Diversicon: Pluggable Lexical Domain Knowledge

Gábor Bella Fiona McNeill David Leoni Francisco José Quesada Real
Fausto Giunchiglia

{gabor.bella,david.leoni,fausto.giunchiglia}@unitn.it, f.mcneill@hw.ac.uk@unitn.it,
francisjqr@gmail.com

Abstract

Natural language understanding is a key task in a wide range of applications targeting data interoperability or analytics. For the analysis of domain-specific data, specialised knowledge resources (terminologies, grammars, word vector models, lexical databases) are necessary. The heterogeneity of such resources is, however, a major obstacle to their efficient use, especially in combination. This paper presents the open-source Diversicon Framework that helps application developers in finding, integrating, and accessing lexical domain knowledge, both symbolic and statistical, in a unified manner. The major components of the framework are: (1) an *API* and *domain knowledge model* that allow applications to retrieve domain knowledge through a common interface from a diversity of resource types, (2) implementations of the API for some of the most commonly used symbolic and statistical knowledge sources, (3) a domain-aware *knowledge base* that helps integrate static lexico-semantic resources, and (4) an online *catalogue* that either hosts or links to existing resources from multiple domains. Support for Diversicon is already integrated into two of the most popular ontology matcher applications, a fact that we exploit to validate the framework and demonstrate its use on a example study that evaluates the effect of several common-sense and domain knowledge resources on a medical ontology matching task.

1 Introduction

Both unstructured and structured data tend to contain large amounts of natural language text, shorter or longer depending on the nature of the data or metadata (e.g., schemas, queries, classification labels, structured data values, semi-structured or unstructured text streams). A certain level of formal understanding of such pieces of text is a necessary prerequisite to a wide range of applications targeting data interoperability (ontology or schema alignment, data integration, query rewriting, entity linking, etc.) or analytics (information retrieval, semantic annotation). Tools for *natural language understanding* (NLU) typically exploit some form of *background knowledge* for the semantic analysis of text, where we mean *knowledge* in the largest possible sense: vocabularies, lexical databases, but also corpus-based resources such as word embeddings, grammars, or machine learning models in general.

Often data and metadata pertain to specific application domains, with text obeying the conventions of specialised domain terminology and grammar. In order to maintain the efficiency of natural language understanding methods for these use cases, the use of common-sense lexical resources, such as WordNet, Wiktionary, or word embeddings trained on general corpora, is generally not sufficient and need to be replaced or extended by domain-specific terms and grammar. This is usually achieved by using domain terminologies or statistical models trained on domain-specific corpora.

In the last decade, deep learning approaches relying solely on unstructured textual input have met considerable success in several areas of NLU and have also been successfully applied in domain contexts using *domain adaptation* or *transfer learning* techniques [25]. More recent research, however, highlighted inherent limitations of fully unsupervised approaches in terms of their ability actually to understand meaning in text [20]. Consequently, the computational linguistics and NLP communities are starting to turn towards hybrid approaches that use structured and unstructured lexical knowledge in com-

bination [32]. Inversely, the *Semantic Web* and *knowledge graph* communities are increasingly incorporating corpus-based approaches into solving traditionally knowledge-driven problems such as entity linking and data integration [34].

The effective use of domain-specific background knowledge resources, however, is subject to numerous challenges.

Finding knowledge. Knowledge resources need to be found online or offline and their relevance to the application evaluated, a time-consuming task for those not familiar with the whereabouts of existing resources.

Plugging in knowledge. The resources need to be plugged into the application, which typically involves interfacing with each resource individually through bespoke implementation. Beyond the problem of interfacing, in the case of large-scale resources—which is typical for domain knowledge—the application may also need to handle performance and memory management issues (e.g., caching). For these reasons, exploiting several resources, whether as part of an experimentation process or for fusion-based approaches, may prove to be prohibitive.

Integrating knowledge. The simultaneous use of complementary yet interrelated domain resources may be necessary, as a single resource may not provide sufficient coverage of a domain or because the application pertains to multiple domains. Mature disciplines have produced domain terminologies that spread through subdomains, go deep into specialised areas, and are diverse due to the varying contexts and points of view of domain experts.[1] Not only may such resources overlap or be related through specialisation, they may also represent knowledge differently (e.g., wordnets, domain terminologies, and word embeddings adopt distinct knowledge models that are hard to reconcile), making integration a prerequisite before their effective use.

Filtering knowledge. Large-scale knowledge bases often cover multiple domains, either because they incorporate domain-specific resources or because their common-sense knowledge is annotated by domain information (as in the case of Princeton WordNet). For high-precision reasoning with domain knowledge, in other words to avoid noise originating from out-of-domain content, it is important to be able to filter query results by domain. However, as resources model domains in different ways (if at all), the filtering mechanism also needs to be resource-specific.

Processing labels. Tasks such as semantic text annotation and indexing, entity linking, or the alignment of schema labels or ontologies all need to apply some form of NLP in order to extract terms in their canonical forms for subsequent lookup, a non-trivial task whose complexity depends both on the language and the domain used.

In this article we present *Diversicon*, a domain-aware, open-source lexical knowledge framework that facilitates access to domain knowledge resources for semantic analysis tasks involving natural language understanding on the lexical level. The framework consists of a common domain-aware knowledge model as well as a set of components that respond to the challenges cited above. On top of the framework itself, we release a set of knowledge resources, adaptor implementations to some well-known online and offline lexical databases and resource types such as *WordNet*, *BabelNet*, and the widely used word embedding models *word2vec* and *GloVe*. We also provide extended versions to two of the most popular and powerful ontology matcher applications, *SMATCH* and *LogMap*, that are equipped to use Diversicon as their underlying knowledge provider. As also demonstrated in our example study, due to the easy integration of domain resources through Diversicon, these matchers can be more effectively used for the alignment of domain ontologies, schemas, and classifications.

The rest of the paper is organised as follows. Section 2 provides an overview of the architecture of the framework and its underlying knowledge model. Sections 3 and 4 present the needs of the most important semantic interoperability applications targeted by the framework and, accordingly, the *Diversicon API* designed to meet these requirements. Section 5 presents the *Diversicon Knowledge Base* and *Catalogue* as background resource components of the framework. Section 6 demonstrates the use of the framework on a example study on matching medical ontologies. Section 7 presents the related work and the paper concludes in section 8.

[1] For example, the SNOMED ontology of medical terms contains over 500,000 concepts and 1,500,000 labels in English only.

2 Framework Overview

Our goal is to provide support for the efficient finding, integration, and querying of domain knowledge resources through a *pluggable knowledge framework*. The main components of the framework as shown in fig. 1, from left to right, are:

- *Static Resources:* downloadable resources on the web, containing dumps of lexical-semantic knowledge in the form of wordnets, domain terminologies, lexicons, and vector space models such as word embeddings;
- *the Diversicon Catalogue:* a metadata catalogue that indexes static lexical domain resources and provides some of them for download in a format directly pluggable into the framework;
- *Knowledge Services:* database systems that provide query services over lexical-semantic knowledge, such as the *Diversicon KB* introduced in this paper, the *Unified Medical Language System*, commonsense multi-domain services such as *DBpedia*, *BabelNet*, and libraries that extract lexical knowledge from word vector models;
- *the Diversicon KB:* a knowledge base that facilitates the integration of static domain knowledge resources under the common *Diversicon Knowledge Model*;
- *the Diversicon API (DivAPI):* a unified knowledge retrieval interface for NLU applications, with implementations towards a variety of knowledge bases;
- *Applications:* solutions for data analysis and interoperability, such as data integration tools, NLP tools, ontology matchers, or search engines, that consume domain knowledge from various sources through calling the DivAPI.

Once set up, the framework provides applications with a unified access to multiple resources describing multiple domains of knowledge. The DivAPI provides language- and domain-aware knowledge lookup methods tailored to the needs of NLU applications. It hides implementation details and modelling heterogeneities of knowledge bases and resources, allowing a seamless switch from one resource to another for application developers. The Diversicon KB provides integrated access to resources otherwise available as static files, separating the responsibility of loading and integrating heterogeneous knowledge resources from their use, taking care of the former and letting applications concentrate on the latter.

3 Target Applications

In order to motivate the design of the framework, we briefly evoke the requirements of an example set of target applications. These applications all involve a natural language understanding task that benefit from the use of lexical domain knowledge. Given the diversity of approaches to NLU, it would be impossible to foresee an exhaustive set of services that cater to all possible needs; however, the interfaces and implementation of Diversicon remain open to assure their extensibility.

Domain Ontology Alignment. This task also covers schema mapping and query rewriting in data integration tasks, as well as the matching of domain classifications. Most alignment methods involve the comparison of node labels, based on various forms of evidence such as string similarity or ontological relatedness. The most relevant case to us, however, is the use of lexico-semantic similarity that relies on the equivalence, similarity, or relatedness of the meanings of words in the label. Evidence for such relations is typically obtained either through *lexical expansion*, i.e., the retrieval of synonymous, similar, or related words, or through reasoning with lexico-semantic axioms (e.g., hypernymy, semantic similarity, derivational relatedness). Clearly, such evidence can be retrieved from both symbolic and statistical domain resources, such as domain terminologies or word embeddings trained on domain corpora. As indicated in fig. 1, support for the DivAPI and the Diversicon Framework has already been implemented for *LogMap* [16], one of the best-faring matchers in multiple tasks of the yearly *Ontology Alignment Evaluation Initiative*[2], and *SMATCH* [12], a popular matcher with thousands of downloads, and derived applications such as schema matching [11] and dynamic query alignment

[2] http://oaei.ontologymatching.org/

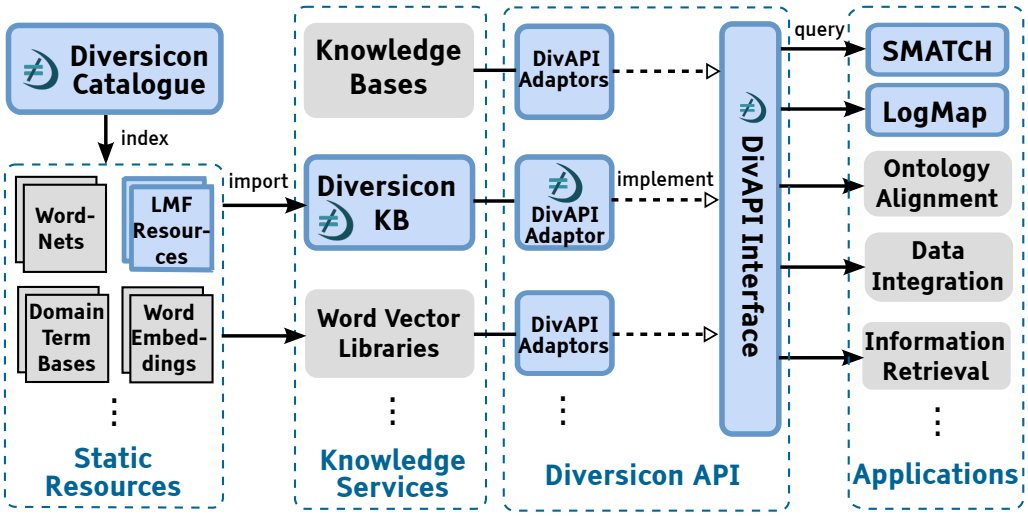

Figure 1: High-level architecture of interoperability applications using background knowledge resources through the DivAPI and the Diversicon Knowledge Base. Components contributed (created or extended) by the paper have been highlighted in blue with thick framing.

[22]. Integration with Diversicon provides these matchers with access to domain knowledge, improving their precision and recall (cf. section 6.3).

Semantic Annotation. Semantic annotation is used in a vast range of applications fulfilling diverse goals and approaches. We cannot attempt an exhaustive coverage of them; in particular, we do not consider tasks that are commonly solved by end-to-end machine learning approaches, such as sentiment analysis. Rather, we concentrate on a subset of usages where symbolic domain knowledge is used for annotation, such as *semantic indexing* for information retrieval, *entity linking*, or *information extraction* for data integration [28]. In all of these cases, the task consists of identifying labels within the text that refer to a specific category of domain concepts or entities. Semantic annotation is typically divided into a prior *recognition* step that identifies the subset of words and expressions to be annotated (e.g., named entity recognition), followed by a *disambiguation* step that selects the most likely meaning for each item recognised. In domain applications, the base set of meanings usually consists of domain terminologies. Diversicon can be used to query such terminologies in a uniform manner, either separately or simultaneously. Disambiguation algorithms, in turn, often rely on a relatedness measure, either computed within the textual context (semantic relatedness across words or across meanings) or with respect to a global domain context (relevance of a meaning with respect to a given domain), as in [2]. The Diversicon Knowledge Model explicitly represents word–word, meaning–meaning, and meaning–domain relations and is able to retrieve them from a combination of resources (respectively: word vector models such as *GloVe* or *word2vec*, terminological DBs such as [3], and domain DBs such as [18, 14]) in order to support disambiguation algorithms.

Classification. In the context of NLU, automated classification is used to categorise large amounts of text—documents or data records—under a varying number of classes. However, as shown in [8], *classification* is an umbrella term that denotes very different cognitive mechanisms that may resemble both a similarity-based form of *recognition* and a form of *reasoning* based on concept subsumption (e.g., *pizzeria* is classified under the broader term *restaurant*). Classic machine-learning-based classification typically solves the former recognitional task, while formal-knowledge-based approaches are better

suited to reasoning. Depending on the underlying knowledge resource, the API of Diversicon can provide either approach with the right kind of lexical-level knowledge—similarity or subsumption—or, due to the unified access, allow for the combined use of both, e.g., by mapping subsumption relations to similarity.

Resource Evaluation. A 'meta-task' that greatly benefits from Diversicon is the comparison of knowledge resources given their performance on a downstream NLU task. Such an evaluation can help in choosing the best resource(s) for a task but also to accelerate the cycles of resource development. Our case study in section 6 presents a large-scale example evaluation using five different resources and two downstream applications, in no less than fourteen (!) setups. The role played by Diversicon is to unify the access to heterogeneous (and sometimes fundamentally diverse) resources and allow switching among them in a trivially simple way (i.e., changing a single line of source code).

4 The Diversicon API and Knowledge Model

The DivAPI is a minimal yet powerful, Java-based collection of services, data structures, and constants, in a way that corresponds to the typical use of domain background knowledge by applications requiring semantic interoperability over natural language text. The API acts as an abstraction layer towards a variety of knowledge resources, leading to a pluggable architecture as shown in fig. 1. It consists of the *DivAPI Interface* that defines and exposes the services to applications in a unified manner, and of a set of *adaptors* that implement the interface towards various knowledge resources, including the Diversicon KB.

A prerequisite of providing a unified query interface for a heterogeneous set of domain resources is to expose a common knowledge model to matchers, to which each of the resources needs to be mapped.

4.1 Knowledge Model

The Diversicon Knowledge Model serves as an intermediary between heterogeneous knowledge resources on the one hand, and applications with differing needs and approaches on the other hand. Thus, the design challenge was to find a suitable compromise between generality and representational power in view of the targeted knowledge sources and applications. Our example study (section 6) found the current model to be an adequate interface between the applications and resources evaluated; nevertheless, a more widespread use of the framework may lead to the fine-tuning or evolution of the model.

The heterogeneity of knowledge bases and resources manifests itself in multiple ways:

- they may be mono- or multilingual;
- they may relate to single or multiple domains, with the notion of domain modelled in various ways (or not at all);
- they may represent meaning in different manners: algebraically (as numerical vectors) or symbolically;
- they may represent different kinds of meaning: lexico-semantic (e.g., wordnets or domain terminologies) or ontological.

Furthermore, for the majority of knowledge resources, domain categories and the notion of domain itself are outside the scope of modelling, either because the resources are domain-agnostic or because they relate to a single domain and thus the categorisation is implicit. Explicit domain categories become necessary, however, when a knowledge base, resource, or application covers multiple domains and intends to model this phenomenon in a formal way. Several computational approaches to domain modelling have been explored in scientific literature. Corpus-based approaches derive domains from words and their context [13], an early and influential example being the *theory of lexical fields* [30]. Lexicographically created resources, such as domain dictionaries, are also word-based. Other approaches apprehend domains as categories of *lexical meanings* rather than of words [18, 14, 9].

In our model we adopt the semantic interpretation of domain and define domains as sets over lexical meanings, while remaining interoperable to a large extent with corpus-based and lexically-oriented resources. We introduce the fundamental entities and relationships of the model below, also depicted in fig. 2.

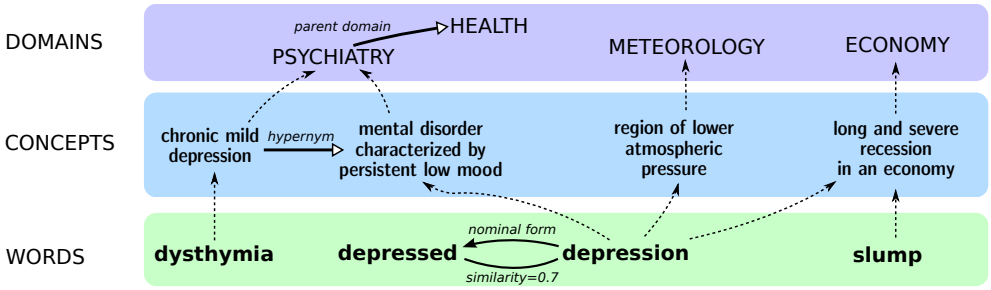

Figure 2: The three-layered, domain-aware *Diversicon Knowledge Model* adopted by the DivAPI

Word: a string of characters that is a lexeme (a lexical item) of a language. In this paper we will also call 'words' multiword expressions (e.g., *hot dog*).

Concept: the unit of lexical meaning. A concept represents the meaning of zero, one, or more words in a given language: our model is thus oriented towards the representation of lexical (as opposed to ontological) meanings.

Domain: following the practice of meaning-based approaches [9, 13, 18], we define domains as sets over lexical concepts. One concept can, in principle, belong to several domains, or its domain may be undefined. The interface defines its own domain hierarchy, represented as string constants, based on the standard *Universal Decimal Classification*.[3]

Word relation: a binary relation of two words, such as weighted string similarity (*depressed* is similar to *depression* with a weight of 0.7) or morphological relatedness (*depression* is the nominal form of *depressed*).

Concept relation: a binary relation of two lexical concepts, such as *equivalence*, *hypernymy*, *meronymy*, or *similarity*.

Domain relation: a binary relation of two domains: the framework currently supports the *parent domain* hierarchical relation which can be used to define a hierarchy of domains: if a concept belongs to the domain of *Psychiatry* then it also belongs to its ancestor domains, e.g., *Healthcare* and *Applied Sciences*.

Word-to-concept relation: indicates that the concept is lexicalised by the word, e.g., the meteorological concept of *region of low atmospheric pressure* is expressed by the word *depression*. The model allows each relation to be weighted by a real number, such as sense frequencies or sense ranks as also used in Princeton WordNet. Depending on whether and how weighting is applied, concepts can be conceived as either classical or fuzzy sets of synonymous words, analogously to WordNet *synsets* (synonym sets).

Concept-to-domain relation: indicates that a concept belongs to a domain, e.g., the concept of *chronic mild depression* belongs to the domain of *psychiatry*. The model also allows these relations to be weighted by real numbers, which means that domains can again be interpreted as classical or fuzzy sets over concepts. These correspond to the domain models provided by well-known domain modelling approaches such as *WordNet Domains* [18] and *Extended WordNet Domains* [14], respectively.

4.2 Interface

The *DivAPI Interface* is called by applications to which it provides background knowledge in a suitable form. Applications, however, process natural language and use background knowledge in diverse manners internally. They may reason quantitatively based on the similarity of strings or concepts. Such applications expect background knowledge to provide similarity measures between units of knowledge (words or concepts):

$$\textit{CVD} \approx_{95\%} \textit{cerebrovascular disease}$$
$$\textit{progressing stroke} \approx_{62\%} \textit{cerebrovascular disease}$$

[3] http://www.udcc.org

Method	Return value	Weighted return value	Input arguments	Explanation
Word-level methods				
getRelatedWords	List(Word)	List(Word, Weight)	Lang, Domain, Word, WordRelation	words that are synonymous, similar, or related to the input word
getRelations	Set(Word-Relation)	List(WordRelation, Weight)	Lang, Domain, Word, Word	relatedness of the two input words
getLanguages	Set(Lang)	∅	Domain, Word	languages that cover the input word
getDomains	Set(Domain)	∅	Lang, Word	domains that cover the input word
getDomainsWeighted	∅	List(Domain, Weight)	Lang, Word, Set(Domain)	degree by which the input word belongs to each domain
Concept-level methods				
getConcepts	Set(Concept)	List(Concept, Weight)	Lang, Domain, Word	concepts expressed by the input word
getConstrained-Concepts	Set(Concept)	List(Concept, Weight)	Lang, Domain, Word, Concept	concepts expressed by the input word that are hyponyms of the input concept
getWords	Set(Word)	List(Word, Weight)	Lang, Concept	words that express the input concept
getGloss	String	∅	Lang, Concept	gloss of the input concept in the input language
getRelatedConcepts	Set(Concept)	List(Concept, Weight)	Concept, Set(ConceptRelation)	concepts that are related to the input concept through the given relations
getRelations	Set(Concept-Relation)	List(ConceptRelation, Weight)	Concept, Concept	lexico-semantic relations that hold between the two input concepts
getLanguages	Set(Lang)	∅	Concept	languages that cover the input concept
getDomains	Set(Domain)	List(Domain, Weight)	Concept	domains that cover the input concept
Global methods				
getLanguages	Set(Lang)	∅	∅	languages supported by the resource
getDomains	Set(Domain)	∅	∅	domains supported by the resource

Table 1: Methods for querying background knowledge provided by the current DivAPI to ontology matchers; most methods have a qualitative and a weighted version as indicated by the return type

A more qualitative form of reasoning (e.g., logic-based) may not need knowledge to be quantified but, in contrast, may require relations to bear explicit semantics (e.g., as equivalence, subsumption, antonymy):

$$CVD \equiv \textit{cerebrovascular disease}$$
$$\textit{progressing stroke} \sqsubset \textit{cerebrovascular disease}$$

Furthermore, applications may or may not have explicit, symbolic internal representations of lexical concepts. In the former case the application may need services such as *return me all meanings (concepts) of the word* `cat` and, subsequently, *return me the similarity of the concepts of 'cat' and 'kitty'*. In the second case, the application may work only with string representations internally and rely entirely on the background knowledge service to provide semantic reasoning capabilities, e.g., for lexical expansion: *return me all synonyms of the word* `cat`.

Accordingly, the Interface is articulated around the following requirements:

- *domain- and language-awareness:* it has the ability to filter results by language and by domain (if such information is provided by the underlying resource);
- *compliance to the Diversicon Knowledge Model:* it is based on an explicit representation of all five formal model elements (as in fig. 2): words, concepts, relations, languages, and domains;
- *support for common semantic interoperability applications:* covering the basic knowledge query needs of the applications presented in section 3;
- *support for both word-level and concept-level reasoning*;
- *support for both similarity-based and logic-based reasoning:* it has the ability to return both qualitative and quantitative results.

The list of background knowledge query methods of the Interface, responding to the requirements above, are listed in table 1. It can be seen from the figure that each method has two versions: a qualitative one and a quantitative weighted one. Thus, the qualitative version of *getRelatedWords* will return a set of words that bear a relation (synonymy or other) with the input word, while its quantitative version will return a map where the degree of relatedness of each word is given by a weight.

Most methods have a *language* or a *domain* input argument, or both. They allow the filtering of the output by language or by domain, e.g., a query

```
getRelatedWords("eng", HEALTH, "cat", SYNONYMY)
```

with `"eng"` indicating English and `HEALTH` the domain, will in principle only return '*computer-aided tomography*' as opposed to terms pertaining to felines. The use of these arguments is optional: either or both can be set as `null`, in which case no filtering is applied.

4.3 Adaptors

The role of *DivAPI Adaptors* is to implement the Interface methods with respect to specific knowledge bases and resources, converting between the internal knowledge representation of each of them and the Diversicon Knowledge Model. However, just like end-user applications, knowledge bases may be very different conceptually and may not be able to provide all types of knowledge supported by the Interface. For example, they may be monolingual, mono-domain or domain-agnostic, and they may or may not have an explicit representation of concepts or concept relations.

The Diversicon Framework currently implements adaptors, freely available as part of the project distribution, both for symbolic lexico-semantic domain knowledge (currently the *Diversicon KB* and *BabelNet*) and for statistical word-vector-based knowledge (two major word vector embedding approaches, *word2vec* and *GloVe*, are currently supported). By design, adaptors are independent of each other and the Framework does not attempt to combine knowledge provided by different sources. Such a behaviour would imply an awareness of the relative authority of resources, as well as algorithmic strategies for knowledge integration and inconsistency resolution. We believe that such logic is safer and more robust to implement in an ad-hoc manner on the application level. However, the *Diversicon KB*, described in section 5, does provide infrastructure for the integration of static knowledge resources within a single knowledge base.

Below we present the challenges and our solutions for implementing adaptors for the most important kinds of background knowledge.

Wordnets. Wordnets cover (a subset of) the general lexicon of a language. Their use in domain-related natural language understanding tasks is justified by the frequent presence of common-sense vocabulary even in highly domain-specific texts. The knowledge model of wordnets is composed of *words* (i.e., lemmas), synonym sets or *synsets* as word meanings, as well as *senses* as the reification of word–synset relations. The WordNet model is straightforward to map to the (somewhat simpler) structure of the Diversicon Knowledge Model: *synset* $\mapsto$ *concept*, *word* $\mapsto$ *word*, *WordNet topic* $\mapsto$ *domain*, *sense relation* $\mapsto$ *word relation*, and *synset relation* $\mapsto$ *concept relation*. As wordnets are distributed in a static, file-based form, some kind of accessing logic is necessary to retrieve its contents. A WordNet DivAPI adaptor can either use existing Java libraries such as *extJWNL* or, as a convenient alternative that we implemented, wordnets can be ingested by the *Diversicon KB* (cf. section 5) which already provides its own adaptor. The English Princeton WordNet, specifically, is included by default in the Diversicon distribution in LMF format and is also downloadable from the Catalogue.

Wordnets enriched with domain information. The domain coverage of Princeton WordNet has been investigated by Magnini et al. [18] and later by Agirre et al. [14]. The rationale behind these efforts is that the division between common-sense and domain-specific knowledge is not clear-cut: rather, there is a large grey area between the two where commonly used words (or more precisely: word senses) can be considered as belonging more to certain domains than to others (e.g., *illness* is an everyday word that belongs more to *medicine* than, say, *transportation*). Both of these efforts define domains as sets over word meanings (in their case, WordNet synsets) as opposed to the words, considering domains as semantic as opposed to lexical objects. There is, however, one major difference in the two models: [18] uses regular sets while [14] uses fuzzy sets where every meaning belongs to every

domain but with a different real-valued weight. The latter is thus able to model the fact that the concept of *river* belongs both to *transportation* and *hydrology* but to different extents. As both approaches are widely used (see, for example, [19, 2]), the Diversicon Knowledge Model, the DivAPI, and the Diversicon KB were all designed to support either approach by providing concept–domain relations as part of the model, allowing them to be qualified by weights.

Static domain terminologies. Domain terminologies are meaning-based and often multilingual resources that pertain to a single domain. They are mostly used to represent synonymy and, in the case of a multilingual resource, also translations. More rarely, they provide relational information such as polysemy or hypernymy. Using static file-based resources (such as domain terminologies represented as spreadsheets or XML trees) as background knowledge necessarily involves a conversion process involving the interpretation of the source format, its transformation to the Diversicon Knowledge Model, and providing fast access to the transformed resource through the DivAPI. While extraction and transformation are resource-specific operations, load and access are shared across resources, which explains the role of *Diversicon KB* within the framework as a database providing a simultaneous and unified storage and view on static resources. Domain terminologies are straightforward to map to the Diversicon Knowledge Model as their notions of *terminological unit* and *term* map to our model's *concept* and *word*, respectively (cf. [33] for more details on the standard modelling of domain terminologies). The domain itself is most often implicit and is not formally represented within terminologies, as their entire content typically pertains to a single domain.

Lexico-semantic knowledge bases. Knowledge bases such as the offline *Diversicon KB* or the online *BabelNet* and *UMLS Metathesaurus* may integrate lexical-semantic knowledge from a variety of sources such as wordnets, online encyclopaedias, or domain term bases. They may be mono- or multilingual and they may be domain-agnostic, domain-specific, or cover multiple domains. They usually adopt a knowledge model very similar to the one of Princeton WordNet presented above, that maps easily to the DivAPI. Non-trivial mappings that adaptors may need to implement on a resource-specific basis are the conversions between the word and concept relation types defined by the knowledge base and by the DivAPI (e.g., *semantic relatedness* of BabelNet is converted into *relatedness* Diversicon word and concept relations) and between elements of the domain set defined by either model (e.g., the DivAPI domain *Healthcare* maps to the BabelNet domain *Health and medicine*). These mappings, including the resolution of potential ambiguities, need to be provided manually as part of the adaptor implementation.

Word vector models. Distributional models and word embeddings associate to each word a numerical vector that, based on a training corpus, represents the context in which the word frequently appears. The vector provides a distributional representation of the 'meaning space' of the word, this interpretation of meaning being, of course, very different from that of symbolic lexical concepts used by, e.g., WordNet. Contrary to the latter, word vector models provide coarser-grained semantics: in their classic implementation, vector distances provide information on *semantic relatedness* as opposed to equivalence or similarity in a strict sense, usually computed as the cosine similarity of word vector pairs.[4] For the purposes of domain-aware applications involving disambiguation or matching tasks, domain-specific embeddings (trained on domain corpora) and embeddings specifically optimised towards semantic similarity are better candidates.

For loading the embedding files and computing similarities we relied on existing Java tools: *Deeplearning4J*[5] for *word2vec* and *Thomas Jungblut's Java wrapper*[6] for *GloVe*. We implemented the adaptors in such a way that the similarity measures output by word embeddings can both be directly used through the weighted DivAPI interface and converted into a pseudo-equivalence relation using a threshold set as an input parameter.

5 The Diversicon KB and Catalogue

The purpose of the *Diversicon KB*[7] is to integrate static lexico-semantic knowledge resources (wordnets, domain

[4] The vectors of antonyms, such as *hot* and *cold*, are typically considered as very closely related by these models.

[5] https://deeplearning4j.org/

[6] https://github.com/thomasjungblut/glove

[7] `http://github.com/diversicon-kb/`

terminologies, lexicons distributed as files) of multiple domains and languages into a single database, providing an efficient and unified interface to manage and retrieve its contents. It implements a pipeline of functionalities, in the following order:

1. validation of a static lexico-semantic resource provided as input;
2. integration of a new resource with existing ones;
3. reasoning over the extended knowledge base;
4. providing the KB contents to external services;
5. upon request, exporting of the KB contents.

The KB is typically installed on the machine running the NLU application and is populated locally by file-based resources.

The Diversicon KB was built upon the UBY *Framework*, a Java framework for creating and accessing sense-linked lexical resources [15]. The main novelty with respect to the latter is an explicit and domain-aware support for the simultaneous use of multiple resources, both separately and in combination. The implementation of this feature made it necessary to improve and extend UBY on several accounts, as follows.

Domains: the system was extended by an explicit support for domains and domain-categorisation of concepts, as explained below.

Namespaces: in order to avoid unique identifier clashes when composing multiple resources (e.g., WordNet extended by domain terminologies), the system supports and enforces the use of namespaces for unique identifiers. This practice helps ensure an internal coherence to the knowledge base and track the provenance of integrated knowledge.

Reasoning: transitive closure is computed over transitive concept and domain relations, each time a resource is updated or combined with another one.

Ingestion mechanism: a user-friendly command-line interface is provided for the ingestion and combination of domain resources.

Input normalisation and strict validation: for the input format, Diversicon provides a DTD as well as an XML schema that are coherent with the importer implementation (which was not the case with UBY). In contrast to UBY, validity with respect to the schema is enforced during importing. Furthermore, input knowledge is normalised: loops are not accepted for transitive relations, inverse relations are replaced with canonical ones (e.g., hypernymy is used throughout instead of hyponymy), representations are simplified, etc.

5.1 Input Format

For the ingestion of static knowledge, the Diversicon KB supports the LMF *(Lexical Markup Framework)* format, an ISO standard for representing lexico-semantic resources [7, 29, 23]. It also supports the direct importing of resources from Wikipedia, Wiktionary, OmegaWiki, and FrameNet, the importer logic for these resources being provided by UBY upon which Diversicon is built.

The LMF format is multilingual and extensible, allowing rich knowledge to be expressed on the lexical, grammatical, semantic, and domain levels. The precise XML schema as well as the DTD, backward compatible with the UBY-LMF implementation, are included in the Diversicon distribution. The elements of the common model were mapped to LMF as follows.

- *Words* are represented as `Lemma` elements.
- *Concepts* correspond to `Synsets`.
- *Word-to-concept relations* are reified as `Senses`, following the structure of Princeton WordNet.
- *Domains* are also represented as `Synsets` that model the meaning of the domain category. The name of the domain, in turn, is represented as a `Lemma` attached to the `Synset` through a `Sense`.
- *Concept relations* are represented as `SynsetRelations`.
- *Domain relations* are also `SynsetRelations` that hold between domain synsets. The only domain relation we have defined so far is the *parent domain* relation that allows us to model domain hierarchies.
- *Concept-to-domain relations* that state that concepts belong to domains are, again, represented as

`SynsetRelations`. The name of the relation is simply `domain`.

Furthermore, as also foreseen in our knowledge model, LMF allows both word-to-concept and concept-to-domain relations (and any entity or relation, in fact) to be weighted by frequency, through the `Frequency` element.

5.2 Importing and Integration

A command-line interface, *DiverCLI*,[8] was developed to support the importing mechanism, including a strict validation of the conformity of the input to LMF specifications. DiverCLI supports the importing of a new lexical resource that will be stored as a separate resource in the database, or the extension of an existing resource. The latter functionality can be used, among others, to integrate common-sense and domain knowledge.

Domain-specific applications can benefit from the simultaneous use of multiple background knowledge resources. For example, in a scenario of medical ontology alignment, a simultaneous use of general common-sense knowledge and domain knowledge may help in computing the semantic relation of subsumption between the labels

Temporary blindness $\sqsubset$ *Transient visual impairment*

where a general common-sense lexical resource can provide the synonymy *transient* $\equiv$ *temporary*, while a medical domain resource may provide that *blindness* is a narrower term than *visual impairment*.

In order to provide a kernel of general common-sense background knowledge to NLU applications, DiverCLI is bundled with the latest Princeton WordNet (currently WordNet 3.1) that we converted into LMF format. The conversion was implemented by means of the *DivMaker* tool[9] that was designed to convert wordnets into LMF and can be used for this purpose for different languages or different wordnet versions in the same language.

The combination of general or domain knowledge resources is supported through an importing mechanism that loads multiple resources into the same database. These resources may be logically independent from each other (i.e., partitions) or, on the contrary, interconnected through relations such as hypernymy–hyponymy. Cross-resource links are taken into account when the KB computes the closures of transitive relations. The problem of potentially clashing unique identifiers is handled through a classic namespace mechanism where each resource is identified by its own namespace. Namespaces also provide a solution for indicating the provenance of each knowledge unit integrated.

Finally, as a way of supporting domain-specific NLU tasks even without the use of domain background knowledge, Diversicon can import WordNet concepts (synsets) annotated by domain information through two distinct mechanisms. The first one consists of exploiting the *topic annotations* already provided by WordNet. When importing the LMF, DiverCLI detects such topic relations and interprets them as domain information. However, in the latest WordNet 3.1, only about 5% of the total 120,000 synsets are annotated by such topic relations. DiverCLI therefore offers a second source of domain information through the importing of the *Extended WordNet Domains* (XWND) resource [14]. XWND annotates each of the 120,000 WordNet synsets with weighted relations to about 180 domain categories, modelling the phenomenon of lexical meanings belonging to several domain categories simultaneously. DiverCLI can import XWND simultaneously with the WordNet LMF and provide these weighted concept-to-domain relations to ontology matchers through the DivAPI.

5.3 Reasoning

As part of the integration process, the Diversicon KB precomputes the transitive closure of the transitive domain and concept relations (such as hypernymy, part–whole, and subdomain) within the resource being imported. If the resource is attached to an existing one (e.g., a domain resource extends WordNet) then the transitive closure is computed over the combined resource. The set of asserted relations is then extended by the inferred relations.

5.4 Querying and Exporting

Besides the low-level query API inherited from UBY and structured around its LMF-based internal data model, the Diversicon KB also provides higher-level, easier-to-use accessor methods to knowledge. In addition, the dedicated

[8] `http://github.com/diversicon-kb/divercli`
[9] `http://github.com/diversicon-kb/divmaker`

DivAPI adaptor implements a further level of abstraction from the output of the KB to the high-level knowledge model and methods offered by the DivAPI Interface.

Exporting functionalities for the knowledge imported are provided in the native Diversicon LMF format, in low-level SQL, as well as in the RDF-based *Lemon* model [21] thanks to an LMF-to-Lemon XSL transformation created for *lemonUby* [5].

5.5 The Diversicon Catalogue

The *Diversicon Catalogue*[10] is a web data catalogue that collects and organises domain-specific and domain-aware lexico-semantic resources. The purpose of the Catalogue is to make it easier for users to share as well as to find such domain resources, especially (but not exclusively) for using them with the Diversicon KB.

The Catalogue uses a DKAN[11] engine and therefore has a look and feel similar to most open data catalogues. It organises resources hierarchically by domain, adopting the upper part of the *Universal Decimal Classification* as its hierarchy.

Among other resources, the LMF-based WordNet presented in section 5.2 is downloadable from the Catalogue.

6 Example Study: Medical Resource Evaluation

The preparation and optimisation of tools for large-scale real-world semantic applications, or the evaluation of solutions in the framework of research often involve the exploration of a large solution space with several possible combinations of tools and resources to compare. As one possible application area among many, this example study addresses the problem of domain ontology alignment, focussing specifically on medical ontologies. Today, a large number of ontology matcher tools are available, many of which are capable of exploiting one or another form of background knowledge for increased precision and recall. In order to find the best-performing solution we wish to compare multiple combinations of matchers and knowledge resources over a common domain-specific evaluation. In our example scenario we evaluate two state-of-the-art open-source ontology matchers, namely SMATCH [12] and LogMap [16], on a biomedical matching task. To do so, we take advantage of the Diversicon framework to reduce the implementational overhead related to supporting a diversity of resources exploited by a diversity of ontology matcher tools. The main message of this example study is the efficient use of Diversicon in order to reduce such efforts, rather than the actual outcome of the evaluations presented below.

6.1 Preparing the Background Knowledge

In terms of background knowledge, we wish to try out a wide range of resources: lexico-semantic and word-vector-based, common-sense and domain-specific. In this example study we consider the five resources shown in table 2 which, with the exception of BabelNet, are either indexed or downloadable from the Diversicon Catalogue. The BabelNet resource is used in two different ways: on the one hand, as a general-purpose knowledge base and, on the other hand, using its subset that pertains to the medical domain, taking advantage of domain-specific filtering exploited by the DivAPI.

The advantage of passing through the DivAPI is easily understood given that, instead of $2 \times 5 = 10$ ad-hoc implementations between the two applications and the five resources, only one API adaptor needs to be implemented per resource type and SMATCH and LogMap only need to connect to the Diversicon API. The major benefit comes from the near-zero-cost reusability of these components when subsequent applications or resources are introduced: the four adaptors used in this example study and presented below are all available online, freely reusable by other applications.

Princeton WordNet. For this resource, no extra work was needed as it is included by default in the Diversicon distribution in LMF format and is also downloadable from the Catalogue.

The SPECIALIST Lexicon. To support the medical matching task with domain knowledge, we integrated the *SPECIALIST Lexicon*[12] into the Diversicon KB which, in turn, provides its content to both matchers. SPECIALIST, a

[10] http://www.diversicon-kb.eu

[11] http://getdkan.org

[12] https://specialist.nlm.nih.gov/lexicon/

Name	Coverage	Kind	Format	Adaptor package
Princeton WordNet	common-sense	lexico-semantic	static files	divapi-diversiconkb
GloVe Common Crawl 840B-300D	common-sense	distributional	word embeddings	divapi-glove
BabelNet	common-sense	lexico-semantic	online KB	divapi-babelnet
BabelNet filtered to health domain	medical	lexico-semantic	online KB	divapi-babelnet
SPECIALIST Lexicon	medical	lexico-semantic	static files	divapi-diversiconkb
word2vec PubMed	medical	distributional	word embeddings	divapi-word2vec

Table 2: List of knowledge resources used for medical ontology matching in the example study

subset of *Unified Medical Language System*,[13] contains morphological, lexical, and lexico-semantic data that support the analysis of English medical text. For our example study we extracted from SPECIALIST the following information:

- lexical variants;
- expansions of acronyms;
- synonymy relations.

For example, ontology matchers may exploit the synonymy relation between *alcoholism* and *alcohol dependence* to produce the mapping

$$\textit{dementia due to alcoholism} \mapsto^{\equiv} \textit{dementia due to alcohol dependence}$$

We converted SPECIALIST into LMF as defined by the Diversicon LMF schema, the conversion being a straightforward process executed by a Python script that grouped the three relation types above into LMF synsets. We share the resulting LMF in the Diversicon Catalogue from where it is freely downloadable.

BabelNet. For integrating BabelNet we used the *divapi-babelnet* adaptor provided by the Diversicon framework. BabelNet is rich in synonyms and semantic relations which, however, vary greatly both in domain coverage and in quality due to the diversity of the source material and BabelNet's fully automated approach to knowledge integration. In order to mitigate the potentially noisy results leading to a disproportionate amount of false positives, the adaptor was finetuned in the following way: firstly, it was restricted to use only high-quality synset relations from BabelNet, by setting the corresponding flag in queries. Secondly, we experimented with domain-based filtering of background knowledge queries made possible by the BabelNet API, using the *domain = healthcare* filtering feature in DivAPI methods. And thirdly, we experimented with exploiting semantic relations of synonymy, similarity, or relatedness by setting the *relation type* parameter in the DivAPI calls.

Word embeddings. We integrated two kinds of word embedding resources, *GloVe* and *word2vec*, each kind through its dedicated adaptor (already presented in section 4.3). With GloVe we used the 840-billion-token *Common Crawl* non-specialised corpus[14] while with word2vec we accessed a domain-specific corpus trained on the *PubMed* resource[15] that consists of more than 28 million citations from biomedical literature. To access these models through the DivAPI it is sufficient to call the constructor of the relevant DivAPI adaptor with the path to the model file as input parameter. For the comparability of results, however, it was necessary to understand the qualitative interpretation of the real-valued vector similarity values provided by the GloVe and the word2vec resource, as their similarity distributions are dependent on their training parameters. The DivAPI adaptors allow a similarity threshold to be set on a per-resource basis, which we used to optimise the results in a simple way.

[13] https://www.nlm.nih.gov/research/umls/

[14] https://nlp.stanford.edu/projects/glove/

[15] https://www.ncbi.nlm.nih.gov/pubmed/

Resource	Type	SMATCH Prec.	SMATCH Recall	SMATCH F1	LogMap Prec.	LogMap Recall	LogMap F1
Baseline (without Diversicon)		70.6%	33.0%	45.0%	93.5%	26.6%	41.4%
Princeton WordNet	common-sense	70.6%	33.0%	45.0%	93.8%	27.5%	42.6%
GloVe Common Crawl	common-sense	75.0%	30.3%	43.1%	90.9%	27.5%	42.3%
BabelNet	common-sense	53.6%	33.9%	41.6%	89.2%	30.3%	45.2%
BabelNet filtered	medical	84.4%	34.9%	49.4%	94.1%	29.6%	45.1%
word2vec PubMed	medical	82.0%	37.6%	51.6%	93.9%	28.4%	43.7%
WordNet + SPECIALIST	medical	79.5%	32.1%	45.8%	94.1%	29.4%	44.8%

Table 3: Example study evaluation results with SMATCH and LogMap, on six different kinds of background knowledge

6.2 Preparing the Matcher Applications

The last step is to implement the DivAPI calls from within the applications, in our case the two matchers SMATCH[16] and LogMap[17]. As most state-of-the-art matchers, they are both implemented in Java, making the calls to the Java-based DivAPI straightforward.

No two applications exploit knowledge in an identical way. In the case of SMATCH and LogMap, a fundamental difference lies in how each one defines a mapping: for LogMap a mapping expresses syntactic or semantic similarity and is quantified by a user-defined threshold over a range of real-valued similarities between 0 and 1. For SMATCH, in contrast, a mapping stands for a logical entailment between ontology nodes previously converted into prepositional logic formulas. The two matchers thus reason in fundamentally different manners: one quantitatively and the other qualitatively.

Another major difference relates to the internal matching strategies: for SMATCH the semantic analysis of labels is a major step of the matching process. It relies heavily on the use of lexical background knowledge: it calls the DivAPI `getConcepts` method to convert words into description logic concepts, and `getRelations` to enrich the set of logical formulas with axioms, reasoning mostly on the concept level. LogMap, on the other hand, has no explicit representation of lexical concepts. It reasons on the word level and uses lexical knowledge as one among many matching strategies. Its use of background knowledge is limited to *lexical expansion*, i.e., the retrieval of synonymous words. The DivAPI supports this operation through its `getRelatedWords` method.

6.3 Running the Evaluations

We finally present evaluation results obtained with the resources and matchers discussed above integrated into Diversicon. The evaluation was not intended as an attempt to advance the state of the art on domain-based ontology alignment, nor to draw definitive conclusions on the performance of various resources and underlying approaches. Rather, our point is the straightfoward feasibility of such a large-scale evaluation.

We used an evaluation corpus from the medical domain: on the one hand, the medical reference ontology *SNOMED CT* [4] and, on the other hand, *ICD-10-CM*, i.e., the *International Classification of Diseases*, version 10, with US-specific extensions. Expert-sourced gold-standard mappings were provided by the *U.S. National Library of Medicine*. As the assessment of results involves some level of manual evaluation (not all true positives appear in the gold stanard mappings), in order to reduce the evaluation efforts over the 14 setups, we randomly extracted a 100-node evaluation corpus that we fed into both SMATCH and LogMap.

As most of the gold-standard mappings are not equivalence but similarity mappings, we considered both equivalence and subsumption mappings output by SMATCH as positives. For LogMap, which outputs similarity mappings, we finetuned input parameters (such as similarity

[16]The Diversicon-based SMATCH extensions are downloadable from https://github.com/s-match/ .

[17]The Diversicon-equipped version of LogMap is downloadable from https://github.com/diversicon-kb/logmap-matcher .

threshold) to maximise the F-score.

The results can be seen in table 3, about which four things need to be noted. Firstly: we consider the experiment a success as it constitutes a first validation of the usability of the entire Diversicon framework (model, API, and adaptors) within real-world applications.

Secondly: while results between SMATCH and LogMap appear to be within the same range, they are not fully cross-comparable given the markedly different ways they define mappings (logical entailment for SMATCH vs similarity for LogMap). They also give different importance to lexical background knowledge: from the very uniform results of LogMap we conclude that background knowledge only plays a relatively minor role in its overall matching method.

Thirdly: comparison to the original, Diversicon-less baseline versions of the matchers shows a small yet consistent (1.2–3.8%) improvement for LogMap when relying on lexical resources. For SMATCH, baseline results are identical to the *Princeton WordNet* scenario: this is a trivial result given that the original SMATCH is already using WordNet in an identical way, just linking to it statically instead of passing through Diversicon.

Lastly: despite the fact that our aim was not to evaluate ontology matchers but rather to demonstrate the ease with which resources can be compared using Diversicon, we nevertheless observe, in accordance with intuitive expectations, a moderate but consistent advantage in using domain-specific background knowledge as opposed to common-sense resources, primarily in precision but also in recall. This fact is especially visible in the case of BabelNet, which is by far the largest resource in our study: when used in an unfiltered way it obtains the lowest precision among all resources because of the overwhelming amount of out-of-domain knowledge it provides. However, with domain-based knowledge filtering—which is obtained simply by setting a domain constant in the DivAPI call—it achieves the highest precision and the second-highest recall for both matchers. We thus present these evaluation results also as an empirical argument for the use of a domain-aware knowledge framework for NLU applications.

7 Related Work

7.1 Domain-Aware Applications

There is ample literature on the use of domain knowledge for domain-specific NLU problems. For NLP and information retrieval, domain-based techniques have been proposed in particular for sense disambiguation tasks [18, 2, 28, 1]. In ontology matching, the AML matcher has used various biomedical ontologies as mediators [6] while SAMBO [17], LogMap, and LogMapBio [16] use online services such as the BioPortal and UMLS. In both application areas, solutions rely on predetermined domain knowledge resources accessed through ad-hoc interfaces. Such solutions could benefit from the pluggability feature of the DivAPI for the extension or added flexibility with respect to knowledge sources.

More recently, there have been attempts to apply *statistical approaches*, such as word embeddings, to domain tasks using domain-specific textual corpora [10, 27, 24]. These efforts all use standard vector models such as *word2vec* or *GloVe* that we are able to exploit by mapping word vectors (embeddings) to concepts of the Diversicon Knowledge Model and serving relevant semantic information (relatedness, similarity) through the weighted methods of the DivAPI. The techniques to train and exploit efficient domain-specific embeddings, however, can be markedly different from the 'vanilla' case of general-purpose corpora due (among others) to corpus sparseness, the potentially much larger size of domain terminologies, and the prevalence of long multiword expressions in domain texts. For this reason, domain-specific NLU applications that adopt a statistical approach often use embeddings combined with domain knowledge [24]. We consider this as an additional application area for the Diversicon framework that provides a unified access to both kinds of resources, keeping in mind nevertheless the wildly differing notion of meaning (concepts vs word vectors) assumed by them.

7.2 Unified Lexico-Semantic Domain Knowledge

The idea of providing unified access to multiple existing knowledge resources has led to a number of large-scale implementations. With respect to lexico-semantic knowl-

edge, past efforts have pursued the integration of wordnets [31, 26]. These resources have remained on the level of common-sense knowledge and have not integrated explicit and comprehensive references to domain categories.

Some more recent efforts have integrated wordnets with encyclopaedic knowledge, e.g., from Wikipedia, such as UBY [15] and BabelNet [19]. A common point of most of these solutions is a focus on cross-lingual knowledge integration, even if the latest versions of both UBY and BabelNet are 'domain-aware' in the sense that they have explicit representations of the domain-relatedness of lexical concepts. Thanks to the breadth and depth of Wikipedia they cover a much larger quantity of domain-specific content.

The *UMLS Metathesaurus* [3] is a continuous large-scale knowledge integration effort specifically for the healthcare domain, and the largest domain knowledge resource we are aware of. The Metathesaurus, as its name suggests, integrates medical thesauri, terminologies, as well as classifications and ontologies with a lexico-semantic focus, and has been maintained for over 20 years. It has been used as an online service providing background knowledge for domain-specific applications such as biomedical ontology matching [17]. It is possible to use the Metathesaurus through the Diversicon framework by implementing its own DivAPI adaptor. However, as is generally the case for large-scale automatically integrated knowledge resources such as the Metathesaurus, but also UBY and BabelNet, automation in building the resource results in limited precision and unavoidably introduces some amount of noise. For applications requiring very-high-quality knowledge, custom integration of hand-picked knowledge sources using the Diversicon KB may be a safer—but admittedly more onerous—approach.

UBY merits a closer look as we also reuse it within the Diversicon KB. In terms of providing a unified, API-based access to large-scale lexico-semantic knowledge for NLP applications, the goals of UBY are similar to ours. There are, however, two main differences. One is in the design and goals of the unified API: the goal of the UBY API is to allow a single access point to the heterogeneous knowledge imported into its database. While we maintain this feature for resources ingested by the Diversicon KB, the DivAPI is intended to harmonise access to a broader spectrum of resources with more diverse underlying knowledge models. The novelty and uniqueness of the DivAPI lies in its domain-awareness and in the ability to provide both a symbolic (knowledge-representation-inspired) and a quantitative (similarity-based) version of the interface, bridging the worlds of symbolic and distributional semantics, and translating knowledge for applications expecting either approach. The second main difference with respect to UBY, as well as to similar efforts such as BabelNet and UMLS, is the focus on the integration *framework* rather than on the integrated content. While UBY provides custom importer logic for resources such as Wikipedia, WordNet, VerbNet, etc., as well as a LMF-based input format, the importing process is not user-friendly and does not seem to consider extensibility as a priority feature. In the Diversicon KB we fixed this shortcoming by providing a robust and validating importing logic.

An alternative approach to representing, storing, and interconnecting lexico-semantic knowledge is the use of the RDF-based *lemon model* [21], which is a simplified and semantic-web-oriented representation of LMF. Lemon-based resources can be linked via standard relations such as `rdfs:equivalentClass` or `rdfs:subClassOf`, augmented using conventional reasoners, and stored in triple stores. We see the support of such an RDF-based knowledge integration as important future work for Diversicon.

8 Conclusions and Future Work

We have presented the open-source *Diversicon* framework that catalogues, integrates, and provides a unified and pluggable access to heterogeneous lexico-semantic knowledge resources for domain-specific applications that require natural language understanding ability. Our motivation was to let these applications access a potentially wide variety of resources through simple service calls to a single API, allowing them to combine forms of knowledge that complement each other but also easily to substitute resources through the life cycle of the application (e.g., in the prototyping stage or to support its evolution).

Domain-awareness, i.e., an explicit and formal support for the notion of *domain*, was one of the prerequisites for the framework. This feature is useful in multi-domain contexts where knowledge from multiple domains or subdomains (including common-sense knowledge) needs to be used simultaneously, but also as a domain-based filtering device for multi-domain knowledge resources. Domain-awareness in Diversicon consists of (a) articulating the

common knowledge model as well as the Diversicon API around domains and the corresponding semantic relations; (b) allowing word senses to be qualified by domains in multiple manners (weighted or unweighted); (c) providing a domain hierarchy as part of the Diversicon API; and (d) providing a knowledge base that is able to ingest and serve large-scale domain resources.

In our example study we successfully used the framework for a comparative evaluation of semantic applications and knowledge resources, demonstrating the simplicity of its use in practice. The API implementations we used in the example study are all freely available from Diversicon's GitHub page.[18]

One important direction in which Diversicon should be extended is the support of a small but important set of linguistic preprocessing features through the API, such as tokenisation, lemmatisation, or stop word detection. These functions enable the subsequent lookup of lexico-semantic information. For domain texts, however, some of these operations may require customised domain-specific approaches due to the potential use of specialised grammar and orthographic conventions. Our plan is to use the Diversicon Catalogue to index relevant domain-specific NLP resources and to support their pluggability through the Diversicon API. Through such NLP functions the API will gain end-to-end NLU support.

Another direction of improvement is the better support of Semantic Web knowledge formats for input, storage, and output. On the import side, support for the *lemon model*, as an RDF-based lexico-semantic representation, is envisaged. *Lemon* is also a good candidate schema for an RDF-based storage backend for the Diversicon KB with a SPARQL interface. Such a solution would extend the possibilities of reasoning over lexico-semantic data and could be exploited more flexibly by standard semantic web applications. Furthermore, while from a theoretical standpoint domain ontologies are not lexico-semantic resources, they are often used as such by NLU applications, and thus the importing of, e.g., OWL *Class* labels as terms and of certain ontological relations such as *subClassOf* or *equivalentClass* would also be a useful addition to the Diversicon KB.

[18] https://github.com/diversicon-kb

References

[1] Bella, G., Giunchiglia, F., McNeill, F.: Language and Domain Aware Lightweight Ontology Matching. Web Semantics: Science, Services and Agents on the World Wide Web **43**(1) (2017)

[2] Bella, G., Zamboni, A., Giunchiglia, F.: Domain-Based Sense Disambiguation in Multilingual Structured Data. In: The Diversity Workshop at the European Conference on Artificial Intelligence (2016)

[3] Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research **32**(suppl_1), D267–D270 (2004)

[4] Donnelly, K.: SNOMED-CT: The advanced terminology and coding system for eHealth. Studies in health technology and informatics **121**, 279 (2006)

[5] Eckle-Kohler, J., McCrae, J.P., Chiarcos, C.: LemonUby–A large, interlinked, syntactically-rich lexical resource for ontologies. Semantic Web **6**(4), 371–378 (2015)

[6] Faria, D., Pesquita, C., Mott, I., Martins, C., Couto, F.M., Cruz, I.F.: Tackling the challenges of matching biomedical ontologies. Journal of biomedical semantics **9**(1), 4 (2018)

[7] Francopoulo, G., George, M., Calzolari, N., Monachini, M., Bel, N., Pet, M., Soria, C.: Lexical markup framework (LMF). In: International Conference on Language Resources and Evaluation-LREC 2006 (2006)

[8] Fumagalli, M., Bella, G., Giunchiglia, F.: Towards Understanding Classification and Identification. In: Proceedings of the 16th Pacific Rim International Conference on Artificial Intelligence (PRICAI) (2019)

[9] Gella, S., Strapparava, C., Nastase, V.: Mapping WordNet Domains, WordNet Topics and Wikipedia Categories to Generate Multilingual Domain

Specific Resources. In: LREC, pp. 1117–1121 (2014)

[10] Ghosh, S., Chakraborty, P., Cohn, E., Brownstein, J.S., Ramakrishnan, N.: Designing Domain Specific Word Embeddings: Applications to Disease Surveillance. arXiv preprint arXiv:1603.00106 (2016)

[11] Giunchiglia, F., McNeill, F., Yatskevich, M., Pane, J., Besana, P., Shvaiko, P.: Approximate structure-preserving semantic matching. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pp. 1217–1234. Springer (2008)

[12] Giunchiglia, F., Yatskevich, M., Shvaiko, P.: Semantic Matching: Algorithms and Implementation. J. Data Semantics **9**, 1–38 (2007)

[13] Gliozzo, A., Strapparava, C.: Semantic Domains in Computational Linguistics. Springer-Verlag Berlin Heidelberg (2009)

[14] González-Agirre, A., Rigau, G., Castillo, M.: A Graph-Based Method to Improve WordNet Domains, pp. 17–28. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). DOI 10.1007/978-3-642-28604-9_2. URL http://dx.doi.org/10.1007/978-3-642-28604-9_2

[15] Gurevych, I., Eckle-Kohler, J., Hartmann, S., Matuschek, M., Meyer, C.M., Wirth, C.: Uby: A Large-Scale Unified Lexical-Semantic Resource based on LMF. In: Proceedings of the 13th EACL Conference, pp. 580–590. Association for Computational Linguistics (2012)

[16] Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-Based and Scalable Ontology Matching. In: The Semantic Web – ISWC 2011, vol. 7031, pp. 273–288 (2011)

[17] Lambrix, P., Tan, H.: SAMBO—a System for Aligning and Merging Biomedical Ontologies. Web Semantics: Science, Services and Agents on the World Wide Web **4**(3), 196–206 (2006)

[18] Magnini, B., Strapparava, C., Pezzulo, G., Gliozzo, A.: Using Domain Information for Word Sense Disambiguation. In: The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, SENSEVAL '01, pp. 111–114. Association for Computational Linguistics, Stroudsburg, PA, USA (2001)

[19] Maud Ehrmann et al.: Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.,

[20] McCoy, R.T., Pavlick, E., Linzen, T.: Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. arXiv preprint arXiv:1902.01007 (2019)

[21] McCrae, J., Spohr, D., Cimiano, P.: Linking lexical resources and ontologies on the semantic web with lemon. In: Extended Semantic Web Conference, pp. 245–259. Springer (2011)

[22] McNeill, F., Gkaniatsou, A., Bundy, A.: Dynamic data sharing for facilitating communication during emergency responses. In: ISCRAM (2014)

[23] Monachini, M., Quochi, V., Del Gratta, R., Calzolari, N.: Using LMF to shape a lexicon for the biomedical domain. LangTech Proceeding, Rome (2007)

[24] Nooralahzadeh, F., Øvrelid, L., Lønning, J.T.: Evaluation of Domain-specific Word Embeddings using Knowledge Resources. In: Proceedings of LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (2018)

[25] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2009)

[26] Pianta, E., Bentivogli, L., Girardi, C.: MultiWordNet: developing an aligned multilingual database. In: Proceedings of the First International Conference on Global WordNet, pp. 21–25 (2002). URL http://multiwordnet.fbk.eu/paper/MWN-India-published.pdf

[27] Pilehvar, M.T., Collier, N.: Improved Semantic Representation for Domain-Specific Entities. In: Proceedings of the 15th Workshop on Biomedical Natural Language Processing, pp. 12–16 (2016)

[28] Rotella, F., Ferilli, S., Leuzzi, F.: A Domain-Based Approach to Information Retrieval in Digital Libraries. In: Italian Research Conference on Digital Libraries, pp. 129–140. Springer (2012)

[29] Toral, A., Monachini, M., Soria, C., Cuadros, M., Rigau, G., Bosma, W., Vossen, P.: Linking a domain thesaurus to WordNet and conversion to WordNet-LMF. In: Proceedings of Second International Conference on Global Interoperability for Language Resources (ICGL2010). Hong Kong (2010)

[30] Trier, J.: Der deutsche Wortschatz im Sinnbezirk des Verstandes: die Geschichte eines sprachlichen Feldes. 1. von den Anfängen bis zum Beginn des 13. Jahrhunderts. Winter (1931)

[31] Vossen, P. (ed.): EuroWordNet: A Multilingual Database with Lexical Semantic Networks. Kluwer Academic Publishers, Norwell, MA, USA (1998)

[32] Vulić, I., Ponzetto, S.P., Glavaš, G.: Multilingual and cross-lingual graded lexical entailment. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, pp. 4963–4974 (2019)

[33] Wright, S., Budin, G.: Handbook of Terminology Management. Application-oriented Terminology Management. J. Benjamins (2001). URL `https://books.google.it/books?id=UYm7XvBXm7QC`

[34] Zwicklbauer, S., Seifert, C., Granitzer, M.: Robust and collective entity disambiguation through semantic embeddings. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 425–434. ACM (2016)