Knowledge Graph generation from two Irish Tune Datasets with SPARQL Anything

Replication Guide

Author: Xiangyue Wei

Supervisor: Dr. Albert Merono Peuela

Student ID: 20041328

Programme of study: Computer Science UBSH3CSCS

April 5, 2023

Contents

Chapter 1

Introduction

This project generated 78,000 knowledge graphs from two Irish tune datasets: CRE and The Session. The evaluation chapter of the report has proven these outputs are clear and correct. This guide aims to help you replicate the results more quickly and easily.

The entire implementation of this project was carried out on Windows. As a result, this replication guide is written after testing on Windows. Therefore, there may be some undetected errors on other operating systems. Please notice, there may be a iteration failed when executing commands. This error is about file name too long, shown in a test in Bash. But in PowerShell on Windows, No error detected.

1.1 Guide Structure

This replication guide consists of five chapters excluding this introduction chapter: Tools, Data Sources, GitHub Repository, Execution Steps and Common Issues.

1. Tools: This chapter will indicate the system used and its version.
2. Data Source: This chapter will provide links to the two original datasets and the altered version. However please notice, this project used SPARQL Anything queried from nearly 40,000 MusicXML files.
3. GitHub Repository: This entire project has been uploaded to GitHub. The link will be included.
4. Execution: This chapter will explain all the steps in the replication process clearly and straightforwardly.

5. Troubleshooting and Common Issues: In this chapter, we will discuss the common issues that might occur during the execution process.

Chapter 2

Tools

In this project, we have chosen to use SPARQL Anything 0.8.1 to query data from MusicXML files. Since this guide will only show the process after removing the files with encoding problems, the EasyABC software will not be included.

2.1 SPARQL Anything

SPARQL Anything 0.8.1 provides a command-line interface and server. In this guide, we will be using the command-line interface to replicate knowledge graphs.

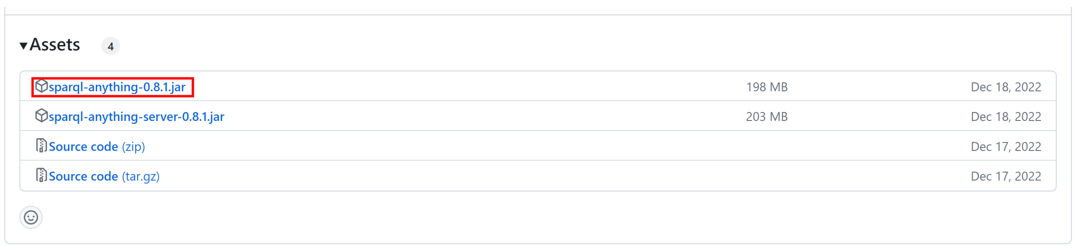
Here are the version, corresponding link, and requirements:

- Command-line interface version: 0.8.1
- Command-line interface link: `https://github.com/SPARQL-Anything/sparql.anything/releases/tag/v0.8.1`
- Requirements: SPARQL Anything requires Java $\geq$ 11 to be installed

If you have trouble copying the links from this file, you can also find a link in the README file of this project from source code submission.

Please scroll down to locate the download section.

Figure 2.1 shows the zip file that needs to be downloaded.


▾Assets 4

sparql-anything-0.8.1.jar	198 MB	Dec 18, 2022
sparql-anything-server-0.8.1.jar	203 MB	Dec 18, 2022
Source code (zip)		Dec 17, 2022
Source code (tar.gz)		Dec 17, 2022

Figure 2.1: the SPARQL Anything command-line interface needed

Chapter 3

Data Sources

This chapter provides the links to the sources of data used in this project, including both cleaned MusicXML files and original MusicXML files. As mentioned in the previous chapter, we will not need ABC Notation files in the replication process. However, to present a clear and complete view, the links to different versions of ABC Notation files will also be included. MusicXML files have been uploaded to both GitHub and Zenodo. Please notice, the MusicXML zip files will be available only on Zenodo.

3.1 ABC Notation Files

The links to ABC Notation files:

- Original ABC Notation File 1 (CRE Dataset):

  `https://github.com/polifonia-project/folk_ngram_analysis/tree/master/cre_corpus/abc`

- Original ABC Notation File 2(The Session Dataset):

  `https://github.com/polifonia-project/folk_ngram_analysis/tree/master/thesession_corpus/abc`

- Altered ABC Notation File:

  `https://github.com/LucyWei88/PRJ2022/tree/main/Altered_ABC`

3.2 MuiscXML files

The links to MusicXML files:

- Original MusicXML Files (GitHub): `https://github.com/LucyWei88/PRJ2022/tree/main/All_XML_Original`
- Cleaned MusicXML Files (GitHub): `https://github.com/LucyWei88/PRJ2022/tree/main/All_XML`
- Original MusicXML Files Zip: `https://zenodo.org/record/7792129#.ZCjWCXbMK5c`
- Cleaned MusicXML Files Zip: `https://zenodo.org/record/7792125#.ZCjWKnbMK5c`

Chapter 4

GitHub Repository

The GitHub repository contains all the components, including both necessary and unnecessary files.

The GitHub repository link of this project:

`https://github.com/LucyWei88/PRJ2022`

Chapter 5

Execution

5.1 Prepare the Directory

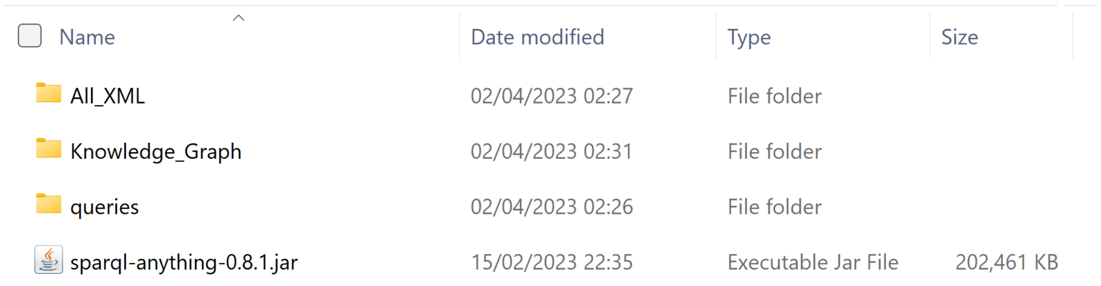
This section will guide you to prepare the directory.

The directory needs to contain the following files and directories:

- queries — a directory, copied from the source code part of the project / downloaded from GitHub
- All_XML — a directory, downloaded from GitHub / `https://zenodo.org/record/7792125#.ZCjWKnbMK5c`
- Knowledge_Graph — a directory that you need to create. This directory contains the four empty directories, which you also need to create:
  - CRE_Music_Notes
  - CRE_Tune_Information
  - The_Session_Music_Notes
  - The_Session_Tune_Information
- sparql-anything-0.8.1.jar — a jar file, downloaded from SPARQL Anything website

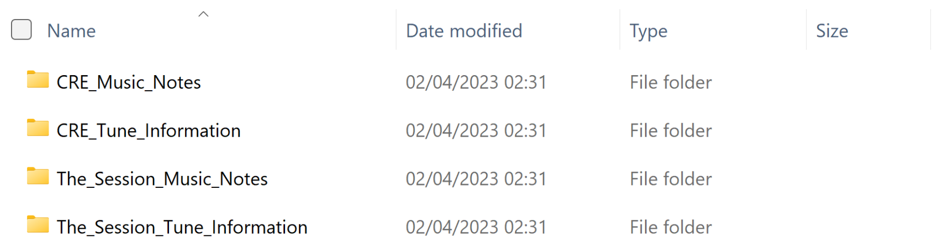
Figure 5.1 shows what the directory looks like.

Figure 5.2 shows what the directory under Knowledge_Graph looks like.



Name	Date modified	Type	Size
All_XML	02/04/2023 02:27	File folder	
Knowledge_Graph	02/04/2023 02:31	File folder	
queries	02/04/2023 02:26	File folder	
sparql-anything-0.8.1.jar	15/02/2023 22:35	Executable Jar File	202,461 KB

Figure 5.1: the directory looks like



Name	Date modified	Type	Size
CRE_Music_Notes	02/04/2023 02:31	File folder	
CRE_Tune_Information	02/04/2023 02:31	File folder	
The_Session_Music_Notes	02/04/2023 02:31	File folder	
The_Session_Tune_Information	02/04/2023 02:31	File folder	

Figure 5.2: the directory under Knowledge_Graph looks like

5.2 Explain the Commands Structure

This section explains the structure of the commands that we will use in the replication process.

Figure 5.3 is the explanation of the possible structure of commands.

Another way to learn the structure is from Command-line interface usage:

1. Open the Terminal
2. Navigate to the directory you stored the SPARQL-Anything jar file
3. Type in: java -jar sparql-anything-0.8.1.jar

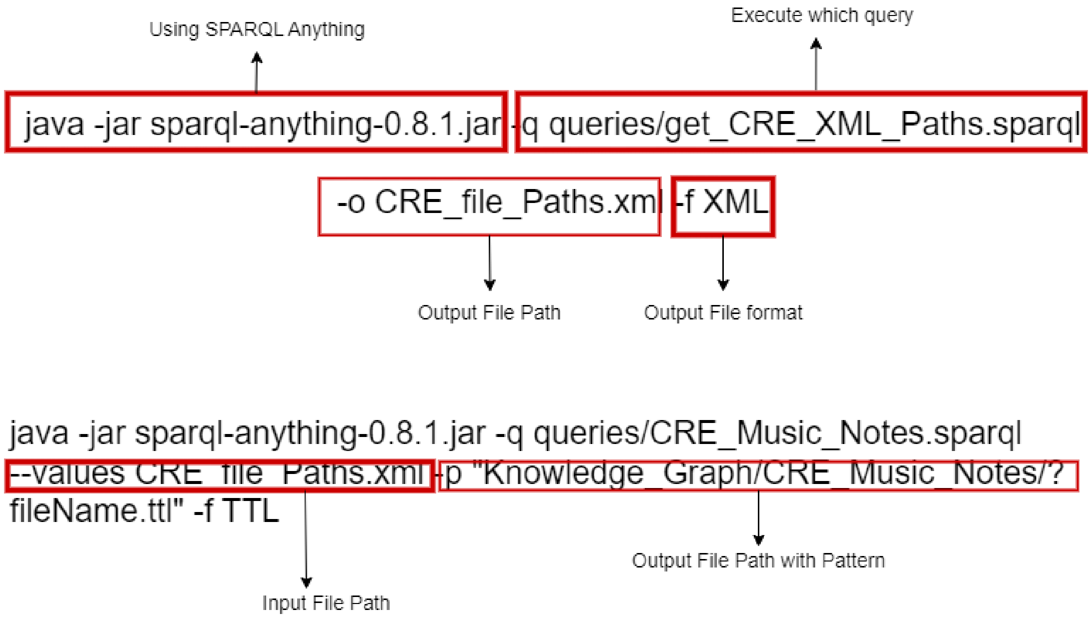
Figure 5.4 shows a part of Command-line interface usage

5.3 Start Replication

The replication consists of three parts: getting the list of MusicXML files, generating the general information from MusicXML files, and generating the music code from MusicXML files. The estimated time is based on a computer with 16GB of RAM.

5.3.1 Generate the lists of MusicXML files

This will generate two XML files. They include the paths and file names of MusicXML files.


```
java -jar sparql-anything-0.8.1.jar -q queries/get_CRE_XML_Paths.sparql
-o CRE_file_Paths.xml -f XML

java -jar sparql-anything-0.8.1.jar -q queries/CRE_Music_Notes.sparql
--values CRE_file_Paths.xml -p "Knowledge_Graph/CRE_Music_Notes/?
fileName.ttl" -f TTL
```

Figure 5.3: the explantion of possible structure of commands

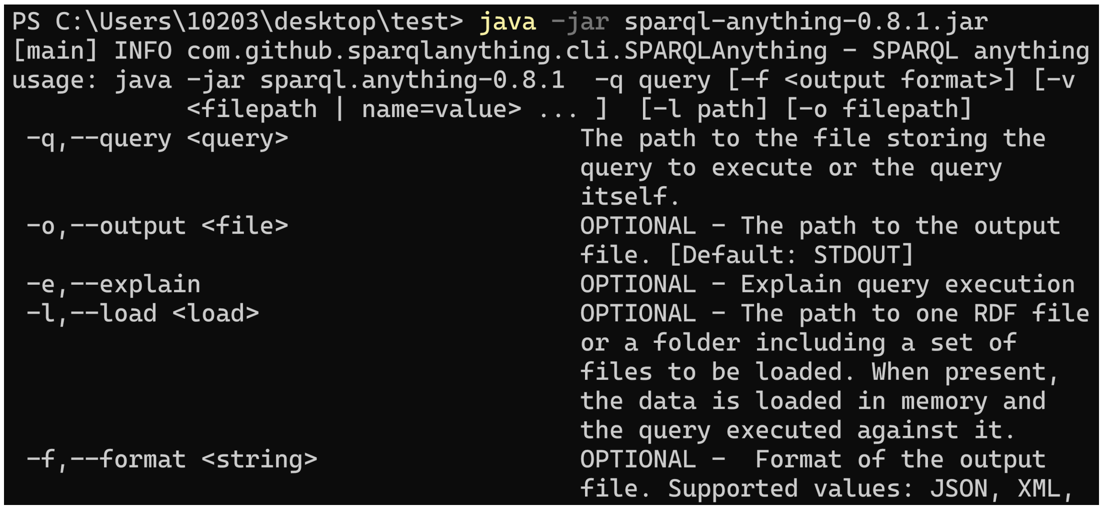
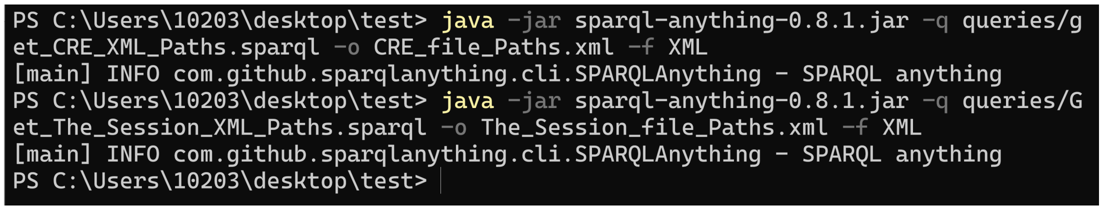
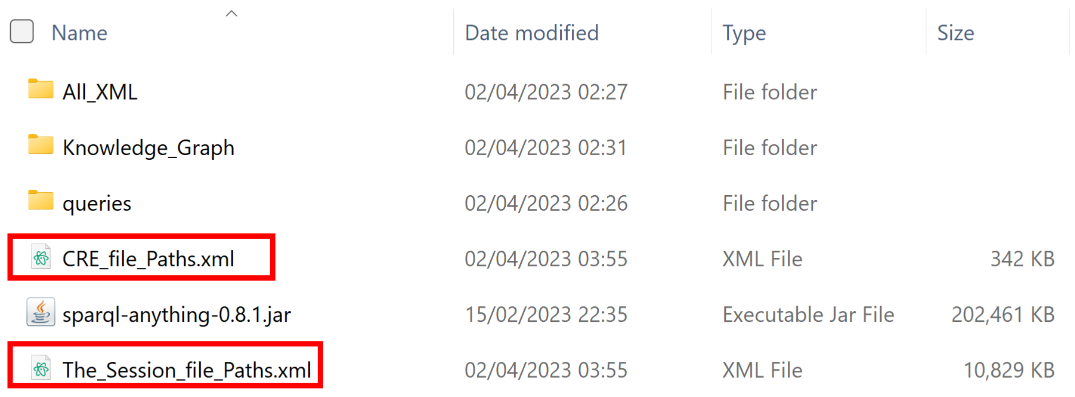

```
PS C:\Users\10203\desktop\test> java -jar sparql-anything-0.8.1.jar
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
usage: java -jar sparql.anything-0.8.1  -q query [-f <output format>] [-v
            <filepath | name=value> ... ]  [-l path] [-o filepath]
 -q,--query <query>                    The path to the file storing the
                                       query to execute or the query
                                       itself.
 -o,--output <file>                    OPTIONAL - The path to the output
                                       file. [Default: STDOUT]
 -e,--explain                          OPTIONAL - Explain query execution
 -l,--load <load>                      OPTIONAL - The path to one RDF file
                                       or a folder including a set of
                                       files to be loaded. When present,
                                       the data is loaded in memory and
                                       the query executed against it.
 -f,--format <string>                  OPTIONAL -  Format of the output
                                       file. Supported values: JSON, XML,
```

Figure 5.4: a part of Command-line interface usage



```
PS C:\Users\10203\desktop\test> java -jar sparql-anything-0.8.1.jar -q queries/g
et_CRE_XML_Paths.sparql -o CRE_file_Paths.xml -f XML
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
PS C:\Users\10203\desktop\test> java -jar sparql-anything-0.8.1.jar -q queries/G
et_The_Session_XML_Paths.sparql -o The_Session_file_Paths.xml -f XML
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
PS C:\Users\10203\desktop\test>
```

Figure 5.5: what the Terminal looks like after the execution of two paths-generation commands



Name	Date modified	Type	Size
All_XML	02/04/2023 02:27	File folder	
Knowledge_Graph	02/04/2023 02:31	File folder	
queries	02/04/2023 02:26	File folder	
CRE_file_Paths.xml	02/04/2023 03:55	XML File	342 KB
sparql-anything-0.8.1.jar	15/02/2023 22:35	Executable Jar File	202,461 KB
The_Session_file_Paths.xml	02/04/2023 03:55	XML File	10,829 KB

Figure 5.6: what the directory looks like after the execution of two paths-generation commands

Each command should not take longer than 10 seconds to execute. The commands directly copied from this PDF file might not be able to be executed in the Terminal. Please adjust the spaces or the lines.

Please follow these steps:

1. Open the Terminal
2. Navigate to the directory you prepared in section 5.1
3. Type in: java -jar sparql-anything-0.8.1.jar -q queries/Get_CRE_XML_Paths.sparql -o CRE_file_Paths.xml -f XML
4. Type in: java -jar sparql-anything-0.8.1.jar -q queries/Get_The_Session_XML_Paths.sparql -o The_Session_file_Paths.xml -f XML

After executing two commands above, two new XML files will appear in the directory.

Figure 5.5 shows what the Terminal looks like after the execution of two paths-generation commands.

Figure 5.6 shows what the directory looks like after the execution of two paths-generation commands. The new XML files are highlighted.

5.3.2 Generate the Knowledge Graphs

Please follow these steps to generate knowledge graphs:

1. Open the Terminal
2. Navigate to the directory you prepared in section 5.1
3. Type in: java -jar sparql-anything-0.8.1.jar -q queries/CRE_Tune_Information.sparql –values CRE_file_Paths.xml -p "Knowledge_Graph/CRE_Tune_Information/?fileName.ttl" -f TTL
4. Type in: java -jar sparql-anything-0.8.1.jar -q queries/The_Session_Tune_Information.sparql –values The_Session_file_Paths.xml -p "Knowledge_Graph/The_Session_Tune_Information/?fileName.ttl" -f TTL
5. Type in: java -jar sparql-anything-0.8.1.jar -q queries/CRE_Music_Notes.sparql –values CRE_file_Paths.xml -p "Knowledge_Graph/CRE_Music_Notes/?fileName.ttl" -f TTL
6. Type in: java -jar sparql-anything-0.8.1.jar -q queries/The_Session_Music_Notes.sparql –values The_Session_file_Paths.xml -p "Knowledge_Graph/The_Session_Music_Notes/?fileName.ttl" -f TTL

In Step 3, the command is to generate the knowledge graphs of general tune information from the CRE dataset. This command should not take longer than 30 seconds to execute. It will generate 1,223 knowledge graphs under CRE_Tune_Information directory.

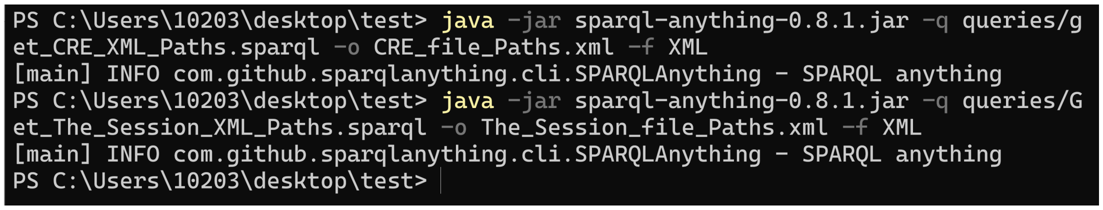
In Step 4, the command is to generate the knowledge graphs of general tune information from the Session dataset. This command should not take longer than 8 minutes to execute. It will generate 38,177 knowledge graphs under The_Session_Tune_Information directory.

In Step 5, the command is to generate the knowledge graphs of music codes from the CRE dataset. This command should not take longer than 1 minute to execute. It will generate 1,223 knowledge graphs under CRE_Music_Notes directory.

In Step 6, the command is to generate the knowledge graphs of music codes from the Session dataset. This command should not take longer than 35 minutes to execute. It will generate 38,177 knowledge graphs under The_Session_Music_Notes directory.

Figure 5.7 shows what the Terminal looks like after the execution of four knowledge-graphs-generation commands.

Here are the links to the generated knowledge graphs:



```
PS C:\Users\10203\desktop\test> java -jar sparql-anything-0.8.1.jar -q queries/g
et_CRE_XML_Paths.sparql -o CRE_file_Paths.xml -f XML
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
PS C:\Users\10203\desktop\test> java -jar sparql-anything-0.8.1.jar -q queries/G
et_The_Session_XML_Paths.sparql -o The_Session_file_Paths.xml -f XML
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
PS C:\Users\10203\desktop\test>
```

Figure 5.7: what the Terminal looks like after the execution of four knowledge-graphs-generation commands

- GitHub:

  `https://github.com/LucyWei88/PRJ2022/tree/main/Knowledge_Graph`

- Zenodo(zip):

  `https://zenodo.org/record/7792131#.ZCoouHbMK5c`

Chapter 6

Common Issues

6.1 File name too long

An issue detected when testing on MacOS using Bash. It relates to the long file namea. However, when using the same datasets and executing the same command, no errors were detected on Windows using PowerShell.

Figure 6.1 shows this error.

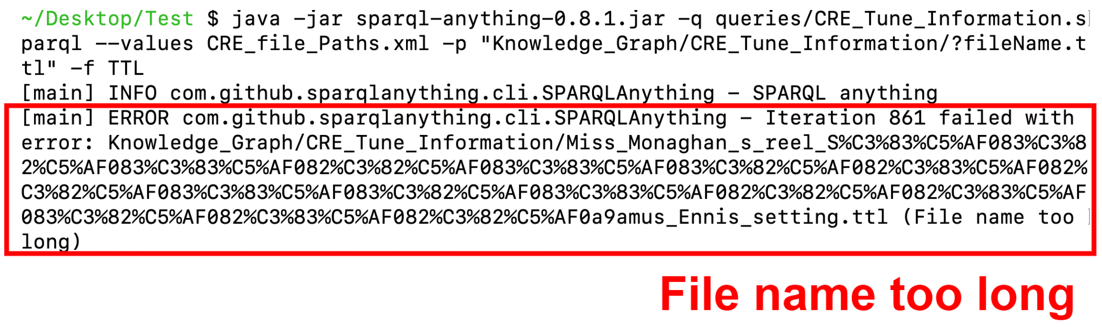
6.2 Making typing errors

It is common to make typing errors when entering the commands.

6.2.1 Wrong File names

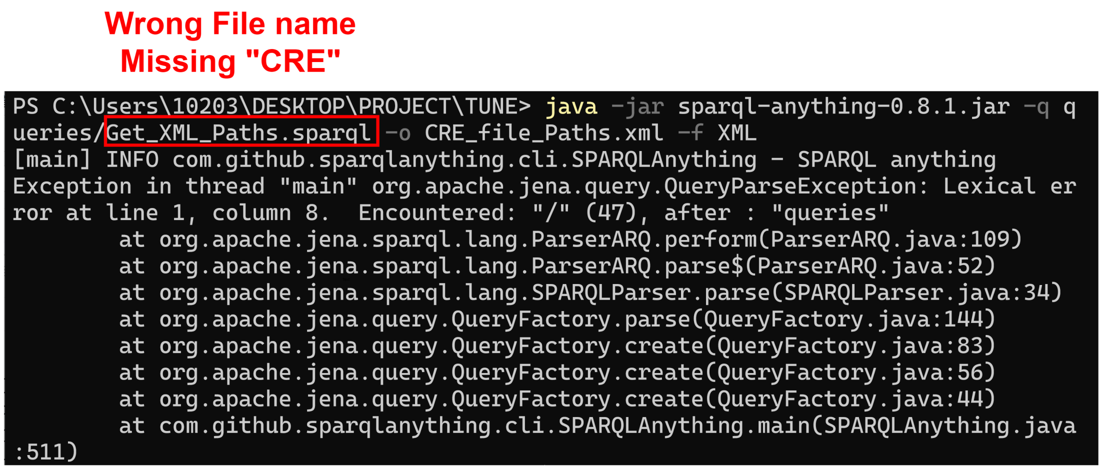
The Terminal will show an error if you enter the wrong file name.

Figure 6.2 shows this error.



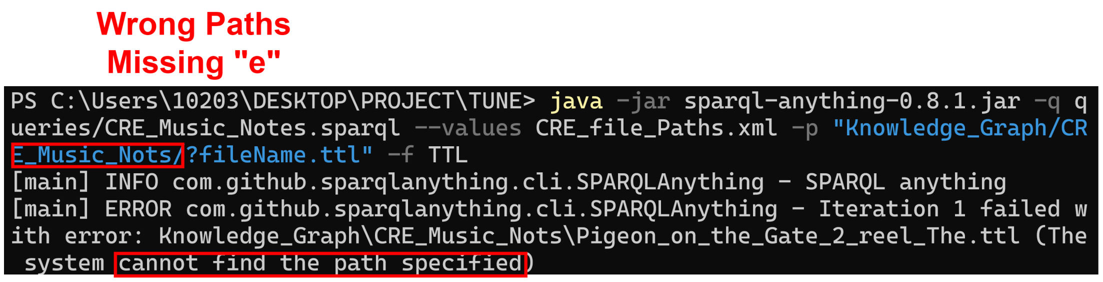
```
~/Desktop/Test $ java -jar sparql-anything-0.8.1.jar -q queries/CRE_Tune_Information.s
parql --values CRE_file_Paths.xml -p "Knowledge_Graph/CRE_Tune_Information/?fileName.t
tl" -f TTL
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
[main] ERROR com.github.sparqlanything.cli.SPARQLAnything - Iteration 861 failed with
error: Knowledge_Graph/CRE_Tune_Information/Miss_Monaghan_s_reel_S%C3%83%C5%AF083%C3%8
2%C5%AF083%C3%83%C5%AF082%C3%82%C5%AF083%C3%83%C5%AF083%C3%82%C5%AF082%C3%83%C5%AF082%
C3%82%C5%AF083%C3%83%C5%AF083%C3%82%C5%AF083%C3%83%C5%AF082%C3%82%C5%AF082%C3%83%C5%AF
083%C3%82%C5%AF082%C3%83%C5%AF082%C3%82%C5%AF0a9amus_Ennis_setting.ttl (File name too
long)
```

Figure 6.1: File-Name-Too-Long Error may happen



```
PS C:\Users\10203\DESKTOP\PROJECT\TUNE> java -jar sparql-anything-0.8.1.jar -q q
ueries/Get_XML_Paths.sparql -o CRE_file_Paths.xml -f XML
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
Exception in thread "main" org.apache.jena.query.QueryParseException: Lexical er
ror at line 1, column 8.  Encountered: "/" (47), after : "queries"
        at org.apache.jena.sparql.lang.ParserARQ.perform(ParserARQ.java:109)
        at org.apache.jena.sparql.lang.ParserARQ.parse$(ParserARQ.java:52)
        at org.apache.jena.sparql.lang.SPARQLParser.parse(SPARQLParser.java:34)
        at org.apache.jena.query.QueryFactory.parse(QueryFactory.java:144)
        at org.apache.jena.query.QueryFactory.create(QueryFactory.java:83)
        at org.apache.jena.query.QueryFactory.create(QueryFactory.java:56)
        at org.apache.jena.query.QueryFactory.create(QueryFactory.java:44)
        at com.github.sparqlanything.cli.SPARQLAnything.main(SPARQLAnything.java
:511)
```

Figure 6.2: Wrong-File-Name error



```
PS C:\Users\10203\DESKTOP\PROJECT\TUNE> java -jar sparql-anything-0.8.1.jar -q q
ueries/CRE_Music_Notes.sparql --values CRE_file_Paths.xml -p "Knowledge_Graph/CR
E_Music_Nots/?fileName.ttl" -f TTL
[main] INFO com.github.sparqlanything.cli.SPARQLAnything - SPARQL anything
[main] ERROR com.github.sparqlanything.cli.SPARQLAnything - Iteration 1 failed w
ith error: Knowledge_Graph\CRE_Music_Nots\Pigeon_on_the_Gate_2_reel_The.ttl (The
 system cannot find the path specified)
```

Figure 6.3: Wrong-File-Path error

6.2.2 Wrong file Paths

The Terminal will show an error if you enter the wrong file path.

Figure 6.3 shows this error.