Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

Text Studio: A Text Processing Architecture to Broaden Access to Natural Language Processing

Author

Tev’n Powers

Thesis Committee

Cecilia Aragon, Department of Human Centered Design & Engineering (HCDE)

Emily Bender, Department of Linguistics

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

Problem Statement

The interdisciplinary field of computational linguistics (CL) and natural language processing (NLP) sits at a unique intersection of two primary domains of knowledge:

1. Understanding of natural languages
2. Proficiency with computational methods (statistics, machine learning, etc.) which can be applied to natural languages

If these domains alone were indeed the entrance criteria for contributing to the CL community or being an active developer of NLP systems, then the field could likely be much larger and richer than it is today. However, in practice, there is an additional domain, that without proficiency, many potential builders and/or beneficiaries of NLP systems are effectively limited in their access to the community. This is the skill of computer programming and more broadly software engineering. In a recent survey that I conducted of 57 individuals, 42 respondents were NLP practitioners of varying skill levels, while 15 respondents can be characterized as hopeful or future practitioners. Of those 15 respondents, 10 individuals noted that a primary blocker to them applying relevant computational methods to their text corpora is a lack of software engineering skill. As I later review popular tools and frameworks which aim to widen the audience of practitioners in the field, we’ll see that a lack of computer programming skill is a major barrier to entry, but in many cases additional software engineering expertise is also an explicit requirement (e.g. setting up servers, installing packages from source code, etc.).

Even for experienced NLP practitioners the current developer experience is inefficient in many ways. For starters, collaboration amongst engineers can be difficult. Out of the 42 respondents that identified as NLP practitioners in the conducted survey, nearly 74% of them noted that they were surprised when the setup required for using another engineer’s NLP source code takes less than an hour. Some of the noted reasons from the survey included a lack of documentation which leads to difficulties installing and ultimately integrating the source code into their work.

The tight coupling of software engineering with language understanding and knowledge of computational methods leads to two opportunities for improvement in the field. First, the barrier to entry for newcomers to CL and NLP requires three unique skill sets. Secondly, the experience for current developers building NLP systems suffers from many engineers not being able to effectively collaborate with or reproduce the work of others.

In my experience as a computational linguistics student, engineer in industry—four years as an Applied Data Scientist on an NLP team in Microsoft Office and nearly two years on NLP teams at Textio—and having read how many companies handle scaling data processing across a variety of worker skill sets and audience contexts, I believe that it can be valuable when building NLP systems to decouple the system architecture design (e.g. feature engineering and model selection) from the programming of the modules/models that are composed to form said system.

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

My hypothesis is that creating Text Studio will promote a richer, more diverse, more collaborative CL/NLP community, that ultimately leads to more people using the valuable tools built in our community to solve their real world problems. First, Text Studio is an application that complements the development workflow of current NLP practitioners, allowing them to collaborate more easily and produce their work more efficiently. Secondly, Text Studio exposes the text processing capabilities developed by practitioners in a way that enables users to then build NLP systems, using one or more of the modules produced by engineers, to apply to their own domains and areas of expertise.

I’d like to explore this hypothesis in my master’s thesis by building Text Studio, the desktop application that enables this decoupling. I have yet to come across a modern and user-friendly open source software application whose core goal is to formalize, support and encourage a workflow that is mutually beneficial for code experts and non-experts alike. In the next section, I review the current flavors of tooling that exist to serve the NLP developer community and increase developer efficiency.

I believe that this approach to building a collaborative NLP community will not only broaden access to new builders of language processing systems, but close the gap between builders and potential consumers of these systems. Most NLP work is done by researchers in academia or in the tech industry, software engineers or data scientists in the industry, or open source software developers. Each of these practitioners has a primary audience that their work aims to serve or advance. For researchers, typically the code they write is to be shared with other researchers in their field, with a goal of making some advancement in their domain. Those who write software at a tech company are building NLP tools specific to the needs of a product that is to be sold or marketed to a particular set of consumers. And lastly, most open source development in NLP either shares a similar interest to the two previous groups or is aiming to create tools that make development more efficient for other developers. If Text Studio can more effectively enable users to discover, adapt, and execute text processing systems, I believe that anyone who has access to text data of their own can leverage the valuable technology built by the NLP community by applying these techniques to their own corpora.

Tev'n Powers
March 1, 2020
CLMS Thesis Proposal

Summary of Approaches

Frameworks & Developer Environments for NLP

GATE: General Architecture for Text Engineering

GATE, the most similar system to my proposed application, was developed to help alleviate the problem of slow takeup of reusable data and algorithmic resources for Natural Language Engineering. This robust desktop application provides a common infrastructure for building Language Engineering (LE) systems, managing Language Resources (LR), and managing Processing Resources (PR). GATE actually offers a line of products for various audiences and use cases. However, with this robustness and comprehensiveness, comes a notoriously complex and steep learning curve for getting started. The GATE developer environment provides aids for constructing, testing and evaluating LE systems. There are three principal elements in the architecture:

1. GATE Document Manager (GDM): a database for storing information about texts and data schema.
2. GATE Graphical Interface (GGI): GUI for launching processing tools on data and viewing and evaluating the results(software lego with LE components).
3. Collection of Reusable Objects for Language Engineering (CREOLE): Collection of wrappers for algorithmic and data resources. These modules handle the heavy lifting.

Module interchangeability or substitution of components is as simple as point-and-click in UI.

GATE's mission and vision is very much aligned with what I hope to accomplish with Text Studio, so it has been worthwhile to closely study their architecture philosophy and identify in which ways I believe it can be improved to provide a better user experience.

WEKA: Waikato Environment for Knowledge Analysis

WEKA describes itself as a unified workbench that would allow researchers easy access to state-of-the-art techniques in machine learning. While it does not have a specific focus on ML for NLP, I find some of its approach and motivation to be comparable to my thesis. At its core it is a collection of machine learning and data preprocessing tools. By providing a toolbox of learning algorithms, paired with a framework that allows researchers to implement new algorithms without being concerned about the infrastructure for data manipulation and scheme evaluation, WEKA has become considered a landmark system for data mining and machine learning.

Data Pipeline Simplification

University of Illinois: NLP Curator

This project from University of Illinois can be cleanly divided into two components. Curator, the processing layer, is an NLP management framework designed to address some common problems and inefficiencies associated with building NLP pipelines. It is a client/server infrastructure that provides a suite of annotators for text processing. Users can also create their own annotators by writing a wrapper or handler for the NLP component of their choice and spinning up a service using some configurations specified in the application's documentation. Secondly, Edison, the data layer of this tool, is the NLP data structure library in Java that provides streamlined interactions with Curator annotators. The goal for Edison is to be a Java library for representing and manipulating various NLP annotations on text data and enabling the quick development of NLP applications.

Collaboration Systems

UIMA: Unstructured Information Management Architecture

IBM's UIMA is middleware architecture for processing unstructured information. It started out to improve IBM Research organizations' ability to discover each other's work, rapidly combine different technologies (with a focus on NLP) and approaches to accelerate scientific advancement. Some of the challenges that the creators of UIMA wanted to address include jointly developing and reusing technology across small geographically dispersed teams, satisfying all audience consumers of their work, reducing inefficiencies of developing against different interfaces and technologies, and transferring lab work to production level software. Similar to GATE, UIMA has a notoriously steep and complex learning curve in order to get started. However, it's two-fold high level objectives are rather straightforward:

1. Accelerate scientific advances by enabling rapid combination of UIM technologies (e.g. nlp, audio and video analysis, information retrieval, etc.)
2. Accelerate transfer of technologies to product by providing an architecture that promotes reuse of tools and supports flexible deployment options.

Project Jupyter

Project Jupyter was borne out of the idea that in order for data, and the computations that process and visualize that data, to be useful to humans, they must be embedded into a computational narrative that tells the story for a particular audience in context. This project offers a set of open-source software tools for interactive and exploratory computing, primarily Jupyter Notebooks: "a web-based interactive computing platform that allows users to author computational narratives that combine live code, equations, narrative text, interactive UI, and other rich media". The high-level and core solution that the team wishes to provide is the collaborative creation of reproducible computational narratives that can be used across a range of audiences and contexts. The three fundamental problems that Jupyter Notebooks aim to address are developing computational narratives that:

1. Span a variety of audiences and contexts (i.e. highly technical paper in an academic journal vs. talk to researcher/non-technical folks vs. non-coding lab scientists who need to perform the same statistical analyses)
2. Are reproducible
3. Are created in collaboration

This collection of approaches is not meant to be exhaustive. For the purposes of this proposal I decided to include only those tools and frameworks which had formal papers in NLP or ML journals and conference proceedings. There are a number of other tools and frameworks from which I will also draw inspiration and look to for comparison to the application I’m building. For all of the coding libraries and even tools for sharing workflows like Project Jupyter, I view my work as complementary to their offerings and it’d be wise to leave room for future iterations of my application to integrate them.

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

Thesis Approach: Text Studio

During my thesis I will develop a desktop text processing application and an accompanying software development kit (SDK). The SDK will enable experienced programmers to write plugins which extend the core capabilities of the application, including the following:

- Data Loaders: data loaders can load data from arbitrary sources into a canonical Text Studio dataset for processing within the application
- Data Annotators: annotators are modules which process input text data and either transforms the input data (e.g. tokenization or part-of-speech tagging) or applies a label to the input data (in the case of a statistical or machine learning model)
- Data Visualizers: visualizers provide visual insight into phenomena in a user’s data

Text Studio will support the various phases of text processing using the notion of the extract, transform, load (ETL) procedure that is common in data processing work. Users can provide text data which will be extracted from their data source(s) into the application. Once the data is in the application dataset format, it can be annotated and transformed by executing any of the available text processing or modeling plugins. And lastly, the transformed text data can be visualized, saved, or exported to a data location for use outside of the application.

The SDK is the key to collaboration and in fact the backbone of the application which will enable users without software engineering backgrounds to participate in building and/or using NLP systems. Programmers with varying levels of experience can use the SDK to write plugins for text processing that adhere to a standardized data schema. These plugins, which extend the functionality of the core application, could range in complexity, as described above. Once written, a user can publish their code to Text Studio’s plugin library, which would then give other users access to using these modules in their own projects. Because there is a standardized data schema that each module must adhere to, users can compose various modules to create a full text processing system (e.g. tokenization from user A, part of speech tagging from user B, a classification module from user C).

Some use cases that have come up in survey data from potential users, who do not currently work in NLP/CL, include summarization of student evaluations from their classes and teachers, identifying trends in vocabulary in social media over time, sentiment analysis of social media text, extracting key doctor notes for input into medical systems. Other use cases that were not explicitly brought up in the survey data, but that I fully intend to enable include language resource management and viewing, data visualization, and evaluation of model performance among others. A non-software challenge for building this application that I will not overlook is how to educate users on the limitations of the computational techniques they may be using, the ethical implication of using machines to speed up work that would otherwise be done by humans, and in general using these tools responsibly.

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

The goal of my thesis approach is to develop an application that expresses a framework for how text processing modules can seamlessly interact with one another and gives users access to extend this framework for their own needs and the needs of other users. The goal of the application is not to solve a specific research goal, but instead provide the framework that enables the community to provide implementations for common NLP tasks, which are then accessible to non-experts in the community to apply to text in their domains.

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

Evaluation Plan

I am approaching this thesis as a human centered design and human-computer interaction (HCI) study. This framing informs my thoughts on how to best evaluate the application that I am building. For my thesis I’m primarily concerned with workflow efficiency for NLP practitioners and measuring if and how much my application can help users more quickly build and iterate on NLP systems. I began pursuing this work by surveying potential users, some of which identified as members of the NLP community and some of which are not. Before building the interface for the application, I’ll also gather feedback on paper prototypes and low fidelity prototypes of interface designs. In order to host these feedback sessions and user experience studies, I’ve received IRB exempt status for “STUDY00009011: Human Centered NLP Study”.

During the development process for Text Studio, I’d like to provide a comparative measure of creating NLP systems of varying levels of complexity by inviting users to create identical NLP systems using a) source code in the programming language of their choice and b) using Text Studio (which in some cases will involve writing code if a needed module doesn’t already exist). This comparison will help me gauge the learning curve of my platform by measuring how quickly users can publish, discover, and compose text processing modules into a working NLP system versus writing the same system themselves from scratch or using the source code of others.

Secondly, I’d like to measure the run time processing performance of an NLP system built in my application compared to executing the source code of a comparable NLP system written outside of my application. I will measure execution time of individual modules and entire systems. Part of this comparison could have significant implications on future designs of the application. For simplicity and time constraints while working on my thesis, I’m expecting the application to execute the processing modules locally on a user’s computer. But I can imagine a more performant option for future iterations of this project allowing code to be executed using cloud computers. This version of the application introduces additional complexity as well as cost implications and limitations that would have to be weighed.

Tev’n Powers
March 1, 2020
CLMS Thesis Proposal

List of Major Papers

Bird, S., & Loper, E. (2004, July). NLTK: the natural language toolkit. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions (p. 31). Association for Computational Linguistics.

Clarke, J., Srikumar, V., Sammons, M., & Roth, D. (2012). An NLP Curator (or: How I Learned to Stop Worrying and Love NLP Pipelines). In LREC(pp. 3276-3283).

Cunningham, H. (2002). GATE, a general architecture for text engineering. Computers and the Humanities, 36(2), 223-254.

Ferrucci, D., & Lally, A. (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering,10(3-4), 327-348.

Hall, M., Frank,E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.

Loper, E., & Bird, S. (2002, July). NLTK: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1 (pp. 63-70). Association for Computational Linguistics.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of the 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,...& Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12 (Oct), 2825-2830.

Perez, F., & Granger, B. E. (2015). Project Jupyter: Computational narratives as the engine of collaborative data science. Retrieved September, 11, 207.