Software Requirements Specification (SRS) Document

Web Application for Ontology Verification

Team 22 (Shikhar, Anirudha, Saujas)

The semantic web refers to websites and online content that is not only interpretable by humans, but by computers as well. To ensure machines can interpret and usefully extract information from these websites (beyond the data analytics possible on the web today), the website has to have information organised in a way that is different compared to existing websites. One way of structuring knowledge available on the internet is to use an ontology, which is a formal way to define concepts and the relationships between different concepts. Concepts refer to classes of objects or ideas, and relations describe how these classes, or instances of these classes, are related to one another.
Building large scale ontologies that accurately capture real world knowledge isn't easy, and to have humans annotate entire ontologies would be highly impractical. Machines are capable of working with large scale data, but don't have the real world knowledge that humans do. A system that takes machine annotated ontologies, and allows humans to look and make corrections where necessary would allow for the best of both worlds — the capacity of a machine and the accuracy of a human. We seek to build an application that allows for humans to review annotations made by machines, paving the way to building knowledge structures that machines can understand.

Brief problem statement:

- To build an application for accepting or rejecting new concepts/relationships and save the new state after accept/reject request is made in the ontology file.
- Build a mechanism for aggregating the decisions made by multiple experts and generate one final ontology that takes into account the decisions made by different users for each new concept/relationship, and decides which decision will reflect in the final ontology

System requirements:

The following languages and technologies will be used to develop the application:

- HTML, CSS, and JavaScript on the frontend:
  - WebVOWL (an existing web application that provides a GUI for viewing ontologies) to visualise ontologies
- Python to build a Flask server for the backend

Functional Requirements:

- A login page and authentication mechanism
- UI elements for experts to accept or reject annotations
- A structure designed for storing OWL files and updating them with expert decisions

- A mechanism to weigh and aggregate decisions by multiple users and store one final decision

Non-functional requirements:

The application will run on Firefox and Chrome without additional plugins/ dependencies

Users profile:

- Domain Experts:
  - Domain experts will be viewing the available ontologies and they will determine whether to accept or reject a particular annotation to an existing ontology file
  - The experts will be using the system in user mode that is they will have a username and a password
  - They are also expected to have basic familiarity with handling computers that is they are comfortable with handling of the graphical interface of commonly used web applications
- Administrators:
  - Administrator will be monitoring over the handling of the existing ontologies and they will be using the system in admin mode.
  - The admin is expected to be proficient at handling computers.
  - The admin should also be able to add annotations to the existing stored ontology files.

Use Case Table:

No.	Use Case Name	Description	Release
1	Load and Visualise an Ontology	• User should be able to load an ontology of their choice and view • The user should be able to see newly added relationships in a different colour • The user should see buttons to accept or reject a new relationship appear in the sidebar when they click the relationship	R1
2	Accept/Reject a Relationship	• Once the user clicks the accept/reject button, the decision should be stored in the database • The colour of the relationship should change once the user has decided it	• Decision storage by R1 • Colour change by R2

3	Accept/Reject a class	• Once the user clicks the accept/reject button, the decision should be stored in the database • The colour of the class should change once the user has decided it	• Decision storage by R1 • Colour change by R2
4	Get verified ontology	• The admin should be able to aggregate the decisions made by multiple users into one final decision • The admin should be able to generate a file containing with all the aggregated decisions having taken effect	R2
5	Store ontology	• The admin should be able to store an ontology file with all the new relationships specified for verification by experts	R2