Selecting and Tailoring Ontologies with JOYCE

Erik Faessler[1⋆], Friederike Klan[2⋆], Alsayed Algergawy[2], Birgitta König-Ries[2], and Udo Hahn[1]

[1] Jena University Language & Information Engineering (JULIE) Lab
[2] Heinz-Nixdorf Endowed Chair for Distributed Information Systems
Friedrich-Schiller-Universität Jena, Jena, Germany

firstname.lastname@uni-jena.de

Abstract. We present JOYCE, a scalable tool for identifying and assembling relevant (pieces of) ontologies from a repository of source ontologies, thus enabling the effective and efficient reuse of formalized domain knowledge. JOYCE includes a conceptual filter to identify relevant classes, minimizes unintended redundancies, i.e. concept duplicates, and excludes knowledge considered irrelevant for the specific conceptual design task.

Keywords: ontology selection, ontology modularization, ontology reuse

1 Introduction

The rise of the Semantic Web has spurred the development of a plethora of ontologies for diverse application domains. This is particularly evident in the life sciences where portals such as BIOPORTAL serve as repositories for hundreds of domain-specific ontologies. In the light of the considerable efforts invested in the creation of these pieces of knowledge, it is desirable that these precious resoures are not just shared, but actually reused in whole or parts as a knowledge base for future applications. However, identifying relevant ontologies for a specific application and adapting them to ones own needs is a challenging task requiring a tremendous amount of time, intellectual effort and expertise. In this demo, we showcase the **J**ena **O**ntolog**Y** **C**ustomization **E**ngine (JOYCE), a scalable tool that alleviates these problems by supporting knowledge engineers in selecting and tailoring ontologies for reuse. It improves on the ONTOLOGY RECOMMENDER provided on BIOPORTAL by enabling the recommendation of ontology modules, by minimizing irrelevant and duplicate knowledge and by being more scalable in terms of the number of ontologies that can be combined.

2 Tool Overview

Input to the system can either be a set of keywords deemed relevant by domain experts or indicative paragraphs extracted from relevant scientific publications

⋆ Joint First Authors

The final publication is available at Springer via https://link.springer.com/chapter/10.1007/978-3-319-58694-6_12.

(which undergo shallow linguistic analysis). Based on such an interest profile, JOYCE matches keywords with concept names to identify relevant (portions of) ontologies and tries to determine optimal combinations of these. It thereby aims at (a) maximizing the number of covered keywords and at minimizing (b) irrelevant and (c) duplicate knowledge. Objective (a) ensures that JOYCE assembles sets of subontologies that jointly form a reasonable basis for a knowledge base fitting the interest profile, (b) avoids superfluous representation structures in the resulting knowledge base, and (c) decreases the risk of logical inconsistencies due to incompatible conceptual models of the same concept. The latter condition ensures that the effort for integrating the ontologies or parts of them into a logically consistent knowledge base (not yet supported) by aligning semantically equivalent concepts is kept as low as possible. The importance of the criteria (a), (b) and (c) can be adjusted.

Ontology Repository and Preprocessing. We use BIOPORTAL as our base repository since it offers a wide range of (519, as of October 2016) ontologies. In order to create a local repository for a particular build of the tool we download all ontologies from BIOPORTAL. Since BIOPORTAL also hosts private ontologies, not available for public download, and conversion or parsing problems occurred, the maximum number of ontologies decreases substantially. In total, our local repository currently contains 323 ontologies. From these ontologies, we generate a dictionary of concept names for all ontology classes. This dictionary serves as an index of class names and all their synonyms as they appear in the ontologies. At the beginning of the selection process, the user-provided terms are matched with this index to determine matching candidate ontologies for the subsequent assembly process. Ontologies with an input coverage of zero are discarded.

Ontology Modularization. Ontology modularization addresses the problem of identifying portions of an ontology. The identification of a single portion given the profile terms is called *module extraction*, whereas the process that modularizes the ontology into a set of partitions is called *ontology partitioning*. While the former finds a subset of an ontology that is relevant to the terms from the interest profile, the latter reflects ontological connectivity criteria. JOYCE is able to handle both types of modules. In its current version, it incorporates a logic-based module extractor [2] (locality-based modules) and a partitioning-based modularizer [1] (partitioning-based modules). Since the latter do not depend on the interest profile, partitioning is done once when bootstrapping the system. The resulting modules are stored in a repository similar to the original ontologies. If extraction-based modules shall be combined, the tool first identifies relevant ontologies (without modifications) and then extracts locality-based modules from these relevant ontologies for subsequent assembly.

Optimization. During the assembly of candidate ontologies or ontology modules, JOYCE uses the following three optimization criteria to measure the fitness of a certain combination of ontologies or ontology modules: the proportion of irrelevant concepts (*overhead*)[3], the fraction of concepts appearing in two or more

[3] Modules shall comprise relevant concepts only, but often include irrelevant classes.

ontologies (*overlap*) and *term coverage*, measuring the fraction of input terms for which a matching concept was found, all relative to the number of input terms. These criteria comply with the objectives (a) – (c) discussed above. The overall goal is to maximize coverage and minimize overlap and overhead.

Identifying Optimal Combinations. We determine sets of ontology modules that are *pareto-optimal* w.r.t. the criteria coverage, overhead, and overlap. This means that they are not dominated by other solutions that are better with respect to at least one criterion and equally well with regard to the others. We also take preferences regarding the importance of these criteria into account. A knowledge engineer might, e.g., indicate that coverage and overlap are equally important. Yet, both are less important than overhead. Since computing and evaluating all possible combinations of ontology modules is infeasible (the problem is a variant of the well-known set cover problem), we use a greedy algorithm to assemble promising sets in an incremental way. We start with singleton sets of ontology modules (one for each candidate module) and add another ontology module from the candidate modules during each round. In each step, we remove those sets that are dominated by others and focus on those that fit the given preferences. To avoid the generation of a huge number of solutions, we consider just a random sample of the ontology modules that might potentially be combined with the candidate module sets determined in the previous step. The sampling process accounts for the potential of a candidate module to maximally improve the candidate sets with respect to the optimization criteria when added and also accounts for the chosen preferences. The sample size can be configured. We consider a newly discovered combination, if it is better w.r.t. at least one criterion (since otherwise it is dominated by the original set), and if its coverage increases compared to the original set. Newly encountered combinations are added to the pool of candidate sets which are input to the next round. The algorithm stops, if no new combinations are encountered or a given maximum number of iterations is reached. The same procedure works for entire ontologies.

3 User Interface and Demonstration

Joyce comes with a Web interface based on Apache Tapestry 5, which is divided into two parts: a configuration section (Fig. 1 upper part) and a results section (Fig. 1 lower part). In the configuration section, the user enters a set of keywords or a text fragment that describes the area of interest. The maximum number and the type of objects to be combined (either entire ontologies, extraction-based modules or partition-based modules), the size of the random sample taken in each assembly round, and the importance of each of the optimization criteria, *coverage*, *overlap* and *overhead*, can be adjusted appropriately. Preference values can be *low*, *medium* or *high*. The results view provides a list of the ontology (module) combinations suggested by Joyce, sorted by coverage. For each combination, a list of the assembled ontologies or modules, their number, as well as specific values for coverage, overhead and overlap for each

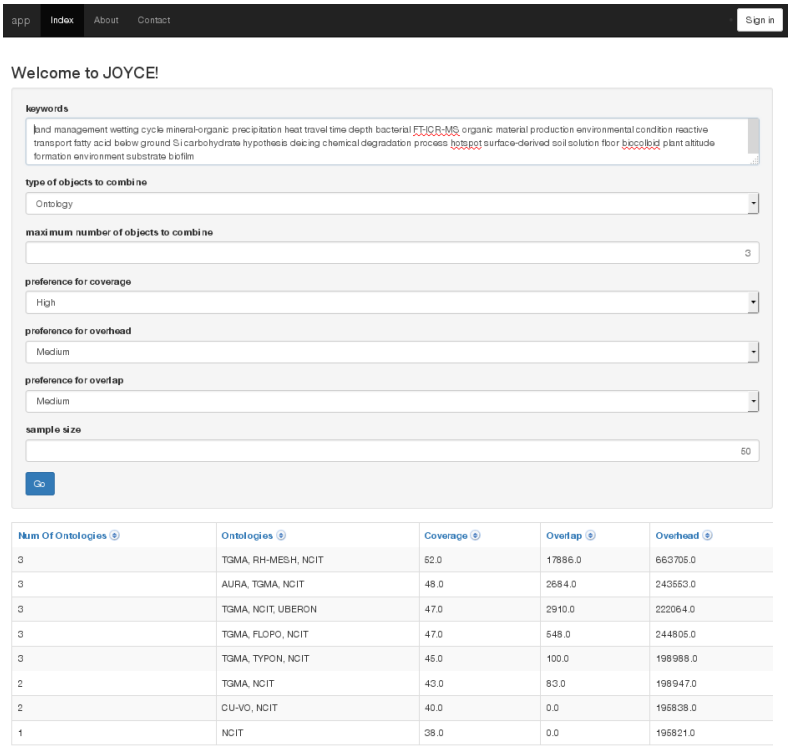

Fig. 1. JOYCE – Graphical User Interface

combination are displayed, thus informing the knowledge engineer about the impact of the criteria applied for tailoring. In the demo (available online on the JOYCE Web page), we will briefly introduce the user interface and the configuration parameters of JOYCE. The system will be run with various inputs and different parameter configurations to demonstrate their effects on the suggested combinations. In particular, we will change the types of objects to be combined to reveal the implications for overhead. We will gradually increase the number of objects to be assembled and look how this affects coverage. We will showcase that JOYCE can even deal with a large number of input terms and objects. Finally, we play with different preference settings for coverage, overlap and overhead, and investigate how these parameters influence the suggested combinations.

Acknowledgments. The work has been partly funded by the *Deutsche Forschungsgemeinschaft (DFG)* as part of the CRC 1076 AQUADIVA.

References

1. Algergawy, A., Babalou, S., Klan, F., König-Ries, B.: OAPT: A tool for ontology analysis and partitioning. In: Intl. Conf. on Extending Database Technology. pp. 644–647 (2016)

2. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: Intl. World Wide Web Conference. pp. 717–726 (2007)