Large scale clinical text processing and process optimization

Olga V. Patterson, PhD\textsuperscript{a,b}, Thomas Ginter, BS\textsuperscript{a,b}, Scott L. DuVall, PhD\textsuperscript{a,b}

\textsuperscript{a} VA Salt Lake City Health Care System, Salt Lake City, UT, USA
\textsuperscript{b} University of Utah, Salt Lake City, UT, USA

Abstract and Objective

This tutorial outlines the benefits and challenges of processing large volumes of clinical text with natural language processing (NLP). As NLP becomes more available and is able to tackle more complex problems, the ability to scale to millions of clinical notes must be considered. The Department of Veterans Affairs (VA) has more than 2 billion clinical notes has developed NLP libraries to be able to approach projects of that scale. Participants will be introduced to existing tools and resources for large-scale NLP tasks, including Unstructured Information Management Architecture Asynchronous Scaleout (UIMA AS), the VA Leo NLP libraries, and the JMX Analysis Module (JAM) monitoring tool. The methods of computational performance analysis will be described and process optimization solutions will be demonstrated. Participants will be walked through a scenario of creating and launching an asynchronous NLP pipeline, monitoring it for performance metrics and identifying bottlenecks, and redeploying the pipeline with an optimal configuration. The tutorial will be presented by two instructors involved in the development and use of Leo and JAM in the VA.

Keywords:

Natural Language Processing; Computing Methodologies; Automatic Data Processing.

General topics

The portions of a clinical record that are narrative text contain rich detail. This text can be stored in a variety of forms such as databases and unformatted text files. With the adoption of electronic medical records continually increasing, the amount of patient information available for research, surveillance, and reporting purposes also grows. With time, longitudinal records are accumulated for individual patients encasing an incredible amount of information.

Natural language processing (NLP) can be used to identify and use the information only available in the text records created by care providers. The simplest and most straightforward way of processing narrative text is using single-threaded, sequential processing, which limits the throughput of NLP tasks to the capabilities of the specific machine. Processing large datasets in this fashion requires an enormous amount of time and often is not feasible. A clinical environment presents additional challenges due to data accessibility restrictions. Difficulties in large-scale NLP projects arise at all steps: 1) accessing the data; 2) processing; and 3) capturing and storing results to a database or other structured format. Small datasets can be easily loaded and processed on a single computer. A large dataset has to be accessed from a database and processed in batches.

Flexible load scalability is needed in order to process large amounts of clinical data in a timely manner. System scalability is the ability of a system to handle an increased amount of data without linear increase of processing time. Scalability can be achieved through a so-called “scale out” approach, which is based on a distributed architecture that facilitates parallel processing.

There are two ways to perform parallel batch processing. One approach is to access the data in batches, and send the batches to be processed sequentially, each on a single node, such that the output is produced independently between the batches. A challenge arises when processed data has to be aggregated across the full dataset. The second approach is to use multiple nodes to perform processing, but use a single node to keep track of the input and output components.

One tool that enables scalable text processing is Unstructured Information Management Architecture Asynchronous Scaleout (UIMA AS). [1] UIMA provides capability to create, configure and deploy language-processing pipelines. Additionally, UIMA AS provides a mechanism for asynchronous communication between components that is seamless to the user. This powerful functionality allows deploying processing services remotely either in a different process space on the same machine or over multiple machines within the same network.

UIMA AS functionality relies on three main components: a service, a client and a queue broker. A service specifies the modules and processing pathway included in a pipeline. A client specifies input and output components. Services and clients define the specifics of data processing, but queue brokers are the main enablers of distributed processing. Different queuing mechanisms are used for co-located and remote components. A UIMA AS pipeline can be deployed asynchronously or synchronously. Asynchronous deployment is helpful when one component of the pipeline is much slower than the others – such as a concept mapper. Asynchronous deployment provides each component in the pipeline with a queue and various components of a single instance of a service can process multiple documents in parallel. Synchronous deployment treats each NLP service in the same way that the single-threaded, sequential processing is done, except it creates multiple instances, each with a queue, and coordinates them in parallel.

Memory and processing limitations of a single computer require large datasets to be processed in batches. The batches can be as small as a single document or may contain thousands of documents, depending on the source of data, speed of access and transmission of the data, size of the full dataset, and
average size of each document. In order to determine an optimal batch size, several iterations of test runs of the same service at different batch sizes is required. If the documents are stored in a remote relational database, a larger batch size might be desired because of the relatively slow query transaction and data transmission. Asynchronous processing allows specifying different batch sizes for input and output.

The VA Leo NLP libraries are a set of services and libraries that facilitate the rapid creation and deployment of UIMA AS annotators. Leo was developed at the Department of Veterans Affairs (VA) to enable research groups without enterprise programming experience to take advantage of the scalability provided by UIMA AS [2]. Leo eliminates the maintenance required by UIMA descriptor files and generates pipelines based on configurable parameters and type definitions provided programmatically. By providing a solid infrastructure, Leo changes the focus of NLP developers from system architecture to algorithm development.

The UIMA AS underpinning allows Leo to manage the scale needed for parallel text processing. With its developer utilities, functionality can be added and seamlessly integrated with existing NLP services — maximizing reusability of previously developed pipelines. The main advantages of using Leo for large-scale deployment are: a) ability to launch services and clients in UIMA AS programmatically; b) automatic deployment descriptor generation; c) plug and play annotators; and d) simplified NLP algorithm and pipeline development.

UIMA AS exposes numerous performance parameters via Java Management Extensions (JMX), a standard part of the Java platform, which can be queried to determine component activity. We developed the JMX Analysis Module (JAM) monitoring tool to take advantage of these JMX hooks and remotely monitor performance of individual UIMA AS components and provide aggregate measures for pipelines [3]. While JAM can be used with UIMA AS components directly, it is also integrated with Leo by providing functionality to register a service with JAM during run-time. Performance visualizations provided by JAM allow different components to be compared at a glance, which is needed to identify bottlenecks in complex deployments. Users can view statistics for single runs or runs across time, identify bottlenecks, and determine optimal system configuration through a set of web reports.

Any system optimization process is iterative. The scale out capability of UIMA AS allows replicating components and full pipeline during run-time. Monitoring running pipelines with JAM and reconfiguring using Leo simplifies the optimization process.

This tutorial introduces the challenges of large data processing and describes a set of tools and approaches that can be employed for clinical text processing.

**Tutorial structure**

**Clinical NLP overview**

Participants will be introduced to the goals, tasks, and methods of text processing. Characteristics that distinguish clinical text from general language and complicate text processing will be outlined.

**Challenges of large scale NLP**

The time required to perform different NLP tasks will be presented. The logistics of storing, transporting, and processing will be discussed. The challenges of processing large amounts of data will be described.

**Scalable design**

Participants will be introduced to scalable design terminology, characteristics, and variety of approaches. The differences between synchronous and asynchronous processing will be described.

**Overview of tools**

Participants will have the ability to use several open-source tools as they are introduced and demonstrated. UIMA AS will be introduced along with a description of how it facilitates the creation of NLP pipelines. Jargen specific to UIMA will be presented and explained. Synchronous versus asynchronous processing with UIMA will be explained. The steps of configuring and deploying an asynchronous pipeline in UIMA will be discussed. Leo will then be introduced to demonstrate the convenience and efficiencies of configuring and launching pipelines programmatically. Leo capabilities will be demonstrated using examples. JAM monitoring capabilities and performance metrics will be described and the JAM user interface and reporting service will be demonstrated.

**Scalable deployment**

Participants will be walked through the steps of deploying an asynchronous pipeline. The pipeline will be a representative example of a document classification task using several analytical components. Defining the input and output of the system along with how the modules interact will be demonstrated. Pipeline configuration and different deployment approaches will be presented.

**System monitoring**

Participants will be walked through the process of monitoring and analyzing performance characteristics of a deployed parallelized pipeline with JAM. The specific performance measures will be explained and discussed. Participants will be tasked to identify specific components that act as processing bottlenecks.

**System reconfiguration and optimization**

Participants will be walked through the steps of performance optimization using component and pipeline replication and alternate deployment approaches. Performance improvement will be measured to ensure bottlenecks are resolved. The iterative nature of performance optimization will be demonstrated.

**Specific educational goals**

At the end of the tutorial, participants will have:

- Understanding of the challenges of large-scale NLP
- Understanding of distributed processing architecture
- Ability to deploy a data processing pipeline in a distributed environment
- Familiarity with tools need for process monitoring
- Skills to recognize processing bottlenecks
• Understanding of approaches for performance optimization

Expected attendees

This tutorial is intended for informaticians, application programmers in clinical settings, and researchers with an interest in implementing NLP tools for processing of large datasets. Familiarity with high performance and distributed computing systems is helpful, but not required. Familiarity with Java programming language is helpful.

Tutorial teacher

This tutorial will be taught by three instructors experienced as researchers, developers, and users of NLP tools for large-scale clinical projects. This tutorial is geared towards hands-on learning and the instructors will interact with the attendees to more effectively facilitate the learning experience.

Olga V. Patterson is a Research Associate with the VA Salt Lake City Health Care System in Salt Lake City, Utah. She received her PhD in the Department of Biomedical Informatics, University of Utah. Her research experience includes information extraction and document classification of clinical text.

Thomas Ginter is the NLP Programming Project Lead in the Department of Epidemiology at the University of Utah School of Medicine. Thomas has a Bachelors Degree in Computer Science from Brigham Young University and over 15 years of software development and architecture experience.

Scott L. DuVall is the Director of the VA Informatics and Computing Infrastructure in the VA Salt Lake City Health Care System in Salt Lake City, Utah responsible for development and application of natural language processing and annotation systems. He leads the VINCI Services team which has completed more than 100 NLP tasks for the VA and other healthcare institutions. He received his PhD in Biomedical Informatics from the University of Utah and is a Research Assistant Professor with a joint appointment in the University of Utah School of Medicine and the College of Pharmacy.

References