OpenBudgets.eu: Fighting Corruption with Fiscal Transparency

Project Number: 645833 Start Date of Project: 01.05.2015 Duration: 30 months

Deliverable 1.9

Linking code lists to external datasets

Dissemination Level	Public
Due Date of Deliverable	Month 10, 29.2.2016
Actual Submission Date	22.03.2016
Work Package	WP 1, Data Structure Definition for Budgets and Public Spending
Task	T 1.3
Type	Demonstrator
Approval Status	Final
Version	1.0
Number of Pages	17
Filename	D1.9 Linking code lists to external datasets.docx

Abstract: This deliverable demonstrates how code lists used in budget and spending data can be linked to third-party datasets. The main outcome of this effort comprises the produced sets of links along with the linkage rules used to generate the links. The accompanying text expounds the motivation of linking and describes the automatic approaches used for linking the considered code lists. Most of the links were produced via SPARQL 1.1 Update operations using exact matches on codes of code list concepts. We also experimented with the Silk link discovery framework, various approaches to data pre-processing, and automated translation. Additionally, we link several code lists manually and reuse existing links for others. Quality of links generated via automatic fuzzy heuristics is manually evaluated by domain experts using a sample of links.

Project funded by the European Union's Horizon 2020 Research and Innovation Programme (2014 – 2020)

History

Version	Date	Reason	Revised by
0.1	04.03.2016	Version for internal review	Jindřich Mynarz
0.2	21.03.2016	Version for external review	Nicolas Kayser-Bril
1.0	22.03.2016	Final version for submission	Jakub Klímek

Author List

Organisation	Name	Contact Information
OKGR	Lazaros Ioannidis	larjohn@gmail.com
UEP	Jakub Klímek	klimek@opendata.cz
IAIS	Fathoni Musyaffa	fathoni.am@gmail.com
UEP	Jindřich Mynarz	mynarzjindrich@gmail.com
UEP	Lucie Sedmihradská	sedmih@vse.cz
UEP	Jaroslav Zbranek	zbranek.jaroslav@gmail.com

Executive Summary

This deliverable follows up on D1.7 (Karampatakis et al., 2015) and links the previously extracted code lists with external datasets. The result of this work is a set of publicly available linksets. We created links between 16 pairs of datasets amounting to the total of 20 975 links. The generated linksets along with linkage rules used to produce them are available from https://github.com/openbudgets/linksets. In most cases, linking was based on exact matches over shared codes from code lists and was automated via SPARQL 1.1 Update operations. However, we experimented also with more sophisticated approaches including fuzzy matching and automatic translation. We discovered that the main bottleneck for linking large datasets is the difficulty of loading data from RDF stores to link discovery tools. To avoid this issue, we used data materialization and executed the linking procedures directly in RDF stores. These efforts typically resulted in partial alignments between code lists, which made them unfit for automated data migration. Hence, we concluded that the main value of linked data lies in data enrichment for human consumers interactively exploring the linked datasets.

Abbreviations and Acronyms

API	Application Programming Interface
CL-GEO	Geographical Standard Code List
CPA	Statistical classification of products by activity
ETL	Extract Transform Load
GML	Geography Markup Language
IRI	Internationalized resource identifier
KML	Keyhole Markup Language
NACE	Statistical Classification of Economic Activities in the European Community
NUTS	Nomenclature of Territorial Units for Statistics
RDF	Resource Description Framework
SKOS	Simple Knowledge Organization System

Table of Contents

1 Introduction

Thanks to various open data initiatives that arose in the recent years there is a plethora of openly available datasets that can be reused without seeking a permission. A large share of these datasets is available in RDF, making it feasible to link the data directly without prior pre-processing, since the entities in the data are already identified via IRIs that can be directly linked. We surveyed the available linked open datasets and chosen several of them that exhibit overlaps with the code lists extracted for the deliverable 1.7 (Karampatakis et al., 2015). Subsequently we prioritized linking datasets that promise to add more value to the code lists, offsetting their value be the complexity of linking them. We also link the previously extracted code lists among themselves. Before we describe how we linked the chosen datasets, let us review our motivation to do so.

1.1 Motivation

Linking makes code lists comparable. In turn, if we make the relations between code list concepts explicit, it improves comparability of the datasets described by the concepts. For example, if code list concepts are marked as equivalent, they can be used interchangeably. In this way, data described by linked concepts can be converted to a more homogeneous representation, which eases its use in combination. In the terminology of ETL this process is also known as data migration.

Linking can also enrich code lists with external data. Linked data may put the code lists in context, such as by providing population counts for municipalities governed by budgetary units, or enable new uses of budget data, such as by linking geographic code lists to geospatial data with geometries that make map visualizations possible. The need for enrichment is particularly apparent in code lists. Descriptions of code list concepts are typically terse. Code lists usually contain only labels with codes, in some cases organized in hierarchical relations. Definitions, scope notes, related links, and the like are often missing in code lists. Encyclopaedic data, in particular, provides this kind of data. Viewed from this perspective, DBpedia,[1] for instance, appears to be a good linking target, since it contains such data extracted from Wikipedia. Moreover, since DBpedia is multilingual, it can provide labels in several languages to concepts from monolingual code lists. Due to these reasons DBpedia is one of the datasets included in the following motivating example.

1.1.1 Motivating example

To illustrate the benefits of linking code lists we show an example of linking the CL-GEO code list concept for the Ústecký region from the Czech Republic (IRI `http://data.openbudgets.eu/resource/codelist/cl-geo/CZ042`). This is one of the regions in which the OpenBudgets.eu project will track the use of EU funds. Using the data from the CL-GEO code list we know the label of this region, its NUTS code[2], and its level in the NUTS hierarchy. If we link this resource to other datasets, we can discover more data.

Linking the concept for Ústecký region to LinkedGeoData (IRI `http://linkedgeodata.org/triplify/relation442452`) provides access to translations of the region's label. For example, we learn that the German name of the region is "Region Aussig". An example use of these translations is to display them in localized visualizations. Following the links in LinkedGeoData[3] we can retrieve the geographic geometry

[1] http://dbpedia.org

[2] https://en.wikipedia.org/wiki/Nomenclature_of_Territorial_Units_for_Statistics

[3] More precisely, SPARQL 1.1 property path `lgdo:members/rdf:rest*/rdf:first/lgdo:ref/geom:geometry/ogc:asWKT`.

of this region in the WKT format.[4] The geometry can be used in map visualizations to render the polygon representing the region (see Figure 1).

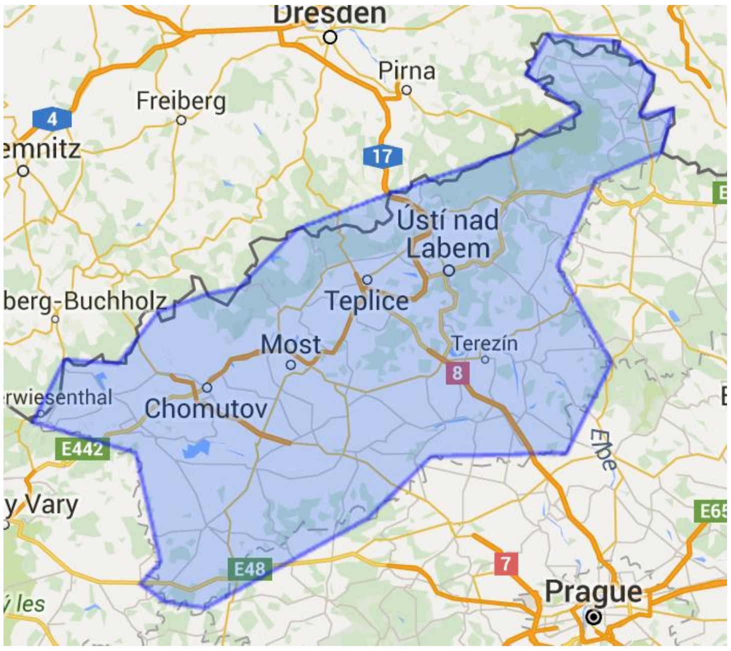

Figure 1: Simplified geometry of the Ústecký region

Link to LinkedStatistics' Geo code list (IRI `http://eurostat.linked-statistics.org/dic/geo#CZ042`) yields no new data, but serves as an alignment that makes datasets described by the linked code lists better comparable, which is true for other links as well.

Link to the NUTS Geovocab (IRI `http://nuts.geovocab.org/id/CZ042`) has similar benefits as the link to LinkedGeoData. This dataset can provide us with the geographic geometry of the region (in RDF, GML, or KML). Additionally, it gives us links to other datasets, such as the European Environment Information and Observation Network (IRI `http://rdfdata.eionet.europa.eu/ramon/nuts2008/CZ042`), which can be traversed further to discover more data.

Link to DBpedia (IRI `http://dbpedia.org/resource/Ústí_nad_Labem_Region`) provides us with useful contextual data including the region's area, population, or links to related resources. Example facts about Ústecký region retrieved from DBpedia are shown in Table 1. This type of data can be used to frame analyses of budget or spending into the local context. For example, population count can be used to compute spending per capita. However, when reusing data from DBpedia, we should carefully examine its reliability. Since this is a dataset derived from the community-created Wikipedia, it may contain errors or outdated figures. A better source for the demographic data would be Eurostat, but it has not released its data in a linkable format, even though there are several unofficial wrappers that provide its data as linked data, such as the one produced by the LATC project.[5] For instance, if we compare the population count from DBpedia with data from the 2011 census carried out by the Czech Statistical Office, we find a potential discrepancy, since the official population as of 2011 was 808 961.

[4] https://en.wikipedia.org/wiki/Well-known_text

[5] http://eurostat.linked-statistics.org

Property	Value
Area in km^2	5334.52
Leader name	Oldřich Bubeníček
Population	852554
Website	http://www.kr-ustecky.cz/

Table 1: Example facts about Ústecký kraj retrieved from DBpedia

Before we proceed with describing how the links from the example were generated, we cover the linking properties that were used to capture the semantics of the generated links.

1.2 Linking properties

Simple Knowledge Organization System (SKOS)[6] offers several linking properties that cover diverse kinds of relations between concepts from knowledge organization systems, such as code lists. These linking properties include `skos:closeMatch`, `skos:exactMatch`, `skos:broadMatch`, `skos:narrowMatch` and `skos:relatedMatch`.[7]

`skos:broadMatch` and `skos:narrowMatch` indicate a hierarchical relation between the linked resources. They are defined as subproperties of `skos:broader` and `skos:narrower` respectively and can be considered as their analogues for use between datasets. Similarly, `skos:relatedMatch` is an analogue of `skos:related` for associative links between datasets. `skos:exactMatch` and `skos:closeMatch` express equality relation. While `skos:exactMatch` is a transitive property that indicates that the resources it links can be used interchangeably, `skos:closeMatch` in an intransitive property with a weaker equality semantics that indicates interchangeability only in some applications. We used `skos:exactMatch` in cases when one-to-one links were discovered, whereas `skos:closeMatch` was used if multiple links for the same resource were found or if the linking was done in a fuzzy manner, possibly indicating an approximate alignment.

With the preliminaries covered we now turn to a discussion of the generated links.

2 Linked datasets

We paired the datasets in tables 2 and 3 and attempted to discover links between the pairs. Some of these datasets are the code lists extracted in deliverable D1.7, while others are external. There are several well-known linked datasets, including DBpedia and LinkedGeoData. A significant target of the generated links are the code lists from the Metadata Registry[8] maintained by the EU Publications Office, including code lists for currencies or budget amount statuses. For each dataset we list its name by which it is referred to further in this text, the URL where it can be found, and an indication whether the dataset is external to OpenBudgets.eu or not.

[6] https://www.w3.org/TR/skos-reference/

[7] http://www.w3.org/TR/skos-reference/#mapping

[8] http://publications.europa.eu/mdr

Dataset	URL
Central Product Classification (CPC)	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/cpc
Classification of Products by Activity (CPA)	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/cpa
Geographical Standard Code List (CL-GEO)	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/cl_geo
ESA 2010 Financial assets	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/ESA2010_financial_assets
ESA 2010 Transactions in financial assets and liabilities	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/ESA2010_assets_and_liabilities
ESIF funds	https://github.com/openbudgets/datasets/blob/master/ESIF/2014/codelists/esif-funds.ttl
ESIF member states	https://github.com/openbudgets/datasets/blob/master/ESIF/2014/codelists/esif-member-states.ttl
ESIF programs	https://github.com/openbudgets/datasets/blob/master/ESIF/2014/codelists/esif-program.ttl
OBEU budget phases	https://github.com/openbudgets/data-model/blob/master/budget/budget-codelists.ttl
OBEU currencies	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/currencies
OENACE	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/skosified/oenace/oeanace.ttl
PRODCOM list (list of products of the European Community)	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/prodcom
STAKOD	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/stakod-greek
Standard International Trade Classification (SITC)	https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified/sitc

Table 2: Linked internal datasets

Dataset	URL
DBpedia	http://dbpedia.org
EU budget amount status	http://publications.europa.eu/mdr/authority/eu-budget-amount-status/index.html
EU currencies	http://publications.europa.eu/mdr/authority/currency/index.html
EU programs	http://publications.europa.eu/mdr/resource/authority/eu-programme/skos/eu-programme-skos.rdf
LinkedGeoData	http://linkedgeodata.org
LinkedStatistics GEO	http://eurostat.linked-statistics.org/dic/geo
NUTS Geovocab	http://nuts.geovocab.org

Table 3: Linked external datasets

Links between these datasets were created either automatically or manually. In several cases we reused existing links and converted them to RDF.

3 Automatic linking

Automatic linking compares the descriptions of candidate resource pairs and determines if their similarity exceeds a threshold that is required for the pair to be linked. Similarity can be based on exact or fuzzy matches. For example, if 2 resources share the same code, they can be considered equal. This is often the case for code lists, which can be linked via notations of their concepts. We used linking via exact match of concept notations (attached via the `skos:notation` property) in most of the linking tasks described in this deliverable.

If shared concept notations were not available in the interlinked datasets, we resorted to linking via labels in natural language. This approach is fraught with numerous issues as labels are not necessarily unique identifiers of the concept they label and their specificity is low (Mynarz, 2012). Labels can be ambiguous due to homonymy, or there can be multiple synonymous labels for the same concept. Unfortunately, code lists rarely offer more than labels for their concepts and linking code lists is thus inherently imprecise. Hierarchical organization of code lists can help provide some context for ambiguous labels but it requires the broader concepts to have already been matched, i.e. iterative processing would be needed.

We used SPARQL 1.1 Update[9] and Silk link discovery framework[10] for automatic linking. Additionally, we experimented with OpenRefine's[11] reconciliation extension to explore the feasibility of diverse data pre-processing options.

[9] https://www.w3.org/TR/sparql11-update

[10] http://silkframework.org

[11] http://openrefine.org

In most cases we created links based on shared codes, which allowed us to link data simply by exact joins over the codes. For example, LinkedGeoData was linked to CL-GEO via common NUTS codes. First, we extracted the resources with NUTS codes from LinkedGeoData and then we linked them using the SPARQL Update operation listed in Example 1. The list of dataset pairs for which we performed automated linking is shown in Table 4.

```
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>


INSERT {
  GRAPH <http://data.openbudgets.eu/resource/linkset/cl-geo-to-linkedgeodata> {
    ?source skos:exactMatch ?target .
  }
}
WHERE {
  GRAPH <http://data.openbudgets.eu/resource/codelist/cl-geo> {
    ?source a skos:Concept ;
      skos:notation ?code .
  }
  GRAPH <http://linkedgeodata.org> {
    ?target lgdo:ref%3ANUTS ?code .
  }
}
```

Example 1: SPARQL Update operation generating links via shared codes

Source	Target	Linking predicate	# of links
CL-GEO	NUTS Geovocab	`skos:exactMatch`	1407
CL-GEO	LinkedGeoData	`skos:exactMatch`	175
CL-GEO	LinkedStatistics GEO	`skos:exactMatch`	1844
CL-GEO	DBpedia	`skos:closeMatch`	1090
CL-GEO	ESIF member states	`skos:exactMatch`	30
CPA	DBpedia	`skos:exactMatch`, `skos:narrowMatch`	2806
ESIF programs	ESIF funds	`skos:broadMatch`	677
EU currencies	OBEU currencies	`skos:exactMatch`	160

Table 4: Datasets linked automatically

3.1 Linking CPA to DBpedia

While DBpedia can provide rich encyclopaedic data, linking it is difficult due to its size and messiness. We decided to link the CPA 2008 code list to DBpedia. We used the October 2015 dumps of DBpedia.[12] Due to the size of DBpedia we limited the datasets loaded into a local mirror to those that were needed for the linking task. In particular, we used the English labels, disambiguation links and redirects.

Due to the generality of CPA it is not possible to restrict DBpedia to a narrower subset, e.g., via classes. Since instances of most classes in DBpedia can be potential matches for CPA concepts, we need to load all the resources from DBpedia. This amounts to 12 million triples for the October 2015 version of DBpedia. Additionally, we used 14M triples of disambiguation links and 7M triples of redirects. Using the disambiguation links we removed ambiguous resources from the dataset. Labels of redirects were added to their target resources and the redirecting resources without labels were deleted. Since we found out that acronyms tend to be ambiguous, we used a heuristic that removed all resources labelled with a term made of 2 or more uppercase characters. This pre-processing left us with a 11M triples dataset.

Labels in DBpedia are ambiguous. Some ambiguous labels are distinguished explicitly via disambiguation links. In some cases, DBpedia labels are distinguished only by character case. For instance, DBpedia contains resource labelled "Coins" (http://dbpedia.org/resource/Coins, a redirect for "Coin"), "COinS" (http://dbpedia.org/resource/COinS, ContextObjects in Spans), and "COINS" (http://dbpedia.org/resource/COINS, Combined Online Information System). Existence of such resources limits the use of normalization of labels. For example, lower-casing such labels would lead to false positives, since all these resources would be linked to CPA concept "Coins" (http://data.openbudgets.eu/resource/codelist/cpa/32.11).

Runtime of linking to DBpedia via Silk is dominated by the data loading time. While Silk allows to parallelize the linking execution by setting the number of threads it uses, loading data from RDF stores tends to be a bottleneck. The fundamental problem of loading data having the size of DBpedia is that retrieving data chunked to smaller subsets requires stable ordering over the whole dataset. Sorting a large set of triples is a computationally expensive operation. We experimented with 2 RDF stores, OpenLink's Virtuoso[13] and Blazegraph,[14] for local mirrors of DBpedia. However, even on a strong server (64 GB RAM) we experienced out-of-memory errors and other failures when loading DBpedia for linking. Both of the tested RDF stores became unstable under heavy load and frequently halted. Therefore, we had to abandon this approach and come up with a computationally feasible solution.

We used semi-automatic linking to explore what approach can be used to link CPA to DBpedia. In this case, we employed OpenRefine with its RDF extension to pre-process CPA and reconcile it with DBpedia using its English labels. The adopted semi-automated approach consisted of automated reconciliation followed by manual refinement of the proposed alignments.

Prior to reconciliation we normalized the labels from CPA to improve the recall of string matching. We also removed the words that can be considered as "stop words" in the context of CPA, such as "services" used as a suffix, since these words do not distinguish the CPA's terms. For this experiment we limited the linked CPA concepts to the 3 topmost levels in the CPA's hierarchy (371 concepts) to avoid overly specific concepts, such as *"Undifferentiated services produced by private households for own use"*, which are less likely to yield a match in DBpedia.

The CPA code list contains compound concepts, such as *"Coal and lignite"*, while Wikipedia, and thus DBpedia in turn, instead contains basic-level categories such as *"Coal"* and *"Lignite"*.

[12] http://vmdbpedia.informatik.uni-leipzig.de/2015-10/core-i18n/en

[13] http://virtuoso.openlinksw.com

[14] https://www.blazegraph.com

This mismatch makes linking more difficult. To solve this problem, we split the CPA labels containing terms joined by conjunctions and reconciled the constituent terms separately. The links from the compound concepts in CPA were typed as `skos:narrowMatch`. For example, *“Coal”* from DBpedia is a narrower concept to *“Coal and lignite”* from CPA.

Following the approach tried using OpenRefine we converted the linking procedure into a series of SPARQL 1.1 Update operations so that its execution could be automated. Instead of using on-the-fly normalization we materialized the normalized versions of CPA and DBpedia labels as objects of `skos:hiddenLabel`. However, due to limited expressivity of SPARQL we could not completely replicate the character normalization and tokenization that was feasible in OpenRefine.

We created 2 types of links for this pair of datasets. We used `skos:closeMatch` for matches based on complete labels of CPA concepts, while `skos:narrowMatch` was used for matching tokens in compound labels. In case of exact matches, we post-processed the generated links via SPARQL Update to keep only the links from the most specific CPA concepts if they had parents sharing the same label that were linked to the same DBpedia resources. This step was motivated by a common guideline in subject indexing that prescribes to use the most specific concept available.

The runtime of this task was dominated by execution of SPARQL Update operations used for data pre-processing. However, it was still several orders of magnitude faster than loading data to link to Silk. The execution time of the actual linking was negligible compared to the duration of data pre-processing. Since all the necessary data was already transformed into the desired format and materialized, linking only needs to perform a join, which is a well-optimized operation in RDF stores.

3.2 Linking CL-GEO to DBpedia

We also linked the CL-GEO code list to DBpedia, using its pre-processed version of the latter described above. We formalized the linking procedure using SPARQL Update operations. The links were created by matching the labels to CL-GEO concepts. The resulting links were subsequently post-processed. If several CL-GEO concepts linking to the same DBpedia resource were found, we retained the links from capital cities denoted by codes ending with “_CAP” and removed others. This decision was motivated by observing that DBpedia usually describes capitals instead of the regions with the same name. Following similar reasoning we preferred the links to the most specific CL-GEO concepts and removed the duplicated links connecting their parent concepts.

3.3 Automated translation

When two code lists do not share a common element, similarities can be searched across the labels that describe each term. As term description words may be ordered differently, the similarity checking tool can tokenize each term to words that are then compared separately. While tokenization may work in some cases, it usually increases the rate of false positives, so that it should not be recommended across the board. In the worst case, the descriptions of two similar terms might be expressed with completely different yet synonymous words. While string similarity cannot discover this kind of similarity, other tools may be used, as long as they can semantically analyse and compare two terms.

Such issues may arise in the common case of the automatic linking of two code lists expressed in different languages. To automate the creation of links for these code lists, a cross-language semantic similarity engine would ideally “understand” and compare the meaning of each term. Using a more straightforward approach, one of the code lists should be first translated into the language of the other. Afterwards, the process described previously would follow: either string similarity or single-language semantic comparison.

Apart from the different wording issue, the translation part of the process can further deteriorate the result of the automatic linking process because the translator is not always able to understand the semantic context of the terms and introduces new, foreign concepts in translation.

For experimentation purposes, a Microsoft Translator plugin for the UnifiedViews ETL framework[15] was developed in this deliverable. The Microsoft Translator API was selected, among other reasons, because of the 2 million characters free tier offering. A quick evaluation was performed, using the NACE code list and its Greek counterpart (STAKOD). The two code lists are actually already linked, as there is a corresponding term in STAKOD for each NACE term. This fact offers a testbed in order to measure how many term pairs would be recognised similar, just by translating the Greek terms and then comparing them to the English ones, without taking into account the existing links.

Using a simple setup in Silk, with string equality comparison, 417 out of the 997 terms found in NACE, were found similar to their counterparts in STAKOD. We experimented with fuzzy string distance metrics, such as the Levenshtein distance, and used more tolerant similarity thresholds, but doing so led to an increase of the false positives rate without better recall of true positives. The failure to discover the rest of the links from the testbed can be attributed to a variety of factors. First, the quality of the translation is not always predictable. Despite that the translation of the mostly noun-based terms was not of such low quality and could be characterized as moderate to good. The second factor is that there are terms that are translated into expressions with words synonymous to the ones in the target code list. For instance, *“manufacture of plastic packaging items”* means the same as *“manufacture of plastic packaging goods”* but the terms are treated as dissimilar using string equality. Increasing the similarity threshold using fuzzy distance metrics reveals more of the expected links but also introduces many false positives, due to the nature of the code lists – many child terms differ from their parent terms only by a few words.

The third and rare factor is the difference caused by local spelling variations (e.g., homogenised vs. homogenized), which might be solved by selecting the appropriate locality in the translation API calls in the first place, given that the code lists use spelling consistently throughout.

An additional linking method could also utilize a semantic similarity comparison to cover the terms for which no matches were found. We tried the online demo of Dandelion.eu[16] to assess the performance of such solutions. The goods vs. items example mentioned before scored 92 % similarity. A falsely similar term *“manufacture of plastics products”* scored 86 %. In that case, the selection of the correct term to link might be based on the highest score. The highest score would be meaningful if it is actually higher than a threshold. For instance, a highest score of 60 % would not be considered high enough to produce a link. Nevertheless, the complexity of this solution is the main reason it is considered out of the scope of this deliverable and remains to be a future research topic.

4 Manual linking

Manual linking was used for small code lists, for which the overhead associated with linking automation would not be justified.

Source	Target	Linking predicate	# of links
EU budget amount status	OBEU budget phases	`skos:broadMatch`	4
ESIF programs	DBpedia	`skos:exactMatch`	4

[15] http://www.unifiedviews.eu

[16] https://dandelion.eu

ESIF EU programs	EU programs	`skos:exactMatch`	6
ESIF EU funds	EU programs	`skos:exactMatch`	6

Table 5: Datasets linked manually

5 Reused links

We reused the mappings provided by Eurostat for several pairs of code lists. For example, we used the CPA-CPC mapping tables.[17] The mappings are available in CSV, so we converted them to RDF using OpenRefine's RDF extension.[18]

Source	Target	Linking predicate	# of links
Central Product Classification (CPC)	Classification of Products by Activity (CPA)	`skos:narrowMatch`	3851
ESA 2010 Financial assets	ESA 2010 Transactions in financial assets and liabilities	`skos:exactMatch`	38
PRODCOM	Classification of Products by Activity (CPA)	`skos:exactMatch`	5567
Standard International Trade Classification (SITC)	Classification of Products by Activity (CPA)	`skos:narrowMatch`	3310

Table 6: Datasets, for which existing links were reused

6 Evaluation

We performed manual validation of a sample of the links produced using automated heuristics. We chose to evaluate the links generated between CPA and DBpedia, since in effect it employed fuzzy matching, albeit its approximate nature was caused by normalization done during data pre-processing instead of the use of fuzzy matching techniques during runtime.

The generated links were evaluated by 2 domain experts. Each of the experts received a random sample of 200 links and was asked to mark the links either as correct or incorrect, or skip the judgement if the correctness of a link cannot be determined. A half of these links was shared among the experts and the other half was unique per expert. We involved 2 domain experts to evaluate the correctness of links. The experts worked independently, so that we could examine the consistency of their evaluation of the shared links.

We used precision to evaluate the quality of the generated links. Precision was computed as the ratio of true positives to all positives. Measures that take true negatives into account, such as accuracy, are not suitable for evaluation of instance matching because of the class

17 http://ec.europa.eu/eurostat/ramon/relations/index.cfm?TargetUrl=LST_LINK&StrNomRelCode=CPA%202008%20-%20CPC%202&StrLanguageCode=EN&StrOrder=2&CboSourceNomElt=&CboTargetNomElt=

18 http://refine.deri.ie

imbalance caused by the number of true negatives significantly exceeding the number of true positives (Christen, 2012, p. 165). Measures that require false negatives are also not available since we only know if the evaluated positives are true or false.

Precision was evaluated using the 200 links that were judged by a single domain expert. Out of these links 20 were marked as undecidable due to lack of information necessary to establish whether a link is correct or not. These links were left out of the sample used for computing precision. Precision of the remaining 180 links was 0.683 (0.8 for `skos:closeMatch` and 0.665 for `skos:narrowMatch`). Low precision may indicate that the employed data pre-processing was too aggressive and introduced false positives. The domain experts achieved 68.48 % consistency in evaluation of the shared links, excluding those that either expert judged as undecidable. The low consistency may be attributed to vague and ambiguous descriptions of the linked concepts involved in the evaluation.

7 Conclusions

We created links between 16 pairs of datasets, both among the code lists used in OpenBudgets.eu and third-party datasets. The generated links in RDF along with the linking rules employed are openly available from https://github.com/openbudgets/linksets. The objectives of linking are dataset alignment and enrichment of code lists, so that further contextual data becomes available. In this deliverable, we described the approaches we used for automatic linking, including automatic translation. The most common and reliable approach for linking we used is based on shared codes identifying the code list concepts.

Although the task of linking data may seem deceptively easy from the outset, aiming for high-quality results is difficult. This rings especially true for code lists, the description of which is usually too short to enable to automatically discover reliable links; they are actually sometimes too ambiguous even for manual linking, and automation can hardly produce more correct links than the manual approach. While linking automation significantly improves the speed of linking execution, it is not worth pursuing if good results cannot be achieved manually.

Most of the problems encountered in this work were technical. While the performance of link discovery tools is good, the performance of loading data from RDF stores is not. In fact, we identified it as the main bottleneck. Nevertheless, it is unclear whether this is an issue of link discovery tools or of RDF stores. Linking tools can optimize their way of querying large data (e.g., using scrollable cursors), while RDF stores can optimize the sequential access to larger datasets. Ultimately, we avoided problems associated with loading data from RDF stores by executing linking procedures directly in RDF stores. We achieved the best performance when data materialization was used.

A fundamental shortcoming of most of the generated linksets is that their linkage is partial. Typically, there are resources left unlinked in either one of both linked datasets. This makes the links unsuitable for data migration and alignment, because doing so requires a complete mapping between the paired datasets. However, even incomplete linksets have value. In particular, human users can benefit from them as they interactively explore data and navigate to related data sources.

Linking code lists will continue in the OpenBudgets.eu projects to cater for the needs identified by the project's use cases. In such cases, we will benefit from the lessons learnt that we presented in this deliverable.

8 References

- Christen, Peter. Evaluation of matching quality and complexity (2012). In: Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Berlin; Heidelberg: Springer, pp. 163–184. DOI 10.1007/978-3-642-31164-2_7.

- Karampatakis, Sotiris; Ioannidis, Lazaros; Philippides, Panagiotis Marios; Bratsas, Charalampos (2015). Deliverable D1.7: Extraction and transformation of relevant code lists. https://openbudgets.atlassian.net/browse/OB-18
- Mynarz, Jindřich (2012). Computing label specificity. http://blog.mynarz.net/2012/01/computing-label-specificity.html