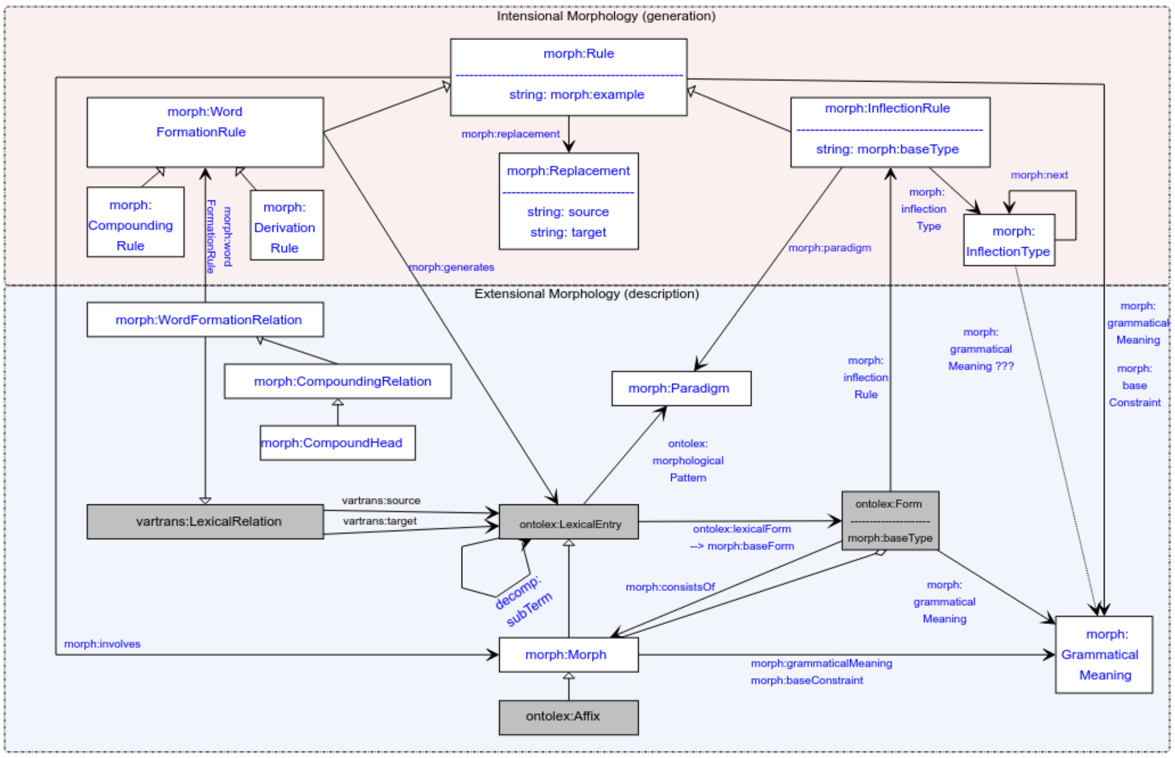

Figure 2: OntoLex-Morph draft model

generated inflected forms with their assigned grammatical categories (see Table 2).[7]

entry	form	gram. cat.	value
parlare	parlo	person number	firstPerson singular
parlare	parli	person number	secondPerson singular
parlare	parla	person number	thirdPerson singular
...	...	...	...

Table 2: Generation of inflected forms of *parlare*

Alternatively, it is possible to modify it to a SPARQL CONSTRUCT query which outputs RDF with the generated forms that can be used saved in a local graph and used in subsequent queries.

5.2. Analysis

An additional level of interoperability, task interoperability, is provided both by the way the Morph module was designed and the nature of RDF technologies: with a slight modification of the SPARQL query used for generation, we can revert the process and get possible morphological analyses for a word, without having these forms pre-generated in advance:

entry	form	gram. cat.	value
parlare	parlo	person number	firstPerson singular

Table 3: Analyses for a wordform *parlo*

Note that this procedure differs from a simple table lookup: the final list of forms is never created, but each is dynamically computed and tested against search criteria. In most cases, it might be impractical (especially since SPARQL endpoints are generally slow and unreliable), but this might be useful, for example, when the rules are not stable or come from multiple external sources.

This also differs from generational approaches, whether statistical or rule-based (e.g., finite-state transducers). While most generational approaches operate with strings, this procedure finds URIs of lexical entries,[8] which, in turn, may contain more information, both paradigmatic and syntagmatic.

[7]All queries and the full version of the outputs are available at `https://github.com/max-ionov/ldl-2024-morph-interoperability/blob/main/sparql/`.

[8]URIs are omitted in the tables above for formatting reasons.

6. Linguistic Interoperability

Having established task interoperability, we turn to argue similarly for *linguistic* interoperability. In our previous research, we have shown that OntoLex-Morph can encode inflectional rules for languages with different types of word formation, i.e., non-concatenative morphology of the Maltese language (Ionov and Rosner, 2023). Further examples exist showing the applicability of the model to agglutinative and polysynthetic languages.[9]

This alone is an example of linguistic interoperability. Furthermore, we can apply the queries that we applied to the Italian dataset in the previous section to the aforementioned Maltese dataset[10] almost without any adaptation:

entry	form	gram. cat.	value
kiteb	ktibt	person	firstPerson
		number	singular
		aspect	perfective
kiteb	ktibt	person	secondPerson
		number	singular
		aspect	perfective

Table 4: Analyses of a Maltese wordform *ktibt*

In this way, we can use the same machinery for both generation and analysis for both languages, and generally this should be extensible for any other language.

However, there are some caveats. Most importantly, the queries we present only account for cases where a form is created by adding only one affix: we do not account for agglutination, where multiple rules can be applied to a single entry to create a wordform (cf. Finnish noun inflection, with separate suffixes for number and grammatical case).

Another problem with the queries we used with regard to linguistic interoperabilty is that there are character classes for vowels and consonants hard-coded into them, and they only account for the Maltese alphabets since the classes are used only in the Maltese set of rules.

Finally, there are some minor inconsistencies in the way different SPARQL engines implement the standards, which leads to slightly different outputs. But all these are limitations of the specific (quite rudimentary) implementation and can be solved by a more complex way of applying the rules, with pre- and post-processing.

7. Domain Interoperability

To show an example of domain interoperability, we need another example. We will focus on chemical nomenclature, developed to facilitate communication by providing a methodology for assigning descriptors to chemical substances so that they can be identified without ambiguity. This domain exhibits a three-level structure: (i) actual *chemical compounds* with a definite molecular structure which is the underlying semantic interpretation, (ii) a *formula* which notates that structure and (iii) terms which are composite strings with their own systematic morphological structure. A few simple examples from *Nomenclature of Inorganic Chemistry* (Connelly et al., 2005), the so-called "Red Book" illustrate this.

Chemical Term	Formula
trioxygen	O_3
sodium chloride	$NaCl$
iron dichloride	$FeCL_2$
trisodium pentabismuthide	Na_3Bi_5
magnesium chloride hydroxide	$MgCl(OH)$

The wording describing the principles of nomenclature is highly reminiscent of that used in linguistic morphology:

> *Generally, nomenclature systems require a root [...] Names are constructed by joining other units to these roots. Among the most important units are affixes. These are syllables added to words or roots and can be suffixes, prefixes or infixes according to whether they are placed after, before or within a word or root.* (Connelly et al., 2005, p. 5)

For example, the name iron dichloride for the substance $FeCl_2$ involves the juxtaposition of element names (iron, chlorine), their ordering in a specific way (electropositive before electronegative), the modification of an element name to indicate charge (the 'ide' ending designates an elementary anion and, more generally, an element being treated formally as an anion), and the use of the multiplicative prefix 'di'.

In practice, we might utilise this knowledge in one of the two (non-exclusive) ways: we might extend a general-purpose lexicon that does not have some of these terms, or we can add information about word relations between the terms. In both cases, it is possible to utilise OntoLex-Morph. As an example, we are going to look at derivates of the word *chlorine*: *chloride* and *dichloride* and multiword expressions (MWE) that contain them: *sodium chloride* and *iron dichloride*.

We start with definitions of the lexical entry *chlorine* and its canonical form:

```
:chlorine a ontolex:LexicalEntry ;
          ontolex:canonicalForm :chlorine_form .

:chlorine_form a ontolex:Form ;
               ontolex:writtenRep "chlorine"@en-GB .
```

[9] https://github.com/ontolex/morph/tree/master/data.

[10] https://raw.githubusercontent.com/max-ionov/maltese-morph/main/lexical-entries-small.ttl.

To add information on how to generate new derivate entries, we add the following instances of WordFormationRelation and DerivationRule:

```
:rel_chlorine_ide a morph:WordFormationRelation ;
              vartrans:source :chlorine ;
              vartrans:target :chloride ;
              morph:WordFormationRule :ine_ide_rule .

:ine_ide_rule a morph:DerivationRule ;
              morph:replacement [
                morph:source "ine$" ;
                morph:target "ide"
              ] .

:di_rule a morph:DerivationRule ;
         morph:replacement [
           morph:source "^" ;
           morph:target "di"
         ] .

:rel_di_chloride a morph:WordFormationRelation ;
              vartrans:source :chloride ;
              vartrans:target :dichloride ;
              morph:WordFormationRule :di_rule .
```

This additional information can be created and stored independently from the main lexicon and can be used to extend the original data with the new domain-specific words and constructions. Using SPARQL federated queries, it can be queried together with the main lexicon, providing the desired results without changing the original dataset. The data can be further expanded by adding string representations in chemical notation:

```
:chlorine_form a ontolex:Form ;
               ontolex:writtenRep "chlorine"@en-GB,
                                  "Cl"@en-x-chem .

:di_form a ontolex:Form ;
         ontolex:writtenRep "di-"@en-GB,
                            "_2"@en-x-chem .
```

With this modelling, it is possible to use SPARQL to generate (a small subset of) chemical formulas from its components, as long as we use written representations with the corresponding language tag. In addition, it is also possible to translate chemical formulas from their notation to natural language and back, regardless of the natural language in question, which is a combination of all types of interoperability discussed in this and the previous sections.

8. Discussion and future work

8.1. Discussion

A key issue is the practical feasibility of using OntoLex-Morph in the way presented in this paper. The performance of SPARQL and the relatively low adoption of LLOD technologies make it a bad contender for an interchange format used on the level of UD or UniMorph. On the other hand, more and more people are becoming familiar with the field, and small and medium-sized datasets work relatively well, even under pressure.

For large datasets, or for services where availability is paramount, pre-generated tables are potentially a better alternative. There are also hybrid solutions such as aggressive caching.

Another problem that we have not touched on in this paper is the heterogeneity of OntoLex datasets. According to Bosque-Gil et al. (2018), this can be because “authors have developed their own ad-hoc extensions due to the actual lack of existing models that account for the specific features of the resource they aim to convert, due to the lack of awareness of a partially similar resource, or even due to the difficulty of finding the appropriate documentation”. Inconsistencies between the datasets require querying the datasets separately, creating more complex queries that account for all the possible configurations, or inserting additional unifying statements into the datasets.

Of course, sometimes this might be due to underspecification by design. One example is representation of grammatical categories in OntoLex: Although the *LexInfo* vocabulary is recommended, it is not required to use it, since there might be categories that are not represented there.

8.2. Future work

At the outset of this paper we suggested that interoperability is widely cited but rarely exemplified quality of LOD. In the preceding sections we have tried to demonstrate that the three types of interoperability described actually make sense and can be illustrated concretely at least for LLOD in the domain of morphology. The main lesson is that it is difficult to demonstrate interoperability without narrowing the focus of application precisely because the inherent chacteristics of LOD – linkedness and openness – are very general, giving rise to a very general and therefore weak notion of interoperability. We feel that we have progressed by narrowing the application to linked language data in the particular area of description systems for the morphological structure of terms.

We foresee two main directions of interest for future research. The first is to deepen the coverage by advancing from a few choice examples to a deeper and wider coverage of tasks, languages and domains. This would bring some much needed detail concerning the adequacy of the underlying descriptive framework. There is clearly a lot of work involved in any of these.

The second direction concerns the inherently useful idea of measuring the degree of interoperability displayed by a given set of resources and associated tools. Such measurement would enable us to investigate whether one system of description is more interoperable than another or whether there is a measureable tradeoff between interoperability and efficiency. At present, these questions are too complex to answer in any exact sense.

One possible direction would be to look at the

amount or the complexity of queries required to extract all the relevant concepts from two or more datasets (e.g. lexical entries, forms, written representations, etc.), or the intersection between these queries for each: the more there is in common, the more interoperability there is.
However, we need to be careful to distinguish between interoperability and lack of flexibility. Having different datasets fully interoperable is not very useful if this comes at a cost of them not representing the data properly.

9. Bibliographical References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Khuyagbaatar Batsuren, Omer Goldman, Salam Khalifa, Nizar Habash, Witold Kieraś, Gábor Bella, Brian Leonard, Garrett Nicolai, Kyle Gorman, Yustinus Ghanggo Ate, Maria Ryskina, Sabrina Mielke, Elena Budianskaya, Charbel El-Khaissi, Tiago Pimentel, Michael Gasser, William Abbott Lane, Mohit Raj, Matt Coler, Jaime Rafael Montoya Samame, Delio Siticonatzi Camaiteri, Esaú Zumaeta Rojas, Didier López Francis, Arturo Oncevay, Juan López Bautista, Gema Celeste Silva Villegas, Lucas Torroba Hennigen, Adam Ek, David Guriel, Peter Dirix, Jean-Philippe Bernardy, Andrey Scherbakov, Aziyana Bayyr-ool, Antonios Anastasopoulos, Roberto Zariquiey, Karina Sheifer, Sofya Ganieva, Hilaria Cruz, Ritván Karahóǧa, Stella Markantonatou, George Pavlidis, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Candy Angulo, Jatayu Baxi, Andrew Krizhanovsky, Natalia Krizhanovskaya, Elizabeth Salesky, Clara Vania, Sardana Ivanova, Jennifer White, Rowan Hall Maudslay, Josef Valvoda, Ran Zmigrod, Paula Czarnowska, Irene Nikkarinen, Aelita Salchak, Brijesh Bhatt, Christopher Straughn, Zoey Liu, Jonathan North Washington, Yuval Pinter, Duygu Ataman, Marcin Wolinski, Totok Suhardijanto, Anna Yablonskaya, Niklas Stoehr, Hossep Dolatian, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Aryaman Arora, Richard J. Hatcher, Ritesh Kumar, Jeremiah Young, Daria Rodionova, Anastasia Yemelina, Taras Andrushko, Igor Marchenko, Polina Mashkovtseva, Alexandra Serova, Emily Prud'hommeaux, Maria Nepomniashchaya, Fausto Giunchiglia, Eleanor Chodroff, Mans Hulden, Miikka Silfverberg, Arya D. McCarthy, David Yarowsky, Ryan Cotterell, Reut Tsarfaty, and Ekaterina Vylomova. 2022. UniMorph 4.0: Universal Morphology. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 840–855, Marseille, France. European Language Resources Association.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications.

Mike Bennett. 2013. The financial industry business ontology: Best practice for big data. *Journal of Banking Regulation*, 14.

Julia Bosque-Gil, Asuncion Gómez-Pérez, Elena Montiel-Ponsoda, and Jorge Gracia. 2018. Models to represent linguistic linked data. *Natural Language Engineering*, 24(6):811–859.

Christian Chiarcos, Christian Fäth, and Maxim Ionov. 2022. Unifying morphology resources with OntoLex-morph. a case study in German. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4842–4850, Marseille, France. European Language Resources Association.

Neil G. Connelly, Ture Damhus, Richard M. Hartshorn, and Alan T. Hutton. 2005. Nomenclature of inorganic chemistry – iupac recommendations 2005. *Chemistry International – Newsmagazine for IUPAC*, 27(6).

Tom Engers, Alexander Boer, Joost Breuker, Andre Valente, and Radboud Winkels. 2008. *Ontologies in the Legal Domain*, pages 233–261. Springer.

Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Xiaohua Hu. 2006. Natural language processing and ontology-enhanced biomedical literature mining for systems biology. *Computational Systems Biology*, pages 39–56.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, pages 29–32, Athens, Greece. Association for Computational Linguistics.

Maxim Ionov and Mike Rosner. 2023. Beyond concatenative morphology: Applying OntoLex-morph to Maltese. In *Proceedings of the 4th Conference on Language, Data and Knowledge*, pages 385–391, Vienna, Austria. NOVA CLUNL, Portugal.

John P McCrae, Julia Bosque-Gil, Jorge Gracia, Paul Buitelaar, and Philipp Cimiano. 2017. The OntoLex-Lemon Model: Development and Applications. In *Proceedings of eLex-2017*, pages 19–21.

Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. Universal Dependencies. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain. Association for Computational Linguistics.

Gilles Sérasset. 2015. DBnary: Wiktionary as a Lemon-based multilingual lexical resource in RDF. *Semantic Web*, 6(4):355–361.