Project Progress Report

Enhance MeTA Toolkit and Metapy usability

David McGuire and Jose Cols

{dmcguire,josec5}@illinois.edu

University of Illinois, Urbana-Champaign

CS 410: Text Information Systems

November 13, 2022

Completed Tasks

Build Metapy locally

The first step, as is common with legacy codebases, is to set up and build the application locally before committing any modifications to the codebase so that other developers may follow. To accomplish this, we created *forks* of the original MeTA and Metapy repositories (Massung et al., 2016). It is crucial to note that these original repositories have many branches with varying commit counts, making it non-obvious to choose the base branch. For this project, we chose the *master* branch of the Metapy repository and a specific commit hash for the MeTA repository, which is referenced by Metapy's submodule dependency. This task entails solving the following issues:

- **Fix MeTA's ICU dependency**. ICU is a set of C/C++ libraries providing Unicode and Globalization support for software applications (The Unicode Consortium, 2022). The reference to this library was broken in the MeTA build pipeline, and there are many issues related to this problem in the original GitHub repository.
    - *Reference*: https://github.com/dmcguire81/meta/pull/1 and https://github.com/dmcguire81/metapy/pull/1
- **Build Metapy for Linux.** After dealing with the dependency issues, building Metapy in Linux depends on configuring the environment with the appropriate compilers and standard library version.
- **Build Metapy for macOS x86_64.** This phase entails updating the tooling and configurations to develop the Python Wheel for this platform in its most recent versions, similar to the process of building on Linux.

Patch existing multi-platform builds based on Travis CI

To generate Python Wheels for most platforms and architectures, the original Metapy repository uses Travis CI, a continuous integration service. This task aims to extend this configuration for newer versions of Python and address current problems that result in pipeline failures, which prevent the *metapy* package from continuously being published in PyPI. This task applies what was learned in the previous task to improve the continuous integration pipeline in charge of packaging and publishing the Python library.

- **Fix Metapy's CI build dependencies for Linux.** This step updates the shell scripts that automatically install the library's dependencies and the tooling required to compile it, including removing deprecated versions. Because this subtask is platform-specific, a similar debugging process is used for other operating systems.
    - *Reference*: https://github.com/dmcguire81/metapy/pull/2
- **Fix Metapy's CI build time-outs for macOS.** Although the overall process of fixing the build is similar across platforms, each has its unique set of issues. For instance, in macOS, changing the compilation tooling and build flags. Along with the removal of deprecated packages and updating target release versions.
    - *Reference*: https://github.com/dmcguire81/metapy/pull/4
- **Add support for all currently-maintained Python versions (3.7 - 3.11).** One of the main issues with the current Metapy distribution is the lack of support for newer versions of Python. The only Python version currently supported by MeTA and the Python Foundation is 3.7, which will be phased out soon. As a result, this subtask makes the changes required to support maintained versions up to 3.11.
    - *Reference*: https://github.com/dmcguire81/metapy/pull/5

Add support for Docker workflows to MeTA

Docker makes it significantly simpler to deliver software packages in containers. This simplification is necessary to assist users who are unfamiliar with the complexities of compiling a C++ library such as MeTA. This is especially relevant for students enrolled in CS 410 Text Information Systems, which does not require C/C++ experience.

- **Create a Dockerfile.** Although introducing Docker support is mostly used to deliver a pre-built image, a Dockerfile provides two additional significant use cases. First, it serves as a starting point for more advanced users who may choose to extend or modify it to include additional capabilities. However, it still abstracts the complexity of installing the necessary tooling and dependencies to compile MeTA. Second, it includes precise configuration instructions that users can utilize to reproduce the build in systems other than Docker.
    - *Reference*: https://github.com/dmcguire81/meta/pull/2
- **Publish a Docker image with MeTA.** Building the Dockerfile and making the resulting image public is critical for reducing the effort required to get MeTA installed and running. A MeTA container with support for *arm64* (including Apple Silicon) and *amd64* architectures is made accessible on Docker Hub under the ID *josecols/meta*. This ARM support is especially beneficial because users using MacBooks with M1 CPUs frequently have difficulty compiling the toolkit.
    - *Reference*: https://github.com/dmcguire81/meta/pull/2
- **Add automated Docker image generation.** The automated Docker image generation ensures continuous integration for the modifications anticipated for the project's second

half, such as automatically upgrading the operating system packages. This process also makes it easier for other maintainers to deliver future versions.

- *Reference*: https://github.com/dmcguire81/meta/pull/2

Pending Tasks

The first half of the project focused on solving the difficult problems related to the library's compilation and adding automation pipelines to streamline the publication process. The pending tasks for the second half, iterate on this by improving the obtained support and writing documentation and tutorials so that future users can benefit from the changes. Table 1 depicts the remaining tasks, along with the person in charge and the expected completion date.

Table 1

List of pending tasks.

Owner	Task	Completion Date
David	Improve multi-platform build by vetting packages generated by Travis CI before publishing.	November 20, 2022
Jose	Improve dockerized workflow by adding periodic OS updates and dependency checks to the Docker image build.	November 20, 2022
David	Publish new artifacts (e.g., Python wheels) for use in CS 410 Text Information Systems.	November 27, 2022
Jose	Update *pybind11* version for better Python support in newer versions.	November 27, 2022
Jose	Add Google Colab integration for Jupyter Notebooks serving as tutorials.	December 3, 2022

Challenges

The main difficulties arise from the codebase's age, which dates back to 2018. The underlying platforms have evolved, resulting in deprecation and dependency problems. Each one takes time to identify because it demands debugging and understanding the underlying cause of the problem. Furthermore, new platforms, such as Apple Silicon, have emerged and are gaining market share (International Data Corporation, 2022). As a result, they must be supported so that future users can access this toolkit. Building MeTA for ARM chips like the Apple M1 was particularly challenging because this chip did not exist at the moment when the library was created.

Moreover, automating workflows with freely and publicly available resources requires efficient setups to avoid incurring computational costs. This is demonstrated by the GitHub action that builds and publishes the MeTA Docker image. With small computing instances, GitHub offers 2000 free minutes per month. However, each compilation can take several hours. As a result, using pre-built libraries and tooling is preferable, though it may be less reliable.

Similarly, building for multiple platforms on Travis CI, which requires parallel builds, can quickly deplete free credits and entail a paid subscription unless the project is documented as open-source software. Given the current maintenance barrier, adding a build that incurs costs will most likely negate the purpose of these changes. Some tradeoffs are thus required to ensure continuous integration.

References

Sean Massung, Chase Geigle, and ChengXiang Zhai. 2016. MeTA: A Unified Toolkit for Text Retrieval and Analysis. In *Proceedings of ACL-2016 System Demonstrations*, pages 91–96, Berlin, Germany. Association for Computational Linguistics.

International Data Corporation. (2022, October 9). *Worldwide PC Shipments Decline Another 15.0% in the Third Quarter of 2022, According to IDC Tracker*. IDC. Retrieved November 13, 2022, from https://www.idc.com/getdoc.jsp?containerId=prUS49755822

The Unicode Consortium. (2022). *International Components for Unicode: ICU*. International Components for Unicode: ICU. Retrieved November 13, 2022, from https://icu.unicode.org/home