A Text Similarity-based Process for Extracting JSON Conceptual Schemas

Fhabiana T. Machado, Deise Saccol, Eduardo Piveta, Renata Padilha and Ezequiel Ribeiro
Federal University of Santa Maria, Santa Maria, Brazil

Keywords: JSON, Schema Extraction, Information Integration.

Abstract: NoSQL (Not Only SQL) document-oriented databases stand out because of the need for scalability. This storage model promises flexibility in documents, using files and data sources in JSON (JavaScript Object Notation) format. It also allows documents within the same collection to have different fields. Such differences occur in database integration scenarios. When the user needs to access different datasources in an unified way, it can be troublesome, as there is no standardization in the structures. In this sense, this work presents a process for conceptual schema extraction in JSON datasets. Our proposal analyzes fields representing the same information, but written differently. In the context of this work, differences in writing are related to treatment of synonyms and character. To perform this analysis, techniques such as character-based and knowledge-based similarity functions, as well as stemming are used. Therefore, we specify a process to extract the implicit schema present in these data sources, applying different textual equivalence techniques in field names. We applied the process in an experiment from the scientific publications domain, correctly identifying 80% of the equivalent terms. This process outputs an unified conceptual schema and the respective mappings for the equivalent terms contributing to the schema integration's problem.

1 INTRODUCTION

Due to the increasing volume of data generated by various applications, NoSQL document-oriented databases models were created. Their main characteristics are schemaless and there are no complex relationships.

Despite the allowed schema flexibility, it is a misconception to state that a schema does not exist. When using an application to access a NoSQL database, it is assumed that certain fields exist with a certain meaning and type. In this sense, there is an implicit database schema: a set of assumptions about the data structure in the code that manipulates it.

This storage model allows, for example, that a field can be present in some documents and in others not, or that there are fields with distinct names, including between documents belonging to the same collection. In this way, there may exist different fields of the same domain that represent the same information, as occurs in integration scenarios of JSON datasets, as presented at Figure 1.

The example in Figure 1 points to some information with the same meaning, but represented quite differently: (1) `scores`, line 4A, and `score`, line 4B,

Document A

```
1 {
2   "student":0,
3   "class_id": 2,
4   "scores": [{
5     "average": 8.7
6   }]
7 }
```

Document B

```
1 {
2   "learner": 1,
3   "id_class": 2,
4   "score": [{
5     "average": 8.7
6   }]
7 }
```

Figure 1: Motivating example.

present only one difference in the plural. This type of change can occur with other suffixes and language prefixes; (2) `class_id`, line 3A, and `id_class`, line 3B. The difference is that their terms are written in reverse order, that is, one with the word "id" at the beginning and another at the end of the word; (3) `student`, line 2A, and `learner`, line 2B, however, have different spellings, i.e., with synonymous words.

Other works on schema extraction in JSON data sources, described in Section 3, are not concerned with different spellings for equivalent fields or not produces a conceptual schema. This issue become important once a document oriented NoSQL database has a flexible schema and could not have standardized dataset.

Therefore, the purpose of this work is to extract

Machado, F., Saccol, D., Piveta, E., Padilha, R. and Ribeiro, E.
A Text Similarity-based Process for Extracting JSON Conceptual Schemas.
DOI: 10.5220/0010475102640271
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 1*, pages 264-271
ISBN: 978-989-758-509-8

the implicit schema in JSON data sources, identifying equivalences between fields that are written differently but representing the same information. In the context of this work, being equivalent covers linguistic similarity like characters, word stems and semantic approach like synonyms resulting a unified conceptual scheme and mappings. This work it is validated through an experiment that has as input documents from digital libraries belonging to the domain of scientific publications. At the end, recall and precision indexes are discussed.

This paper is organized as follows. Section 2 describes the background. Section 3 related works. Section 4 defines the process for extracting conceptual schema from JSON datasets. Section 5 specifies an experiment. Finally, Section 6 presents the conclusions.

2 BACKGROUND

Schemas in NoSQL Datastores. The concepts of entity, attribute and relationship have some specificities: an entity can be an abstraction of the real world (Varga et al., 2016), as well as correspond to a JSON document or object. An attribute can be represented by fields of mono or multivalued arrays. NoSQL databases have no explicit relationships; they use a nested entity model, inferring implicit relationships.

Conceptual representation for NoSQL. The IDEF1X (Integrated DEFinition for Information Modelling) notation with adaptations characterize the modeling suitable for NoSQL models (Benson, 2014) (Jovanovic and Benson, 2013). This is done through rules for representation. Its model is distinguished by the fact that entities derive from a root.

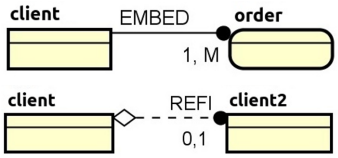

Figure 2: Example of notation for aggregate-oriented data models.

First item in Figure 2 points to a built-in aggregate of one-to-many $1 : M$ relationship. The second item illustrate cardinality relationships $0 : 1$. The "REFI" operator is positioned in the entity where only one reference value will be "copied", that is, referenced in the other entity, thus generating a kind of connection between the entities.

Techniques for Identification of Textual Equivalence. The following measures of text similarity calculation are quite distinct, and, in general, calculate a score in the range $\{0, 1\}$ for a pair of values.

String-based similarity measures operate on sequences and character composition, comparing the distance between two texts. The Levenshtein measure (Levenshtein, 1966) was chosen, as it is widely used. The process, however, can be used with other character-based similarity measures.

Knowledge-based measure is based on the similarity degree according to the information meaning in a semantic network. One of the most popular tools in this regard is Wordnet (Miller, 1995). In this work, we use the Lin (Lin, 1998) measure, which applies the information content of two nodes using the LCS (Lowest Common Subsumer).

Although stemming is not a measure of similarity, the Porter Stemming Algorithm (Porter, 2006) is applied through the equivalence of their stem indicating equivalence or not.

3 RELATED WORK

Related works on schema extraction in JSON data sources are concerned with version inference (Ruiz et al., 2015), extraction of a schema with cardinalities in a proper representation format (Klettke et al., 2015) and a unique semantic approach (Kettouch et al., 2017). However, they do not address the issue of different spellings for equivalent fields. This issue become important in integration scenarios, scheme flexibility or for lack of standardization. In (Blaselbauer and Josko, 2020) work, a linguistic approach is used but generates only similarity graphics.

The work methodology uses a four-step process that starts with JSON documents. Searches for fields that have different spelling, but represent the same information in the schema. This is accomplished through techniques that identify equivalence in field names, generating as output a unified conceptual representation of the implicit schema present in the database.

4 A PROCESS FOR SCHEMA EXTRACTION

In order to access JSON documents in a unified way, we propose a process to schema extraction analyzing textual similarity of its fields. The process is applicable to documents from the same domain to identify

fields representing the same information but written differently.

As input, the process receives documents stored in NoSQL in JSON format. As output, it generates the conceptual representation of an unified schema.

Some definitions are given as follow:

Definition 1. Structure. In a json format $Json := [key_a : value_a, ...]$, structure refers to the fields of a JSON document where $Struc := [key_a, key_b, ...]$, in the sense of differentiating what is not a value.

Definition 2. Schema. Schema has a more comprehensive sense than structure, as it is usually related to entities, relationships and attributes.

Definition 3. Delimiters. A set of JSON grammar symbols that mark the beginning and the end of an object or array, this is $Delimiter := ([,],\{,\})$.

Definition 3. Block. A block is a set of attributes.

The extraction process process is divided into **four** main steps.

The **pre-processing** in Section 4.1, stage initially aim at preparing the files, eliminating the data, keeping only the fields. After pre-processing, the word set receives similarity techniques: knowledge based string, character based string and stemming. Its aim is identifying equivalences in fields that represent the same information but are written differently. This is the **similarity analysis** step, Section 4.2.

After the similarity analysis, the next step is the **identification of equivalences**, Section 4.3. This phase is responsible for analyzing the results of the resulting measures, testing and inferring the equivalence between the terms. Finally, a visual representation of the conceptual schema is generated along with a table representing the mappings of the terms that have been consolidated. This is the step of **structure representation** in Section 4.4.

4.1 Pre-processing

The pre-processing step, shown in Figure 3, prepares the documents for the following steps. All the files in the collection are traversed by joining the fields in a single file. The repeated terms are eliminated by keeping only the distinct fields. This process aims to decrease the number of comparisons, improving efficiency.

The extract fields activity is divided into two steps. The first one separates only the fields, that is, the $[key_a, key_b, ...]$ portion of the $Json := [key_a : value_a, ...]$. The second one merges the fields into a single file. The source document is stored in a list of references to enable future mappings. The following step is described:

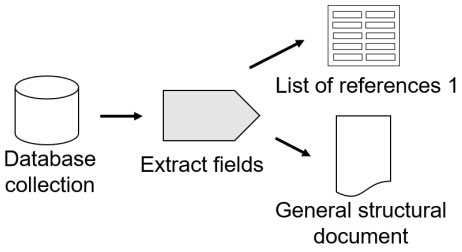

Figure 3: Pre-processing step.

Input. An existing JSON document collection belonging to the same domain. Each document must be validated by the grammar.

Extract Fields. This activity has the purpose of traversing the documents by separating only the distinct fields, merging into a single file and saving a list of references.

The extract fields activity is detailed in other subactivities:

Separate Fields from the Data. This activity processes the documents keeping only the field names and the delimiter symbols $doc_i[key|delimiters]$. This is done for each document, generating a unique text file. At this stage no comparison is made.

Merge Structure. In this step, the intention is to avoid element redundancy $[key_a, key_a, key_b, ..]$, to reduce the number of entries for analysis of textual similarity. Thus, it performs tests and comparisons to merge and to maintain only the distinct fields. In this process, the source reference of the different fields is saved $value_a := [doc_i, doc_j]$.

Output. The first output artifact is a text-format file containing a single document representing the collection called the *general structural document* $Out_1 := doc_i[key|delimiters], doc_j[key|delimiters], ...$ This should have, besides the delimiters that will assist in the future steps, the fields with distinct names. The hierarchy remains the same.

The second artifact is a list that contains the term and the reference to which original documents it contains $Out_2 := distinct\_value_a[doc_i, doc_k], ...$

4.2 Similarity Analysis

The purpose of the similarity analysis step, indicated in Figure 4 is to apply different techniques of text similarity analysis to identify words that represent the same information, but have different spellings.

There are applied three different techniques to each pair of words that compose the general structural document.

Input. General structural document.

Similarity Analysis. In this activity, the input file is copied and serves as the initial artifact for the applica-

Figure 4: Similarity analysis step.

tion of the three different techniques. These are executed in parallel. The input passes through each technique and generates result matrices with values indicating the similarity. The comparisons are made in the form all with all, for earch pair $[key_a, key_b] \in Out_1$, assuming that all can be equivalent.

The similarity analysis is detailed in other sub-activities:

Analyze stem of the word. This sub-activity refers to the technique of Stemming, that is, stem extractor. It is executed using the Porter Algorithm (Porter, 2006). By applying rules to remove suffixes and prefixes from the English language, the result is a radical one. This is not necessarily a valid word and may have no meaning.

In the case of the stemming technique, the stem of the words $stem\_key_a$ and $stem\_key_b$ are compared to define equivalence. If it is identical, the result will be 1, as shown by Equation 1, otherwise it will be 0. Thus, in this technique, the resulting values will always be integers 0 or 1.

$$Stem_{sim} : (stem\_key_a == stem\_key_b) \rightarrow 1 \quad (1)$$

Analyze Character-based Similarity. It consists of applying some similarity technique. The Levenshtein (Levenshtein, 1966) function was chosen because it was applied to short texts, through the minimum number of operations required to transform one string into another and is represented by Equation 2.

$$Lev_{sim} : (key_a, key_b) \rightarrow \{0,1\} \quad (2)$$

Analyze Knowledge-based Similarity. It targets the semantic focus. The implementation of the Wordnet Similarity for Java is used applying the semantic measure Lin (Lin, 1998) that is based on information content and represented by Equation 3.

$$Lin_{sim} : (key_a, key_b) \rightarrow \{0,1\} \quad (3)$$

In this case, the word indicated by key_i must be valid in Wordnet, otherwise the result will be assigned a zero value.

Output. Three result arrays are generated with integer values in the range from zero to one. Each array contains information about the result of a given similarity function for each pair of words. The corresponding output $Out_3 := M_{stem}, M_{lev}, M_{lin}$ is represented in an array whose main diagonal will always be one. This also has the characteristic of being symmetrical both below and above the main diagonal.

After the end of the similarity analysis step, the three generated arrays follow to identify equivalences.

4.3 Identification of Equivalence

The step of identifying equivalences, shown in Figure 5, is decisive in the schema unification process.

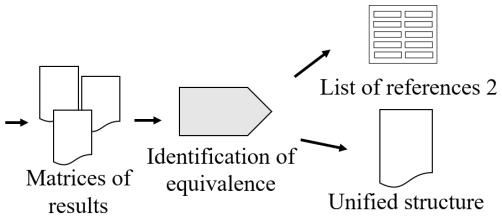

Figure 5: Identification of equivalence step.

Its purpose is to perform tests and comparisons to define the equivalence between fields. It uses the values resulting from each measure applied in the previous step.

Input. Matrices of results of applied techniques.

Identify Equivalence. In this activity, calculations and tests are performed from a threshold. Afterward, these results are synthesized in a unified structure. In this same process are stored the consolidated fields and to which others correspond.

The identify equivalence activity is detailed in other sub-activities:

Calculate Equivalence. This sub-activity represented by Algorithm 1, is performed for each pair values. Tests, comparisons and calculations are performed and generate a single matrix M_{res} where $M_{res} \in \{0,1\}$.

Algorithm 1: Calculate equivalence.

```
   input : M_stem, M_lev, M_lin
   output: M_res
1  for e_ij ∈ M_stem and M_lev and ,M_lin do
2      if e_stem or e_lev or e_lin == 1 then e_res = 1 ;
3      if e_stem or e_lev or e_lin == 0 then e_res = 0 ;
4      if e_stem == 0 and e_lin == 0 then
5          if e_lin > T_a then e_res = 1 ;
6      end
7      if e_avg_res > T_b then e_res = 1 ;
8  end
```

This sub-activity is based on the following premises:

- When a word is not valid according the the grammar, it will not have a value in the radical and synonym measures, that is, $Stem_{sim} : (a,b) = 0$ and

$Lin_{sim} : (a,b) = 0$. In this case, the threshold T_a is defined to the character measure.

- When the three measures have values in the $\{0,1\}$ interval, the equivalence is given through a weighted average Avg with threshold T_b where weights are an arbitrary choice.
- For the definition of the threshold, the following test was performed: given a set of 14 word pairs, where 10 are equivalent and 4 are not, the recall and precision indices were calculated for combinations of the thresholds T_a and T_b, according to Table 1.

Table 1: Threshold definition tests.

T_a	T_b	Recall	Precision
0,75	0,50	0,636	0,700
0,70	0,50	0,909	0,714
0,70	0,45	0,909	0,667
0,60	0,50	1,18	0,591

Thus the defined threshold were $T_a = 0,70$ and $T_b = 0,50$.

- When the weighted average is applied, it is represented by the Equation 4. If $Avg \geq T_b$ then its considered equivalent.

$$Avg = (1 * Stem_{sim} + 2 * Lev_{sim} + 3 * Lin_{sim})/6 \quad (4)$$

Consolidate Structure. This sub-activity aims to generate a single listing of fields unifying those that were considered equivalent in the previous step.

It has as input a M_{res} analyzing each pair of words (key_a, key_b) corresponding to the element $e_{res} \in \{0,1\}$. When $e_{res} = 1$ the key_a is kept and the mapping is saved. The element to be maintained is chosen arbitrarily, being considered the first occurrence.

This sub-activity generates a second list of references with the terms and their equivalents $Out_4 := key_b \iff key_a$.

Rebuild Structure. This sub-activity rearranges the consolidated terms into a single document, called unified structure, using Out_4. This sub-activity is based on the following premises:

- The unified structure is built from a collection document considered base.
- The largest document belonging to the entry collection is chosen as the basis, as it contains the largest number of fields.
- Base document delimiters are kept/added in the unified structure and field values are ignored.

From the base document, each term is analyzed and replaced by its equivalent if necessary. Terms that have been consolidated but are not present in the base document are added to the end of the structure. Its generates the unified structure, that is, $Out_5 := uniq\_doc[keys, delimiters]$.

Output. The activity identify equivalence consists of more extensive and represents the core of the work. Two outputs are generated: Out_4 - a second list of references containing equivalences between fields and Out_5 - a unified structure.

After this step, it is necessary to generate a visual conceptual representation, as well as to organize the mappings.

4.4 Structure Representation

The structure representation step, shown in Figure 6, aims to generate a visual notation and the mappings. The first is accomplished through the unified structure by applying conversion rules. The second generates the mappings of the consolidated terms to their equivalents and their source documents.

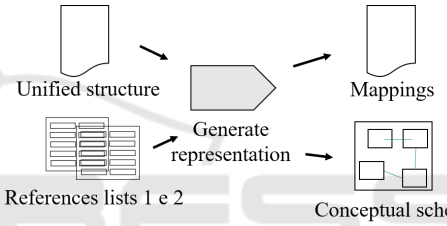

Figure 6: Structure representation step.

Input. Unified structure, that is Out_5 and references lists corresponding to the Out_2 and Out_4.

Generate Representation. This activity produces the conceptual schema and the mappings of the consolidated terms to the equivalents identified in the process. The generate representation activity is detailed in other sub-activities:

Generate Mappings. This sub-activity aims to consult both lists of references to infer new equivalences between the fields, generating a final table with the consolidated term path represented by the JSONPath notation (Goessner, 2007), the other matching terms and occurrences in documents.

This activity is based on the following premises:

- The consolidated term path column is generated from the unified structure.
- The matching terms column is generated based on Out_4 and are checked new equivalences like $A = B, B = C \Rightarrow A = C$
- The occurrences in documents column is generated based on Out_2.

Adapt to Aggregate Notation. This sub-activity create the representation through IDEF1X notation adapted

to the NoSQL model. It applies the defined rules of conversion, described in Section 4.4.1, in a text file that corresponds to the unified structure.

An algorithm, represented by Algorithm 2, similar to a parser is applied whose delimiters indicate the transformation into a class or attribute, for example.

Algorithm 2: Adapt to aggregate notation.

input : $Uniq\_doc[key, delimiters]$ and Rules
output: Schema
for $event \in Uniq\_doc$ **do**
  switch *event* **do**
    case { **do** $R1 \Rightarrow$ new class;
    case key_i **do** $R2 \Rightarrow$ attribute;
    case *[* **do** $R3 \Rightarrow$ new class with one attribute;
    case *[{* **do** $R4 \Rightarrow$ new class;
  end
end

This sub-activity is based on the following premises:

- Some rules have been defined for converting the unified structure into a conceptual scheme.
- The first object in unified structure is considered root entity.
- The delimiters assist in the assembling the conceptual scheme and also indicate the type of relationship.

4.4.1 Rules

This new artifact, also represented by Algorithm 2, presents some rules defined during the work, to generate the conceptual schema and a visual representation. These are used in the identification of the blocks in objects or arrays that help in the definition of classes, attributes and relationships.

The following are briefly described:

- The *R1 Rule* is applied to generate a new class with relationship $0:1$. This rule occurs when identifying '{', i. e., the beginning of a JSON object.
- The *R2 Rule* transforms fields into corresponding attributes of a class.
- The *R3 Rule* is applied to generate a new class with relationship $O:M$. This class contains only a single attribute represented by *att*. Occurs when was found '[', i.e., an array with enumeration of items inside is opened.
- The *R4 Rule* also generates a new class with relationship $0:M$. Occurs when identifying '{[', i.e., the beginning of an object with many arrays.

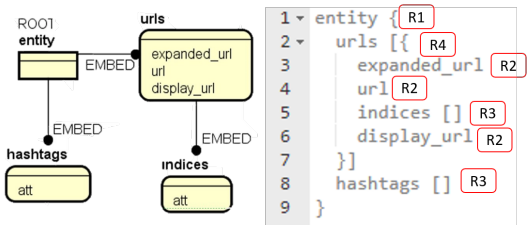

Figure 7: Example of applying the rules.

To illustrate the application of the rules, the Figure 7 is displayed that demonstrates each case described.

Output. This activity ends the extraction process generating the two final artifacts: the mappings and the unified conceptual schema.

5 RESULTS

This section presents the extraction process in an experiment and the results found. The application domain is related to scientific publications; entries are exported references from academic libraries files in JSON format. The execution of this extraction process treats fields as a whole.

Evaluation. Precision and recall measures are used. The recall is the proportion of the total of similar existing pair that appears in the final result. On the other hand, precision indicates the proportion of pairs of values correctly identified as similar that appear in the result.

5.1 Execution of the Extraction Process

The process consists of executing the sequence of sub-activities presented in the four steps: preprocessing, similarity analysis, equivalence identification and structure representation.

A implementation was developed in Java language with libraries such as Simmetrics-core and Wordnet. The project, executable and test files are available on GitHub[1]. It accept as input, the folder containing the JSON documents and the thresholds definition. Outputs a list of references, that is Out_4 and a matrix results M_{res}. The process is terminated manually and produces a visual representation.

The extraction process is based on the following premises:

- It is considered that the fields in general mode can be equivalent to any other, regardless of the hierarchy level that they are and that represent information of the same context.

[1]https://github.com/ftmachado/schema-similarity

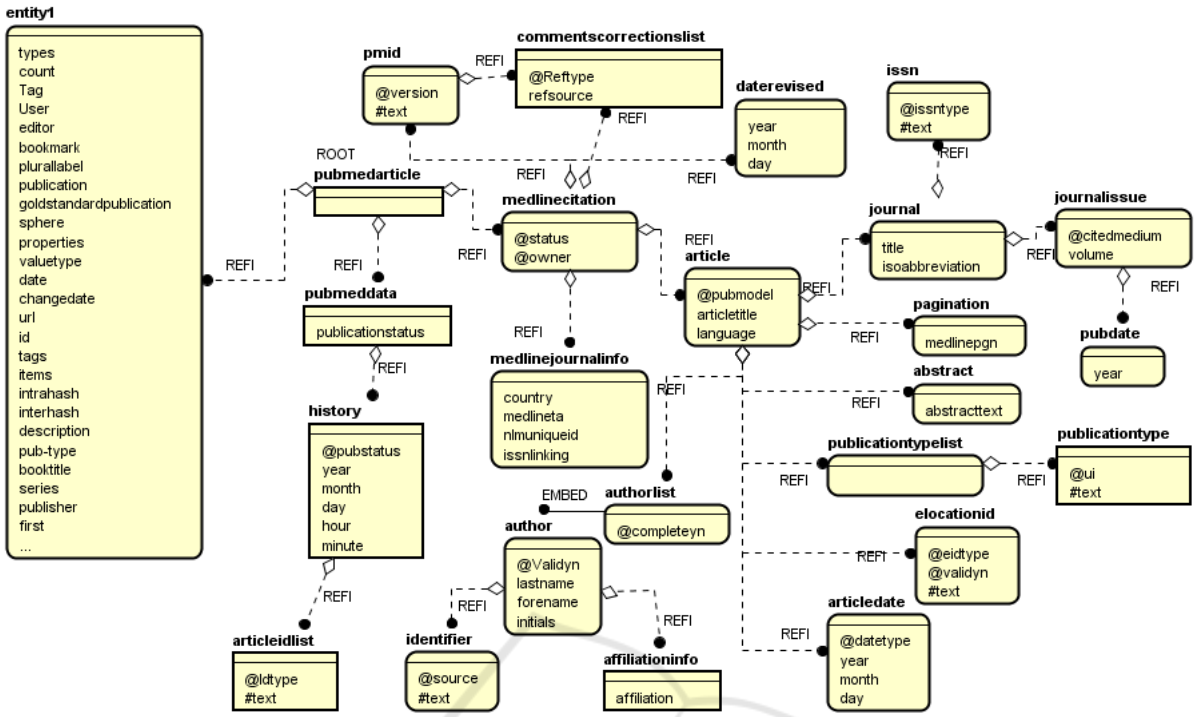

Figure 8: Conceptual schema extracted by the process.

- The data hierarchy in this case is not being considered, as it would be necessary to have a domain expert to assist the matching.

This experiment was carried out with the extraction of 50 files from each library, namely: Bibsonomy, DBLP and Pubmed. These inputs and the thresholds were submitted to our implementation, that shows which fields are considered equivalent.

In the **pre-processing step**, the fields are separated and consolidated into a single text file, called the general structural document. Also generate the list of references that contains the fields and the reference of which documents occur.

Thus, in the **similarity analysis** step, matrices are generated for each analysis. This step results in three matrices with the results of each field pair textual similarity technique applied.

The next step of the process is **identify equivalences**. It calculates equivalences resulting an matrix with zero or one values and a second list of references containing terms that were considered equivalent.

Table 2: Output list of references 2.

types = type	Publication = Issue	Tag = label
count = number	Hour = Minute	User = user
author = authors	author = Author	title = Title
editor = editors	volume = Volume	year = Year
journal = Journal	number = Issue	

Table 2 is the output of list of references. The subactivities of consolidating structure and rebuilding the structure are performed manually, where the source document containing the largest number of fields is chosen for base.

Thus, the **structure representation** phase generates the two final artifacts of the extraction process: the mappings and the conceptual schema.

Table 3: Mapping result example.

Consolidated term path	Matching term	Occurr
$..pubmedarticle.		Doc_1
medlinecitation.	Year	Doc_2
datecreated.year		Doc_3

The mappings, exemplified in Table 3 present the consolidated term path, the matching term, and the occurrence in original dataset.

The conceptual schema, shown in Figure 8, is genereated based on the defined rules starting from a root element identified by the ROOT label, in this case, the term *pubmedarticle*.

In Figure 8, the entities at the right side of the root term represent nested arrays with relationships of type $1 : M$, identified by `EMBED`, and $1 : 1$ indicated by `REFI`. Fields that are not found in the structure of the chosen base document are considered as a separate block, '*entity1*', at the left of the root element.

5.2 Evaluation

In this experiment, 15 fields could be considered as relevant, that is, that represent the same information. A total of 14 was retrieved according to Table 2. The unidentified terms would be $Tags \Rightarrow tags$.

The revocation and precision rates are shown in Table 4 together with the number of terms retrieved, terms relevant retrieved and terms relevant represented by *Ret, Rel Ret and Rel* respectively.

Table 4: Recall and precision values.

Ret	Rel Ret	Rel	Recall	Precision
14	12	15	80%	85,71%

Among the terms retrieved, two were identified incorrectly: $Publication \Rightarrow Issue$ and $Hour \Rightarrow Minute$. This happened because the semantic metric found high values for these cases. Some difficulties encountered in correctly identifying the equivalent terms are due to particularities of the chosen techniques.

Some words are not valid in the language and therefore can not be analyzed semantically, just as they can not be radical in Porter's algorithm. However, they will always be analyzed for the variation of characters having an appropriate threshold for each case. Thus, from the indexes found, the process precision is considered good.

6 CONCLUSION

The main contribution concerns the process for extracting conceptual schemas that has as input a collection of documents in JSON format. These files may be stored in a NoSQL database or in the Web in general.

In order to exploit the flexibility of schemas, the process aims to identify equivalences in the fields that are written differently or at integration scenarios, either for lack of standardization or for misunderstanding, but that represent the same information. In this way, it uses similarity techniques that cover similar spelling, synonyms and radical equivalence of the word. The process is applied between documents and within the document, generating relationships of type $1:M$ or $0:M$, once in an NoSQL model these are indicated about a entity nested in a root element.

The tests indicate that the process has more scope as a greater number of variations, maintaining good rates of revocation and precision. Inconsistencies occurred in cases of words that even have the same spelling, have different meanings, or questions of the Wordnet library.

The extraction of a unified schema can also be useful in future work to allow the submission of queries about it, since a mapping indicates to which other terms that consolidated term refers and points the respective origin documents of the corresponding terms. A future solution could be investigating the use of algorithms to deal with homonyms.

With the growth in the volume of data and the popularization of the data of the mono structured as JSON, it is need to be concerned about schemes so that it can develop applications that access them in a coherent way. This proposal differs by exploring the flexibility of schemas, identifying equivalent fields in terms of synonymous, word radical and character generating a unified schema.

REFERENCES

Benson, S. R. (2014). Polymorphic data modeling. Master's thesis, Georgia Southern University.

Blaselbauer, V. M. and Josko, J. M. B. (2020). Jsonglue: A hybrid matcher for json schema matching. *Proceedings of the Brazilian Symposium on Databases*.

Goessner, S. (2007). Jsonpath - xpath for json. http://goessner.net/articles/JsonPath/. Acessed in 2016, November.

Jovanovic, V. and Benson, S. (2013). Aggregate data modeling style. *SAIS 2013*, pages 70–75.

Kettouch, M. S., Luca, C., Hobbs, M., and Dascalu, S. (2017). Using semantic similarity for schema matching of semi-structured and linked data. In *2017 Internet technologies and applications (ITA)*, pages 128–133. IEEE.

Klettke, M., Störl, U., Scherzinger, S., and Regensburg, O. (2015). Schema extraction and structural outlier detection for json-based nosql data stores. In *BTW*, volume 2105, pages 425–444.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Lin, D. (1998). An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304. Citeseer.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Porter, M. (2006). An algorithm for suffix stripping. *Program: electronic library and information systems*, 40(3):211–218. https://doi.org/10.1108/00330330610681286.

Ruiz, D. S., Morales, S. F., and Molina, J. G. (2015). Inferring versioned schemas from nosql databases and its applications. In *International Conference on Conceptual Modeling*, pages 467–480. Springer.

Varga, V., Jánosi-Rancz, K. T., and Kálmán, B. (2016). Conceptual design of document nosql database with formal concept analysis. *Acta Polytechnica Hungarica*, 13(2).