Comparative Analysis of Methods for Open Data Catalog Similarity

Jorge Martinez-Gil

Software Competence Center Hagenberg GmbH
Softwarepark 32a, 4232 Hagenberg, Austria
jorge.martinez-gil@scch.at

Abstract

Open data initiatives continue to grow, and with them, the number of independent catalogs published across various platforms. Comparing these catalogs helps with tasks such as improving search quality, linking related datasets, and maintaining consistent metadata. This paper surveys methods for measuring similarity between open data catalogs. Approaches include matching structured elements like triples, applying hashing techniques to domain-specific terms, and using semantic representations to align catalog contents. Each method comes with different assumptions about how metadata is structured and how records can be matched. We also examine practical limitations, such as poor metadata quality, differences in schema, and limited availability of shared benchmarks. The final section outlines directions for future work, including cross-lingual catalog matching, automated metadata enrichment, and techniques that better scale to large catalog collections.

Keywords: Open Data, Data Catalogs, Semantic Similarity Measurement

1. Introduction

In times when the use of data has exploded exponentially, open data catalogs (ODCs) have emerged as an indispensable source of information that facilitates transparency, innovation, and multidisciplinary collaboration [1]. These catalogs, which often must be manually curated through government agencies, research organizations, or even non-profit organizations, offer invaluable knowledge spanning a wide range of disciplines, industries, and geographies [8]. However, as the volume of ODCs skyrockets, methods and tools are needed to facilitate their proper processing. For example, to assess their quality [2], to quantify their similarity [16], to use them in an effective way [22], and so on.

The concept of catalog similarity goes far beyond the mere comparison of metadata. The reason is that behind this concept lies a significant challenge in interconnecting different data repositories.

Therefore, the possibility of automatically determining the similarity of ODC can positively impact a large group of professionals involved in data economics, for example, researchers, policymakers, scientists, etc. In this way, providing methods and tools that facilitate the effective and efficient comparison of ODCs can facilitate decision-making processes and multidisciplinary collaborations, and it holds great potential for the collective use and analysis of open data.

This research is designed to examine how similarity between ODCs can be measured using a variety of methods. We aim to present the main approaches available, covering a range of computational strategies that have been proposed or applied to this problem. The objective is to organize these options in a way that helps users understand their differences and practical use. We also intend to give users a way to decide which method fits best depending on the structure, size, and purpose of the catalogs being compared. The broader aim is to make open data easier to connect and reuse in practical settings across science, policy, and digital services.

The remainder of this paper is structured as follows: Section 2 introduces related work concerning similarity between ODCs and challenges that remain pending. Section 3 provides an illustrative overview of existing methods to address the challenge of ODC similarity using different computational approaches. Section 4 presents the lessons that can be drawn from this research and possible future research directions.

2. Related Works

ODCs are typically used to describe and organize data assets, making them more discoverable and accessible for the stakeholders of an organization. The standard DCAT (Data Catalog Vocabulary) [9] is a widely used specification for describing data catalogs, often used with government and open data portals [13]. When comparing two DCAT data catalogs, it is possible to find similarities in various aspects inherent to them.

The aspects that should be considered in a specific situation are an open discussion. However, some viewpoints suggest assessing metadata elements for similarities, including title, description, keywords, publisher, contact information, access URLs, data formats, and licenses, should work fine in general cases. The organizational structure of the catalogs, whether hierarchical or based on tags and keywords, is also a widely suggested option for cases of this kind.

Other aspects that could be considered are compliance with standards like DCAT and additional metadata elements or support for standards like CKAN[1] or Dublin Core[2]. Also, the data

[1] https://ckan.org/
[2] https://www.dublincore.org/

format and licensing model compatibility suggest similarities in ODCs. The search and discovery capabilities, including filters and faceted search, and other advanced options could also be evaluated, along with methods for accessing data such as direct downloads or APIs. Additionally, the sources and diversity of data, data coverage, versioning support, metadata quality, update frequency, and the use of linked data principles for semantic relationships can also be sources of great value to be investigated.

However, several challenges remain open in the field, and there is still a need for solutions to handle them, including standardized metadata schemas, addressing data quality issues [15], and managing evolving catalogs. Additionally, as open data initiatives evolve, future research should focus on dynamic and real-time similarity assessment, multi-modal data catalogs, and cross-domain catalog comparisons. In this work, we try to shed light on the topic of similarity [19], as we try to identify ways in which similarity could be determined automatically [10, 17]. These directions may help support more scalable and interoperable infrastructures for data reuse across different domains.

3. Similarity Methods for Open Data Catalogs

Similarity methods for ODCs play a crucial role in enhancing the discoverability [14] of vast and diverse datasets available in the public domain. These methods could, for example, employ techniques such as natural language processing [4, 20], metadata analysis through machine learning [6, 7], and other kind of semantic-based techniques, or even sophisticated combinations of all of these methods [5], to establish connections and relationships between datasets based on their content, structure, and context. Although this topic is still very incipient, some authors have some works that focus additionally on the interpretability [18] to help stakeholders understand how the similarity value is calculated.

In this work, we focus on calculating similarity scores [11] since this should enable users to find datasets relevant to their specific needs, even when they may not know the exact dataset names or categories. The rationale behind this approach is to facilitate more efficient data discovery and encourages data reuse and integration across domains. This can lead to increased collaboration, innovation, and democratizing data-driven insights [12]. As the volume of open data grows, similarity methods remain essential to work with ODCs and address complex challenges and data-driven decision-making processes.

In the context of this work, we are going to focus on an illustrative technique for each of the existing approaches:

1. Considering the two ODCs as two repositories of triples.
2. Considering the two ODCs as two repositories of tokens.
3. Considering the two ODCs as two character sequences.
4. Considering the two ODCs as two documents written in a common purpose-specific language.

We will now go deeper into the technical details of these approaches, exploring their unique attributes, advantages, and potential challenges to gain a comprehensive understanding of their applicability and potential.

3.1. Considering the two ODCs as two repositories of triples

Calculating the similarity between two ODCs, considering them as repositories of triples, involves comparing the triples associated with the datasets in each ODC to determine their similarity. This process can be helpful in various data-related tasks, such as data integration or alignment, where the people work with a granularity at the level of triples.

The idea is simple and based on calculating the similarity between two sets of triples by iterating through each triple in both ODCs and counting the number of triples common to both ODCs. Algorithm 1 shows us how an algorithm could implement this. The significant advantage of this method is its simplicity and ease of understanding.

Algorithm 1 Calculate Similarity Between Two Repositories of Triples

Initialize an empty RDF graph g
Parse ODCs into g
Initialize two empty sets $triples$ and $triples2$
for each triple (s, p, o) in g where $s = dataset\_001$ **do**
    Add o to $triples$
end for
for each triple (s, p, o) in g where $s = dataset\_002$ **do**
    Add o to $triples2$
end for
Initialize $similarity$ to 0
for each triple $p1$ in $triples$ **do**
    for each triple $p2$ in $triples2$ **do**
        if $p1 = p2$ **then**
            Increment $similarity$ by 1
        end if
    end for
end for
Calculate $similarity$ as $similarity / \max(length\ of\ triples, length\ of\ triples2)$

3.2. Considering the two ODCs as two repositories of tokens

Calculating the similarity of two ODCs as two repositories of tokens, whether ordered or unordered, can offer several advantages. For example, tokenizing and comparing data catalogs based on their tokens allows us to identify similarities without loading and processing the entire datasets quickly. Moreover, the calculations to be performed are highly scalable. The reason is that, as the size of the ODC grows, the computational cost of comparing tokens remains relatively low.

Algorithm 2 encapsulates the *CheckSimilarity* function to calculate the similarity between two ODCs (converted into sets of RDF data) using TF-IDF vectorization and cosine similarity. The idea is first to convert the ODC into TF-IDF feature vectors and then compute the cosine similarity between these vectors, representing the similarity between the two ODCs. The result is scaled to a percentage and returned as the final similarity score. Comparing the similarity is commonly used in many natural language processing tasks.

Algorithm 2 Calculate Similarity Between Two Repositories of Tokens

function SIMILARITY(odc, odc2)
  $vectorizer \leftarrow TfidfVectorizer()$
  $tfidf\_matrix \leftarrow vectorizer.fit\_transform([odc, odc2])$
  $cosine\_sim \leftarrow cosine\_similarity(tfidf\_matrix[0], tfidf\_matrix[1])[0][0]$
  return $cosine\_sim$
end function

3.3. Considering the two ODCs as two character sequences

Calculating the similarity of two ODCs using the Longest Common Subsequence (LCS) [3] method involves finding the longest sequence of items common to both catalogs. LCS provides a structured approach to aligning sequences of characters. This alignment helps understand how data items in one catalog correspond to those in the other, which can be crucial for data matching and integration. The LCS method is often used in various fields to measure the similarity between sequences, such as information retrieval, text analysis, etc.

Algorithm 3 is intended to construct a matrix dynamically and iteratively comparing their elements. It utilizes dynamic programming to find the maximum common subsequence length and returns this value.

Algorithm 4 calculates the similarity between two ODCs. The idea is first to determine their LCS's length and then normalize it by dividing it by the maximum length of the input sequences. The result is a similarity score between 0 and 1, where 1 indicates complete similarity, and 0 indicates no commonality.

Algorithm 3 Calculation of the Longest Common Subsequence

function LCS($odc, odc2$)
  $m \leftarrow length(odc)$
  $n \leftarrow length(odc2)$
  $lcs\_matrix \leftarrow$ initialize a 2D array of size $(m+1) \times (n+1)$ filled with zeros
  for $i \leftarrow 1$ to m **do**
    for $j \leftarrow 1$ to n **do**
      if $odc[i-1] = odc2[j-1]$ **then**
        $lcs\_matrix[i][j] \leftarrow lcs\_matrix[i-1][j-1] + 1$
      else
        $lcs\_matrix[i][j] \leftarrow \max(lcs\_matrix[i-1][j], lcs\_matrix[i][j-1])$
      end if
    end for
  end for
  return $lcs\_matrix[m][n]$
end function

Algorithm 4 ODC Similarity Calculation using LCS

function SIMILARITY($odc, odc2$)
  $lcs\_length \leftarrow LCS(odc, odc2)$
  return $\frac{lcs\_length}{\max(length(odc), length(odc2))}$
end function

The great advantage of the LCS method is that it primarily measures structural similarity based on the order of items in the catalogs. If the order of items is crucial for the analysis, the LCS method can be suitable. However, in cases where the item order is less important, it is better to consider other token-based measures, which focus on item presence or vector representations of the ODCs.

3.4. Considering the two ODCs as two documents written in a common purpose-specific language

These methods are known for their computational efficiency, especially when working with huge volumes of text data. It can quickly process large volumes of data, making it a potentially fast approach for performing calculations. To illustrate this approach, we have chosen the Winnow hashing algorithm [21], which works very well in identifying similarities between specific-purpose languages.

The Algorithm 5 adapts the famous Winnow algorithm that takes a text and an integer k, splitting the text into k-grams, hashing them, and finding the minimum hash value within sliding windows. This process creates a very valuable fingerprint of the text for later processing and comparison.

Algorithm 5 Winnow Algorithm

function WINNOW($text, k$)
  $k{-}grams \leftarrow emptylist$
  for $i \leftarrow 0$ to $len(text) - k$ **do**
    $append\,(k{-}grams, text[i : i + k])$
  end for
  $hashes \leftarrow emptylist$
  for $k - gramink{-}grams$ **do**
    $append\,(hashes, hash(k - gram))$
  end for
  $w \leftarrow 10$
  $min\_hashes \leftarrow emptylist$
  for $i \leftarrow 0$ to $len(hashes) - w$ **do**
    $min\_hash \leftarrow min(hashes[i : i + w])$
    $append\,(min\_hashes, min\_hash)$
  end for
  return min_hashes
end function

As a second step, Algorithm 6 compares two fingerprints previously calculated using Algorithm 5 by calculating the Jaccard similarity coefficient between their fingerprints. As with previous cases, a higher Jaccard similarity indicates greater similarity between the texts, making these algorithms useful for measuring ODC similarity efficiently.

Algorithm 6 ODC Similarity Calculation using Winnow Hashing

function SIMILARITY($odc, odc2$)
  $k \leftarrow 5$
  $fingerprint1 \leftarrow set(Winnow(odc, k))$
  $fingerprint2 \leftarrow set(Winnow(odc2, k))$
  return $\frac{len}{(fingerprint1 \cap fingerprint2)} len(fingerprint1 \cup fingerprint2)$
end function

4. Conclusion

This research surveys several approaches to measuring similarity between Open Data Catalogs (ODCs), showing the range of techniques applied to this task. Methods include simple triple-based comparisons as well as approaches that use semantic signals adapted to specific data vocabularies. These options differ in what they capture, how scalable they are, and how well they generalize across types of data. Choosing a method requires attention to the structure and goals of the catalogs being compared.

Different types of catalogs require different strategies. Domain-specific metadata, the level of structure in descriptions, and the presence of standardized vocabularies all affect which methods are suitable. The number of entries, the consistency of metadata, and the language used across entries can also affect similarity results. Technical constraints such as indexing efficiency or available memory may limit which techniques are feasible in larger deployments.

Future work should continue exploring this direction, focusing on scalable, adaptive, and context-aware methods for catalog similarity. There is also room for hybrid approaches that combine structural, lexical, and semantic signals, as well as methods that incorporate user interaction or relevance feedback to refine similarity estimates. As open data initiatives expand, and cross-catalog data integration becomes more common, reliable similarity measures will be essential for enhancing data interoperability, supporting intelligent catalog navigation, and enabling more effective dataset recommendation and reuse across diverse domains and organizational contexts.

References

[1] Albertoni, R., Browning, D., Cox, S., González-Beltrán, A. N., Perego, A., & Winstanley, P. (2023). The W3C data catalog vocabulary, version 2: Rationale, design principles, and uptake. *CoRR*, *abs/2303.08883*. doi:`10.48550/arXiv.2303.08883`. `arXiv:2303.08883`.

[2] Albertoni, R., & Isaac, A. (2021). Introducing the data quality vocabulary (DQV). *Semantic Web*, *12*, 81–97. doi:`10.3233/SW-200382`.

[3] Bergroth, L., Hakonen, H., & Raita, T. (2000). A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000* (pp. 39–48). IEEE.

[4] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. doi:`10.18653/v1/n19-1423`.

[5] Dzeroski, S., & Zenko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, *54*, 255–273. doi:`10.1023/B:MACH.0000015881.36452.6e`.

[6] English, T. M., & Gotesman, M. (1995). Stacked generalization and fitness ranking in evolutionary algorithms. In J. R. McDonnell, R. G. Reynolds, & D. B. Fogel (Eds.), *Proceedings of the Fourth Annual Conference on Evolutionary Programming, EP 1995, San Diego, CA, USA, March 1-3, 1995* (pp. 205–218). A Bradford Book, MIT Press. Cambridge, Massachusetts.

[7] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*, 1–309.

[8] Lakomaa, E., & Kallberg, J. (2013). Open data as a foundation for innovation: The enabling effect of free public sector information for entrepreneurs. *IEEE Access*, *1*, 558–563. doi:`10.1109/ACCESS.2013.2279164`.

[9] Maali, F., Erickson, J., & Archer, P. (2014). Data catalog vocabulary (dcat). w3c recommendation. *World Wide Web Consortium*, (pp. 29–126).

[10] Martinez-Gil, J. (2014). An overview of textual semantic similarity measures based on web intelligence. *Artif. Intell. Rev.*, *42*, 935–943. URL: `https://doi.org/10.1007/s10462-012-9349-8`. doi:`10.1007/S10462-012-9349-8`.

[11] Martinez-Gil, J. (2022). A comprehensive review of stacking methods for semantic similarity measurement. *Machine Learning with Applications*, *10*, 100423.

[12] Martinez-Gil, J. (2023). A comparative study of ensemble techniques based on genetic programming: A case study in semantic similarity assessment. *Int. J. Softw. Eng. Knowl. Eng.*, *33*, 289–312. doi:`10.1142/S0218194022500772`.

[13] Martinez-Gil, J. (2023). A survey on legal question-answering systems. *Comput. Sci. Rev.*, *48*, 100552. URL: `https://doi.org/10.1016/j.cosrev.2023.100552`. doi:`10.1016/J.COSREV.2023.100552`.

[14] Martinez-Gil, J., Alba, E., & Aldana-Montes, J. F. (2008). Optimizing ontology alignments by using genetic algorithms, . *419*. URL: `https://ceur-ws.org/Vol-419/paper2.pdf`.

[15] Martinez-Gil, J., & Aldana-Montes, J. F. (2011). Evaluation of two heuristic approaches to solve the ontology meta-matching problem. *Knowl. Inf. Syst.*, *26*, 225–247. URL: `https://doi.org/10.1007/s10115-009-0277-0`. doi:`10.1007/S10115-009-0277-0`.

[16] Martinez-Gil, J., & Aldana-Montes, J. F. (2012). An overview of current ontology meta-matching solutions. *Knowl. Eng. Rev.*, *27*, 393–412. URL: `https://doi.org/10.1017/S0269888912000288`. doi:`10.1017/S0269888912000288`.

[17] Martinez-Gil, J., & Aldana-Montes, J. F. (2013). Semantic similarity measurement using historical google search patterns. *Inf. Syst. Frontiers*, *15*, 399–410. URL: `https://doi.org/10.1007/s10796-012-9404-7`. doi:`10.1007/S10796-012-9404-7`.

[18] Martinez-Gil, J., & Chaves-Gonzalez, J. M. (2021). Semantic similarity controllers: On the trade-off between accuracy and interpretability. *Knowl. Based Syst.*, *234*, 107609. doi:`10.1016/j.knosys.2021.107609`.

[19] Martinez-Gil, J., Paoletti, A. L., Rácz, G., Sali, A., & Schewe, K. (2018). Accurate and efficient profile matching in knowledge bases. *Data Knowl. Eng.*, *117*, 195–215. URL: `https://doi.org/10.1016/j.datak.2018.07.010`. doi:`10.1016/J.DATAK.2018.07.010`.

[20] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* (pp. 3111–3119).

[21] Schleimer, S., Wilkerson, D. S., & Aiken, A. (2003). Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (pp. 76–85).

[22] Subramaniam, P., Ma, Y., Li, C., Mohanty, I., & Fernandez, R. C. (2021). Comprehensive and comprehensible data catalogs: The what, who, where, when, why, and how of metadata management. *CoRR*, *abs/2103.07532*. `arXiv:2103.07532`.