CSC2001F 2024 Data Structures Assignment 1

Instructions

Artificial Intelligence systems need to acquire general or common sense knowledge about the world in order to answer questions from users (e.g. “do Asian beetles have spots?”) or to perform basic reasoning (e.g. “what weights more, a ton of bricks or a ton of feathers?”). Large Language Models such as ChatGPT are “trained” (acquire knowledge) primarily from text on the web, which is not guaranteed to be factually correct or representative of the real world. Therefore researchers have developed datasets containing verified general knowledge which can be used to improve AI systems. One of these datasets is GenericsKB, a knowledge base of “generic” sentences about the world. Generic statements (not to be confused with generics in Java) are knowledge about kinds of things (including e.g. animals, objects, or abstract concepts) that are generally true (although there might be exceptions). The full GenericsKB contains 3.4M+ sentences expressing general truths such as "Dogs bark," and "Trees remove carbon dioxide from the atmosphere."

The goal of this assignment is to build a proof-of-concept Java program for querying the knowledge in GenericsKB, as well as to add additional knowledge or to update the knowledge base. We assume for this assignment that the data is stored in memory while the application performs multiple functions on-demand until the user exits. A more elaborate application could offer additional functionality but we focus on just the core functionality for this assignment.

Dataset

You will be working with a pre-processed subset of 100 000 statements (the file `GenericsKB.txt` is attached), derived from GeneticsKB (https://allenai.org/data/genericskb). The order of statements has been randomized. An additional file (`GenericsKB-additional.txt)`can be used to test adding additional statements to the knowledge base.

(where some items or statements might overlap with the initial knowledge base).

Each entry in the knowledge base has 3 data fields: The “term” (the thing that the statement is about, e.g. “tree”), the sentence (e.g. “Trees remove carbon dioxide from the atmosphere.") and a confidence score (between 0 and 1, where 1 represents complete confidence). The reason for the confidence score is that the dataset was constructed semi-automatically, so the statements were not all manually verified. Note that terms may consist of multiple (space-separated) words, e.g. “cellular mechanism”. The data is given in a text file with one statement per line; the 3 data fields (term, sentence, score) are separated by tabs. Therefore when reading in the data line by line, the first step is to split each line by tabs. While the original GenericsKB contains multiple statements corresponding to the same term (e.g. “Bee pollen has benefits” and “Bee pollen comes from flowers”), your program only has to be able to store a single statement per term. The main dataset only contains in single statement per term, but the additional file may have multiple statements corresponding to the same term, which should be interpreted as update operations, as explained below.

Here is a sample of what the data looks like:

```
internment	Internment is captivity	1.0
chocolate mousseChocolate mousse is mousse	1.0
chocolate syrup	Chocolate syrup is a sauce	1.0
bolete	Boletes are mycorrhizal species that grow in a mutually beneficial association with tree roots.	0.8370600342750549
bee pollen	Bee pollen comes from flowers.	1.0
```

Study the data carefully. The data loading should be relatively straight-forward and you must write your own code to read in the text file.

Application

Your application must include at least functionality to do the following:

1. Read in an initial knowledge base from a file to populate the (in memory) knowledge base
2. Allow the user to add new statements to the knowledge base. The new statements can be about items already in the knowledge base or about new items. If a new statement is about an item already in the knowledge base then the entry for that item should be updated with the new statement (i.e., update the sentence and confidence score), unless the new statement has a lower confidence score. This update functionality should be supported both through the user interface directly and through loading a knowledge base file (in which the statements are treated as updates executed in the order that they appear in the file).
3. Display information from the knowledge base: For a give item, display the statement about that item if there is one in the knowledge base. For a given item and sentence, check whether the statement is in the knowledge base or not, and return the confidence score if it is.

You may use any user interface for the application - at least a text menu is required but the interface can be graphical or GUI-based. Here is an example text-based interaction:

```
Choose an action from the menu:
1. Load a knowledge base from a file
2. Add a new statement to the knowledge base
3. Search for an item in the knowledge base by term
4. Search for a item in the knowledge base by term and sentence
5. Quit
Enter your choice: 1

Enter file name: GenericsKB.txt

Knowledge base loaded successfully.

Choose an action from the menu:
1. Load a knowledge base from a file
2. Add a new statement to the knowledge base
3. Search for an item in the knowledge base by term
4. Search for a item in the knowledge base by term and sentence
```

```
5. Quit
Enter your choice: 3

Enter the term to search: maple syrup

Statement found: Maple syrup is made from the sweet sap that is stored
in the trunk of the sugar maple. (Confidence score: 0.75)

Choose an action from the menu:
1. Load a knowledge base from a file
2. Add a new statement to the knowledge base
3. Search for an item in the knowledge base by term
4. Search for a item in the knowledge base by term and sentence
5. Quit

Enter your choice: 4

Enter the term: maple syrup
Enter the statement to search for: Maple syrup is made from the sweet
sap that is stored in the trunk of the sugar maple.

The statement was found and has a confidence score of 0.75.

Choose an action from the menu:
1. Load a knowledge base from a file
2. Add a new statement to the knowledge base
3. Search for an item in the knowledge base by term
4. Search for a item in the knowledge base by term and sentence
5. Quit

Enter your choice: 2

Enter the term: maple syrup
Enter the statement: Maple syrup is the product made when water is
evaporated out of maple sap.
Enter the confidence score: 0.777

Statement for term maple syrup has been updated.
```

Data structures

You should write **two** versions of the application: `GenericsKbArrayApp` and `GenericsKbBSTApp`. You have to use the specified data structures to store the statements, with one entry per item.

In `GenericsKbArrayApp` you must use a traditional array (a single array of objects) to store the statements. You may use a fixed-size array or try to determine the size programmatically. Do not use a LinkedList, ArrayList or other advanced data structure and do not sort the data. In this version, the functionality to add new statements to the knowledge base does not have to include adding new items, only to update the statements associated with items in the initial knowledge base.

In GenericsKbBSTApp you must use a Binary Search Tree (BST) instead of an array. Your BST implementation can be created from scratch or re-used from anywhere. You may NOT replace the BST with a different data structure and you may not use a balanced tree. The same functionality should be supported, except that the BST has to support adding additional items as well. Statement deletion does not have to be supported.

Test your applications to load 3 versions of the dataset with different sizes.

Hint: Do this assignment incrementally. First create a data structure with only items and get the related functions working one at a time. Then add in the functions related to full statements. Finally, write the code to load in the external file.

Report

Write a report that includes the following:

- What your OO design is: what classes you created, why, and how they interact (at most 1 page).
- What test values you used during testing and what the output was in each case (use output redirection or cut-and-paste or take screenshots) (at most 10 pages).
- A statement of what you included in your application(s) that constitutes creativity - how you went beyond the basic requirements of the assignment (at most 1 page). Examples of creativity include:
    - Designing a visual/multimedia interface
    - Enhancing the search functionality to include matching items containing single whole words (e.g. “cellular” matching “cellular mechanism” and “cellular respiration” but not “unicellular”) or other partial matches of items and/or sentences.
- Summary statistics from your use of git to demonstrate usage. Print out the first 10 lines and last 10 lines from "git log" , with line numbers added. You can use a Unix command such as:

```
git log | (ln=0; while read l; do echo $ln\: $l; ln=$((ln+1));
done) | (head -10; echo ...; tail -10)
```

Dev requirements

As a software developer, you are required to make appropriate use of the following tools:

- `git`, for source code management
- `javadoc`, for documentation generation
- `make`, for automation of compilation and documentation generation

Submission requirements

Submit a .tar.gz compressed archive containing:

- Makefile
- src/
    - all source code
- bin/
    - all class files
- doc/
    - javadoc output
- report.pdf

Your report must be in PDF format. Do not submit the git repository.

Marking Guidelines

Your assignment will be marked by tutors, using the following marking guide.

Artefact	*Aspect*	*Mark*
Report (24)	Appropriate design and implementation of OOP and data structures:	
	Overall OOP design	3
	Traditional Array implementation	3
	BST implementation	3
	Experimental tests (core):	
	Description of the testing protocol	3
	Test populating the knowledge base	3
	Test updating the knowledge base	3
	Test searching the knowledge base	3
	Description of creativity	3
Code (35)		
	OOP design	8
	Interactive interface	3
	Reading data from file	3
	Populating the knowledge base (array)	3
	Updating the knowledge base (array)	3
	Searching the knowledge base (array)	3
	Populating the knowledge base (BST)	3
	Updating the knowledge base (BST)	3
	Searching the knowledge base (BST)	3
	Implementation of creativity	3
Dev (11)	Git usage log	2
	Documentation (javadoc)	6
	Makefile: make and clean targets	3
Total (70)		

Additional resources for assignment

- GenericsKB.txt
- GenericsKB-additional.txt