Received: 18 December 2020 | Revised: 27 April 2021 | Accepted: 28 April 2021
DOI: 10.1002/ail2.23

LETTER

WILEY

Heritage connector: A machine learning framework for building linked open data from museum collections

Kalyan Dutia | **John Stack**

Science Museum Group, London, UK

Correspondence
Kalyan Dutia and John Stack, Science Museum Group, London, UK.
Email: kalyan.dutia@pm.me; john.stack@sciencemuseum.ac.uk

Funding information
Arts and Humanities Research Council

Abstract
As with almost all data, museum collection catalogues are largely unstructured, variable in consistency and overwhelmingly composed of thin records. The form of these catalogues means that the potential for new forms of research, access and scholarly enquiry that range across multiple collections and related datasets remains dormant. In the project *Heritage Connector: Transforming text into data to extract meaning and make connections*, we are applying a battery of digital techniques to connect similar, identical and related objects within and across collections and other publications. In this article, we describe a framework to create a Linked Open Data knowledge graph from digital museum catalogues, perform record linkage to Wikidata, and add new entities to this graph from textual catalogue record descriptions (information retrieval). We focus on the use of machine learning to create these links at scale with a small amount of labelled data, and models which are small enough to run inference on datasets the size of museum collections on a mid-range laptop or a small cloud virtual machine. Our method for record linkage against Wikidata achieves 85%+ precision with the Science Museum Group (SMG) collection, and our method for information retrieval is shown to improve NER performance compared with pretrained models on the SMG collection with no labelled training data. We publish open-source software providing tools to perform these tasks.

KEYWORDS
information retrieval, knowledge graphs, museums, record linkage

1 | INTRODUCTION

Museum collections contain a vast array of objects, documents, artworks, samples of the natural world, digital files and more. It is common for large museums to have substantial collections numbering in the hundreds of thousands or millions of objects of which only a small portion is on public display. The remaining collections are stored and conserved for future display, research and loan. At the Science Museum Group (SMG), less than 10% of the estimated 425 000 collection objects is on public display in the museums.

Since the late 1990s the affordances of the web and other technologies, such as digital imaging, have led to increasing volumes of collection content being published online, making the stored collections, previously only accessible to a small number of visiting researchers, available to a global audience and providing new ways for museums to deliver their missions.

Even though enormous volumes of collection objects have been digitised, there remain barriers to their discovery and exploration, meaning that the full potential of the digitised collections for research, access and scholarly enquiry remains dormant. It is this challenge that the *Heritage Connector: Transforming text into data to extract meaning and make connections* project seeks to address.

Discovery and exploration of digitised collections is driven by collection catalogue records: a dataset that began with handwritten ledgers then moved to index cards, and is now created and managed in specialised software tools (collection management systems). These records are usually stored by the collection management system in a relational database whose structure and contents is used to deliver online search discovery and browsing features. For the most part, these records pre-date digitisation and their origins lie in the needs of museum collection management: accession records, locations, loans, conservation reports, hazard analyses and so on. A subset of these fields are used to populate the object's online record. The records may also have been augmented by new cataloguing data which has been specifically implemented to improve the online experience: indexing, descriptive and contextualising texts, manually created links to other resources, etc. In spite of this, discovery and exploration of those objects remains primarily driven by the museum's catalogue. As with almost all data, and although managed in a specialised system, museum collection catalogues are largely unstructured, variable in consistency and overwhelmingly composed of thin records.

In the Heritage Connector project, we are applying a battery of digital techniques to connect similar, identical and related objects within and across two collections (SMG and Victoria and Albert Museum [V&A]) and other datasets and publications. The project explores a range of data analysis approaches that analyse collections catalogues and external knowledge graphs[1] – primarily Wikidata – and build links at scale between these. The affordances of the resulting new dataset includes[2,3]:

- Improving keyword search results.
- Additional dimensions by which to filter and sort the collection.
- Macro-views of the entire collection so that users can grasp the contents overall of the collection.
- Serendipitous discovery through linking of previously isolated records.
- New entry points into the collection via newly generated groupings of records.
- Richer onward journeys from records to records of related objects, people, organisations and so on.
- New forms of interface which provide an alternative to keyword search discovery.

Rather than attempt to create a comprehensive, cross-collection federated search or aggregator (such as Europeana, Cornucopia, ArchivesHub or ArtUK) or manually construct a Linked Open Data (LOD)[4,5] dataset (such as American Art Collaborative Linked Open Data initiative or Linked Art), Heritage Connector attempts to demonstrate that use of AI-generated knowledge graphs using LOD can provide useful functionality that can sit alongside or augment existing collection websites, aggregators or other LOD implementations. Furthermore, Heritage Connector explores what level of accuracy in the links and functionality derived from the knowledge graph is – although not perfect – good enough to provide benefit to users.

1.1 | Heritage Connector software

The software described in this article[6,7] consists of a Python library that enables museum developers to load existing collection catalogues into a LOD knowledge graph and create links between entities in this knowledge graph and to Wikidata using machine learning techniques. The rest of this article is set out as follows. Section 2 describes the data sources we have used to develop Heritage Connector. Section 3 describes the components of the Heritage Connector software. Section 4 describes our approach to record linkage: creating links between pairs of museum records and Wikidata records which represent the same real-world entity. Section 5 describes our approach to named entity recognition as part of the process of creating new knowledge graph links from text fields. Finally, in Section 6 we state our conclusion and our plan for future work.

2 | DESCRIPTION OF DATA SOURCES

2.1 | Wikidata

Wikidata[8] is the free, open, linked, multilingual and structured database which underpins Wikipedia but which is also a tool in its own right. Today, Wikidata contains over 91 million records[*] structured as linked data. Wikidata was selected for this project because it includes references to dozens of external data points in cultural heritage collections and other linked datasets, and because it covers a vast range of subject domains that extend far beyond those traditionally covered by museum collection catalogues, enabling diverse data to be trialled in the project. Once linked to Wikidata, museum collections can leverage this additional data to build new forms of exploration, discovery and research. The full affordances of connecting the SMG and V&A Collection catalogues to Wikidata remain to be explored by the project, but emerging opportunities that are shaping our approach include:

- Extending the Heritage Connector knowledge graph to include "facts" not in the collection catalogue and presenting them to users.
- Using links between collection records and Wikidata records to provide users onward links to other museum collections and data sources.
- Ingesting data from Wikidata or from other places via Wikidata into the Heritage Connector knowledge graph to improve discovery of collection records.
- Using data ingested from Wikidata to infer and present new entry points into the collections such as themes, events and topics.
- Using Wikidata as a route to Wikipedia entries for articles associated with collections.

Wikidata, as a knowledge graph, is made up of entities (records) and properties (predicates) describing the relationships between these entities. Entities are uniquely identified by their Q identifier (QID), and properties by their P identifier (PID). As well as a list of "claims" – properties and their values – each entity also has a label (title), multiple aliases (alternative titles) and a description (short piece of text describing the entity). There exist hierarchies of both entities and properties in Wikidata. In order to navigate this hierarchy, the properties *instance of* (*P31*) and *subclass of* (*P279*) are used. The property *equivalent property* (*P1628*) enables mapping between Wikidata properties and RDF[9] predicates as further described in Subsection 4.1.3.

To avoid confusion between records (a collection of database fields about the same real-world thing in a database, which here is a knowledge graph) and entities, which are both knowledge graph components and spans of text extracted using Named Entity Recognition (NER), we hereon refer to knowledge graph entities as 'records' and identified text spans which are added to the knowledge graph as 'entities'. We make an exception when specifically referring to the *entities* and *predicates* in a knowledge graph.

2.2 | Science Museum Group collection

The Science Museum Group (SMG) is a family of five museums: Science Museum, London; National Science and Media Museum, Bradford; Science and Industry Museum, Manchester; National Railway Museum, York; and Locomotion, Shildon. It is one of the UK's national museums and received 5 million visitors and 11 million online visitors in 2019-2020.[10] The Group's collection traces its origins back to the 1852 Great Exhibition and covers numerous aspects of science, technology, design, transport, medicine, media and art.

Historically, the collections were managed by the different museums in the group and presented online in a number of different interfaces. Since 2016, the collection has been available online in a unified web interface[11] and significant volumes of objects continue to be photographed and published online. There are, however, some limitations to the current online presentation of the collection and these limitations are common across the cultural heritage sector:

- The collection is large and extremely diverse in content.
- The disparate content types benefit from specific approaches to cataloguing.
- Cataloguing standards have grown organically, often over decades, and legacy cataloguing data is, therefore, inconsistent.

- Catalogue records are generally thin, and many objects are similarly catalogued.
- Application of controlled vocabularies[12] is limited and inconsistent.
- There are very few or no links to external sources including other collections or related material.

SMG's digital collection records are stored in a relational database managed by a collection management system. These records describe objects (including archival material) present in the physical collection, and related people and organisations (such as makers or users of objects). There are two tables which store object records and people/organisation records, one which stores archival documents and two join tables which describe person-object and organisation-object relationships (commonly person A *used/made* object B). For the purpose of this article we use CSV exports from this database, split into three subgroups with their numbers of records in brackets: people (10 352), objects (282 259) and organisations (7743). We use the join tables to add relationships between records to the knowledge graph.

Each SMG record has a label (title), description and about 4-6 fields that describe properties of the record such as a person's occupation or the date of formation of a company. The label field is free text and the description field is longer-form free text. On average there are 0 to 1 fields connecting each record directly to another, but the description fields may contain additional information about links between records which cannot be used effectively to explore the collection as they only exist in paragraphs of text, rather than explicit links. When we describe records as 'thin' in this article we refer to the small number of fields within and the limited number of links between them. Readers who are keen to explore the existing format of the collection are encouraged to visit its web interface.[11]

3 | TECHNICAL ARCHITECTURE AND APPROACH

The open-source software published alongside this project is a set of Python libraries with which a software developer at a museum can create and enrich LOD from digital museum collections, which were originally stored in a set of relational tables (likely within a collection management system). First, a developer ingests a collection catalogue which comprises multiple relational tables into a knowledge graph: the 'Heritage Connector' (HC). With this knowledge graph they can then use the Record Linkage component to create links between HC records and Wikidata records which refer to the same real-world entity, and the Information Retrieval tools to add records to the knowledge graph from text passages such as collection record descriptions, blog posts and articles. Figure 1 shows the components of the Heritage Connector software.

In a survey taken at the project's June 2020 webinar[13] 59% of respondents from cultural heritage institutions said that a major hurdle to them adopting Wikidata QIDs in their collection was time, resources, or the large amount of work required. Therefore, for every component in Heritage Connector, we design approaches that account for the constraints on time and budget that are common in the heritage sector. Each machine learning model is designed to work with just a small amount of labelled data, using rules and dictionaries instead to integrate expert knowledge into the

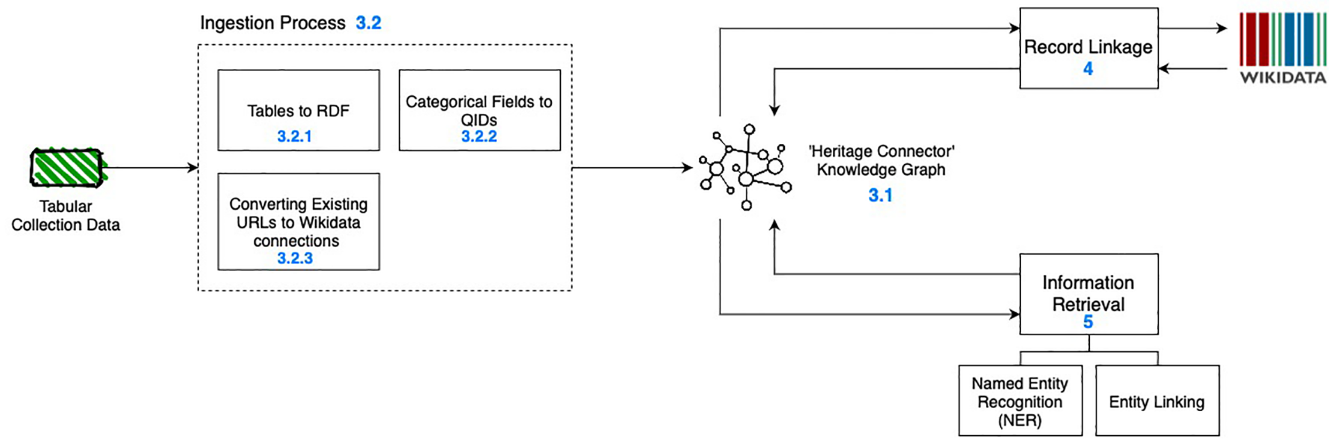

Ingestion Process 3.2
Tables to RDF 3.2.1
Categorical Fields to QIDs 3.2.2
Converting Existing URLs to Wikidata connections 3.2.3
Tabular Collection Data
'Heritage Connector' Knowledge Graph 3.1
Record Linkage 4
WIKIDATA
Information Retrieval 5
Named Entity Recognition (NER)
Entity Linking

FIGURE 1 Diagram showing the components of the Heritage Connector software. The numbers in blue correspond to sections in this article

learning process where possible. All systems are designed to work on a mid-range laptop or a small cloud virtual machine. We test them on a Macbook Pro with a 2.3 GHz Quad-Core Intel i5 and 8 GB RAM.

In the remainder of this section we describe each of the components in turn, except the Record Linkage and Information Retrieval components which are described in Sections 4 and 5, respectively.

3.1 | Knowledge graph

The knowledge graph is primarily stored in JSON-LD format in an Elasticsearch index, with a clone maintained in an Apache Jena Fuseki[14] triplestore. JSON-LD in Elasticsearch has been chosen as the primary storage method as it is generally more familiar to developers than triplestores, and as we plan to use the dense vector field type[†] for lookup of graph and text embeddings later on in the project.

All data in the knowledge graph is stored according to Linked Data standards,[4] meaning that each entity and predicate is stored as an HTTP URI. URIs are commonly represented in shortened prefix: predicate form, for example, http://xmlns.com/foaf/0.1/maker as foaf:maker, where the prefix represents an RDF[9] *namespace*. Each namespace has an RDF vocabulary: a list of allowed predicates for that namespace. We do not generally enforce a particular set of RDF namespaces or vocabularies for our knowledge graph, and instead advise developers to choose predicates based on whether they can be linked to a Wikidata property (see Subsection 4.1.3). However, we specifically use some RDF predicates in our software as defined in Table 1.

Only first-degree links are stored in the knowledge graph, that is, '*Sony* manufactured the *Walkman*' and not '*Panasonic* were competitors to Sony, who manufactured the *Walkman*'. The links in the knowledge graph are of one of the following types (see Table 1): existing links between records and existing fields for each record from the collection; owl:sameAs links between knowledge graph records and Wikidata records which describe the same real-world entity; and hc:entTYPE links between knowledge graph records and entities (where TYPE is replaced by the predicted entity type) which have been identified in the description field of each record using the Information Retrieval process (Section 5). Computational tools such as knowledge graph embeddings[15] and other graph machine learning methods can then be applied to this data to retrieve records for similar and related objects. We refrain from defining the terms 'similar' and 'related' here as these qualities are largely use-case driven: one definition of similarity may work well for visualisation, but may not work well for a 'more like this' search.

TABLE 1 RDF predicates that serve a particular purpose in Heritage Connector

RDF predicate	Usage	Example values (SMG)
rdfs:label	Label of the record	*'Babbage, Charles'*
rdf:type	Type of the record, corresponding to Wikidata P31. Values must be converted to a Wikidata QID to use the record linkage component.	*Human (Q5)*
skos:hasTopConcept	A pointer to the source table of the record in the original collection. For SMG we use 'PERSON', 'OBJECT' and 'ORGANISATION'.	*'PERSON'*
sdo:isPartOf	Stores the collection membership of a particular object.	*'SCM – Art'*
owl:sameAs	Stores links between HC and Wikidata records produced by the record linkage component.	https://www.wikidata.org/wiki/Q46633
hc:entityLOC, hc:entityORG, hc:entityPERSON, hc:entityFAC, hc:entityNORP, hc:entityLANGUAGE, hc:entityDATE, hc:entityOBJECT	Stores links between a record and entities found in its description field using the information retrieval component. Values can either be URIs or strings, depending whether the found entities have been linked to a knowledge graph record.	*'Royal Victoria Hotel'* (for entityFAC)

Note: All other predicates are the choice of the developer or user. All prefixes follow linked data principles except for hc, a custom prefix which at the time of writing has been assigned a dummy (instead of public) URI.

3.2 | Ingestion process

3.2.1 | Tables to RDF

This process allows relational tables loaded into Pandas DataFrames to be imported into the knowledge graph via a mapping of DataFrame column names to predicates in an RDF vocabulary. Joins between tables are ingested using a method which adds a triple (*subject-predicate-object*) for each row of a join table with the corresponding predicate. For example, for a join table in the original relational database which contains objects and the companies which manufactured them, the ingestion component can be used to add <object_URI> <http://xmlns.com/foaf/0.1/maker><company_uri> for each object-company relationship, where the predicate http://xmlns.com/foaf/0.1/maker has been specified by the developer.

3.2.2 | Converting categorical fields to Wikidata QIDs

Heritage Connector includes a component to convert values in any categorical text field in the HC knowledge graph to Wikidata QIDs, optionally using a provided mapping of that field to the Wikidata ontology to constrain which QIDs can be returned. For example, the values 'engineer', 'pilot' and 'medical instrument maker' in an *occupation* field which is defined as all entities in the subclass tree of *occupation* (*Q12737077*) become Wikidata QIDs Q81096, Q2095549 and Q66060301. Conversion to QIDs is obligatory for the rdf:type of each record in order for the record linkage component to be used.

For SMG data we use the GeoNames API[16] to standardise place names and Wikidata property *GeoNames ID* (*P1566*) to convert them to Wikidata QIDs. A Jupyter notebook to do this is available in the main project repository.[6]

3.2.3 | Converting existing URLs to Wikidata links

It is common for museum cataloguers (including curators and archivists) to add URLs to the description fields of collection records that point to external pages which describe the same thing as the record, or something related. These external pages are usually records in online knowledge bases such as Wikipedia, Grace's Guide[17] or Oxford Dictionary of National Biography,[18] whose IDs can be converted to Wikidata QIDs using Wikidata's external identifiers. For example, a Grace's Guide ID can be converted to a Wikidata QID by using the Wikidata external identifier *Grace's Guide ID* (*P3074*) as in the SPARQL query in Listing 1.

Listing 1: SPARQL query to resolve the Grace's Guide ID *Isambard_Kingdom_Brunel* to its equivalent Wikidata ID

```
  PREFIX wdt: <http://www.wikidata.org/prop/direct/>
  SELECT? item WHERE {
     ? item wdt: P3074 'Isambard_Kingdom_Brunel'
}
```

Heritage Connector contains a component which converts URLs and IDs found in fields of collection records to owl:sameAs links to Wikidata, where the found URLs and IDs in a record point to pages which refer to the same real-world entity as the collection record. This is a three-step process with steps as follows:

1. *Find URL and ID patterns* through predefined regular expressions.
2. *Convert these to Wikidata QIDs* using a SPARQL query as in Listing 1, or the Wikipedia API[19] to resolve a Wikipedia URL.
3. *Filter found Wikidata QIDs to those which match the collection record.* We implement a set of filters in Python designed to be used interactively, where Wikidata records with different values of specified fields to the collection record (type, significant dates and label) can be filtered out of the list of matches.

Applying this process to SMG data we have made 2925 (28.3% of total) unambiguous links for people records, 697 (9.0% of total) links for organisation records, and 0 links for object records. Using a form added to the top of the collection website for internal museum users we obtained a further 157 links for people records, 39 links for organisation records and 30 links (0.01% of total) for object records.

4 | RECORD LINKAGE TO WIKIDATA

A record linkage system is one that tries to find all pairs of records (a, b), where a and b refer to the same real-world entity but are from two different data sources. In our case, the first data source is the HC knowledge graph and the second data source is Wikidata: a knowledge graph with over 91 million records. We describe two records which refer to the same real-world entity as being 'matched'. We face the following constraints when working specifically with museum data and Wikidata:

- Thin records, with few complete fields on average.
- A small amount of labelled data (number of [a, b] pairs that have been labelled as matching by a human annotator), owing to constraints on time and resources.
- Many records in the museum collection are likely to be in the long tail of Wikidata, meaning that a record's Wikidata equivalent is likely to be thin, or even that it may not exist at all.

As shown by the SemTab challenge on tabular to knowledge graph matching,[20] many approaches that have had recent success with this problem use algorithmic rather than deep learning methods. MTab[21] uses the joint probability distribution of a number of different signals including language and record type prediction to predict matches. TabularISI[22] uses TF-IDF on direct comparisons between the corresponding fields of two records to account for sparse data. Magellan,[23] a system for producing record linkage systems, helps developers produce a pipeline of candidate generation, feature engineering, then training a machine learning classifier such as a decision tree or Naive Bayes.

As well as algorithmic approaches, knowledge graph embeddings have recently been used to create effective record linkage systems.[24,25] Our experiments with training TransE[26] embeddings on a knowledge graph built from SMG's data led to embeddings that were too poor in quality to be used in a record linkage system, due to the data's 'thinness' – small number of properties per record and links between records – that the software is designed to overcome. Knowledge graph embeddings which incorporate information from literals[27] may well produce better results on similar knowledge graphs, but at the time of writing there are no mature libraries that can be used to produce any such embeddings.

4.1 | Our approach

A standard record linkage pipeline is shown in Figure 2 alongside our approach. To prevent the unfeasibly large computation of performing blocking (generating a list of candidate record pairs) against Wikidata, we replace the blocking and sampling steps with a search step which generates candidate matches for each HC record. We also implement a function that computes a suitable similarity score $s(a, b) \in [0, 1]$ based on the types (string, float, URI) of a and b, where a is a value from the HC knowledge graph, b is a value from a corresponding field in Wikidata, and both (a, b) are represented either by their URI or label. We show that a decision tree performs well as a classifier on SMG data, which brings the additional benefit that the classifier is explainable out-of-the-box.

The rest of this subsection describes each part of our record linkage approach in detail. Subsection 4.1.5 details results on SMG data.

4.1.1 | Loading records

At this stage, a SPARQL query is run against the HC knowledge graph to return all the records from a particular source, where the source is specified by a value for the predicate skos:hasTopConcept in the query. The data load has two modes: training, where only records with an existing owl:sameAs link to a Wikidata record are returned; and inference,

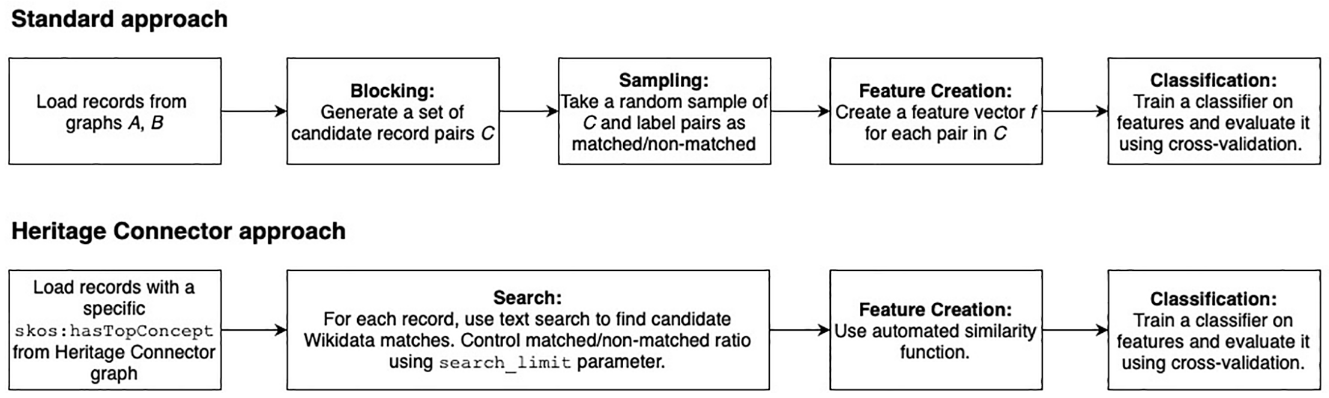

Standard approach
Load records from graphs *A*, *B*
Blocking: Generate a set of candidate record pairs *C*
Sampling: Take a random sample of *C* and label pairs as matched/non-matched
Feature Creation: Create a feature vector *f* for each pair in *C*
Classification: Train a classifier on features and evaluate it using cross-validation.
Heritage Connector approach
Load records with a specific skos:hasTopConcept from Heritage Connector graph
Search: For each record, use text search to find candidate Wikidata matches. Control matched/non-matched ratio using search_limit parameter.
Feature Creation: Use automated similarity function.
Classification: Train a classifier on features and evaluate it using cross-validation.

FIGURE 2 Comparison between a standard record linkage pipeline (Adapted from Magellan[23]) and our approach designed for Heritage Connector and Wikidata

where only records without such a link to Wikidata are returned, and the goal is to predict these links. There is an optional limit parameter to allow a developer to select only a subset of the available records.

The remainder of the steps up to and including feature creation are performed in batches with default size 100 (*batch_size* = 100).

4.1.2 | Search

We replace the *blocking* and *sampling* steps here with a step that performs a text search against Wikidata using the label of a HC record and returns the top N results. N can be specified by a developer by a search_limit parameter. Although Wikidata supports text search through the MediaWiki API,[19] the speed and stability of this service was insufficient in testing us to run searches for tens of thousands of collection records. To bypass the need to search Wikidata directly, we have developed a tool to load a subset of the Wikidata JSON dump[‡] into an Elasticsearch index. This tool has been open-sourced as *elastic-wikidata*.[28]

For each record the search method queries the label of the record against the *labels_aliases* field in the Elasticsearch index, which is a join of the Wikidata label and aliases for each Wikidata record. The QID, label, aliases and types of each returned Wikidata record are stored in memory for the batch.

In training mode the matching Wikidata record for each HC record is already known, so a y vector of length *batch_size* * N is created with values 1 for the *batch_size* pairs known to match, and 0 for the $(N-1)$ * *batch_size* pairs not labelled as matching. Retrieval of all record pairs known to match requires setting N so that the matching Wikidata record always appears in the top N search results for each HC record. In experiments we found that $N = 30$ gave 100% recall of known matches whilst keeping the data size small enough that the speed of the feature creation step was not impacted.

4.1.3 | Feature creation

The result of this step for each batch is a feature matrix $X = [f_1; f_2; \cdots; f_N]$, $X \in \mathbb{R}^{N \times m}$ where, N is batch size and each $f_i(a, b) = [s_1, s_2, ..., s_m]$ is a vector containing the similarity $s_j(a, b)$ between the value a of a property of a HC record and the value b of an equivalent property of a Wikidata record. There are m total features for HC records with the chosen skos:hasTopConcept.

To calculate these similarities for tabular data it is common to have to pre-specify or predict (as in SemTab[20]) both a mapping between equivalent fields in the two datasets and the data type of each field. We remove these requirements for our record linkage approach by:

- Using the Wikidata property *equivalent property* (*P1628*) to map between RDF predicates and Wikidata predicates.

- Designing a similarity function which uses Python type coercion to decide whether to apply a string or numeric similarity measure. If a HC value can be coerced to float then a numeric similarity measure is used, otherwise a string similarity measure is used.

There are two exceptions for which we use more bespoke similarity methods. We measure record type similarity – when the predicate is rdf:type – using the scoring function $1/(1 + d)$, where d is the shortest path between the types of the two records along the direction of the *subclass of* (*P279*) property calculated using Blazegraph's RDF GAS API.[29] To compare geographical values, where HC geographical values have already been converted to QIDs as in Section 3.2.2 and Wikidata geographical values are identified as being Wikidata records that are in the subclass tree of *Wikidata property to indicate a location* (*Q18615777*), we apply the same scoring function but along the direction of the *located in the administrative territorial entity* (*P131*) property.

For an HC record to pass through this stage of the pipeline the software enforces that it must have at least a label (compared against the label of each Wikidata record) and an rdf:type (compared against value of *instance of* (*P31*) for each Wikidata record).

4.1.4 | Classification

We formulate the record linkage task as a binary classification problem: predicting whether a pair of records (a, b) are matched or not based on their feature vector f. Given the small amount of data and number of features this is a simple machine learning problem; however, we benchmark a few different classifiers against taking the sum of each feature vector (the sum of the similarities) in the next subsection.

4.1.5 | Results

Ten-fold cross validation for three classification models as well as a “sum of scores” benchmark are shown in Table 2. The sum of scores predictions are generated for each record i as

$$y_{\text{pred}}(i) = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} f_i \geq T \\ 0, & \text{otherwise} \end{cases}$$

where, T is chosen separately for each classifier to maximise precision.

Separate classifiers were trained for *people* and *organisations*. Results for objects are not reported due to a prohibitively small amount of data to perform cross-validation. Training data proved difficult to collect for objects due to a lack of Wikidata records to describe the majority of the objects in the SMG collection. Our early experiments have shown that information retrieval (Section 5) rather than record linkage methods are better suited to linking objects to Wikidata, through linking concepts (such as events, locations and brand names) mentioned in object descriptions to the HC knowledge graph and to Wikidata. Record linkage of people and organisations who created, used or were depicted in these objects helps to further contextualise the objects in the collection using Wikidata.

We have implemented a decision tree as the record linkage classifier in HC as it performs well for SMG data. This has the additional benefit of being easily explainable: a developer can visualise a trained decision tree classifier to check

TABLE 2 Record linkage results for people and organisations in the SMG collections

	People			Organisations		
Model	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Sum of scores	0.874	0.558	0.681	0.253	0.664	0.367
Logistic regression	0.555	0.919	0.692	0.361	0.774	0.492
Decision tree	**0.953**	**0.958**	**0.956**	**0.851**	**0.383**	**0.528**
SVM	0.867	0.645	0.740	0.459	0.191	0.271

Note: Results for the model with the highest precision are highlighted.

it is operating as expected, without having to manually find and inspect examples of correctly and incorrectly matched record pairs.

Using this record linkage method we created 551 (0.2% of total) Wikidata links for objects (running inference on 15 000/3000 00 objects), 5343 (51.6%) links for people and 1692 (21.9%) links for organisations. We expect these numbers to increase as we add more metadata to our records using the methods described in Section 5.

5 | CREATING LINKS WITHIN THE COLLECTION (INFORMATION RETRIEVAL)

The approach we take to creating links within the collection is that of information retrieval (IR) on longer-form text in museum collection catalogues: extracting named entities from these passages of text and linking them to records in the HC knowledge graph and/or Wikidata. For the experiments in this section we apply IR methods to the description field of each record in the HC knowledge graph, but this same approach could be used for archival documents or blog posts related to a collection, for example. We break the process of IR down into two tasks:

- *Named entity recognition (NER)*: predicting named entities and their types in text spans;
- *Entity linking*: predicting links between these named entities and records in a knowledge base (here the HC knowledge graph and/or Wikidata), where both the named entity and the knowledge base record refer to the same real-world entity.

We are still developing an approach to entity linking based on existing methods[30,31] which will benefit from its own detailed analysis in future, so we leave it outside the scope of this article. Here we present an approach to NER as part of the IR process which can adapt to different collections without relying on any labelled data.

NER works via a machine learning model trained to identify and classify spans (entities) in text belonging to predefined categories such as *PERSON*, *ORGANISATION*, *EVENT*, *DATE* or *OBJECT*. These models are typically neural models trained on a large text corpus such as OntoNotes 5.0.[32] To adapt such a model to a new domain (such as a new museum collection) a technique called transfer learning is used, where the existing model is trained further on a dataset representing this domain. This new dataset typically consists of hundreds or thousands of sentences which have been manually labelled with correct entity annotations.[33] To avoid a museum having to undertake this considerable labelling task we present the following methods which work alongside an NER model (in an entity recognition 'pipeline'), to enable it to adapt to a new collection, and to the heritage domain, in the absence of labelled data:

1. Using the labels and types of records from the HC knowledge graph as a dictionary with which to annotate entity spans. Implemented in a custom spaCy[34] component, *DictionaryMatcher* (Subsection 5.1).
2. Filtering out entity mentions after annotation which are not likely to be entities based on their form using a custom spaCy component, *EntityFilter* (Subsection 5.2).
3. Annotating text spans as date entities by matching them to date formats commonly found in heritage collections using token patterns and regular expressions, which are stored in and shared from a central repository. Implemented in a custom spaCy component, *DateMatcher* (Subsection 5.3).

The custom spaCy components that enable these methods are available in *heritage-connector-nlp*,[7] alongside configuration for the annotation tool Label Studio[35] and Python code to test the performance of spaCy NER models.

5.1 | Using the labels of collection records as an entity annotator

The labels and types of records in the HC knowledge graph provide a lot of collection-specific information about named entities that are likely to be mentioned in the descriptions of collection records. For example, in the description of a pacemaker[§] in the SMG collection '*Cardiac pacemaker, induction type, comprises power unit and two induction coils, by Joseph Lucas of Birmingham in conjunction with Queen Elizabeth Hospital*', spaCy's en_core_web_lg model identifies 'Joseph Lucas' as having type PERSON, whereas from the collection records we can identify that 'Joseph Lucas' actually refers to engineering manufacturer 'Joseph Lucas Limited'[¶] which has type ORG.

We propose that the labels and types of collection records can be used as a dictionary to improve the performance of the entity recognition pipeline for a collection. To evaluate this, we extract the labels and types of all entities in the HC knowledge graph to a file that can be ingested by a custom spaCy component, *DictionaryMatcher*. For SMG data we use the skos: hasTopConcept value for each entity (*PERSON*/*ORGANISATION*/*OBJECT*) to map to NER entity type, but for another collection another source of this mapping could be needed. We exclude objects from the dictionary as many of their labels would not be considered named entities, for example, 'Microscope'. Labels which are duplicated across more than one record where those records have different types are removed from the export entirely, to avoid introducing ambiguity into entity type annotation.

5.2 | Filtering out false positive entities based on their form

Creating entity annotations using a dictionary can easily generate false positives. The *EntityFilter* component is designed to be used after any dictionary-based annotation method in an entity recognition pipeline, and removes found entities if they are all uppercase, all lowercase, or shorter than a predefined number of characters (default 3). An exclusion is made for tokens containing three or more numbers in row, as these could be dates. An example of the effect of using this component is shown in Figure 3.

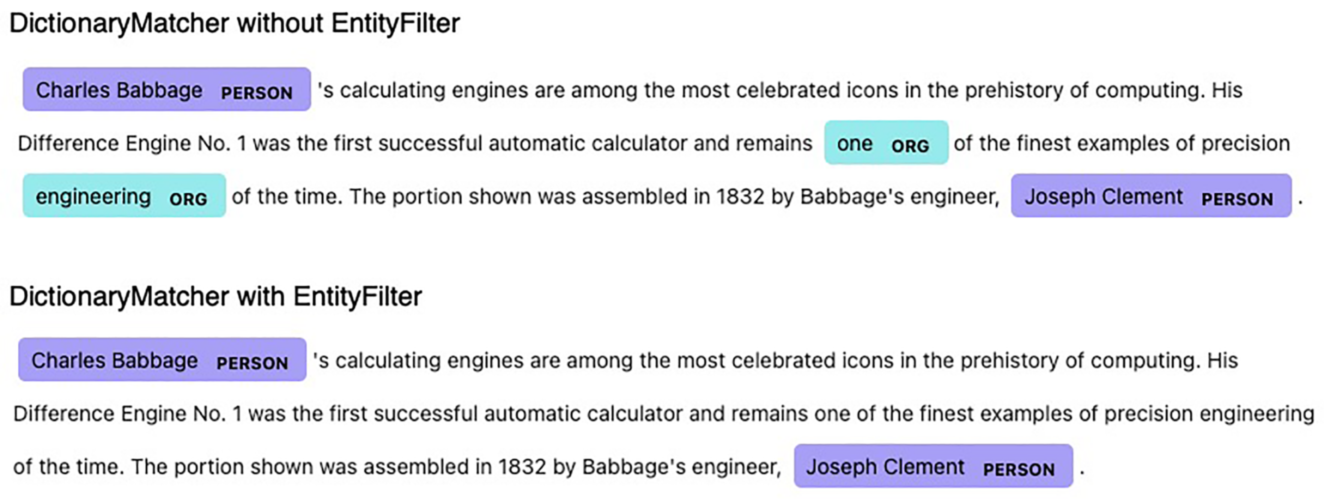
Whilst simple, this approach has proven effective for SMG data. Adjustments can easily be made to this method to accommodate for different characteristics of other museum collections, for example to disable or tweak any of the existing rules, specifically accept proper nouns where the first character is lowercase (eg, 'iPad'), or specifically accept spans which end in an abbreviation for a currency (eg, '24 USD').

5.3 | Rule-based annotation of catalogued dates

DateMatcher implements spaCy's *EntityRuler* component to annotate dates, specifically those that are common in museum catalogues but not reliably captured by spaCy's smaller models. Examples are 'c.1300-1370', or 'late 2nd century BC'. It is designed both to compensate for some of the shortfalls of NER models on heritage text and to act as a demonstrator for developers who may wish to add patterns they see in their own catalogues to the NER pipeline, which can then be shared with the wider community through the *heritage-connector-nlp* repository.

5.4 | Results

The results of using our dictionary-based and rule-based annotators at various points in the entity recognition pipeline are shown in Table 3. We show results on two different spaCy models: en_core_web_lg, a CNN trained


DictionaryMatcher without EntityFilter

Charles Babbage PERSON 's calculating engines are among the most celebrated icons in the prehistory of computing. His Difference Engine No. 1 was the first successful automatic calculator and remains one ORG of the finest examples of precision engineering ORG of the time. The portion shown was assembled in 1832 by Babbage's engineer, Joseph Clement PERSON .

DictionaryMatcher with EntityFilter

Charles Babbage PERSON 's calculating engines are among the most celebrated icons in the prehistory of computing. His Difference Engine No. 1 was the first successful automatic calculator and remains one of the finest examples of precision engineering of the time. The portion shown was assembled in 1832 by Babbage's engineer, Joseph Clement PERSON .

FIGURE 3 Dictionary-based entity annotation using DictionaryMatcher with and without the EntityFilter component, which removes phrases that have been tagged as entities that are unlikely to be entities based on their form

TABLE 3 Results for pretrained NER models and the same models augmented with rule-based and dictionary-based annotators, for both CNN (en_core_web_lg) and transformer (en_core_web_trf) models

Metric	NER model	NER model + collection labels	NER model + date annotation	NER model + collection labels + date annotation	NER model + collection labels + date annotation (overwrite allowed)
en_core_web_lg					
Precision	0.5138	0.5595	0.5239	0.5701	**0.5780**
Recall	0.6779	0.6683	**0.6992**	0.6896	0.6951
F1	0.5845	0.6091	0.5990	0.6242	**0.6311**
LOC F1	0.7894	0.7980	0.7894	0.7980	0.8068
ORG F1	0.5639	0.5532	0.5639	0.5532	0.5797
DATE F1	0.7977	0.7977	0.8571	0.8571	0.8583
OBJECT F1	0.2805	0.2840	0.2805	0.2840	0.2733
PERSON F1	0.6519	0.6637	0.6519	0.6637	0.6505
NORP F1	0.6771	0.6771	0.6632	0.6632	0.6374
EVENT F1	0.2759	0.2857	0.2759	0.2857	0.2857
en_core_web_trf					
Precision	0.5938	**0.6403**	0.5938	**0.6403**	0.6259
Recall	**0.7333**	0.7250	**0.7333**	0.7250	0.7092
F1	0.6562	**0.6800**	0.6562	**0.6800**	0.6649
LOC F1	0.8121	0.8099	0.8121	0.8099	0.8093
ORG F1	0.6625	0.6453	0.6625	0.6453	0.6078
DATE F1	0.9056	0.9056	0.9064	0.9064	0.9049
OBJECT F1	0.1932	0.1970	0.1932	0.1970	0.1949
PERSON F1	0.7464	0.7464	0.7464	0.7464	0.6964
NORP F1	0.8257	0.8257	0.8093	0.8093	0.7847
EVENT F1	0.2500	0.2500	0.2500	0.2500	0.2500

Note: 'Collection Labels' refers to the joint use of the methods in Sections 5.1 and 5.2, and 'Date Annotation' refers to the method in Section 5.3. The suffix '(overwrite allowed)' signifies that the annotator described in Section 5.1 was allowed to overwrite entity annotations set by the NER model, and its absence means that the dictionary annotator was only allowed to annotate text spans which had not already been annotated. Precision, recall and F1 are at individual entity level.

on GloVe[36] vectors; and en_core_web_trf, a transformer-based RoBERTa[37] model. Both models are trained on the OntoNotes 5.0 corpus.[32] We combine the labels GPE and LOC into LOC; PRODUCT and WORK_OF_ART into OBJECT; and ignore all other labels in the OntoNotes corpus not listed in Table 3.

Our results show that using all components as additional annotators improves overall model performance with no manually-labelled data, with the pipelines which use all of *DictionaryMatcher*, *DateMatcher* and *EntityFilter* showing an increase in F1 score of 4.6 points for en_core_web_lg and 2.4 points for en_core_web_trf above the pretrained NER model baseline.

The improvement when these components are used, however, is smaller for the transformer model compared with the CNN model – for some entity types there is a small positive or negative change in performance. This may indicate that the transformer model alone is performant enough to capture the 'expert knowledge' we are trying to inject using these extra components.

Another consideration is inference time: on a Macbook Pro with an Intel i5 CPU and 8GB of RAM the average inference time for the transformer-based model is 0.13it/s, compared with 0.01it/s for the CNN-based model. Scaling this to the 350 000 records in the SMG collection means that running inference for the entire collection would take around 50 minutes for the CNN-based model compared with approximately 12 hours for the transformer-based model. This difference would become more apparent on larger collections such as the Victoria and Albert Museum's, which holds over 2.3 million objects,[38] as well as people and organisation records.

6 | CONCLUSION AND FURTHER WORK

In this article we show that small machine learning models which can be run on large datasets on a mid-performance consumer laptop are useful tools for creating Linked Open Data from museum collections, particularly in tackling the problems of record linkage and information retrieval. We show that small machine learning models are effective for performing record linkage between museum collections and Wikidata, using only a small amount of labelled training data. Using the SMG collection we also show that both using labels of collection records as a dictionary and encoding some heritage-specific knowledge (in this case date formats common in museum catalogues) as rules can improve the performance of NER methods as part of the information retrieval process with no manually annotated data. This performance increase is particularly significant on smaller models which museum developers are able to run on their own laptops on datasets the size of digital museum collections.

In the remainder of the Heritage Connector project we plan to keep developing our approach to information retrieval for museum catalogues, apply these same approaches to the collection of the Victoria and Albert Museum and build a set of prototype interfaces onto the knowledge graph to enable exploration and discovery of collections and related content by users.

ACKNOWLEDGMENTS

We would like to thank the members of the Heritage Connector project team for their input: Jane Winters, Jamie Unwin, Rhiannon Lewis, Dr Tim Boon, Angela Wolff and Richard Palmer. We would additionally like to thank James Clemoes, Rhiannon Lewis, Jamie Unwin, Jane Winters and Tim Boon for reviewing this article before submission.

Heritage Connector is funded by the Arts and Humanities Research Council's *Towards a National Collection: Opening UK Heritage to the World*** programme.

DATA AVAILABILITY STATEMENT

The data that support the findings of this work are available from the authors upon reasonable request.

ORCID

Kalyan Dutia https://orcid.org/0000-0003-4678-1597

John Stack https://orcid.org/0000-0003-2514-429X

ENDNOTES

* https://www.wikidata.org/wiki/Wikidata:Statistics

† https://www.elastic.co/guide/en/elasticsearch/reference/master/dense-vector.html

‡ Available at https://dumps.wikimedia.org/wikidatawiki/entities/

§ https://collection.sciencemuseumgroup.org.uk/objects/co137358/cardiac-pacemaker-birmingham-england-1960-1970-cardiac-pacemaker

¶ https://collection.sciencemuseumgroup.org.uk/people/cp80874/joseph-lucas-limited

** https://ahrc.ukri.org/research/fundedthemesandprogrammes/tanc-opening-uk-heritage-to-the-world/

REFERENCES

1. Hogan A, Blomqvist E, Cochez M, et al. Knowledge graphs. *arXiv Preprint*, 2020:arXiv:2003.02320.
2. Tharani K. Much more than a mere technology: a systematic review of Wikidata in libraries. *J Acad Librariansh*. 2021;47(2):102326.
3. de Boer V, de Bruyn T, Brooks J, de Vos J. The benefits of linking metadata for internal and external users of an audiovisual archive. In: *Metadata and Semantic Research*. Cham: Springer International Publishing, Cham; 2019:212-223.
4. Berners-Lee Tim. *Linked Data - Design Issues*. (2006). https://www.w3.org/DesignIssues/LinkedData.html. Accessed December 10, 2020.
5. 5-star Open Data. https://5stardata.info/en/. Accessed December 10, 2020.
6. Dutia Kalyan, Unwin Jamie, Stack John. Heritage Connector – GitHub. https://github.com/TheScienceMuseum/heritage-connector. Accessed December 15, 2020.
7. Dutia Kalyan. Heritage Connector NLP – GitHub. https://github.com/TheScienceMuseum/heritage-connector-nlp. Accessed December 15, 2020.
8. Denny V, Markus K. Wikidata: a free collaborative knowledgebase. *Commun ACM*. 2014;57(10):78-85.
9. W3C. RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/. Accessed December 15, 2020.
10. Science Museum Group. Annual Review. https://www.sciencemuseumgroup.org.uk/about-us/annual-review/. Accessed December 10, 2020.

11. Science Museum Group. Science Museum Group Collection. https://collection.sciencemuseumgroup.org.uk/. Accessed December 10, 2020.
12. Collections Trust. Terminologies - Collections Trust. https://collectionstrust.org.uk/terminologies/. Accessed March 8, 2021.
13. Heritage Connector Blog. Webinar: Wikidata and Cultural Heritage Collections. https://thesciencemuseum.github.io/heritageconnector/events/2020/06/22/wikidata-and-cultural-heritage-collections-webinar/. Accessed December 10, 2020.
14. Apache. Apache Jena - Apache Jena Fuseki. https://jena.apache.org/documentation/fuseki2/. Accessed December 15, 2020.
15. Wang Q, Mao Z, Wang B, Guo L. Knowledge graph embedding: a survey of approaches and applications. *IEEE Trans Knowl Data Eng*. 2017;29(12):2724-2743.
16. Geonames. Geonames. http://www.geonames.org/. Accessed December 15, 2020.
17. Grace's Guide Ltd. Grace's Guide to British Industrial History. https://gracesguide.co.uk/Main_Page. Accessed December 15, 2020.
18. Oxford University Press. Oxford Dictionary of National Biography. https://www.oxforddnb.com/. Accessed December 15, 2020.
19. MediaWiki. API:Main Page. https://www.mediawiki.org/wiki/API:Main_page. Accessed December 18, 2020.
20. Hassanzadeh O, Efthymiou V, Chen J, Jiménez-Ruiz E, Srinivas K. *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2019)*. Zenodo. 2019.
21. Nguyen P, Kertkeidkachorn N, Ichise R, Takeda H. MTab: matching tabular data to knowledge graph using probability models. *In SemTab, ISWC Challenge, CEUR-WS.org*, 2019;2553:7–14.
22. Thawani A, Hu M, Hu E, et al. Entity linking to knowledge graphs to infer column types and properties. *In SemTab, ISWC Challenge, CEUR-WS.org*. 2019;2553:25-32.
23. Konda Pradap, Das Sanjib, Suganthan GC Paul, et al. Magellan: toward building entity matching management systems. In Proceedings of the VLDB Endowment. 2016;9(12):1197–1208.
24. Azmy M, Shi P, Lin J, Ilyas IF. Matching entities across different knowledge graphs with graph embeddings. *arXiv Preprint*. 2019;arXiv:1903.06607.
25. Chabot Yoan, Labbe Thomas, Liu Jixiong, Troncy Raphaël. DAGOBAH: an end-to-end context-free tabular data semantic annotation system. In The 18th International Semantic Web Conference. 2019:41–48.
26. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. *Adv Neural Inf Process Syst*. 2013;26:2787-2795.
27. Asefa Gesese G, Biswas R, Alam M, Sack H. A survey on knowledge graph embeddings with literals: which model links better literal-ly? *arXiv Preprint*. 2019;arXiv:1910.12507.
28. Dutia Kalyan. Elastic Wikidata – GitHub. https://github.com/TheScienceMuseum/elastic-wikidata. Accessed December 15, 2020.
29. BlazeGraph. RDF GAS API. Blazegraph Wiki. https://github.com/blazegraph/database/wiki/RDF_GAS_API. Accessed December 16, 2020.
30. Klie Jan-Christoph, Castilho Richard Eckart, Gurevych Iryna. From zero to hero: Human-in-the-loop entity linking in low resource domains, S. 6982–6993. In The 58th annual meeting of the Association for Computational Linguistics (ACL 2020), Virtual Conference; 2020.
31. Wu Ledell, Petroni Fabio, Josifoski Martin, Riedel Sebastian, Zettlemoyer Luke. Zero-shot entity linking with dense entity retrieval. 2020.
32. Linguistic Data Consortium. OntoNotes Release 5.0. https://doi.org/10.35111/xmhb-2b84. Accessed December 17, 2020.
33. Rodriguez Juan Diego, Caldwell Adam, Liu Alex. Transfer learning for entity recognition of novel classes. In Proceedings of the 27th International Conference on Computational Linguistics; 2018:1974–1985.
34. Honnibal Matthew, Montani Ines, Van Landeghem Sofie, Boyd Adriane. spaCy: Industrial-strength Natural Language Processing in Python. 2020.
35. Tkachenko Maxim, Malyuk Mikhail, Shevchenko Nikita, Holmanyuk Andrey, Liubimov Nikolai. *Label Studio: Data labeling software*; 2020. https://github.com/heartexlabs/label-studio.
36. Pennington Jeffrey, Socher Richard, Manning Christopher D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014:1532–1543.
37. Liu Y, Ott M, Goyal N, et al. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint*. 2019:arXiv:1907.11692.
38. Victoria and Albert Museum. About Us. https://www.vam.ac.uk/info/about-us. Accessed December 10, 2020.

How to cite this article: Dutia K, Stack J. Heritage connector: A machine learning framework for building linked open data from museum collections. *Applied AI Letters*. 2021;2(2):e23. https://doi.org/10.1002/ail2.23