MERMAID: Production-level Knowledge Base Construction System

Anonymous authors

Abstract

Knowledge bases play crucial roles in a wide variety of information systems, such as search engines and intelligent personal assistants. For responding constantly fluctuating user information demands, we aim to construct a large-scale and well-structured global knowledge base from the world’s evolving data. In this paper, we discuss enterprise-specific issues with knowledge base construction and present how to deal with these issues in our construction system called “MERMAID”. To maintain the quality of our knowledge base at the production-level, MERMAID is carefully designed to incorporate various automatic and manual validation methods. We partly leverage manual validation methods to deal with business requirements and user feedback quickly since it is difficult to filter out all incorrect facts automatically in practice. Moreover, we propose a novel information extraction method that obtains reliable factual information from Web-crawled data on the basis of distant supervision. Our constructed knowledge base is already utilized in real-world Japanese Web services, and the number of entities in it keeps growing steadily.

1. Introduction

Knowledge Bases (KBs) have been supporting of many user activities such as browsing, searching or buying experiences on the Web. Well-known search engines, such as Google, Bing and Yahoo! Inc., show not only the search results, called 10 blues links, but the information as entity panels provided by their own KBs, i.e., Google KG\(^1\), Satori\(^2\) and Yahoo! KG [Blanco et al., 2013], respectively. Our Japanese KB (JKB) has the same uses as them. Figure 1 depicts the relation between the JKB and the entity panel of “Attack on Titan”, which is a famous Japanese manga and anime.

There are various KBs and systems of constructing them; thus, we distinguish between global KBs and domain-specific KBs as stated in [Deshpande et al., 2013]. Domain-specific KBs capture specific area information, for example [Iannacone et al., 2015], mentioned a cybersecurity KB, EntityCube [Zhu et al., 2009] shows the relationships among people and

---

1. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
DBLife [DeRose et al., 2007] formats database communities. Unlike domain-specific KBs, global KBs, such as YAGO [Suchanek et al., 2007, 2008], Freebase [Bollacker et al., 2008], DBpedia [Auer et al., 2008], Cyc [Lenat, 1995] and Wikidata [Erxleben et al., 2014] attempt to represent any fact and relation to benefit any information system, as described in [Färber et al., 2015].

KB-construction systems differ depending on the application, for example, NOUS [Choudhury et al., 2017] is a domain-specific KB construction system, and many machine-learning-based approaches, such as Fonduer [Wu et al., 2018], have been proposed to maximize the $F_1$ score on the quality of the output knowledge base. We attempted to capture all significant information, such as facts and relations between entities to show the entity panels illustrated in Figure 1 by using our JKB, a global and production-level curated KB which meets a standard of quality. As [G.C. et al., 2015] indicated, industrial systems differ greatly compared to those in academia regarding objectives; therefore, we cannot merely adopt such one-shot KB construction systems. We should merge and match large (hundreds of millions of entities) and heterogeneous (many data sources including in-house, external and structured, and semi-structured) data by using our global-domain ontology. WOO [Bellare et al., 2013] and Kosmix [Deshpande et al., 2013] are end-to-end systems of constructing, maintaining, curating, and using the production-level KBs in Yahoo! Inc. and WalmartLabs, respectively. Similar to WOO and Kosmix, we propose a KB construction
system called Merge and Match Interspersed Data (*MERMAID*). The architecture of this system is shown in Figure 2.

*MERMAID* is designed to handle hundreds of millions of entities covering our wide-ranging domain services (books, movies, companies, landmarks and so on) and constructs a large production-level KB every day. Different productions or applications have different interpretations of production-level KBs; thus we conduct quantitative and qualitative evaluations for every application. On the whole, a production must maintain high precision, and the input data, especially linked open data (LOD), have many incorrect facts and relations (e.g., many facts and interlinks between Freebase and DBpedia are incorrect [Zaveri et al., 2013]); therefore, we carefully designed methods of automatically validating and filtering incorrect facts and manually incorporating curated results.

With our automatic validation and filtering methods, we compare actual and defined domains in our ontology in the same manner as in [Péron et al., 2011] and filter facts and entities by their certainties. One of the difficulties of automatical error checking, broadly speaking, is requiring a rich and formal ontology [Paulheim, 2017]. For this reason, we constantly refined our global-domain ontology to be useful for validation. *MERMAID* constructs a KB whose scheme is similar to Resource Description Framework (RDF) format and has the certainty of each fact. Since the certainties will be summed up by matching
and merging the same entities or facts, MERMAID filters facts and entities by referring to their certainties.

However, it is too difficult to validate and refine using only automatic methods. Since the types of error facts are diverse, we use both automatic and manual validation and filtering methods. With manual methods, we mainly check important entities (i.e., these entities are often displayed as the corresponding entity panels), add the erroneous facts to a blacklist and remove the facts by referring the list. Both automatic and manual validation and filtering methods are essential when constructing KBs in an enterprise because we should accurately validate to maintain the high precision of the JKB.

Besides maintaining the high accuracy, we should improve the number of facts and entities from various types of input data; thus, integrating and de-duplicating entities of the JKB are equally essential and thought-provoking tasks [Köpcke and Rahm, 2010]. We use rule-based and graph-based matching approaches. Rule-based entity matching approaches using the unique and accurate identifiers, such as ISBN, have high accuracy, and many entities do not have any unique identifier. We can match entities, such as those described above, by using graph-based approaches. Graph-based approaches tend to be lower accuracy compared to rule-based ones; thus, we introduce a new clique-based method to maintain matching results at production-level.

Constructing a KB is a constant challenge [Bordes and Gabrilovich, 2014] and importing data from only LOD and content provider (CP) data that private companies manage and have their own right is not sufficient; we take in the information from Web pages to further increase the scale. NELL [Carlson et al., 2010] and Knowledge Vault [Dong et al., 2014] are KBs constructed from Web pages. We introduce a new method for automatic extracting information from semi-structured Websites based on distant supervision [Mintz et al., 2009]. The information populates the JKB.

MERMAID uses many automatic validation, matching and information-extraction methods and imports many types of data such as LOD, CP data, and Web pages. Therefore, manually finding erroneous data is essential because there are various types of errors that automatic methods cannot handle [Paulheim, 2017]. We continuously check the entity panel and exploit our chatbot to answer our simple question, for example, we ask “What is the height of Tokyo Tower?” then the chatbot answer “333m”, for an organization to test the JKB. Since MERMAID is a modify-friendly environment, we can rapidly modify issues due to the dogfooding.

To summarize, our main contributions are as follows:

- We propose a production-level KB construction system: MERMAID, and the JKB constructed with this system is one of the largest in Japanese.
- We describe the challenging issues of constructing KBs, especially in an enterprise, and show how to solve them.
- We introduce and formulate the applicability of a new method of information extraction from Web pages.

This paper is organized as follows: in the following section, we introduce general issues for constructing KBs in an enterprise. In Section 3, we explain, MERMAID, our JKB constructed with this system, and the correspondence between each component of the system and the issues. In Section 4, we give the details of the new method for extracting
information from Web pages. In Section 5, we discuss the detailed metrics of the JKB and the impact of each component. In the last section, we conclude our paper and explain our future work.

2. General Issues Requirements

We should address the following general issues when constructing the JKB by using MERMAID.

I1: Importing large, heterogeneous, and multiple domain data

According to previous studies [Dalvi et al., 2012, Bellare et al., 2013], constructing a comprehensive KB, whose entities and their attributes with high coverage, requires integrating information from a large amount of data sources. We should uniformly incorporate heterogeneous schemed and different quality data.

I2: Improving entity identifiability

If multiple entities are representing the same object in a KB, users cannot identify that object. Therefore many entity-matching systems have been proposed (e.g., [Konda et al., 2016]); however, it is difficult to directly adopt a machine-learning based approach to avoid as much incorrect matching as possible.

I3: Keeping consistent the same entities across time

JKB users expect the previous entity-id to be the same as the current version even though the matching algorithms or the input data change. Due to the assurance, users can associate their objects with the JKB ID.

I4: Automatically improving quality

There are two aspects for improving quality, i.e., removing incorrect facts or adding and complementing correct facts. Since public data contain a large amount of incorrect data, we cannot find most incorrect facts manually. Moreover, different data sources have different granularity of the concept, for example, Wikidata has the written-work entity regarding comics, and CP data have the exact product entity regarding these comics. Thus, linking the two entities of different concepts is useful to use and improve quality.

I5: Constructing a modify-friendly environment

We cannot cover all errors because the input data change by the minute. Therefore, we should set up an environment in which we can use the JKB and check the quality of facts in the JKB regularly, such as dogfooding. Furthermore, rapidly incorporating our and editorial refinements is equally significant.

I6: Handling business requirements and manually examined results

We may face different business issues, such as copyright problems, for different applications; thus, we should extract accredited data.

3. MERMAID

MERMAID works on the Apache Spark\(^3\); thus, we take the operations on distributed computing environment into account. In this section, we briefly describe the system architecture.

\(^3\) https://spark.apache.org/
Table 1: Comparison of the input data regarding the primary information.

<table>
<thead>
<tr>
<th>Data sources</th>
<th>Data amount</th>
<th>Accuracy of extracted data</th>
<th>Complexity to extract data</th>
<th>Update frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikidata</td>
<td>30 GB</td>
<td>Fair</td>
<td>Low (Json)</td>
<td>about every two weeks</td>
</tr>
<tr>
<td>DBpedia (Japanese)</td>
<td>120 GB</td>
<td>Fair</td>
<td>Low (N-Triple)</td>
<td>about once a year</td>
</tr>
<tr>
<td>Freebase</td>
<td>380 GB</td>
<td>Fair</td>
<td>Low (N-Triple)</td>
<td>End at June 30, 2015</td>
</tr>
<tr>
<td>Wikipedia (Japanese)</td>
<td>2.6 GB</td>
<td>Good</td>
<td>Medium</td>
<td>about every two weeks</td>
</tr>
<tr>
<td>CP data (Japanese)</td>
<td>≈ 100 GB</td>
<td>Excellent</td>
<td>Low (Json, tsv, xml, etc.)</td>
<td>every day</td>
</tr>
<tr>
<td>Crawled data</td>
<td>≈ 1000 GB</td>
<td>Bad</td>
<td>High</td>
<td>every day</td>
</tr>
</tbody>
</table>

3.1 Overall Architecture

MERMAID is mainly composed of twelve components shown in Figure 2, and we roughly divide the roles into two groups whether input data are Web crawled data or not. We daily collect and update structured, or semi-structured data such as LOD (e.g., Wikidata, Wikipedia, DBpedia, or Freebase), in-house CP data (e.g., landmark, movie or book data), and Web-crawled data by using our Web crawler. These three types of data have different characteristics regarding their accuracy and the complexity of how to extract key-value information, as shown in Table 1. The system dumps and imports such structured and semi-structured data, other than crawled data, from the Importer to the Exporter through other ten primary components. As shown in Table 1, extracting key-value information from crawled data is another challenge due to the combination of various input documents and variation in extraction targets [Chang et al., 2006]; thus, we design our information extraction (IE) method from Web-crawled data separately from the main components and retrieved the IE data as additional data.

3.2 The JKB Scheme

We introduce a new scheme for KBs and explain its advantages. The JKB is a set of entities. An entity is composed of a unique id, types (e.g., PERSON and WRITTENWORK), and a set of Statements. A Statement is composed of the predicate and a set of Triples. A Triple is similar to the RDF scheme that is composed of subject, predicate and object, and has its certainty and the type of data. Moreover, all the structures preserve the data sources as meta information.

There are two advantages with this scheme as follows: first, we can treat a series of data as one fact because of the Statement structure. For example, a coordinates Statement is composed of a latitude and a longitude Triple. We can also add meta information to each Statement as a Triple, for example, the date when the Triple is added to the JKB. Second, we can maintain the quality of the JKB by filtering these data following the certainty and the type of data. For example, we filter Statements whose certainties are less than 0.3 and values does not match with the defined type of data.
Table 2: Correspondence between issues discussed in Section 2 and primary components of MERMAID. Checkmarks mean component addresses corresponding issue.

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>I6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Entity Matcher</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Entity Linker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ID Assigner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Additional Data Combiner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Attribute Completer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Validator</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Exporter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Information Extractor</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Manual Refiner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JKB scheme</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

3.3 Comparison between Issues and MERMAID Components

We describe the correspondence between the issues discussed in Section 2 and the components of MERMAID. Table 2 summarizes this correspondence. Some components address more than one issue.

3.4 Elementary Functions of Each Component

We explain twelve components of MERMAID and the Manual Refiner briefly. **Importer** is a data feed component. It supports arbitrary input data scheme, except for crawled data, as shown in Table 1, and unifies with the JKB scheme with certainty values. We define the certainty values for every data source or predicate, for example, the certainty of Freebase is lower than Wikidata due to the update frequency. The Importer ensures the id uniqueness.

**Attribute Converter** converts the types and predicates of the input data to our ontology by using each mapping file. For example, we define a mapping from the Wikidata-entity type such as [https://www.wikidata.org/wiki/Q215627](https://www.wikidata.org/wiki/Q215627), to the type defined in our ontology such as PERSON.

**Entity Matcher** outputs entity clusters, which groups the same entities. The Entity Matcher is mainly composed of three steps, rule-based matching, graph-based matching and filtering unnecessary entity clusters and their attributes. First, accurate matching is conducted by rule-based matching. Second, we create blocks of candidate entity clusters with weak matching methods (e.g., name matching) to reduce the computation time. We connect edges between related entity clusters of each block with more strict matching methods (e.g., name and birthdate) than blocking matching and extract cliques from each entity cluster-to-entity cluster graph. Third, the Entity Matcher removes entity clusters whose attribute certainties are zero and unmapped attributes at the Attribute Converter. We describe the detail of the component in Section 5.3.

---

4. [https://plus.google.com/u/0/109936836907132434202/posts/bu3z2wVqcQc](https://plus.google.com/u/0/109936836907132434202/posts/bu3z2wVqcQc)

5. [https://www.w3.org/TR/rdf11-concepts/](https://www.w3.org/TR/rdf11-concepts/)
**Entity Linker** links between two entities where they do not connect each other in spite of their intercorrelations. For example, the Entity Matcher does not match the “Attack on Titan” entity with the Written-Work type from Wikidata and that with Film from CP data because their types differ. The Entity Linker solves the problem by using their attributes and connects them as shown in Figure 1. The difficulty comes from the concept discordance. In a previous study [He et al., 2013], books and written works between two data sources were identified. Our ontology covers the relation between works and products; therefore, we can generalize and solve this problem.

**Id Assigner** assigns a unique ID to each entity cluster based on the set of data sources of entities in the cluster. For example, if the id of an entity cluster composed of two entities derived from “Wikidata Id: 100” and “DBpedia Id: 2000” is “JKB Id: 300”, we preserve two relations as follows: “Wikidata Id: 100” → “JKB Id: 300” and “DBpedia Id: 2000” → “JKB Id: 300”.

We preserve the relations between the id and the data source on the Apache HBase⁶, which is the Hadoop database, as the **Id Table**. The Id Assigner ensures that the same entity clusters compared to the past ones inherit to the past id in the following steps:

1. Extract all past JKB ids of entities in an entity cluster.
2. Extract all past data sources from the above JKB ids.
3. Calculate the ratio of the number of the intersections of past and current data sources to the number of past data sources.
4. If the ratio is greater than 0.5, assign the past id to the entity cluster; otherwise, assign a new id and update the Id Table.

The id is persistent across the time with the Id Assigner.

**Additional Data Combiner** combines entities from the additional data whose scheme is the same as the JKB’s; thus, the Additional Data Combiner incorporates these entities into entity clusters from the Id Assigner by using their ids.

**Entity Merger** merges one or more entities in each entity cluster into one entity whose certainties of all attributes are summed up by certainties of the same attributes. The Entity Merger is a simple but powerful component because the Validator can filter untrust attributes accurately due to the summed certainties.

**Object Converter** converts the object of each Triple to the corresponding entity-id by referring the Id Table. If the Triple is derived from linked data and the object is described as the specific identifier, it is easy for the Object Converter to convert the object to the corresponding entity-id by using the Id Table. Since these literals represent any object, linking such objects to the corresponding entities is similar to entity-disambiguation problems.

Therefore, we should resolve the entity ambiguity from two viewpoints, (1) the type consistency between the predicate and our ontology and (2) distinguishing different objects with the same name and type, such as a person’s name. We solve the first problem by comparing our ontology with the range type of the predicate and second problem by narrowing down the target to accelerate and reduce the number of incorrect interlinks. This is our long-term challenge.

---

⁶. [https://hbase.apache.org/](https://hbase.apache.org/)
**Attribute Completer** complements attributes to entities based on our ontology and their URL attributes. First, it complements attributes by using a symmetric property defined in our ontology such as the `inverseOf` property of `owl:inverseOf`. Second, it extracts useful information from the entity-related URL, for example, OGP images are useful attributes of these entities. The Attribute Completer partially addresses a well-known challenge; knowledge base completion [Socher et al., 2013].

**Validator** removes invalid data based on blacklists created via the Manual Reﬁner, inconsistency between Triples and our ontology, and fact checks the results using crawl data (e.g., URLs are deadlink or not). It also rewrites values to standardize deﬁnitions of the JKB, for example, the phonetic characters are uniﬁed to hiragana (one of the writing systems of Japanese). We describe the details of the component in Section 5.4.

**Exporter** ﬁlters and corrects Triples to avoid service-speciﬁc issue, such as copyright problems, and outputs the JKB to a well-known format (e.g., JSON or N-Triples).

**Information Extractor** extracts factual information from a large set of Web-crawled data. There has been extensive work on IE from Web data [Chang et al., 2006]. We regard each Web page as a Document Object Model (DOM) Tree and collect all DOM path patterns related to a predicate by using the JKB.

We now explain details of the Information Extractor. The Information Extractor uses novel method for extracting information from semi-structured data based on distant supervision [Mintz et al., 2009]. First, we ﬁnd path patterns from the subject to the object with conﬁdence scores. Path patterns are extracted from DOM trees of a large amount of Web-crawled data and the JKB. Second, we extract new information from the extracted

---

7. https://www.w3.org/TR/owl-ref/#inverseOf-def
8. http://ogp.me/
Anonymous authors

path patterns and merge the confidence score with our defined function. Figure 3 illustrates an overview of the method of the Information Extractor. We describe the details of this method in Section 4.

Manual Refiner is not one of the components of MERMAID but it is equally significant system for maintaining high accuracy and constructing a modify-friendly environment for I5 and I6. It can handle corner cases that are difficult to remove or refine automatically. First, we find the incorrect facts from user feedback and our quantitative evaluation and examine if these facts are derived from a business requirement or rare cases or not. If these facts are corner cases and required rapid modification, we add them to the blacklist. Second, we create a new Triple with which we cannot import data due to the lack of information, such as images, without any additional information. The quality of JKB improves with the Manual Refiner. In the future, we use machine-learning based approaches by using the manually annotated data as training data sets.

4. Information Extraction from Web Crawled Data

In this section, we discuss our IE method. We use Web pages as DOM trees, as in [Lockard et al., 2018], and extract useful paths from all DOM trees with the confidence score. Next, we extract a new Triple by using the extracted paths and merge the confidence scores with our merge function. Figure 3 illustrates the overview of our IE method.

4.1 DOM Path Extraction from a Web page

We regard a Web page as a sequence of tags by depth-first search of the DOM tree. We define a kth Web page as a sequence of tuples \( W_k = ((t_{k,1}, v_{k,1}), \ldots), \) where \( t_{k,i} \) and \( v_{k,i} \) are defined as the ith tag and text of the kth Web page, respectively. For example, Web page 1 in Figure 3 is represented as \( W_1 = ((dl, ""), (dt, "Person A"), (dd, ""), (a, "http://XXX/AAA"), \ldots). \)

We define a sequence of tags as \( W_T_k = (t_{k,1}^g; \ldots). \) All seed data \( SD \) are defined as a set of Triples denoted as \( SD = \bigcup_m (s_m^g, o_m^g) \) where \( s_m^g \) and \( o_m^g \) are mth subject and object texts, respectively. Given a kth Web page and mth Triple data, we define the information path \( P_{k,m} \) as follows:

\[
P_{k,m} = \begin{cases} (t_{k,i}, t_{k,i+\delta(i,j)+1}, \ldots, t_{k,j}) & \text{if } \exists i, j \in \mathbb{Z} \text{ s.t. } (v_{k,i}, v_{k,j}) \in SD, \\ () & \text{otherwise,} \end{cases}
\]

where

\[
\delta(i, j) = \begin{cases} 1 & i \geq j, \\ -1 & i < j. \end{cases}
\]

Given the kth Web page, a set and a multiset of information paths are defined as \( P_k = \bigcup_m P_{k,m} \setminus \{(\)\} and \( P_k^m = \bigcup_m P_{k,m}, \) respectively. We define the tuple of an information path and the confidence score as follows:

\[
PC_k = \{(p', \frac{n(p', P_k^m)}{n(p, C_{P'}(WT_k))}) \mid p' \in P_k\}
\]

\[
= \{(p', cs_{k,p'}) \mid p' \in P_k\},
\]

Anonymous authors
where \( n(u, S) \) is number of occurrences of a content \( u \) in a set or a multiset \( S = \{s_1, \ldots, s_{|S|}\}, \) 
\( C_n(S) = \{(s_i, \ldots, s_{i+n-1}) | i \in \{1, \ldots, |S|\}\}, \) and \( cs_{p,k} \) is the confidence score of path \( p \) in the \( k \)th Web page. For example, \( PC_1 \) in Figure 3 is \{((dl, dt, dd, a), \frac{1}{2})\}.

4.2 New Information Extraction from Web pages

Given two confidence scores \( P_A \) and \( P_B \), we define the new associative binary relation between these scores as follows:

\[
P_A \oplus P_B = \frac{P_A P_B}{P_A P_B + \alpha(1 - P_A)(1 - P_B)},
\]

where \( \alpha \) is a hyperparameter for the probability that two paths from two other Web pages are the same and the extracted fact is incorrect. We set, in practice, the hyperparameter \( \alpha \) to 0.1 by the preliminary experimental results. The above binary relation satisfies an associative law. Given a set of scores \( SC \), we define the sum of \( \oplus \) among \( SC \) as \( \bigoplus_{i,j \in SC} i \oplus j \).

We define a multiset of all pairs of a path and the confidence score as \( PC_m = \bigcup_k PC_k \).

The set of new data from a Web page \( W_k = (\{t_{k,1}, v_{k,1}\}, \ldots) \) and a \( p \) is denoted as \( D_k(p) = \{(t_{k,i}, v_{k,i}) | (t_{k,i}, \ldots, t_{k,j}) \in W_k \text{ s.t. } p = (t_{k,i}, \ldots, t_{k,j})\} \setminus SD \). All multisets of a new Triple and the merged confidence score is defined as follows:

\[
TCS = \{(t, cs) | (p, cs) \in PC_m, t \in \bigcup_k D_k(p)\},
\]

for example, \( TCS \) in Figure 3 is \{((Person C), \frac{1}{2}), ((Person C), \frac{2}{3})\}. We extract a set of new Triples and the merged confidence score as follows:

\[
ND = \{\left( t, \bigoplus_{t'=t'} cs_{t'} \right) | (t', cs_{t'}) \in TCS, t \in \bigcup_k D_k(p)\},
\]

for example, \( ND \) in Figure 3 is \{((Person C, https://ZZZ/CCC), 0.9523)\}.

The new data \( D_k(p) \) does not extract Triple from the Web page containing only one fact, because \( D_k(p) \) depends on a Web page and an information path. We share the path patterns from a Web page with other Web pages whose URL host is the same. The amount of extracted Triple data with the same accuracy as our method without sharing among the same host URLs increases by about 75%.

5. Experimental Results

In this section, we explain the results from experiments we conducted regarding issues discussed in Section 2. First, for I1, we compared the size of other popular global knowledge bases and show the results from our IE methods, which are presented in Section 4. Second, we show the impact of the Entity Matcher and persistence of the Id Assigner for I2 and I3, respectively. Third, we evaluated the increase and decrease in the number of entities and facts for improving the quality I4.

5.1 Overall JKB Results

Figure 4 shows the transitive graphs of the number of entities and Triples of the JKB each month for eighteen months. The JKB steadily increased the number of entities. We took in
Figure 4: Both of figures are transitive graphs for 18 months. Left is the number of entities in the JKB. Right is histories of the number of Triple in the JKB and excluded invalid Triple data.

Figure 5: Distribution of the number of extracted information every the confidence score. A large CP data between 14 and 15th month; thus, the transitive graph in Figure 4 shows the sharp growth. Since the quality of validation and filtering methods has been increasing, the number of Triples changes monthly.

5.2 Information Extraction Results

We extract official websites from crawled Web data by using the Information Extractor described in Section 4. Extracted information have their confidence scores, and Figure 5 depicts the distribution of the number of extracted information divided by their confidence scores.

We evaluate extracted websites whose confidence score is 0.8 and over to our products (e.g., entity panel as shown in Figure 1), Moreover, we select the results that do not match with the existing JKB, namely, newly extracted websites as shown in Table 3 as the target for evaluation. We check the selected results are official or not with our strict guarantee of quality. For example, if the extracted website is closed or is a part of the official site,
Table 3: The number of extracted official websites whose confidence score is greater than 0.8 by the Information Extractor and the precision of them. We evaluate the precision about the extracted websites which do not appear in the JKB and the precision of the whole extracted websites.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision of all values</th>
<th>Precision of newly crawled data</th>
<th># of extracted values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landmark</td>
<td>89.0%</td>
<td>67.0%</td>
<td>9,022</td>
</tr>
<tr>
<td>Person</td>
<td>77.3%</td>
<td>60.0%</td>
<td>5,530</td>
</tr>
</tbody>
</table>

Figure 6: History of id-consistency ratio between the current and the past JKBs.

Figure 7: Comparing with the number of matched entities after rule-based matcher and graph-based matcher.

we regard the website is incorrect. The precision does not satisfy our production-level; however, the Validator and the Exporter accurately remove such incorrect data. For this reason, the JKB achieve the 99% precision.

In the future, we improve the precision and the number of extracted values with high confidence scores; besides, we plan to extract various types of values from Web-crawled data.

5.3 Identifiability Results

We first compared two JKBs separated by a week and confirmed that only 0.0004% of entities changed their ids. We also observed that more than 94% of the entities did not change their ids, as shown in Figure 6. Since we stopped importing some data sources and the Validator filters more and more invalid entities, some entities have deleted from the JKB. That is the reason why there are about 6% of entity-ids are inconsistent to the current entity-ids.

Second, we show the number of matched entities. The Entity Matcher uses two algorithms as follows; (1) Rule-based matching matches entities whose Wikipedia URL or some identifiers, such as IMDb and (2) Graph-based matching matches entities with their types, names, and reliable attributes (e.g., birthdate or coordinates). We show the number of
matched entities in Figure 7. Due to the graph-based matching, the number of matched entities increases. Graph-based matching does not match entities derived from the same data sources to improve the precision. For this reason, the precision of matching results is about 99%.

5.4 Automatic Validation and Completion Results

The Validator automatically removes many invalid Triples to maintain the quality of the JKB, as shown in the right side of Figure 4. The Validator filters (1) facts whose domain types are inconsistent with the type of objects, (2) facts that are functional\(^9\), (3) facts whose data types do not match objects (e.g., if the data type is URL, the value must start with “http”), and (4) facts whose values do not satisfy the format (e.g., date or ISBN). We observed that 97.7, 2.0, 0.2, 0.1% of all validation data are (1), (2), (3), and (4), respectively. Note that a large amount of the invalid data of (1) is mainly derived from unmapped entities from the LOD to our ontology; thus, we can reduce the number of filtering entities, and this validation is one of the reasons that the JKB maintains high accuracy.

The Entity Linker and Attribute Completer complement facts to the JKB. Complemented facts account for about 1.4% of all facts in the JKB. About 10% of complemented facts from the Entity Linker links two related entities, and others are derived from the Attribute Completer.

6. Conclusion and Future Work

We presented Merge and Match Interspersed Data system that constructs scalable and production-level curated KB. The system address enterprise-specific issues, namely, importing large and heterogeneous data, such as LOD, CP data and Web-crawled data, increasing the number of entities with high precision, and satisfying business requirements. We show how to address the issues, and the impact of each component of MERMAID. Our constructed knowledge base, JKB, is one of the largest KB in Japanese and already utilized in

---

8. [https://www.w3.org/TR/owl2-syntax/#Functional_Data_Properties](https://www.w3.org/TR/owl2-syntax/#Functional_Data_Properties)

14
real-world Japanese Web services. Figure 8 depicts the relation among types of the JKB related to the PERSON type.

In the future, we plan to solve the following challenge: (1) Give more proper certainties to all facts and improve the automatical fact-checking systems. (2) Understand how each component affects the quality of the JKB. (3) Improve the accuracy and recall of the Entity Matcher and the Information Extractor using more sophisticated methods.

References


Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data 


