Rens Mester & Jim Lemmers

The goal, domain and scope:

We will provide a bicyclist or pedestrian with a more scenic route than the shortest route based on the starting point and endpoint in Amsterdam. This scenic route can for example consist of bridges (nice overview along the water), monumental buildings, monumental nature and squares.

The scope of the ontology is bound to representing places in Amsterdam with a latitudinal and longitudinal coordinate. We subclass these places to what they represent in their basic form in the real world, e.g., bridges, buildings, squares and nature. These are then further subclassed with their purpose in mind, e.g., a heritage building, religious building, a museum, a tree or a pond. These subclasses are used to infer what a scenic route should contain. We would like to make the restrictions on a Scenic Route as broad as possible, so that we can infer a scenic route from as many combinations of types of places.

The first screen presented to the user is a form where the starting point and endpoint of the route are to be entered. When the user has submitted this form, the application will use the google maps API to receive the shortest route. The json response from Google Maps will be dynamically converted to RDF and classified as an individual with the ‘ShortestRoute’ class with a ‘hasPlace’ property pointing to the steps in the route.

Using the route graph we created, we will query the database for objects close to the places in the ‘ShortestRoute’ and insert them in the route graph. This is achieved by drawing an imaginary square around each step and checking which objects fall into these squares. These objects are then classified as ‘InterestingPlace’ and linked to the Route instance with the ‘hasPlace’ property. The Scenicroute will be inferred by, e.g., a set of minimal 2 bridges, minimal 2 monuments and some monumental trees.

If a ‘ScenicRoute’ is inferred in the dynamic graph, we make an API call to Google Maps in order to calculate the shortest route with the interesting objects as waypoints. Google provides an optimization in calculating the route between the waypoints, so we don’t have to worry about that. The response we get back from the second query to Google Maps is then drawn onto a map for the user to see.