Above the Surface

Organizing the digitally processed archaeological finds of the north/southline

Abhilash M. Abhilash 12345678
Camilla Santos Andersen 12345678
Chris Atherton 12345678
Radu David 12345678
Danny de Vries 14495643
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

1 INTRODUCTION

From 2003 to 2012 excavations took place for the creation of the North/South metro line in Amsterdam. At Damrak and Rokin, which are unlikely archaeological sites due to being in the city center, archaeologists had a chance to physically access the riverbed. During these excavations in the Amstel over 700,000 objects were preserved which resulted in the archaeological collection called 'Below the Surface' [1] commissioned by the Municipality of Amsterdam.

The collection has a great variety of objects, from tools over centuries old to credit cards recently lost which makes the collection a rare source of urban history. All objects are digitally processed (e.g. photographed, labeled, metadata added) and displayed on a front-end website at *belowthesurface.amsterdam*. This website shows an overview of all the objects and a detail page with metadata of a particular object but no further categorization or classifications. This research aims to further organize this collection of objects with a focus on grouping the items by *functional properties*, determining *cultural relevance*, and researching *object relationships*.

2 RELATED WORK - CHRIS

2.1 Below the Surface

The starting point for re-organizing the collection, creating aggregated datasets, and exploring overview data is the dataset downloadable in *.csv* format on the Below the Surface project website. During the excavations around 700,000 objects were found, the dataset is a subset and contains around 20,000 objects that are digitally processed and rudimentary labeled. A separate data field description [2] file can be downloaded which further explains the controlled vocabulary used for the dataset.

[1] https://belowthesurface.amsterdam/en
[2] https://belowthesurface.amsterdam/en/pagina/publicaties-en-datasets

2.2 Museum research - Desk Research - Chris

2.3 Antique collectors - Chris

2.4 Archaeologists - Chris

2.5 Academic Research - Chris

2.6 Machine Learning - Radu

3 METHODOLOGY - ABHILASH

3.1 Glushko's Questions on Information Organisation

3.1.1 What is the main purpose of the website? The above the surface website focus to provide an interactive platform for users to explore the archaeological finds from the excavation of Amsterdam's North-South metro line.

3.1.2 Who is it for? The website is for anyone interested in archaeology, history, and Amsterdam's heritage. The website has collections of items from different time period so can act as a gateway for collectors of ancient antique artifacts.

3.1.3 Why it is being organized? Users can use the website to learn about the archaeological finds from the excavation of Amsterdam's North-South metro line, view images of the artifacts, and read about their historical significance.

3.1.4 Where it is being organized? The website provides information about the project, a timeline of the excavation, and a gallery of images of the artifacts found during the excavation

3.1.5 How it is being organized? The website is organized into four main sections: Home, Project, Timeline, and Gallery. Each section has sub-sections that provide more detailed information.

3.2 Data Dictionary and Metadata - Chris

3.3 Ontology - Chris

3.4 Data Model

The data model for the website is designed to manage the archaeological finds efficiently. The primary goals of this data model are:

- Normalize the data to minimize redundancy and maintain data integrity.
- Provide a unified structure for different material categories.
- Enable the storage of common attributes shared by all archaeological finds.

The data model as shows in Figure 1 is structured into following main parts:

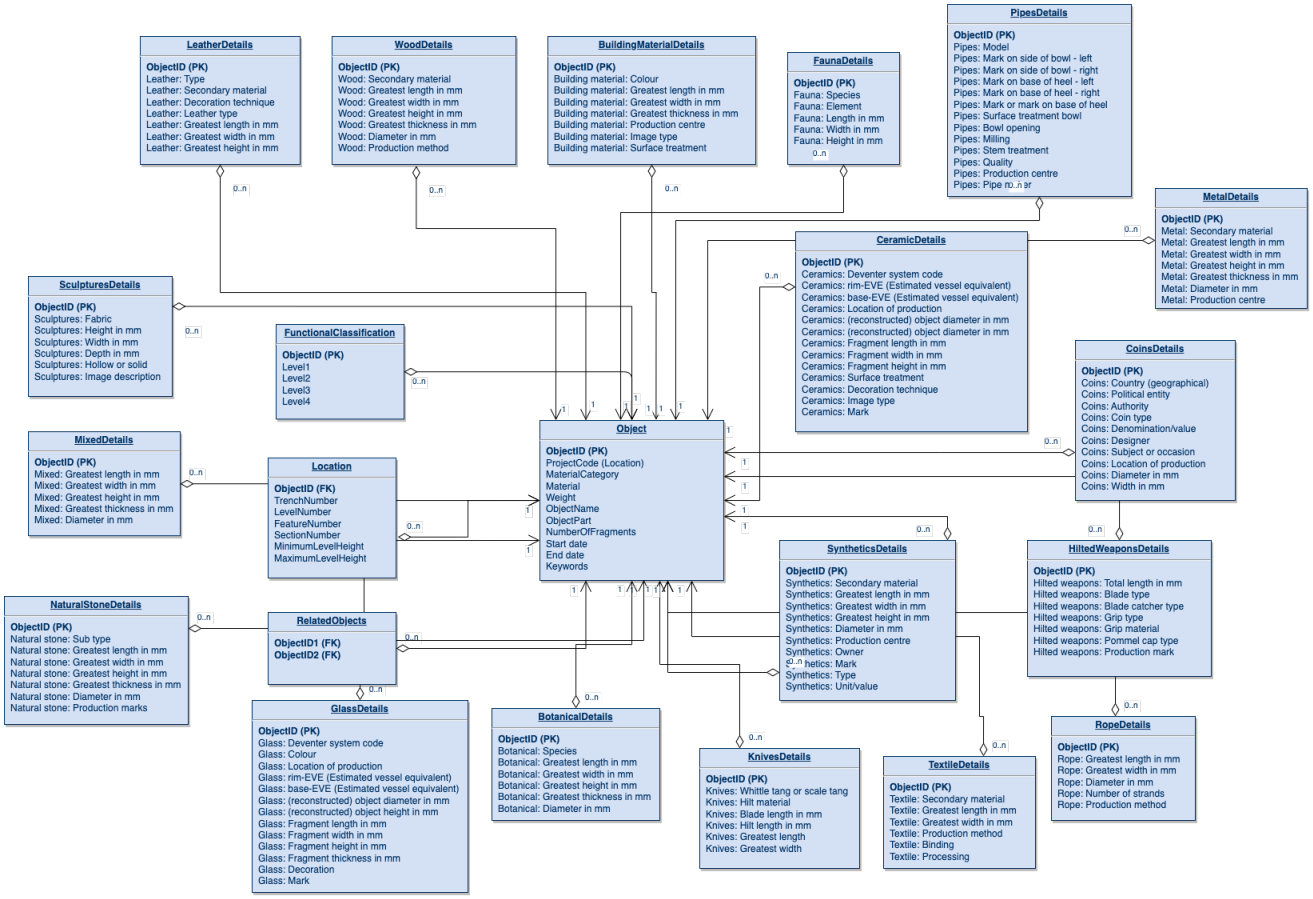

Figure 1: Data Model

3.4.1 Object. This table is central to the data organisation for common attributes of all archaeological finds. It includes information such as the ObjectID, project code, category, subcategory, weight, object name, object part, number of fragments, and temporal data (start date, end date). The ObjectID field acts as the identifier for all objects and is the primary key.

3.4.2 Location. This table contains fields for location information (trench number, level number, feature number, and section number), as well as keywords and classification levels. A binary field indicates whether the find is listed on the website. ObjectID is relating the location with Object as foreign key.

3.4.3 Functional classification. This table contains the main functional usage of the object. It is divided into catergories and level of functional classification between Level 1 to Level 4.

3.4.4 Related Objects. The "Related Objects" table is a critical component of the data model, as it enables the establishment of relationships between different archaeological finds. This table serves as a bridge to link findings that may have connections between each other. In the context of the model, it enables the website to visualize associations between artifacts and materials within and across categories.

3.4.5 Material-Specific Tables. Each material category (e.g., Ceramics, Glass, Building Materials, etc.) is represented by a dedicated table linked to the Object Table through a foreign key relationship. These material-specific tables store attributes that are unique to each category, including dimensions, production details, and any other specialized data relevant to the category.

3.5 Technical Implementation of the website

The website is a custom front-end mostly using web standards and open-source software and libraries. The assumption is that it will mainly be used in a desktop environment by the users to explore on larger screens so the website is not fully responsive and thus not mobile-optimized.

3.5.1 Front-end frameworks. The web application is created with the open-source front-end framework Svelte [3] and UI framework SvelteKit which allows the application to be built-in interface components, each chart is rendered separately making it more efficient to add functionality (e.g. add datasets, render different chart types) in the future but also makes the website performant when more data and charts are added. Svelte can be downloaded as a module (package)

[3] https://svelte.dev

from NPM [4] and uses the JavaScript back-end run-time Node.js [5]. For the charts the JavaScript charting library Chart.js [6] is integrated into the components which allows charts to be rendered in HTML5 Canvas without much configuration.

3.5.2 Dataloading. The processed and transformed dataset was used as a primary data source of which subsets of the dataset in *.csv* format, roughly one per year chart and section, are converted to *.json*. These data files are loaded on page load of the browser. For this prototype, no back-end was set up and no database queries are being made. Any filter options and updating of the charts are more custom, it uses JavaScript utility functions to allow the data to be pre-processed and have only the data change.

3.6 Machine Learning Model

3.6.1 Data-set presentation. Bellow the surface provides a dataset[**?**] of all the objects resulting from the excavations. The data is provided in the form of a .csv file, with 139190 rows and 163 columns. Each row corresponds to an object. Describing each object is well outside of the scope of the purposes of this section, however, an explanation of the relevant columns is necessary.
The following columns are relevant for the purposes of the ML model:

- vondstnummer - represents a unique inventory number, in the form of a string. Every object has a vondstnummer. Example: "NZC1.00001MTL001".
- object - a description of the contents of the object. Example: "sieve residue"
- subcategorie - a categorisation of the object material. Example: "metal: copper alloy"
- objectdeel - describes the object type morphologically (if it is part of a bigger object, a set, etc). Example: "fragment"
- vlak_min - Describes the minimum depth at which the object might have been found. Example: "-22.0"
- vlak_max - Describes the maximum depth at which the object might have been found. Example: "-22.01"
- begin_dat - The beginning of the interval of the estimated year of the object. Example: "1675.0"
- eind_dat - End of the interval of the estimated year of the object. Example: "1725.0"
- niveau1 - The category in which the object is placed. Example: "Communication & Exchange"

For the columns *object, subcategorie, objectdeel, vlak_min, vlak_max, begin_dat, eind_dat, niveau1* there are rows in the dataset in which one or more of these columns are blank.
The column *niveau1* can take the value of one of 12 pre-determined categories, as well as the value "Not classified". As previously mentioned, there are rows where this column is blank.

3.6.2 Objectives. Our objective is to create a machine-learning model that will complete the missing data for the "niveau1" column. This means that our model will predict a value in the niveau1 column, for the rows where currently that column is blank or has the value "Not classified". The prediction will be based on the values in the *object, subcategorie, objectdeel, vlak_min, vlak_max, begin_dat, eind_dat, niveau1* columns, which will act like input to the machine-learning model.

3.6.3 Deliverables. In order to achieve our objectives, the following files are delivered:

- process_dataset.py - a simple python script that takes the original 163 column .csv files and consolidates it into another .csv files that only contains the columns of interest. The name of this .csv file is "selected_dataset.csv"
- machine_learning.py - this python script is the backbone of the machine-learning process. It is a more-complex script that does the following steps:
  - Loads the "selected_dataset.csv" dataset
  - Preprocess the data (completes the values with 0 or placeholders here they are blank", etc)
  - Converts text stings to vectors
  - Splits the data into unlabeled and labelled data based on the values in the "niveau1" column
  - Splits the labelled data using a training, testing, and validation split
  - Builds the ML model
  - Compiles the model
  - Trains the model
  - Tests the model
  - Predicts the values of niveau1 for the unlabelled data
  - Saves the updated dataset into a file named predicted_dataset.csv
- predicted_dataset.csv - a file containing the dataset in selected_dataset.csv, but with the column selected_dataset.csv fully completed.

3.6.4 Model description. Given the problem, a relatively simple neural network model was chosen. It consists of 1 input layer, 2 hidden layers and one output layer. The input layer has 7 input neurons, coresponding to the following variables:

- subcategorie
- object
- objectdeel
- vlak_min
- vlak_max
- begin_dat
- eind_dat

For the text fields, a Text-to-Vector conversion was necesarry. We used the word2vec_model for that. The data of the other columns was normalised.
The hidden layers were composed of 10 neurons (dense layers) with a relu activation function. The outputted has 12 neurons, with a softmax activation function, each neuron corresponding to one of the 12 category values niveau1 can take.
The neural network was trained on the labelled data, using a 80 - 10 - 10 training - validation - testing split.

4 RESULTS - ABHILASH

4.1 Website

The original website has a lot of individual objects and detailed metadata about physical properties. The aim of the website was mainly to *summarise* the collection to allow the before-mentioned

[4] https://www.npmjs.com
[5] https://nodejs.org/
[6] https://www.chartjs.org

users to explore the broad dataset and find interesting patterns. Then the user would be able to *create a subset* of the dataset based on physical properties and categories by further filtering the dataset. The design is based on the branding of the original below the surface project and follows standard information architecture (e.g. primary navigation, form filters) and visual design principles.

4.1.1 Overview summary page. On loading of the website the user is greeted with a 'summary' overview page (shown in Figure 2) with introductory text and three visualiations that display; *time of origin* in horizontal stacked bar charts, *functional properties* in a polar chart and *material usage* in a doughnut chart. The page has a primary top navigation to 'smooth-scroll' to each section and a 'create a subset' floating action button to navigate to the collection detail page.

4.1.2 Collection detail subset. Based on the categories of the summary page a user can create a subset of the collection on the 'create a subset' detail page. Above the fold the user has the option to use the same *category filters* from the summary page. Then a user additionally has the option to click on the filter drowpdown to expose more form filter options using checkbox with more granular options such as location found, size and dimensions and material technique. A 'download subset' button allows the user to download the generated subset in .csv format.

4.2 Machine Learning Model

Following training the neural network over 50 ephocs, the accuracy of the neural network was tested. An accuracy of aprox. 75% was obtained over the test data, which the neural network has never seen before.The accuracy is considered satisfactory for the given application. Furthermore, the neural network was used to clasify the remaining unlabelled data, the results being demed

4.3 Dataset

4.3.1 Initial Dataset. The initial dataset as provided in belowthesurface website was structured as a flat model with each row representing an individual archaeological object. This flat structure included many columns, each corresponding to specific attributes associated with the objects. These attributes stored a wide range of material categories, such as ceramics, building materials, fauna, glass, and more. While the flat model provided an accessible format for data entry, it led to redundant storage of certain information, challenging in visualization, particularly common attributes that applied to all finds.

4.3.2 Data Transformation and Normalization. As the dataset was provided in a flat model, which presented certain challenges in terms of data redundancy and efficient visualization. To address these issues and enhance the integrity of the data, a normalization process was done to fit the data into the relational model as described in the Data Model section.

4.3.3 Normalized datasets. All the datasets in accordance with the datamodel is uploaded into the datasets repository of GitHub. A csv file for each table of the data model is created based on the initial dataset.

4.4 Ontology and Data Dictionary - Chris

5 CONCLUSION - CAMILLA

6 DISCUSSION - CAMILLA

6.1 Reflection

7 FUTURE WORK - CAMILLA

7.0.1 Website. Additional visualization features and filter options need to be added to the website in the next iteration of the website. Currently, the website only visualizes broad categories but there are more subcategories in the dataset that the user might be able to click on to get more detailed views and overlays. From a user experience point of view, further usability testing needs to be done to validate the User Experience (UX) and User Interface (UI) of the prototype website. Direct feedback from the users would further validate the workings of the website and uncover hidden interface and interaction problems. From a technical perspective, the website currently relies on exported data that is then loaded into the web visualization. A further enhancement for the website, and to make it more dynamic, is to have the dataset exposed through a Query-like API (e.g. GraphQL [7]) with an underlying 'headless' database (e.g. MongoDB [8]) in which the website is able to fetch up-to-date collection real-time data. This also allows for more performant and dynamic filter options on the 'create a subset page' through the use of a back-end.

8 CONTRIBUTION OF AUTHORS

- *Abhilash:* did stuff
- *Camilla:* did stuff
- *Chris:* did stuff
- *Radu:* did stuff
- *Danny:* programmed the website and visualizations (front-end development) as well as conceptualized the interface interactions (interaction design) and look and feel (visual design).

9 ACKNOWLEDGEMENTS

We thank lecturer dr. V.O. Degeler (University of Amsterdam) for providing guidance and assistance during the project and MsC. A. Fleck (University of Amsterdam) who provided valuable feedback and answered our questions during the seminars which helped us further expand our research.

10 APPENDIX

10.1 Hosted source code

The source code of the web-based visualization, Python notebooks of the machine learning model and datasets used are hosted on GitHub using the MIT License. Under the *uvaio* username we have several code repositories:

(1) Notebooks: Source Code for the Jupyter Notebooks for data processing and machine learning. https://github.com/uvaio/notebooks
(2) Website: Source Code for the custom front-end website and interface. https://github.com/uvaio/website

[7] https://graphql.org/
[8] https://www.mongodb.com/

(3) Datasets: The processed and modeled datasets and the ontology. https://github.com/uvaio/datasets

10.2 Live version

A live demo version of the front-end website and visualization (desktop only) is hosted on Netlify and can be viewed using the following link https://uvaio.netlify.app

10.3 Website screenshots

Shown in Figures 2 and 3.

REFERENCES

[1]]DatasetBelow [n. d.]. Bellow the surface dataset. https://statics.belowthesurface.amsterdam/downloadbare-datasets/Downloadtabel_EN.csv. Accessed: 2023-10-08.

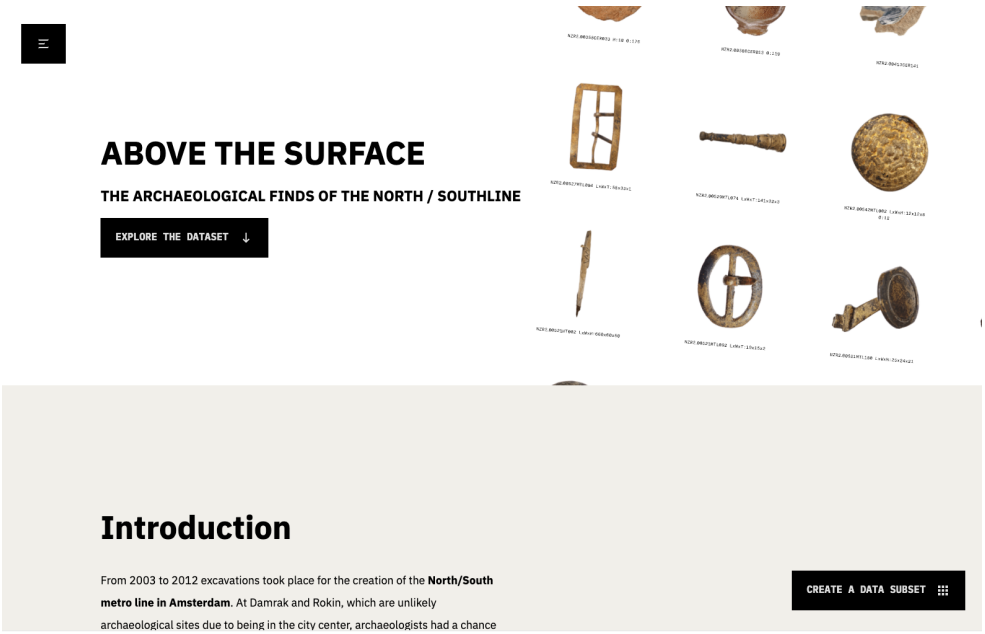

Figure 2: Screenshot of the 'overview landing page' of the website

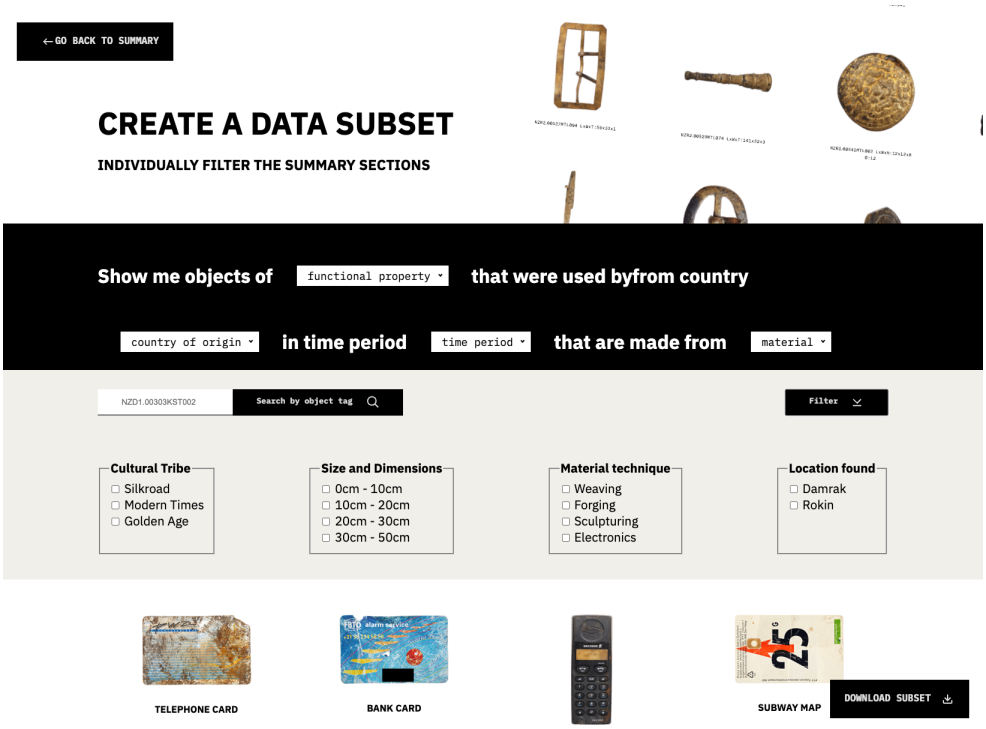

Figure 3: Screenshot of the detail 'create a subset' page of the website