OD2WD: From Open Data to Wikidata through Patterns*

Muhammad Faiz, Gibran M.F. Wisesa, Adila Krisnadhi, and Fariz Darari

Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia
{muhammad.faiz52,gibran.muhammad}@ui.ac.id, {adila,fariz}@cs.ui.ac.id

Abstract. We present OD2WD, a semi-automatic pattern-based framework for populating the Wikidata knowledge graph with (originally tabular) data from Open Data portals. The motivation is twofold. One, our framework can further enrich Wikidata as a central RDF-oriented knowledge graph with a large amount of data coming from public or government sources. Two, our framework can help the discovery of data from such Open Data portals and its integration with Linked Data without forcing the Open Data portals to provide Linked Data infrastructure by themselves, as the latter may not always be feasible due to various technical, budgetary, or policy reasons. Throughout the transformation process, we identify several reengineering and alignment patterns. We implement the framework as an online service and API accessible at `http://od2wd.id`, currently tailored for three Indonesian Open Data portals: Satu Data Indonesia, Jakarta Open Data, and Bandung Open Data.

Keywords: Open Data · Wikidata · Reengineering Pattern · Alignment Pattern

1 Introduction

The Open Data initiative has been adopted in various degrees by many countries around the world [15] due to a number of benefits it offers, incl. increasing citizen participation and boosting economic growth. However, challenges remain on the findability as well as usability of such data particularly due to poor data publishing practices. Even when data publishing is facilitated by a national or regional Open Data portal, it is done in a format such as the tabular, comma-separated value (CSV) format, which does not easily support interlinking and integration. Viewing this from the 5-star rating model of Open Data [1], a huge amount of data has only 3-star rating, rather than 5-star rating.

Fig. 1 shows an example of tabular data about public schools in Jakarta stored in the Jakarta Open Data portal at `http://data.jakarta.go.id/`. One could conceivably think that the portal also possesses related CSV files containing, e.g., the number of students of the schools mentioned in Fig. 1. Since

	A	B	C	D	E
1	Nama Sekolah	Alamat	Wilayah	Telepon	Jenis Sekolah
2	SMPN 3	JL. MANGGARAI UTARA IV/6 MANGGARAI	Jakarta Selatan	021 8303844	SMP
3	SMPN 11	JLN.KERINCI BLOK.E KEB.BARU JAK-SEL	Jakarta Selatan	021 7221665	SMP
4	SMPN 12	JLN.WIJAYA IX NO.50 KEB.BARU JAK-SEL	Jakarta Selatan	021 7208095	SMP
5	SMPN 13	JL.TIRTAYASA RAYA	Jakarta Selatan	021 7262939	SMP
6	SMPN 15	JL. PROF. DR. SOEPOMO, SH MENTENG DALAM	Jakarta Selatan	021 8312669	SMP
7	SMPN 16	JL.PALMERAH BARAT NO.59	Jakarta Selatan	021 5483415	SMP
8	SMPN 19	JLN.BUMI BLOK.E NO.21 KEB.BARU JAK-SEL	Jakarta Selatan	021 7250219	SMP
9	SMPN 29	JLN.BUMI BLOK.E	Jakarta Selatan	021 7247493	SMP
10	SMPN 31	JL.PENINGGARAN BARAT III	Jakarta Selatan	021 7239730	SMP

Figure 1: Tabular data about public schools in Jakarta with columns representing school name, address, sub-district, phone number, and school type.

CSV or similar formats do not support interlinking, the integration of such data becomes more complicated.

A solution to improve this situation involves the cooperation of Open Data providers (i.e., governments and public institutions) to provide Linked Data infrastructure beyond just putting CSV files online. However, technical, budgetary, or policy reasons may prevent this to be realized. So, we advocate an alternative approach in which we employ an already existing public Linked Data (or knowledge graph) repository to host a Linked Data version of those CSV files. Provided that the data licensing policies of the Open Data portals and the repository are not in conflict, this approach then only requires, in principle, a technological solution of transforming the data from Open Data portals into data in the public Linked Data repository. The benefits of this approach have been pointed out by van der Waal et al. [14], namely discoverability, harvesting, interoperability, and community engagement.

We thus implement OD2WD, a semi-automated framework for transforming and publishing tabular, CSV-formatted data in Open Data portals to Wikidata. We chose Wikidata as the target repository not just because it is one of the most prominent Linked Data repositories, but more importantly, it allows public to add and edit data in it. From a Wikidata perspective, our effort can also be viewed as enriching Wikidata content. For example, from the public school data in Fig. 1, we can obtain connections between entities in it to existing entities such as the sub-districts. We describe the challenges in realizing such a transformation framework in Section 2 and detail the transformation workflow in Section 3. We also discover a few recurring patterns, described in Section 4, in different phases of the transformation framework, namely two reengineering patterns in datatype detection and *protagonist* column (i.e., main subject column) detection phases, as well as four alignment patterns in property mapping, entity linking, class linking, and instance typing phases. Such patterns can be useful in other similar scenarios of data conversion from a tabular form to Linked Data. For the last two sections of the paper, we report some performance evaluation regarding different phases of the transformation workflow (Section 5) and finish with a brief conclusion of the paper (Section 6).

2 Tabular Data to Wikidata Graph: Challenges

The challenges faced by our approach do not solely arise from the conversion process of tabular data to plain RDF data. Rather, we wish to integrate such data *into* Wikidata. There are a number of tools to solve the former problem, for example, Tarql [4] and RDF123 [7]. In addition, W3C has published a suite of recommendations aimed at generating RDF data from tabular data sources (e.g., Tandy et al. [12]). Our use case, however, requires consideration for the targeted platform (i.e., Wikidata) and its vocabulary. In particular, if a CSV table contains an entity that is semantically equivalent to an existing Wikidata item, then that item and its identifier should be used as the basis for enrichment. In contrast, inventing new identifiers would overpopulate the Wikidata knowledge graph with distinct identifiers that actually refer to the same real-world entity. The D2RQ mapping language [2] and the W3C recommendation of R2RML [11] both provide mapping languages into RDF, yet the data sources must be in the relational database format and also the mapping languages do not concern vocabulary mapping and entity linking aspects. Karma [8] is a semi-automated tool to map structured sources to ontologies and generate RDF data accordingly. However, the tool is not tailored toward Open Data portals with Wikidata as the target knowledge graph. The aforementioned tools are not appropriate for our setting, while the W3C recommendations do not technically specify a recipe that can be used to solve our problem.

Specifically, there are two key challenges in our setting. First, tabular data does not have a strict form that allows immediate determination of the entity to be used as a subject, a predicate, or an object. Note that tabular data can have different forms [3]: vertical listings, horizontal listings, matrices, and enumerations. Among those, data in Open Data portals usually takes the form of vertical listings, whose rows are similar entities with attributes expressed by the columns. The second challenge is the alignment aspect. Every term we extract from a table is aligned to a vocabulary term in Wikidata. Oftentimes, we can find entities with the same label but different context. For example, the term "Depok" in Indonesia may refer to a city in West Java, a district in Cirebon Regency, a district in Sleman Regency, and many more. To obtain good quality of data conversion into Wikidata, we need to resolve this ambiguity challenge.

3 Conversion Flow

The OD2WD system converts and republishes CSV data from Open Data portals to Wikidata. CSV serves as a canonical format for tabular data as other tabular formats such as XLS and ODS can be immediately exported as CSV. The conversion process (Fig. 2) consists of two major parts, namely *triple extraction* (via preprocessing and metadata extraction) and *alignment to Wikidata terms* (via mapping and linking) followed with (re)publishing the triples to Wikidata.

Triple Extraction. Our analysis on the data in the three Open Data portals we use as the basis of evaluation – see Section 5 for further information about

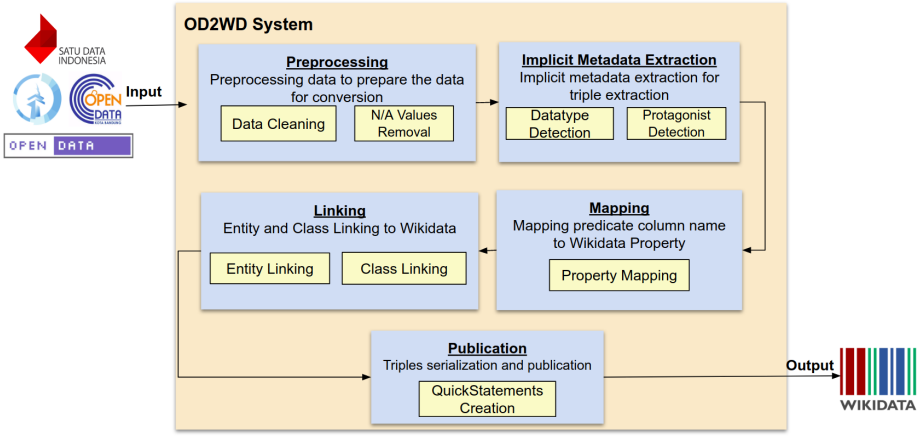

Figure 2: Data Conversion Architecture

the portals – did not find any table with format other than vertical listings. Moreover, more than 95% of those tables have exactly one column designated as the subject or *protagonist* column. So, our current implementation assumes the table to be converted is vertical listing with one protagonist column.

In a vertical listing table, each row represents a bunch of information about a single entity that can be represented as a set of triples. All triples from that set share a subject obtained from the cell value under the protagonist column. Each non-protagonist column header gives a predicate and the corresponding object is the cell value under that column. Starting from a vertical listing table, the triple extraction phase consists of preprocessing and implicit metadata extraction.

Preprocessing. This step consists of data cleaning such as case normalization on column headers as well as N/A values removal. The latter is realized by deleting either the rows or columns depending on which would delete the lowest number of cells. For example, in a table of 50 rows and 4 columns, we would delete rows that contain N/A values, but in a table of 15 columns and 4 rows, we would delete columns that contain N/A values. We also delete the (artificial) index column as it conveys no semantic information useful in our conversion process, e.g., the column with header "No." indicating row numbers.

Implicit Metadata Extraction. This step consists of two substeps: *datatype detection* of each column and *protagonist (column) detection*. In datatype detection, we go through each column and guess the datatype of each cell value under that column. The latter is done by matching the cell value to a bunch of regular expressions listed in Table 1 following the flow given by Fig. 3. The regular expressions used were original, with two exceptions: "Is Globe Coordinate"[1] and

[1] `https://stackoverflow.com/questions/3518504/regular-expression-for-matching-latitude-longitude-coordinates`

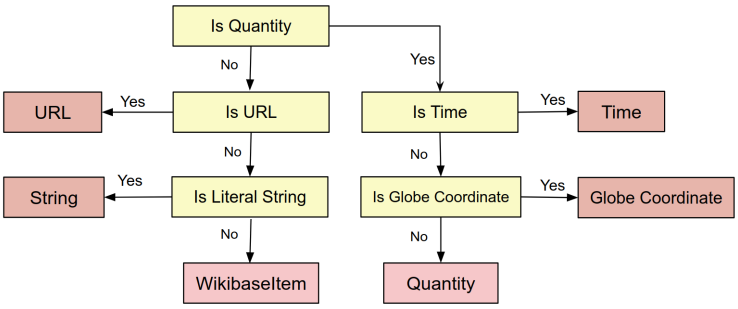

Figure 3: Datatype Detection

Table 1: Regular Expressions for Datatype Detection

Evaluation	Regular Expression
Is Quantity	`[-+.,()0-9]+`
Is Time	`^([0-2][0-9]|(3)[0-1])([\/,-])(((0)[0-9])|((1)[0-2]))([\/,-])\d{4}$`
Is Globe Coordinate	`^[-+]?([1-8]?\d(\.\d+)?|90(\.0+)?),\s*[-+]?(180(\.0+)?|((1[0-7]\d)|([1-9]?\d))(\.\d+)?)$`
Is URL	`^[a-zA-Z0-9_\-\@]+\.[a-zA-Z0-9]_\-\.`
Is Literal String	`[\.\,\!\?\>\<\/\\\)\(\-\_\+\=\*\&\^\%\$\#\@\!\:\;\~]`

"Is URL"[2]. The datatype for the column is decided by a majority rule over datatypes on all cell values under that column.

The protagonist detection step aims to determine which column acts as the subject of all triples induced by a row. We employ the following heuristic: the column with the *highest number of unique cell values* is taken as the protagonist column. Should a tie occur, the *leftmost* column takes precedence. Note that any index column will not be taken as protagonist since it is already removed in the preprocessing step. Our experiment (Section 5) indicates that tie only occurs in less than 7% of the datasets we work with and the above tie-breaking rule shows 100% accuracy in the evaluation. Of course, the heuristic itself may yield inaccurate prediction of protagonist because there are tables with either zero or more than one protagonist columns. Such tables are however in minority (less than 5% of the datasets). Handling such tables involves special steps, e.g., creation of synthetic identifiers, or merging multiple columns into one compound protagonist column, which we leave for future work.

Wikidata Alignment. The triple extraction step generates triples of the form (protagonist cell value, column header, cell value), but not yet RDF triples because no URI is assigned to the triple components. So, the next step is aligning to Wikidata items and properties so that appropriate URIs can be used. There

[2] `https://stackoverflow.com/questions/11724663/regex-for-simple-urls`

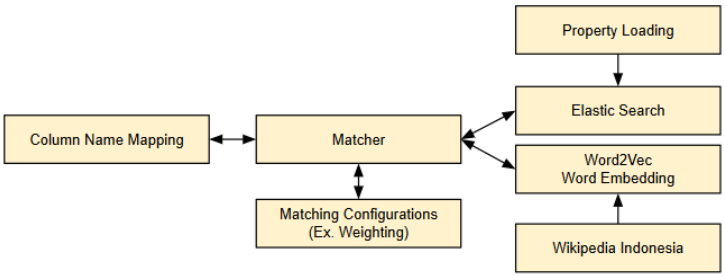

Figure 4: Mapping Architecture

are two substeps in Wikidata alignment step, namely *mapping* and *linking* steps.

Mapping. In this step, non-protagonist column headers are aligned to Wikidata properties. Fig. 4 shows the architecture of our mapping process. We employ an Elasticsearch component[3] backed by an index formed of all Wikidata property IDs, labels, description, alias, and range. Given a column header, this component provides us with a list of candidate properties closely matching it. To choose the semantically closest property, we employ Word2Vec [9] with Indonesian Wikipedia dump[4] as the underlying corpus. We then compute the cosine similarity in the embedding space between the column header and the candidate properties. The closest candidate property is taken as the alignment target for the column header. Note that Elasticsearch may yield no candidate properties. In this case, the triple is either dropped, unless the Wikidata community[5] approves the creation of a new property based on the given column header or an appropriate Wikidata property is manually found.

Linking. As illustrated in Fig. 5, here we perform: (*i*) entity linking from cell values to Wikidata items, and (*ii*) class linking, from protagonist column header to Wikidata class.

Given a cell value in the table whose type was detected as WikibaseItem, entity linking aims to find an appropriate Wikidata item URI for it. Currently, ambiguity is resolved by making use of column name as context. As future work, this can be expanded to include other context information. The process starts with obtaining a list of candidate items for the cell value through the Wikidata Entity Search API together with Wikidata classes the items belong to. To choose which item is the most appropriate to link with the cell value, we look at a combination of two cosine similarity scores computed in the embedding space (via Word2Vec). The first one is similarity between (vector representation of) the cell value and the item. The second one is between the context information (column header) and the class(es) of the item. We can thus rule out the items

[3] `https://www.elastic.co/products/elasticsearch`
[4] `https://dumps.wikimedia.org/idwiki/latest/`
[5] `https://www.wikidata.org/wiki/Wikidata:Property_proposal`

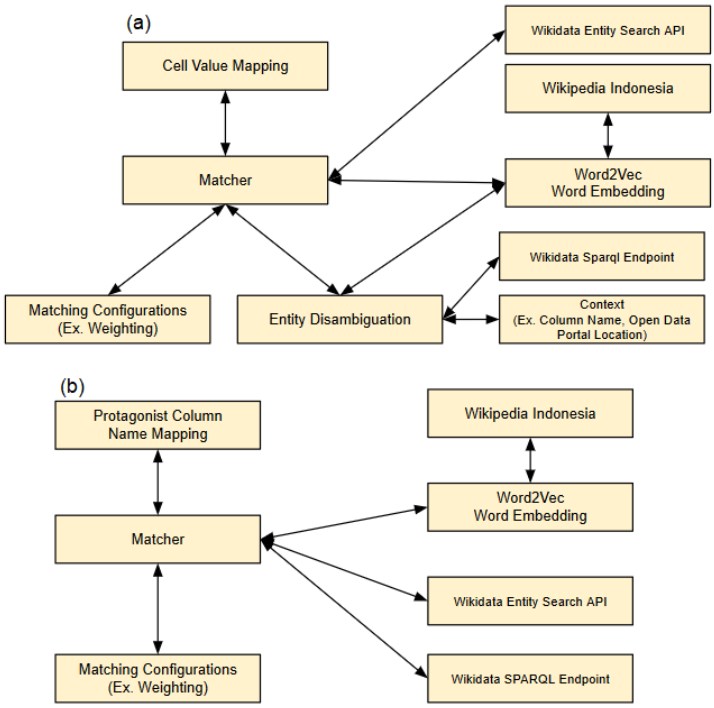

Figure 5: Entity Linking (a) and Class Linking (b) Architecture

whose class does not match with the context. The item with the highest combined score is chosen to link to the given cell value. If no linking is found (because the score is too low), then the cell value is likely to be a new item to be created in Wikidata.

Class linking is performed on protagonist column header. The motivation is that all protagonist entities belong to the same category or class and information from the protagonist column header can be exploited to obtain that class. The process starts with getting candidate items that likely match the protagonist column header via Wikidata Entity Search API. Then, we filter out those items that are not classes (using SPARQL query on Wikidata endpoint). A class is an item occurring at the object position of the "instance of" or "subclass of" relationships. Next, we select the class with the highest similarity score with the protagonist column header, computed in the embedding space.

Publishing The publishing phase aims to publish the conversion result, which is in triple format of protagonist-property-value, to Wikidata. Our current implementation uses the QuickStatements[6] tool to import the result to Wikidata. This

[6] `https://tools.wmflabs.org/quickstatements/`

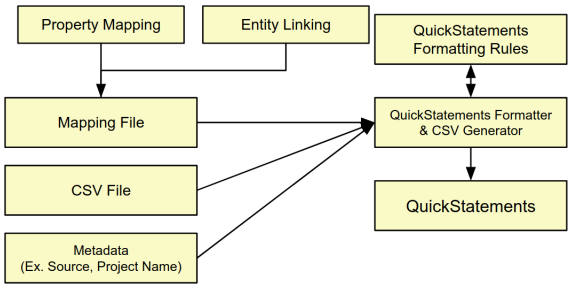

Figure 6: Publishing Architecture

phase makes use of the mapping and linking results from the previous steps, and generates a QuickStatements serialization to be executed for Wikidata importing. Not all results from mapping and linking phase will be published, because there is a possibility of an error in those phases. Hence, a final manual check can be performed to resolve errors or false values in the mapping and linking.

Fig. 6 shows the architecture for the publishing phase. This phase consists of several steps. The first is loading the results of entity linking and property mapping. After that we use the data from the source CSV to fill in the values of literal properties and labels. We then apply the correct formatting based on each column datatype according to the QuickStatements syntax. We also add metadata for the import process, e.g., reference (i.e., Open Data portal links). The final step is simply executing QuickStatements.

4 The Patterns

Ontology Design Patterns (ODPs) are modeling solutions to frequently occurring ontology design problems [6]. Unlike the usual ontology modeling problem whose aim is to design an ontology capturing a particular notion, our work concerns with patterns occurring during transformation process. In the ODP typology, these patterns are reengineering and alignment patterns. Within our data transformation workflow, these patterns are apparent in protagonist and datatype detection phases as well as within vocabulary alignment phases.

4.1 Reengineering patterns

Reengineering patterns operates on a source model to generate a new ontology or knowledge graph (or its parts) as a target model via some transformation procedure. The source model need not be an ontology; other types of resources are possible, e.g., thesaurus, data model patterns or linguistic structures. In our case, the source model is a tabular data model, while the target model is a graph shape that is part of the Wikidata knowledge graph. Metamodel transformation rules can be used to describe reengineering patterns [6, 13].

Given: Schema tuple $T = (C_1, \ldots, C_m)$, $t = (c_1, \ldots, c_m)$ is a row in table T, and $C_k = Prot(T)$.

Generate: Graph with the following form (written in Turtle syntax):

$\texttt{Subj}\ \texttt{Pred}_1\ \texttt{Obj}_1\ ;\ \ldots\ ;\ \texttt{Pred}_{k-1}\ \texttt{Obj}_{k-1}\ ;\ \texttt{Pred}_{k+1}\ \texttt{Obj}_{k+1}\ ;\ \ldots\ ;\ \texttt{Pred}_m\ \texttt{Obj}_m.$

where $\texttt{Subj} = LinkRes(c_k)$, the Wikidata entity corresponding to c_k (as obtained according to the alignment pattern in Fig. 10), and for $1 \leq j \leq m$, $j \neq k$, we have:

- $\texttt{Obj}_j = LinkRes(c_j)$, the Wikidata entity corresponding to c_j;
- $\texttt{Pred}_j = MapRes(C_j)$, the Wikidata property corresponding to column header C_j according to the alignment pattern in Fig. 9.

Figure 7: Reengineering pattern for generating graph shape from table with protagonist column identified

In our case here, the reengineering patterns emerge as we go along the transformation process. They help us streamline our thought process and more importantly, they capture the principles applicable to the transformation of virtually all tables in Open Data portals. As metamodel, however, there is no clear consensus as to how exactly the pattern can be expressed formally. Thus, we conveniently choose a very simple meta-rule expression to describe the patterns as seen below.

The first reengineering pattern here concerns the graph shape obtained from the table by accounting for its protagonist. Let T be a table expressed as a *schema tuple* $T = (C_1, \ldots, C_m)$ with column headers $C_1, \ldots, C_m$. Then, the table can contain up to N *data tuples* of the form $t = (c_1, \ldots, c_m)$ representing a row of the table T where each c_j is a value of column C_j in that row. Let $Prot(T)$ be the column header C_k that is detected as protagonist of T. Then, the reengineering pattern specifies a graph shape according to Fig. 7.

We also identify a reengineering pattern during datatype detection. Specifically, given a column header C_j of a table T and the corresponding values in that column, we obtain the datatype corresponding to the column via reengineering pattern in Fig. 8. Note that procedurally, the datatype detection is done through a flow given in Fig. 3.

4.2 Alignment patterns

Alignment (or *Mapping*) *patterns* express semantic associations between two vocabularies or ontologies [5, 10]. With alignment patterns, one can declaratively express the associations between vocabularies. Such expressions can be captured by some alignment language such as EDOAL.[7] In our case, we slightly relax the definition of vocabularies here to include not just Wikidata properties and items, but also terms appearing as table headers and values. Thus, existing alignment

[7] `http://alignapi.gforge.inria.fr/edoal.html`

Given: A column header C_j of table T containing N rows and $c_j^{(i)}$, $1 \leq i \leq N$ are N (not necessarily unique) values from each row of T at the j-th column.

Generate: A Wikidata datatype `dt` for column C_j if the majority of $c_j^{(i)}$'s satisfy the datatype pattern `dtp`, which is a Boolean combination of regular expressions from Table 1 specified as follows:

- if `dtp` is neither Quantity, URL, nor Literal String, then then `dt` is WikibaseItem;
- if `dtp` is neither Quantity nor URL, but is Literal String then `dt` is String;
- if `dtp` is not Quantity, but is URL, then `dt` is URL;
- if `dtp` is Quantity, but not Date and not Globe Coordinate, then `dt` is Quantity;
- if `dtp` is Quantity, not Date and is Globe Coordinate, then `dt` is Globe Coordinate;
- if `dtp` is Quantity and Date, then `dt` is Time.

Figure 8: Reengineering pattern for datatype detection

language like EDOAL cannot exactly capture the intended alignment. Instead, we express an alignment as a set of RDF triples using our own vocabulary whose intuitive meaning can be easily understood. This format also allows the alignments to be shareable more easily.

During mapping, entity linking, and class linking phases we identify a number of alignment patterns below, expressed declaratively as RDF graph. Below, we use the following URI prefixes: `xsd` for `http://www.w3.org/2001/XMLSchema#`, `wd` for `http://www.wikidata.org/entity/`, `wdt` for `http://www.wikidata.org/prop/direct/`, `skos` for `http://www.w3.org/2008/05/skos#`, `rdf` for `http://www.w3.org/1999/02/22-rdf-syntax-ns#`, `od2wd` for `http://od2wd.id/resource#` and `od2wd-prop` for `http://od2wd.id/property#`.

Alignment pattern in mapping phase. The alignment pattern identified during mapping phase is between a non-protagonist column header of a table and a Wikidata property. Let `ColName` be a column header and `wdt:Y` is the Wikidata property most likely associated with `ColName` according to the mapping procedure in Fig. 4. Then, $MapRes(C) =$ `wdt:Y` expressed as a graph structure in Fig. 9, which also includes mapping relation information (skos:broadMatch), confidence score of the mapping, URI of mapping procedure, an the time when the mapping was computed. We use `skos:broadMatch` because column names in the source table tend to have a semantically narrower meaning than the corresponding Wikidata properties. That is, `ColName` "has broader concept" `wdt:Y`.

Alignment pattern during entity linking phase. During entity linking, we discover an alignment pattern similar to the one in Fig. 9, this time between values in a table and Wikidata entities (Fig. 10). Given a value in the table `EntityName`, the linking phase described in Fig. 5a results in `wd:Y`, the most likely Wikidata entity that matches `EntityName`. Note that this is only done to values of type Wikibaseitem. The provenance information is similar, but with `skos:closeMatch` as the mapping relation.

```
_:link1 od2wd-prop:type skos:broadMatch ;
        od2wd-prop:from "ColName" ;
        od2wd-prop:to   wdt:Y ;
        od2wd-prop:confidence "Num"^^xsd:decimal ;
        od2wd-prop:generated_from od2wd:od2wdapi ;
        od2wd-prop:when "Time"^^xsd:dateTime .
```

Figure 9: Alignment pattern for column headers and Wikidata properties.

```
_:link1 od2wd-prop:type skos:closeMatch ;
        od2wd-prop:from "EntityName" ;
        od2wd-prop:to   wd:Y ;
        od2wd-prop:confidence "Num"^^xsd:decimal ;
        od2wd-prop:generated_from od2wd:od2wdapi ;
        od2wd-prop:when "Time"^^xsd:dateTime .
```

Figure 10: Alignment pattern for table values and Wikidata entities.

```
_:link1 od2wd-prop:type skos:closeMatch ;
        od2wd-prop:from "ColName" ;
        od2wd-prop:to   wd:Y ;
        od2wd-prop:confidence "Num"^^xsd:decimal ;
        od2wd-prop:generated_from od2wd:od2wdapi ;
        od2wd-prop:when "Time"^^xsd:dateTime .
```

Figure 11: Alignment pattern for protagonist column `ColName` and Wikidata class-type entity `wd:Y`

Alignment pattern during class linking phase. Protagonist columns are not mapped to Wikidata properties since their values are the subject entities. Instead, they are linked to class entities, i.e., those occurring as the target of *instance of* or *subclass of* properties in Wikidata, via the procedure given in Fig. 5b. Fig. 11 shows an alignment pattern between protagonist columns and Wikidata class-type entities, expressed similarly as the earlier two alignment patterns.

In addition to the alignment between protagonist columns and Wikidata class-type entities, we also observe an alignment between values in a protagonist column and its Wikidata class-type entity. That is, such values are viewed as instances of the class-type entity. This is described in Fig. 12.

5 System Performance and Evaluation

We measure the accuracy of each conversion step, by comparing the system results with the human-created gold standards. The experiment was done using 50 CSV documents coming from several of Indonesian open data portals: Indonesia Satu Data (`http://data.go.id`), Jakarta Open Data (`http://data.jakarta.go.id`) and Bandung Open Data (`http://data.bandung.go.id`).

```
_:link1 od2wd-prop:type rdf:type ;
        od2wd-prop:type_context "ColName" ;
        od2wd-prop:from wd:Z
        od2wd-prop:to   wd:Y ;
        od2wd-prop:generated_from od2wd:od2wdapi ;
        od2wd-prop:when "Time"^^xsd:dateTime .
```

Figure 12: Alignment pattern asserting `wd:Z` as instance of `wd:Y` where `wd:Z` is the result of $LinkRes(c)$ with c a value under protagonist column `ColName`.

Table 2: Datatype Detection Evaluation Result

Number of Correct Column	Total Number of Column	Accuracy Average per Document	Accuracy Total
286	349	83.5%	81.9%

5.1 Evaluation Result

This section discusses the result of evaluation from several conversion phases, especially those who have been identified to have a pattern.

Datatype Detection Datatype detection is an effort to predict the datatype of each columns of the table, the rules of the datatype are based on the Wikidata datatypes, further information about the datatype rules and this phase as a whole could be seen on previous section. To measure the performance of this phase, we measure the accuracy of system's datatype prediction. We compare the system's prediction with a gold standard we created using the help of Wikidata-educated human judges using a 3-judge system, where, for each column, 3 human judges would evaluate the system prediction, and give a verdict whether the prediction was accurate or not. The results of the experiment are shown in Table 2.

From the table above we could see that the system has managed to obtain quite a satisfactory performance with the total accuracy of 81.9% and 83.5% average accuracy per document.

Protagonist Detection For protagonist detection to check the accuracy of the heuristics, we use 50 CSV files, the same files used in datatype detection evaluation, and we label each CSV with their respective protagonist column, the labelling process was done manually by a researcher and several evaluators using a 3-Judge System similar to the one we used in datatype detection evaluation. Then we compare the result of the heuristics guess of the document's protagonist with the label to rate the heuristics accuracy.

For protagonist detection phase, we managed to obtain accuracy of 88% . As we can see, though it is not perfect, this score is pretty satisfactory.

Table 3: Linking and Mapping Evaluation Result

Phase	Mapping	Entity Linking	Class Linking
Number of Prediction	279	890	100
Number of Correct Prediction	221	787	70
Accuracy	79.21%	88.42%	70%

Mapping and Linking To evaluate mapping and entity linking, we asked human evaluators to rate the correctness of the predicted mapping and entity linking of 50 CSVs with 279 column names and 890 cell values. Each evaluator is given the predicted mapping between column names from CSV and Wikidata properties and the predicted entity linking between cell values from CSV and Wikidata entities. Every prediction is evaluated by 3 different evaluators.

The class linking is similarly evaluated. Here, we asked human evaluators to check the correctness of the predicted Wikidata class for the protagonist column of 100 CSVs.

Table 3 summarizes the result of our evaluation. We achieve an accuracy of 79.21% for mapping, 88.42% for entity linking, and 70% for class linking.

5.2 Result Discussion

The result that we managed to obtain is quite satisfactory. The system managed to obtain a good accuracy score on all phases of the system. However, there are few cases not handled very well and thus result in inaccuracies.

Datatype detection is an early phase in the system. On the current implementation, a set of regular expressions are matched to the cell values of each column in sample rows to determine their datatype. From this method, we managed to obtain an accuracy of 80%. Inaccuracies in this phase are caused by irregularity in the values themselves. As an example, a cell value consisting only of numbers is normally a Quantity, but there are columns whose values are only numbers, but should not have a Quantity datatype, such as the column containing identification code of a cemetery.

We have observed that there are other factors that could potentially be used in determining a column's datatype, especially the column header. There are column header names that sometimes correspond to certain datatypes, such as "nama" (or name) that is more likely to be a WikibaseItem. The incorporation of such a factor is left for a future work.

Inaccuracies were also caused by nested structure in the table. For example, the table 'data-tps-kota-bandung' containing waste collection location data, has a cell value 'Kel. Campaka 2 Rw - Sukaraja 1Rw - Pangkalan Auri' in column 'Sumber Sampah (RW - Kelurahan - Kecamatan)'. The cell actually contains compound information, i.e., waste source, neighbourhood code, subdistrict, and district. A special case of nested structure is where the cell values of a column can be composed of multiple datatypes. The nested structure case occurs in less than 6% of the datasets and their special handling is left for future work.

The protagonist detection phase has also shown good performance with 88% accuracy obtained. Most of the errors were due to carry-on errors from the datatype detection phase in which the correct protagonist column was judged to have a datatype other than WikibaseItem. Consequently, the column was incorrectly not considered to be the protagonist.

For the mapping and linking phases, many cases of errors occur when our model failed to map to the correct property or entity. For example, our prediction failed to map "SMA Negeri 10" from cell value to appropriate entity. Our prediction maps "SMA Negeri 10" to SMA Negeri 10 Padang (Q7391091) although the CSV where we get that cell value talks about high school in Jakarta, not in Padang. We can resolve this case by adding more context to the mapping process. For example because we took that CSV from Jakarta Open Data Portal, we can take that information as a context to filter out any high school outside Jakarta. The mapping and linking phases also dependent on the previous phase such as datatype detection and protagonist detection phase, hence any error occurs in those phases will be affecting the result of our mapping and linking. From this observation, we could understand that one phase could influence other phase's performance. Hence, it is important that we should seek to improve the performance of all phases, especially the early phases of the conversion process.

6 Conclusions and Future Work

OD2WD is a tool to convert tabular data in CSV format to RDF and republish it into Wikidata knowledge graph. The main idea and challenges of this process are two fold: extracting triples from the source table and aligning it with Wikidata vocabulary. We approach the problem using patterns, used as a blueprint for the conversion process. OD2WD system was constrained to a vertical listing table, so While OD2WD system currently only support CSV format, adding support for other data format should not pose a big problem because the conversion process are practically the same as long as they are in vertical listing table category.

To measure the system performance, each phase of this system has been evaluated using tabular data from the Indonesia Open Data Portal, Jakarta Open Data Portal, and Bandung Open Data Portal. We achieve 81.9% accuracy on datatype detection, 87.7% accuracy on protagonist detection, 79.21% accuracy on property mapping phase, 70% in class linking, and 88.42% in entity linking. This results shows that there are some degree of risks of inserting wrong information to Wikidata, this risk, however, were mitigated because user could manually check the conversion results before publishing them to Wikidata. At the end of the research we have published 20256 statements to Wikidata as a result of converting data from open data portals.

For further works we suggest that adding more context such as metadata from open data portal to the mapping and linking phases will increase the accuracy of mapping and linking results.

Acknowledgements. This work is supported by the 2019 PITTA B research grant "Analysis and Enrichment of Wikidata Knowledge Graph" from Universitas In-

donesia and the Wikimedia Indonesia project "Peningkatan Konten Wikidata". We thank students of Faculty of Computer Science Universitas Indonesia for their help in evaluation part of this work, as well as Raisha Abdillah from Wikimedia Indonesia for her assistance during final checking before deploying the conversion results to Wikidata.

References

1. Berners-Lee, T.: Linked Data (2006), https://www.w3.org/DesignIssues/LinkedData.html
2. Bizer, C., Seaborne, A.: D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs. In: ISWC (Posters) (2004)
3. Crestan, E., Pantel, P.: Web-scale table census and classification. In: WSDM (2011)
4. Cyganiak, R.: Tarql (SPARQL for tables) (2019), http://tarql.github.io
5. Gangemi, A.: Ontology design patterns for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3729, pp. 262–276. Springer (2005)
6. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies (2009)
7. Han, L., Finin, T., Parr, C.S., Sachs, J., Joshi, A.: RDF123: from spreadsheets to RDF. In: ISWC (2008)
8. Knoblock, C.A., Szekely, P.A., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: Semi-automatically Mapping Structured Sources into the Semantic Web. In: ESWC (2012)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)
10. Scharffe, F., Zamazal, O., Fensel, D.: Ontology alignment design patterns. Knowl. Inf. Syst. **40**(1), 1–28 (2014). https://doi.org/10.1007/s10115-013-0633-y, https://doi.org/10.1007/s10115-013-0633-y
11. Souripriya Das, Seema Sundara, R.C. (ed.): R2RML: RDB to RDF Mapping Language. W3C Recommendation (27 September 2012), https://www.w3.org/TR/r2rml/
12. Tandy, J., Herman, I., Kellogg, G. (eds.): Generating RDF from Tabular Data on the Web. W3C Recommendation (17 December 2015), https://www.w3.org/TR/csv2rdf/
13. Villazón-Terrazas, B., Priyatna, F.: Building ontologies by using re-engineering patterns and R2RML mappings. In: Blomqvist, E., Gangemi, A., Hammar, K., Suárez-Figueroa, M.C. (eds.) Proceedings of the 3rd Workshop on Ontology Patterns, Boston, USA, November 12, 2012. CEUR Workshop Proceedings, vol. 929. CEUR-WS.org (2012), http://ceur-ws.org/Vol-929/paper10.pdf
14. van der Waal, S., Wecel, K., Ermilov, I., Janev, V., Milosevic, U., Wainwright, M.: Lifting Open Data portals to the data web. In: Auer, S., Bryl, V., Tramp, S. (eds.) Linked Open Data - Creating Knowledge Out of Interlinked Data (2014)
15. World Wide Web Foundation: Open data barometer - leaders edition (2018), http://bit.ly/odbLeadersEdition