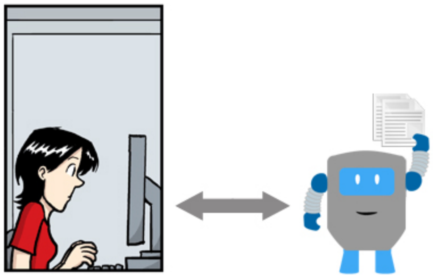
COMP 474/6741 Intelligent Systems (Winter 2024)

Project Assignment #2

Due date (Moodle Submission): Monday, April 15th
Counts for 50% of the course project

In this second part of our *Intelligent Systems* project, your tasks are:

- Fixing any issues that surfaced in Part #1 of your project;
- Populating the knowledge base with the course *content* from the two courses where you collected their material; and
- Adding a natural language interface to make it a real 'chatbot.'



Project #1 Updates. In case there were issues with your first project submission pointed out to your group during the demo, please make sure you address them before working on the new parts.

Knowledge Base Topic Population. We'll continue the automatic knowledge base construction for your bot's brain by adding triples representing the *content* of the two courses you captured in Part #1:

Pre-processing: Using a suitable library,[1] convert the material you collected (slides, web pages, lecture notes, lab documents, etc.) from their source format (e.g., PDF, DOCX) into plain text files that you can further analyse through natural language processing using spaCy.

Entity linking: Your task here is to link entities appearing in the texts to a knowledge base, like DBpedia or Wikidata (or both). For example, the topic *"Chatbot"* appearing on a slide would be linked to http://dbpedia.org/resource/Chatbot. You can use any of the tools mentioned in the lecture or lab to detect and link your entities, such as *DBpedia Spotlight*[2] or *spaCy fishing*.[3]

You must develop a suitable filtering strategy to post-process the results from these entity linkers, so that only named entities are linked (e.g., using POS tags or other linguistic features computed with spaCy).

Triplification: Using your RDF Schema from Part #1 of the project, encode the named entity information you extracted in form of `Topic` triples that link them to the corresponding URI of the course material (lab, lecture, tutorial, etc.) and material URIs[4] and add them to your Knowledge Base served by Fuseki.

Make sure all your URIs are properly connected: you have multiple courses, each course has multiple lectures, each lecture has multiple associated contents (slides, web pages, ...) and each content has (typically) multiple topic mentions.

Write SPARQL queries that can answer the following questions:

1. For a course c, list all covered topics t, printing out their English labels and their DBpedia/Wikidata URI, together with the course event URI (e.g., 'lab3') and resource URI (e.g., 'slides10') where they appeared. Filter out duplicates.
2. For a given topic t (DBpedia or Wikidata URI), list all courses c and their events e where the given topic t appears, along with the count of occurrences, with the results sorted by this count in descending order.

[1] For example, *Apache Tika*: https://tika.apache.org/ (there is a Python `tika` package to access a Tika server)
[2] Note that this requires you to install a local copy of *Spotlight*, as frequent requests to the public Web API will get you blocked.
[3] See https://spacy.io/universe/project/spacyfishing/
[4] Generally the `file://` URIs from Part #1, unless you set up a document server.

3. For a given topic t, list the precise course URI, course event URI and corresponding resource URI where the topic is covered (e.g., “NLP” is covered in COMP474 → Lecture 10→ Lab 10 Notes).
4. Write a SPARQL query to identify any course events or resources within a specific course c that do not have any associated topic entities. This query helps in verifying that all relevant educational materials have been adequately linked to topics in the knowledge base, ensuring completeness and providing insights into potential areas for improvement.

Roboprof Chatbot. For users to be able to interact with your bot, you need to develop a natural language interface to your knowledge base (i.e., a *grounding*-based bot). It has to be able to answer (at least) the following questions:

1. All of the queries from Part #1.
2. *“What is the <course> about?”*
   E.g, “What is COMP 474 about?”: provides the course description as answer.
3. *“Which topics are covered in <course_event>?”*
   E.g, “Which topics are covered in Lab #2 of COMP 474?”: provides the topics (in English), together with their resource URI where they can be found.
4. *“Which course events cover <Topic>?”*
   E.g., “Which course events cover Deep Learning?”: lists all courses with their events (lectures, labs) that include this topic, sorted by frequency of the topic, in descending order.

You have to implement your natural language interface using the Open Source *Rasa* chatbot framework.[5] Implement suitable dialogs that you can map to SPARQL queries and answer using your Fuseki server. You then have to translate the response triples into fluent natural language answers for the user.[6]

- Create an *intent* for each question.
- Setup suitable *actions* to access your Fuseki server using SPARQL queries.
- Develop a suitable strategy for handling out-of-vocabulary (OOV) words in a question. Consider techniques such as synonym mapping or leveraging the NLU model’s built-in capabilities to suggest the closest known entities or intents for unrecognized words.
- Develop a suitable error handling strategy when a user’s question cannot be answered from your knowledge base, e.g., providing helpful feedback or suggesting alternative queries.
- For generating a fluent natural language answer from the results of the SPARQL query, you can use static templates or integrate a pre-trained Large Language Model (LLM) obtained from HuggingFace, suitable for text generation tasks, such as T5-small or DistilGPT-2.[7]

Report. Update your report from Part #1, i.e., submit a single, combined & revised report for your complete project, with the following changes:

Report updates: Fix any mistakes (spelling, structure, etc.) from Part #1. Document major changes you applied from Part #1, such as changes to your RDF Schema or generated triples.

KB population: Add a section (ca. 1–2 pages) describing your process of creating the topic triples for your knowledge base (processing the documents, tool used for linking to LOD, filtering results from the linking tool). Explain how you create the triples for KB population. Create a table providing

[5] See https://rasa.com/docs/rasa/installation/installing-rasa-open-source

[6] When combined with a LLM, this technique is called *Retrieval Augmented Generation (RAG)*, which is important to ensure text generated by LLM is trustworthy, verifiable (through the knowledge base) and can provide content that was not available during LLM training (here, the university-related data, but in general any project, organization or company-specific data).

[7] See https://huggingface.co/docs/transformers/task_summary#natural-language-processing

statistics for the generated triples (total number of triples, number of distinct topics, number of topic instances/course).

Chatbot design: Add a section (ca. 2–3 pages, plus sample output) on how you used Rasa for implementing the natural language conversation interface. Document the intents, stories, entities, and actions created for your bot. Describe the integration process with your Fuseki server and how SPARQL queries are formed and executed based on user inputs. Detail your method for handling out-of-vocabulary (OOV) words and managing error cases. Explain your approach for translating SPARQL query results into natural language answers, highlighting the use of templates or LLM integration.

Include one concrete example from each query type (the 13 queries from Part #1 plus the 3 new ones) that showcases your chatbot's workflow: starting with (1) the initial user input, followed by (2) intent recognition and entity extraction, the (3) generated SPARQL query and (4) its output, and concluding with (5) the translation of query results into fluent natural language responses.

Deliverables. Your Moodle submission must include the following deliverables within a single `.zip` or `.tgz` archive:

Deliverables from A1: All the parts of your project as listed for Part #1 (with any modifications you might have made for Part #2). In other words, submit your complete course project, not just the changes from this part.

README. A comprehensive `readme.txt` or `readme.md` file:

- It must enumerate the contents and describe the purpose of each file in your submission.
- Clearly outline the steps to execute your code for (a) generating triples and (b) running the chatbot.
- If your instructions are incomplete and your code cannot be run you might not receive marks for your work.

Linking Code: Your new Python code for linking & filtering entities from the documents, as well as creating the Topic triples from these linked entities.

New queries & results: Make sure you also add the new topic SPARQL queries described above together with their output.

Chatbot: Your new Rasa chatbot Python code, including any associated resources (configuration files, training data, etc.). Include your trained ML model as well.

Report: The project report, as detailed above, as PDF.

Originality Form. Include an *Expectation of Originality* form[8] for each team member:

This form, attesting to the originality of your work, must be electronically signed and we need a form from each team member. Note: If your group has not changed, you can submit the same forms as for Part #1.

Submission Procedure: You must submit your code electronically on Moodle by the due date (late submission will incur a penalty, see Moodle for details).

Project Demo. We will schedule demo sessions for your project using the Moodle scheduler. The demos will be on campus and all team members must be present for the demo. Guidelines for preparing for the demo will also be posted on Moodle.

Other Guidelines. Please refer to the Project Part #1 document for the *Academic Integrity Guidelines for the Roboprof Project* as well as the *Project Contribution and Grading Policy.*

[8] Available at https://www.concordia.ca/ginacody/students/academic-services/expectation-of-originality.html