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 Wikidata knowledge graph with data from Open Data portals, which are mostly still available in tabular formats. The motivation is twofold. One, this can further enrich Wikidata as a central RDF-oriented knowledge graph with a large amount of data especially from public or government sources. Two, this can help the discovery of data from such Open Data portals and their integration with other data already part of 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 from `http://od2wd.id` currently tailored for three Indonesia Open Data portals: Satu Data Indonesia Portal, Jakarta Open Data Portal, and Bandung Open Data Portal.

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

1 Introduction

Open Data initiative has been adopted in various degrees by many countries around the world [12, 6] due to a number of benefits it offers, including increasing citizen participation and promoting innovation and economic growth through the release of reusable data [9]. Challenges remain however on findability as well as usability of such data due to problems such as the lack of appropriate supporting data infrastructure and poor practices of data publishing. 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 in their publication, rather than the 5-star rating.

Fig. 1 shows an example of tabular data containing basic information about public schools in Jakarta stored in the Jakarta Open Data portal.[1] One could

[1] `http://data.jakarta.go.id/`

	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

Fig. 1. Some public school data in Jakarta with columns from left to right representing school name, address, sub-district, phone number, and type of school.

conceivably think that the portal may also possess CSV files containing the number of students of the schools mentioned in Fig. 1 or mentioning the name of sub-district appearing in this table. Since CSV or similar formats do not support interlinking, integration of such data becomes more complicated.

An obvious way to improve this situation is certainly to push open data providers (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 where 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. [11], namely discoverability, harvesting, interoperability, and community engagement. Publishing to a bigger, more widely known repository with an established community can extend the benefits from just local government and community to the larger national or worlwide community.

Following up this idea, we implement OD2WD, a framework for transforming data from Open Data portals available in tabular format (CSV) into 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 populating or enriching Wikidata content. In the context of public school data from Fig. 1, we can obtain connections between entities in the aforementioned public school data to existing entities such as the sub-districts. Moreover, since Wikidata is multilingual, we essentially allow the data published using Indonesian language to be accessible through other languages. We describe the challenges in realizing such a transformation framework in Section 2 as well as details of the transformation workflow in Section 3.

We have also found a few recurring patterns 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 that can be useful in other similar scenarios of data conversion from a tabular form to linked data. These patterns are described in Section 4.

For the last two sections of the paper, we report some evaluation of the system performance 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

First thing to note here is that our problem is not merely converting tabular data to some plain RDF data. Rather, we wish to integrate such data *into* Wikidata. There exist a number of tools to solve the former problem, for example, Tarql [3], XLWrap [7], and RDF123 [5]. In addition, W3C has published a suite of recommendations aimed at generating RDF data from tabular data sources (Tandy et al. [10] and other documents from the CSV on the Web Working Group[2]). In our case, however, we need to take into account the existing vocabulary in Wikidata, in particular, if a table contains an entity that is semantically equivalent to a Wikidata item or property, then we wish to use the item's identifier as the basis instead of inventing our own identifier. The aforementioned tools are not appropriate for this setting, while the W3C recommendations do not technically specify exactly 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 [2] expressed as knowledge. They can be: *vertical listings* whose rows are similar entities possessing attributes expressed by the columns, e.g., tabular data in Fig. 1; *horizontal listings*, which can be viewed as a transpose of vertical listings (the attributes are the rows instead of columns); *matrix* whose row and column headers are all subjects, while the objects are the combination of a pair of subjects (particular row-column pairs), e.g., distance between all pairs of cities in a province; and *enumeration*, e.g., a list of provinces in Indonesia. Note also that in vertical or horizontal listings, the subject may be omitted, especially when the table talks about only a single entity, e.g., listing of features of a particular model of Toyota car. Fortunately, data in Open Data portals usually take the form of vertical listings with a designated subject column, which may not necessarily be the leftmost column of the table.

The second challenge is the alignment aspect. Every term that we have already extracted from a table will be aligned to a vocabulary term in Wikidata Knowledge Graph. Wikidata entities are user generated, meaning registered users can easily add and edit Wikidata entities. Hence, often we can find entities with the same label but different meaning and context. For example, the term

[2] `https://www.w3.org/2013/csvw/wiki/Main_Page`

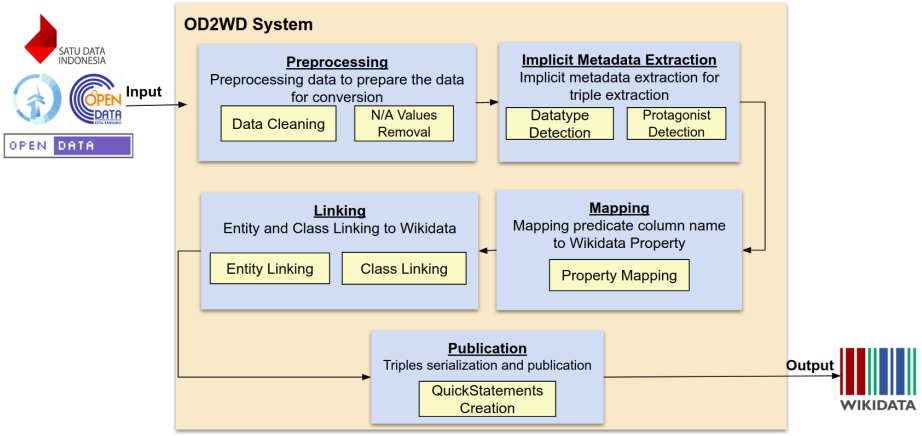

Fig. 2. Data Conversion Architecture

"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 knowledge graph. The main idea of the conversion process is an effort to generate a set of triples from the source CSV document in an accurate and complete manner, that is why we name this step *triple extraction*. The resulting set of triples is subsequently *aligned with the Wikidata vocabulary* in order to republish the extracted triples to Wikidata knowledge graph. Fig. 2 shows the architecture for data conversion on OD2WD.

Our system is designed to convert tables of vertical listings category (as per Crestan's table taxonomy [2]). Two main factors of tables in this category are: (*i*) the presence of protagonist information within the table, in the form of protagonist column or subject column, and (*ii*) the existence and structure of information describing the protagonist in the form of predicate columns. Our system concentrates on handling tables with the presence of these two factors, since such tables occur often in Open Data portals.

Triple Extraction A triple is composed of three elements: subject, predicate, and object. Thus, to create or extract a triple, those three elements must be obtained. Crestan [2] elaborated that the biggest challenge for semantic triple extraction from a table of vertical listing category is determining which columns list the predicates and which list the subject. The process of determining which columns list the subject is called *protagonist detection*. Since there is only one

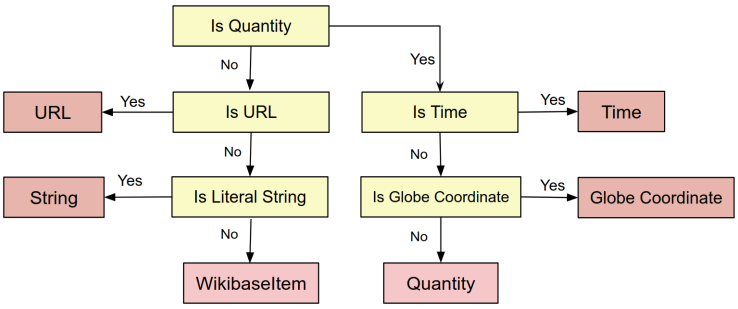

Fig. 3. Datatype Detection

protagonist column in a table, once it is detected, the remaining columns can then be inferred to be predicate columns.

After the subject and predicate information has been obtained, the object of the triple could be retrieved from the cell values. Having all the subject, predicate, and object, we may now create a triple.

Wikidata Alignment The resulting triple is now aligned with the Wikidata vocabulary. Wikidata entities can be further grouped into items and properties. An item refers to a real-world object, concept, or event that is given a unique identifier in Wikidata as well as description about it. A property is the descriptor for a data value. Based on that definition, subjects and objects will be aligned to Wikidata items, which we call *linking phase*, whereas predicates will be aligned to Wikidata properties, which we call *mapping phase*. There is also the *class linking phase* to relate protagonist entities with their respective classes, inferred from the protagonist column name.

3.1 Conversion Flow Implementation

This subsection elaborates each phase in the conversion flow.

Preprocessing The preprocessing phase aims to prepare the data so it can be used for further steps more effectively by performing data cleaning and N/A values removal.

Implicit Metadata Extraction This phase aims to complete the information required to extract triples from the table values. As for the datatype detection, for each column we are evaluating, we take sample rows to be evaluated. We rely on regular expressions to determine column datatypes. If there is any discrepancy of predicted datatypes from the sample rows, we take the majority rule. The flow of the datatype detection is shown in Fig. 3 and the regular expressions used are listed in Table 1. The regular expressions used were mostly created by us, with

Table 1. Datatype Detection Regular Expressions

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	`[\.\,\!\?\>\<\/\\\)\(\-\_\+\=\*\&\^\%\$\#\@\!\:\;\~]`

exceptions on "Is Globe Coordinate"[3] and "Is URL"[4] taken from StackOverflow with some modification.

With respect to protagonist detection, the aim is to determine which column acts as the main node of the graph created. We approach this problem using a heuristic approach of column value uniqueness with column ordering as tie breaker should a tie on uniqueness score occurs. Column uniqueness is calculated by dividing the number of distinct values by the number of rows. The score of each column is then sorted with the one with the highest score assigned to be the table protagonist. Should a tie occurs between several columns sharing the same highest uniqueness score, we will use column ordering as the tie breaker in which the column located on a more left position receives a higher score than those located more to the right.

Mapping The first step of the mapping phase is to identify and map the classes and properties from column names to their Wikidata correspondences. The inputs of this step are column header values and their datatypes (that are obtained from a previous phase).

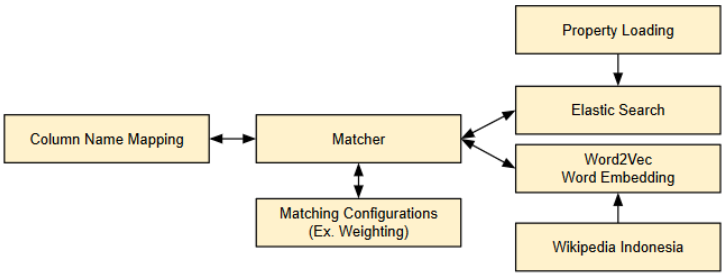

Fig. 4. Mapping Architecture

[3] `https://stackoverflow.com/questions/3518504/regular-expression-for-matching\\-latitude-longitude-coordinates`
[4] `https://stackoverflow.com/questions/11724663/regex-for-simple-urls`

Fig. 4. shows the architecture of our mapping process. In this phase we use two tools: an index of Wikidata properties and a trained word embedding model. We use an indexing method because directly using the Wikidata query service did not satisfy our needs. To make the index, we query all the Wikidata properties and store them in our index. For each property, we obtain its ID, label, description, alias, and range, and store the information in our index. As for the word embedding model, we use the Word2Vec technique [8] applied to the Indonesian Wikipedia dump.[5] Word embeddings are useful to map column header values with Wikidata properties with the closest semantic relation.

To retrieve property candidates, we query using header column values and their datatype as query keywords. Each property candidate will have a score according to the similarity between the word vector of the property candidate and the header column value. We calculate it using cosine similarity. The property candidate with the highest score will be mapped to header name. It is possible that a query returns the empty result, meaning that the corresponding column does not have any matching property in Wikidata.

Linking In this linking step, we perform: (*i*) class linking, from protagonist header value to Wikidata class, and (*ii*) entity linking from cell values to Wikidata entities. The step is illustrated in Fig 5.

Entity linking aims to link cell values from data in Open Data portals to their appropriate Wikidata entities. It requires two inputs: cell values from the CSV data and the context of the cell values. We define context as the column name of that cell value, and use context to resolve ambiguities. First, we call Wikidata Entity Search API using cell values as search keywords. We then rerank the search results based on the context. We utilize the class information for each entity to resolve ambiguities. Each entity will have a score according to its cosine similarity between the word vector of context and its class label. Moreover, for each candidate entity we calculate its cosine similarity score wrt. the cell value. For the final score to decide which entity is to link to, we calculate the mean of those two scores. The entity with the highest score will be linked to the cell value. The output of this phase is a link between the cell value and its appropriate Wikidata entity. In the case of no linking found, we use the special value "null".

Class linking aims to link protagonist column names with Wikidata classes. The process goes as follows. First, we search for Wikidata entities based on the protagonist column name using the Wikidata Entity Search API. Then, we use SPARQL querying to identify whether the obtained entities are classes or not. An entity x is a class if there is an "instance of" or "subclass of" relationship with x acting as the object of the relationship. We then compute and rank the cosine similarity between the class candidates and the protagonist column names. The output of this phase is a link between protagonist column names and Wikidata classes.

[5] `https://dumps.wikimedia.org/idwiki/latest/`

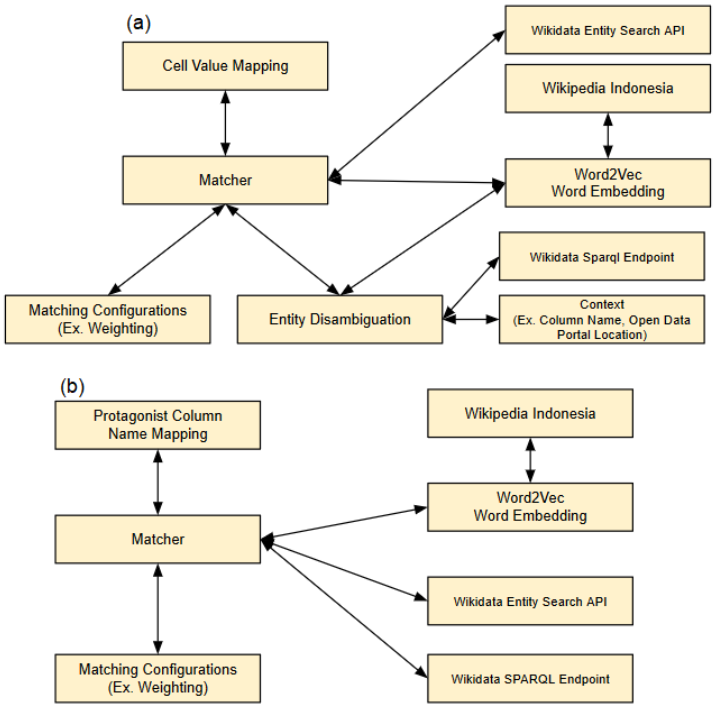

Fig. 5. Entity Linking (a) and Class Linking (b) Architecture

Publishing The publishing phase aims to publish the conversion into Wikidata. The result is in triple format of protagonist-property-value. In the current implementation we use the QuickStatements tool to import the result to Wikidata.[6] This phase makes use of the mapping and linking results from the previous steps, and generates a QuickStatements serialization to be executed for Wikidata importing.

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.

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

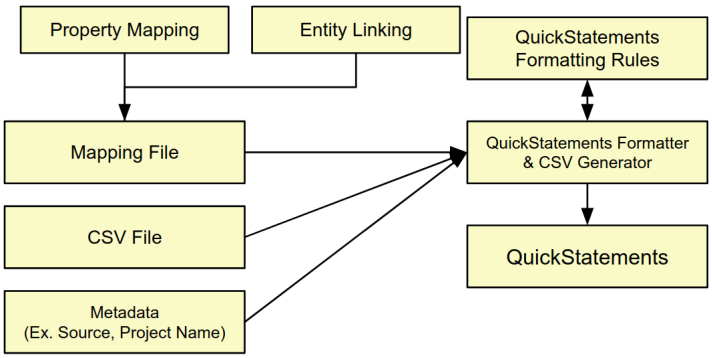

Fig. 6. Publishing Architecture

4 The Patterns

Ontology Design Patterns (ODPs) are modeling solutions to frequently occurring ontology design problems [4]. 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 mapping 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 Re-engineering patterns

Reengineering patterns operates on a particular source model to generate a new ontology or knowledge graph (or its parts) as a target model via some transformation rules or procedures. The source model need not be an ontology; other types of resources are possible, e.g., thesaurus concepts, 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 [4].

The first reengineering pattern we identify concerns the graph shape obtained from the table by accounting for its protagonist. This pattern can be described as follows. 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

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.

Fig. 7. Reengineering pattern for generating graph shape from table with protagonist column identified

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 by Fig. 3.

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 `dtype` if the majority of $c_j^{(i)}$'s satisfy the datatype pattern `dtypePattern`, which is a Boolean combination of regular expressions from Table 1 specified as follows:

- if `dtypePattern` is not Quantity and not URL and not Literal String, then `dtype` is WikibaseItem;
- if `dtypePattern` is not Quantity and not URL, but is Literal String then `dtype` is String;
- if `dtypePattern` is not Quantity, but is URL, then `dtype` is URL;
- if `dtypePattern` is Quantity, but not Date and not Globe Coordinate, then `dtype` is Quantity;
- if `dtypePattern` is Quantity, not Date and is Globe Coordinate, then `dtype` is Globe Coordinate;
- if `dtypePattern` is Quantity and Date, then `dtype` is Time.

Fig. 8. Re-engineering pattern for datatype

4.2 Alignment patterns

Alignment (or *Mapping*) *patterns* express semantic assocciations between two vocabularies or ontologies. 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. 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-prop` for `http://od2wd.id/property#`, and `od2wd` for `http://od2wd.id/resource\#`.

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`.

```
_: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 .
```

Fig. 9. Alignment pattern for column headers and Wikidata properties.

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.

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. Here, we discover the following alignment pattern between protagonist columns to Wikidata class-type entities, expressed similarly as the earlier two alignment patterns (Fig. 11).

```
_: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 .
```

Fig. 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 .
```

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

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.

```
_: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 .
```

Fig. 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`.

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 Portal (`http://data.go.id`), Jakarta Open Data Portal (`http://data.jakarta.go.id`) and Bandung Open Data Portal (`http://data.bandung.go.id`).

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.

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%

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 we use several approaches in determining the protagonist columns. To determine which heuristics that will be used, we looked for heuristics with the best accuracy. To check the accuracy of the heuristics, we pick a sample CSV, and we label each CSV with their respective protagonist column, the labelling process was done manually by the researcher and several evaluators using a 3-Judge System. 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 87.7% for the evaluation dataset. As we can see, though it is not perfect, this score is pretty satisfactory. Most of the wrong cases were caused by a carry-on error from the datatype detection phase, in which the protagonist column were falsely judged to have a datatype other than wikibaseitem, which makes them not considered to be a protagonist column candidate.

Mapping and Linking To evaluate mapping and entity linking phases we conduct experiments using 50 CSVs with 279 column names and 890 cell values. We hired evaluators to assess our prediction by giving them a document that

consists of the predict mapping between column name from CSV and Wikidata Property for mapping phase and predict entity linking between cell value from CSV and Wikidata Entity for entity linking phase. Every prediction will be evaluated by 3 different evaluators who have been given preliminary knowledge about the task before starting the evaluation. The evaluators will be assessing our prediction by giving each prediction a score of 0 (for incorrect predictions) or 1 (for correct predictions). For class linking phase, we conduct experiment using 100 CSVs with 100 protagonist column name. Our benchmark document for class linking consists of the linking between protagonist column name and predicted Wikidata class. If there is no appropriate class, it will be linked to string "NOT FOUND". Every linking result will be given a score of 0 (if false) or 1 (if true).

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%

Table 3 above is the result of our linking and mapping phase. For mapping we get accuracy of 79.21%, entity linking we get accuracy of 88.42% and class linking is 70%. Many cases of errors occur when our mapping and linking prediction 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. Such a case can be resolved by using metadata to filter unwanted entities or properties.

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.

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. At the end of the research we have published 3964 statements to Wikidata as a result of converting data from open data portals.

References

1. Berners-Lee, T.: Linked data (2006), https://www.w3.org/DesignIssues/LinkedData.html
2. Crestan, E., Pantel, P.: Web-scale table census and classification. In: King, I., Nejdl, W., Li, H. (eds.) Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011. pp. 545–554. ACM (2011)
3. Cyganiak, R.: Tarql (SPARQL for tables): Turn CSV into RDF using SPARQL syntax (2019), http://tarql.github.io
4. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 221–243. International Handbooks on Information Systems, Springer (2009)
5. Han, L., Finin, T., Parr, C.S., Sachs, J., Joshi, A.: RDF123: from spreadsheets to RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5318, pp. 451–466. Springer (2008)
6. Lämmerhirt, D., Rubinstein, M., Montiel, O.: The state of open government data in 2017. Tech. rep., Open Knowledge Foundation (2017), https://blog.okfn.org/files/2017/06/FinalreportTheStateofOpenGovernmentDatain2017.pdf
7. Langegger, A., Wöß, W.: XLWrap - Querying and integrating arbitrary spreadsheets with SPARQL. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5823, pp. 359–374. Springer (2009)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013), http://arxiv.org/abs/1301.3781
9. Scassa, T.: Privacy and open government. Future Internet **6**(2), 397–413 (2014)
10. 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/
11. 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 - Results of the LOD2 Project, Lecture Notes in Computer Science, vol. 8661, pp. 175–195. Springer (2014)
12. World Wide Web Foundation: Open data barometer - leaders edition. Tech. rep., World Wide Web Foundation (2018), https://opendatabarometer.org/doc/leadersEdition/ODB-leadersEdition-Report.pdf