NLP-PIER: A Scalable Natural Language Processing, Indexing, and Searching Architecture for Clinical Notes

Reed McEwan, MS, MSSE\textsuperscript{1}, Genevieve B. Melton, MD, PhD\textsuperscript{2,3}, Benjamin C. Knoll\textsuperscript{4}, Yan Wang, PhD\textsuperscript{2}, Gretchen Hultman, MPH\textsuperscript{2}, Justin L. Dale\textsuperscript{1}, Tim Meyer\textsuperscript{1}, Serguei V. Pakhomov, PhD\textsuperscript{2,4}

\textsuperscript{1}Academic Health Center-Information Systems, \textsuperscript{2}Institute for Health Informatics, \textsuperscript{3}Medical School, and \textsuperscript{4}College of Pharmacy, University of Minnesota, Minneapolis, MN

Abstract

Many design considerations must be addressed in order to provide researchers with full text and semantic search of unstructured healthcare data such as clinical notes and reports. Institutions looking at providing this functionality must also address the big data aspects of their unstructured corpora. Because these systems are complex and demand a non-trivial investment, there is an incentive to make the system capable of servicing future needs as well, further complicating the design. We present architectural best practices as lessons learned in the design and implementation NLP-PIER (Patient Information Extraction for Research), a scalable, extensible, and secure system for processing, indexing, and searching clinical notes at the University of Minnesota.

Introduction

Enabling research and discovery for Academic Health Centers and other healthcare delivery systems and associated programs like Clinical Translational Science Award programs requires leveraging large, changing, and disparate data sources via technology platforms and architectural considerations. Besides challenges posed by storing, analyzing, and integrating voluminous and variable structured data (e.g., *omics, clinical data), unlocking information embedded in unstructured data to make it computable and accessible represents an important and somewhat different challenge. Clinical unstructured data includes clinical notes and reports generated primarily by clinicians within electronic health record (EHR) systems. Natural language processing (NLP) systems and methods are often used to extract associated structured data from these documents including named entity recognition to recognize terms and map them to controlled vocabularies, understanding temporal expressions, determining negation and uncertainty, and inferring family or social history information. With these modules, an NLP “self-service” search engine is a valuable researcher tool, typically including structured data extracted from the source text. While the core functionality of free-text and semantic searches does not vary much from one system to another, exactly how a system’s components are orchestrated and deployed can make a big difference.

In this paper, we present an architectural approach, system performance, and lessons learned for the NLP-PIER (Patient Information Extraction for Research) system designed for clinical and translational science researchers which enables fast and responsive searching of clinical notes outside of the EHR within a secure and research protocol-compliant environment. Three architectural principles constrained the overall design of the system: scalability, extensibility, and security. The system is capable of processing 1 million Epic clinical notes every 80 minutes, indexing the results, and providing free-text and semantic search information retrieval functionality as a self-service web application using a familiar search engine paradigm. The architecture presented, although implemented using an identified set of technologies, is a flexible framework transferrable to other institutions implementing clinical notes search in an enterprise data warehouse (EDW) setting. Specific technology choices within the six major architectural components can vary according an institution’s needs while preserving the benefits of the framework’s design.

Methods

Environment and System Purpose

University of Minnesota’s (UMN) Clinical and Translational Research Institute (CTSI) Clinical Data Repository (CDR) represents a partnership with the University’s academic primary healthcare system, Fairview Health Services. The UMN CDR contains clinical data from Fairview’s six hospitals and two practice plans (academic and community-based) with over 115 clinics from the system’s enterprise EHR (Epic systems). The Research
Development and Support (RDS) team on behalf of CTSI performs extract, transform, and load (ETL) processes on this data for a number of downstream applications including the UMN clinical trials management system, OnCore, research requests, and institutional participation in large, multi-center research collaboratives\(^2\), for example the ACT Network\(^3\) aiming to improve patient care and increase clinical trial accrual.

Research data requests filed by an individual researcher or research team are managed by the UMN Informatics Consulting Service (ICS). With knowledge of the research process, clinical data in the CDR, and acting as a liaison for researchers, ICS analysts verify that the request conforms to institutional rules (e.g., Institutional Review Board) and then meet with the researcher to discuss/refine the request. The final request is delivered to the RDS team for fulfillment, which consists of querying the CDR and compiling data sets (including identified or de-identified as appropriate), which are then placed in a restricted access “data shelter” with tools for analysis by the researcher. This approach works well for the vast majority of structured data contained in the CDR. Requests focusing on information contained in unstructured data, like clinical notes or reports, are not well suited to the process outlined above. Performance of free-text search across tens of millions of documents using relational database engines is typically sub-optimal. Even in the best case scenario where notes are linked to structured data, data analysts are forced into laborious, subjective decisions as to whether notes should be included in a data set associated with a request. Secure, self-service searching of clinical notes by the researcher can mitigate manual effort and uncertainty.

Some groups have enabled search by giving providers and researchers a search interface to clinical notes within an EHR system\(^4\). While search functionality implemented within EHR systems has some promise, this approach may not be flexible enough for different research use cases and is dependent upon EHR vendor functionality/capabilities. This approach also does not support our institution’s CDR model similar to other EDW approaches by many institutions. Furthermore, our expectation is to implement an approach enabling enhancements over time driven by use cases, including modules and functionality to include semantic searches leveraging similarity metrics\(^5\) and other specialized functionality\(^6\). On the other hand, designing and implementing a platform outside of the EHR forces information technology professionals and informaticians to confront and balance myriad system implementation choices and challenges. Chief among these challenges is the problem of big data – not knowing which documents will be requested necessitates a design capable of efficient and scalable preprocessing of the entire CDR clinical note corpus, often amounting to tens or hundreds of millions of documents.
**System Architecture**

At a high level, the PIER system architecture is divided into six components (Figure 1): relational database repositories, an NLP processing pipeline, an interface engine, an indexing cluster, a Web application, and researcher access nodes. All of these components are deployed within the University’s high-risk researcher subnet dedicated to CTSI research. Dotted lines around each component serve as a context boundary and a representation that firewalls protect components from unauthorized network access. Except for the CDR, which is an Oracle 11g database on physical hardware, all components are deployed as virtual infrastructure. More detailed descriptions of each component are provided, as follows.

In order to leverage the database repositories for PIER, two sets of database level preprocessing are applied to the notes and note-related metadata in preparation for processing and annotation by the NLP pipeline. First, a series of ETL processes sourcing data from the CDR populates staging tables with clinical notes, their identifiers, and metadata elements used to characterize clinical notes and provide analytic dimensions. Five of these data elements are mapped to the HL7-LOINC Document Ontology: Kind of Document, Type of Service, Setting, Subject Matter Domain, and author Role. The other ten metadata elements include patient identifiers, encounter identifiers, and document author information. Daily triggers surface new notes and metadata, as well as updates to existing notes. Second, documents are then processed and grouped by time interval, typically by year, to boost the performance of the NLP pipeline collection reader client.

The NLP pipeline utilized by PIER is the open source system, BioMedICUS (BioMedical Information Collection and Understanding System), based on the Unstructured Information Management Architecture – Asynchronous Scaleout (UIMA-AS) architecture. Following low-level NLP tasks like tokenization, normalization, and chunking, BioMedICUS performs tasks to extract concepts from the Unified Medical Language System (UMLS) and determine negation context of the identified terms. Denormalized notes and metadata are ingested by a BioMedICUS collection reader client, packaged for consumption by the UIMA framework, and put on a messaging queue where they are picked up by service nodes running the BioMedICUS NLP tasks. Fully annotated notes, represented as XMI documents, are put on a return queue for additional processing.

PIER’s interface engine is a Mirth Connect instance. Its role is to take the XMI documents off the return queue, parse the document, and send the processed data to two types of endpoints: a notes repository (logically part of the CDR) for persisting note-related artifacts and the indexing cluster for use directly by PIER. The entire XMI document is persisted to the CDR notes repository to support NLP-related research activities outside of PIER’s context. Data sent to the indexing cluster includes the following elements: a) text of the note itself, b) document metadata, and c) a subset of the NLP annotations, specifically the UMLS concepts identified for each note, whether that concept is negated, begin and end offsets for each concept, and their UMLS semantic types.

A server cluster consisting of three Elasticsearch nodes indexes the notes and annotations sent as JSON documents to Elasticsearch’s RESTful API. To improve efficiency, note and annotation data is stored in yearly partitions. For example, all notes from the year 2014 are in a single index, and notes from 2015 are in a separate index. Naming of the indices follows the pattern `YYYY_notes_vN`, where `YYYY` is the year and `N` is the version of the index, incremented each time that year’s notes are re-indexed. All notes-related indices share a common schema. Annotation-related indices use the same naming convention as the notes and share a separate annotation schema. Index processing uses a snowball analyzer (grammar-based tokenizer, lowercase filter, stop filter, snowball stemming filter) to process the note text and store the data using Apache Lucene.
Figure 2 is an illustration of the index structures on which our naming schemes are based. Horizontal rows represent individual note and annotation indexes, paired by year in the range 2007-2015. Omitted for simplicity is a small percentage of notes with encounter dates spanning multiple years prior to 2007. Index pairs share a set of unique document identifiers, for example, \{a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, ... an\} for the 2007 indexes. The value of \(n\) for each year (e.g., \(a_n\)) corresponds to that year’s document count. In practice, it is not the same number from year to year. The shaded box with a solid border (\(D\)) represents an individual note whose information is split between two documents, each with the same identifier (\(c_i\)), one in the 2009 \textit{notes_v1} index and one in the 2009 \textit{annotations_v1} index. Shaded boxes with dashed borders (\(R_1, R_2, R_3, \text{and NOTES}\) represent aliases, or virtual indexes. The \(R_1-R_3\) aliases represent note sets associated with data requests. The \textit{NOTES} alias includes all individual note and annotation indexes, simplifying the process of querying the entire corpus.

The web application portion of PIER uses a fairly typical model-view-controller (MVC) web application design. This application provides the self-service user interface in which researchers perform queries against the analyzed and indexed notes stored in the indexing cluster. Grails is used as the MVC web application framework. For the most part, PIER is configured like a typical Grails application. The one deviation from the Grails playbook is that the view layer utilizes Angular-JS which interacts with RESTful Grails endpoints acting as proxies to the indexing cluster. MySQL is used as the backend database, and an Apache Web Server fronts a Tomcat 7 servlet container into which the Grails application is deployed. Use of the HTTPS protocol is required to connect a browser to the application. Role-based access control is provided by the Spring Security plugin for Grails. Users are authenticated against an LDAP directory and the application has four hierarchical authorization roles. From least to most privileged these are: researcher, data analyst, admin, superadmin. Each level in the role hierarchy has all the privileges of the lesser roles. Clinical researchers are the PIER’s target audience and are assigned the researcher security role. This role is restricted to approved note sets based off of an individual ICS request. In addition to issuing search queries and navigating paginated results, researchers can filter results by metadata values presented as search result facets, save queries for running in subsequent sessions, and export search results in CSV and JSON formats. Data analysts are allowed to access any note set in the system for the purpose of helping researchers conduct their work. The admin role also permits the ability to perform global searches and the ability to provision users, their roles and manually configure note sets in the system. Superadmins are permitted to perform application and system level operations such as creating new indices, index mappings, and index cluster monitoring operations from within PIER.

The final component of the architecture is a set of access nodes, termed data shelters. These Windows 2012 servers are available in a HIPAA-compliant environment to researchers to perform analysis on authorized PHI data. Firewalls and group policies restrict network access to a subset of the nodes within the high-risk researcher subnet. Access to the Internet is prohibited, as is taking data out of the data shelter nodes. Identified data generally remains in the shelters and exceptions are handled through a governance process. Adhering to the principle of least privilege, data shelter maintenance and configuration is performed by the RDS team, and server operating system
maintenance and configuration, including local and network firewalls, is handled by separate personnel on the infrastructure operations team. Members of the NLP-IE team implementing the analysis and search architecture are not members of the other teams.

Evaluation

We assessed metrics of the PIER system and environment at the document, patient, encounter, and date level, as well as information about the associated data footprint encompassed by the system. PIER system benchmarks around analysis and query responsiveness were obtained providing further information on system scalability.

We conducted an informal focus group session with six CTSI clinician researchers who utilize clinical notes in their studies. The purpose of the session was to gain feedback on PIER’s initial user interface as well as understand basic use cases of NLP for researchers and their requirements. Participants were given a demonstration of NLP-PIER’s search functionality, filters, and its capabilities to save and re-use queries and export results for analysis. After the demonstration, feedback regarding perceived value, functionality, usability, and ideas for improvement were solicited.

Finally, members of technical teams from CTSI, RDS, and the NLP-IE program collaboratively composed a set of “lessons learned” from the conception of PIER to its initial deployment organized along the architectural requirements of scalability, extensibility, and security.

Results

System Metrics and Performance

The distributed architecture as described has proven to be flexible and efficient at processing the notes. Table 1 summarizes aggregate metrics for the system as a whole including document count for the corpus, distinct counts for clinically pertinent metadata dimensions, and storage footprints.

Table 1. Summary metrics of the overall notes processing system, including patient and encounter counts and data footprints.

| Documents | Clinical Metrics | Storage Footprints |
|-----------|------------------|-------------------|
| Total Notes | 100 M | Distinct Patients | 2 M | Index Storage, Total | 600 GB |
| Average Note Size | ~2 kB | Encounters | 45 M | Index Storage, Notes | 240 GB |
| Annotations | 7 B | Service Dates | 4 M | Index Storage, NLP Annotations | 360 GB |

Out of the box, so to speak, with very little infrastructure performance tuning, a BioMedICUS UIMA client on one node, a message broker on another node, two BioMedICUS service instances deployed to a third node, Mirth on a fourth, and a three-node indexing cluster produces sustained throughput averaging 750K documents per hour, including persisting the BioMedICUS end product to a database repository (Notes Repo in Figure 1).

Information retrieval metrics are quite favorable. Without computing facets, query performance for searches identifying millions (between 2-20 million) notes is on the order of one second the first time a query term is encountered. Subsequent search times measure in the tens to hundreds of milliseconds. In general, completion time for faceted queries scales with the number of matching documents. Faceted queries identifying comparable numbers of documents (2-20 million) complete in 2-5 seconds. In the same way that a smaller set of matching documents improves completion time, filtering by faceted values significantly speeds up queries by reducing the document count prior to computation of facet statistics.

Even with minimal tuning, the system performance has exceeded our expectations. Early analysis indicates that the UIMA client is the gate keeper governing NLP/indexing throughput. This is an area of active investigation. Composite NLP and information retrieval indexing throughput, currently on the order of 1 million documents every 80 minutes, has room for significant improvement by tuning the client and employing more BioMedICUS service instances across several additional nodes. The current system utilizes only two instances on one node.
Focus Group Feedback

All participants expressed positive perceived value, especially when performing chart reviews. Other valuable applications of the tool included quality improvement efforts, establishing cohorts, recruitment, and screening patients for study eligibility. Usability feedback regarding the quantity of facets (too many) and the meaning of their domain value sets was applied to the tool prior to launch. Ideas for improvement include in-tool access to structured patient- and encounter-related CDR data (e.g., medications, labs, demographics, and billing). Other ideas include adding collaborative elements to the tool so that members of a research team can share saved queries as well as notes flagged as relevant to a study, the ability to group and order search results differently, for example by time when grouped by encounter or the patient itself, and the ability to export notes and metadata for analysis by selecting notes across searches instead of the current functionality allowing only the export of an entire set of search results, many of which may not be relevant to the data set used for post-search analysis.

We are planning new features based on the feedback we received. Implementing the suggested features can be accommodated through new functional and minor refactoring isolated to the NLP-PIER Web application component of the overall system. Because of the extensibility of our infrastructure, no changes to the existing architecture, processes, inter-component interfaces, or intra-component processes should be necessary.

Lessons Learned: Scalability

1. Scaling NLP note processing – handling changes in the note corpus and system requirements: An individual note, once finalized in the EHR, can be considered a static document. By extension, the notes corpus is also static at any point in time. Assuming no change over time in the algorithms used to process the notes, once a note is processed by both NLP and the indexing platform, its analysis lifecycle can be considered complete. What changes regularly over time is the daily accretive delta of new notes.

While the notes are static, the algorithms used to analyze them change and improve over time. At some point these improvements are significant enough to warrant reanalyzing the entire corpus to provide improved analytics to researchers utilizing the system. Additionally, NLP processing is the rate-limiting step in the overall processing of a note. Therefore, processing and analyzing clinical notes is a cyclical activity which surfaces concerns about the time and effort that must be expended to completely re-analyze the corpus. NLP and indexing components must be capable of scaling for performance. If they cannot scale, then choices must be made - costly refactoring of major components, bolt-on additions that compromise the integrity of the architecture, or forgoing the benefit of reanalysis and re-indexing.

BioMedICUS provides the NLP pipeline component of the reference architecture. BioMedICUS is implemented using the Unstructured Information Management Architecture (UIMA) framework and deployed in a UIMA-AS (Asynchronous Scaleout) configuration. Regarding scalability, the UIMA-AS exposes numerous client and service component configuration parameters that can be adjusted to tune the deployment for performance. Services, where most of the NLP heavy lifting is done, can be deployed in parallel to further enhance performance”. Couple this configurability with virtual infrastructure, in the cloud or on-premises, and the system becomes highly tunable for performance.

2. Scaling the search engine: Information retrieval technologies support the search engine functionality. In this component of the architecture, scalability is provided by Elasticsearch, an open source indexing framework for search. Early proof of concept (PoC) implementations using one search node yielded sub-optimal query performance using about a third of the total 100M note corpus. Horizontally scaling the search cluster to three nodes, and increasing JVM memory by a factor of two, to 12GB, dramatically improved search performance, speeding up query execution time by a factor of two to three. Available virtual infrastructure minimized the cost and effort to make these changes to the cluster. Automatically rebalancing of the data shards to the new nodes minimized the effort to achieve the new level of performance.

3. Index mappings for minimum search latency – properly associating types: One of the key features made possible by indexing notes along with their metadata is the ability to compute facets and aggregate statistics for each query. To efficiently provide these statistics, Elasticsearch uses an in-memory structure called field data to store computed facets. Statistics are computed for the entire field the first time it is requested. Subsequent statistics are derived from the in-memory representation, eliminating the initial computation cost. Early PoCs,
because they indexed a sufficiently small subset of the overall corpus, hid the upfront expense of computing field data and combing through it to provide query-specifics statistics. As the number of indexed documents grew, query execution times extended into the tens of seconds, approaching a minute in duration even for queries utilizing in-memory data. Detailed analysis exposed an unanticipated combinatorial effect caused by the strategy for relating semantic concepts extracted by the NLP pipeline to the note itself, a nested document relationship. Faceting and aggregating statistics were being computed and processed for not only the parent document (the intended target), but for each of the nested documents, one for each semantic concept associated with the parent document. The 1:70 ratio of notes to annotations resulted in field data nearly two orders of magnitude greater than for notes alone, severely and negatively impacting performance of both the initial field data calculations and subsequent in-memory processing. Index mappings were refactored such that notes and NLP annotations reside in their own indexes, resolving the resource utilization issues. Notes and annotation are associated, so to speak, at runtime by matching on a shared-value, externally-derived document identifier used in the indices. This external identifier is a concatenation of the source system name and the source system identifier, a structure intended to avoid identifier collisions between disparate source systems and has properties similar to a one-to-one primary key/foreign key relation, minus any referential integrity constraints.

**Lessons Learned: Extensibility**

1. **Patterned index naming convention:** The `YYYY_notes_vN` and `YYYY_annotations_vN` patterns for partitioning and naming the notes and annotations indexes helps to isolate the rest of the architecture from ongoing indexing operations. Its year-based partitioning serves as a descriptive organizational structure and a stable interface around which indexing and querying operations can be built. For example, time-based queries (e.g., notes containing a particular UMLS concept between March 2014 and December 2015) can be constructed to declaratively query portions of the corpus (explicitly name the “2014_notes_v*” and “2015_notes_v*” indexes; wildcards are allowed when querying indexes) based on the date range, thus reducing the number of indexes queried and improving query performance. Querying the entire corpus is accomplished by referencing a comprehensive “notes” alias, or synthetic index, defined to include all notes-based indexes. New indexes can be added to the alias, for example at the turn of a calendar year without any changes to the existing search engine web application. A new yearly partition (2015_notes_v2) can be created alongside the existing yearly partition (2015_notes_v1). A couple of simple aliasing commands replaces the old partition with the new one, requiring no changes to the search application and with zero downtime. Index aliasing is also used to provide virtual indexes mapped to individual data requests. These identifier-based aliases (see Security section next) allow for runtime discovery of authorized search contexts, again requiring no changes to the search application.

2. **Inclusion of an interface engine:** An interface engine, Mirth Connect in our architecture, can perform many functions, but its primary purpose in the healthcare industry is to route and transform HL7 messages. Therefore, it is not normally found in NLP and indexing architectures because the reading, parsing, and writing functions that it provides our architecture are already well covered in the UIMA and indexing spaces. Besides fulfilling a future need to process notes as they are delivered in real time as HL7 messages, Mirth Connect serves as a way to keep BioMedICUS development efforts and its interfaces focused on pure NLP processing tasks while providing a loose coupling to post-NLP processes. Routing BioMedICUS’s UIMA-based XMI output to a single messaging queue, regardless of final its final destinations, relieves the NLP framework from having to robustly handle all the tasks necessary for post-NLP processing by three separate endpoints across two APIs, HTTP and JDBC.

3. **Separating the search application and document analysis:** In PIER, the search application is decoupled from the analytical tools and processes responsible for NLP and indexing. This separation layers the search user interface on top of the data, allowing NLP, indexing, and search/user interface components to evolve with minimal dependencies. Choosing a different Web application framework has little, if any, bearing on analytical processes.
Lessons Learned: Security

Unlike i2b2 for cohort discovery using de-identified clinical data, or semantic search systems focused on non-PHI data, protecting PHI within PIER requires security at many levels. Indexing platforms are typically not secure by default. This posed a couple of problems.

1. Mapping requests to a subset of the notes corpus: Because the indexing cluster holds all the notes and a typical data request maps only to a subset of those, a mechanism is needed to securely limit searches to these subsets. The solution spans three different stores of data: CDR, Spring Security data structures, and Elasticsearch. As part of the ICS data request process, individual researchers and a list of note identifiers are associated with each request and persisted to tables within the CDR. Scheduled services within NLP-PIER use the request-researcher association data to populate a security database (MySQL) used by the Grails Spring Security Plugin with a set of authorized users (the union of distinct users across all requests), assigning them to the researcher role. Additional services query the request-note associations to build virtual indexes, one for each request. These virtual indexes are implemented as document identifier filters mapped to an index alias name (Figure 2). When an authorized researcher authenticates to the application, the Spring Security framework decorates the authenticated user with a list of currently active, authorized requests sourced from the from the CDR. This list is presented to the user as a search context option within the NLP-PIER user interface. Search requests packaged in the view layer utilize these contexts (aliases) in their URLs, which are routed to Grails controllers mapped to URL patterns. Services called by the controller validate the index alias portion of the request URL against the decorated request list previously stored by the security framework. Only if the currently requested index is in the list is the request allowed. In this way, if a sophisticated user were to successfully issue an authenticated request to the web application utilizing an unauthorized index alias, analogous to a SQL injection attack, the request would be blocked.

2. Limiting network access to the indexing cluster: In addition to the in-application security measures, network access is limited at multiple levels in a layered security architecture to prevent unauthorized access to the notes. Firewalls protect the index cluster by limiting connections to those originating from a developer VPN, the application server hosting the NLP-PIER web application tier, and from the interface engine component, Mirth, used in the NLP pipeline and indexing process. In this architecture the web application acts as proxy server to the indexing cluster. In the application/web layer, only a handful of service-level methods access the indexing cluster, limiting the number of vectors from the application to the index. These methods are invoked by controller actions protected by role based access control. Additional services ensure that the user has been granted access to the index alias being queried, otherwise the request is not fulfilled. View-level access to the indexing cluster, for example through asynchronous web requests (e.g., AJAX) is not employed, nor would it be allowed. Firewall rules prevent browser clients from directly accessing the indexing cluster.

Connections to the web application server are only allowed from the data shelters. Access to the data shelters requires filing and fulfillment of an access request, and upon approval, manual configuration allowing the researcher access to the data shelter. Data shelter sessions are restricted to a special VPN – only those researchers with approved and active data requests are allowed access. Data governance policies prevent data from leaving the shelter environment. Accordingly, search results exported from PIER must remain in the researcher’s workspace within the shelter.

Discussion

The findings of this study provide insights to architectural best practices for a clinical note NLP delivery platform for researchers at a large academic health system. The findings, while specific to our institution and NLP capabilities for clinical and translational science researchers, are generalizable to other self-service, customized, big data applications leveraging a data warehouse architecture. A number of important lessons learned in the high-level areas of scalability, extensibility, and security were discovered in the development and implementation of PIER.

Systems detailing clinical notes search and self-service models in an EDW setting address some of the same functionality we present here for PIER (self-service, architecture, search functionality, and NLP). Our contribution is to provide documentation of an architectural template and our specific implementation of that template. We discuss issues and solutions others may encounter when trying to design and implement a similar system of the same scale. Additionally, we believe our application of index aliases to be a differentiating component of the overall system, enabling secure access to PHI-containing clinical text while contributing to the scalability and extensibility of the overall architecture.
Systems utilizing Lucene for information retrieval are not new, and ours is not the first in the clinical notes search space. As the Lucene space evolves, in recent years bringing Solr and Elasticsearch into the system architect’s toolbox, it is natural for these frameworks to provide functionality across a range of systems focusing on information retrieval. In our case, Elasticsearch was the best overall fit for the indexing cluster in our architecture, but did require some unexpected effort. In addition to presenting the overall benefits of our architecture, we intend to provide a head start to the reader contemplating using Elasticsearch for clinical document indexing and retrieval. Elasticsearch tends to be used to handle big data where document size is small, with a simple, relatively consistent format (e.g., time series data like server logs or tweets). Tailoring its use to clinical notes took some trial and error, as we discovered issues not covered by the official documentation and numerous helpful, but not directly applicable, blog posts. Relationships between documents can be constructed in a variety of ways: parent-child relationships, nested documents, denormalization, and application-side joins. Application-side joins were initially discounted because we wanted relationships among the data to be manifested by the index, not outside of it. Parent-child index mappings were not semantically appropriate for our data. Data denormalization looked promising, but had the side effect of skewing the size of the index toward the annotations. We preferred to minimize the size of the note indexes assuming this would lead to better search performance. Nested objects most closely represented the semantic relationship between the notes and their annotations. But, as discussed in the section on scalability lessons, nested documents severely degrade the efficient use of system resources (memory) and query performance. In the end, we adopted application-side joins because they maximized system performance at the cost of a minimally more complex web application.

Minimal UIMA-AS scaling was employed – two concurrent BioMedICUS instances were deployed to the same UIMA-AS service node. Although not utilized to produce the system metrics presented in this paper, multithreaded and parallelized UIMA-AS deployments in a PHI-compliant supercomputing environment can be employed to massively scale NLP throughput with only a small increase in the deployment complexity. The increased deployment complexity can be mitigated by judicious use of server scripts and/or DevOps practices.

Current NLP-PIER limitations include a limited initial deployment. Many of the focus group recommendations have already been incorporated into NLP-PIER. Techniques for implementing their sharing-related suggestions are under development. As the number of users grows we plan to conduct a more rigorous usability evaluation to discover improvements we can make to the tool to match user expectations and workflows. Additional search functionality is planned, the goal of which is to provide users with an easy to use semantic search interface based on the mapping of corpus text to clinical concepts (e.g., UMLS concepts) and semantically related terms. Discovery and visualization of extracted family and social history information is also on the list of improvements.

Conclusion

With the increased adoption of EHR systems and desire to leverage information in text for clinical and translational research, there is a need for efforts to explore platforms to effectively handle text and to provide self-service of NLP capabilities for researchers in a scalable, flexible, and secure manner. This study involves examining the architecture of PIER, a platform for clinical researchers in a secure environment and describes the system’s metrics, lessons learned to date, and other best practices. Based on the preliminary findings, implications for designing similar big data systems include properly choosing functional components and interfacing them in ways that provide for current and future scalability in terms of the volume of data to process, and periodically reprocess, based on advances in analytical methods. Loose coupling of system components provides maximum extensibility, protecting the initial investment and providing a solid base capable of changing with minimal effort in response to future requirements. Finally, properly constructed index mappings and their relationships have a big impact on search performance and the ability, when used in conjunction with appropriate safeguards like firewalls, to easily secure data from unauthorized access.

Acknowledgements

The authors would like to acknowledge the teamwork and collaboration provided by the Fairview Health Services Data Reporting and Analytics team, UMN Academic Health Center-Information Services, CTSI ICS staff, Minnesota Supercomputing Institute, and the Natural Language Processing-Information Extraction Program, including technical assistance by Nathan Huff. This research was supported by Fairview Health Services, University of Minnesota Physicians, Agency for Healthcare Research & Quality (#R01HS022085 (GM)), and National Institutes of Health (#R01LM011364 (GM), #R01GM102282 (SP) and #8UL1TR000114 (Blazer)).
References

1. NLP-PIER [Internet]. http://athena.ahc.umn.edu/nlppier/. Accessed September 20, 2015.
2. Waitman LR, Aaronson LS, Nadkarni PM, Connolly DW, Campbell JR. The Greater Plains Collaborative: a PCORNet Clinical Research Data Network. Journal of the American Medical Informatics Association : JAMIA. 2014 Jul-Aug;21(4):637-41.
3. ACT Network [Internet]. Available from: http://act-network.org/. Accessed September 20, 2015.
4. Duke JD, Morea J, Mamlin B, Martin DK, Simonaitis L, Takesue BY, Dixon BE, Dexter PR. Regenstrief Institute's Medical Gopher: a next-generation homegrown electronic medical record system. International journal of medical informatics. 2014 Mar;83(3):170-9.
5. McInnes BT, Pedersen T, Liu Y, Melton GB, Pakhomov SV. Knowledge-based method for determining the meaning of ambiguous biomedical terms using information content measures of similarity. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2011;2011:895-904.
6. Pakhomov S, McInnes B, Adam T, Liu Y, Pedersen T, Melton GB. Semantic Similarity and Relatedness between Clinical Terms: An Experimental Study. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2010;2010:572-6.
7. Bill R, Pakhomov S, Chen ES, Winden TJ, Carter EW, Melton GB. Automated extraction of family history information from clinical notes. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2014;2014:1709-17.
8. Wang Y, Chen ES, Pakhomov S, Arsoniadis E, Carter EW, Lindemann E, Sarkar IN, Melton GB. Automated Extraction of Substance Use Information from Clinical Texts. Proceedings of the American Medical Informatics Association Symposium. 2015. Accepted.
9. Dolin RH, Alschuler L, Boyer S, Beebe C, Behlen FM, Biron PV, Shabo Shvo A. HL7 Clinical Document Architecture, Release 2. Journal of the American Medical Informatics Association : JAMIA. 2006 Jan-Feb;13(1):30-9.
10. Chen ES, Melton GB, Engelstad ME, Sarkar IN. Standardizing Clinical Document Names Using the HL7/LOINC Document Ontology and LOINC Codes. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2010;2010:101-5.
11. Rajamani S, Chen ES, Akre ME, Wang Y, Melton GB. Assessing the adequacy of the HL7/LOINC Document Ontology Role axis. Journal of the American Medical Informatics Association : JAMIA. 2015 May;22(3):615-20.
12. Rajamani S, Chen ES, Wang Y, Melton GB. Extending the HL7/LOINC Document Ontology Settings of Care. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2014;2014:994-1001.
13. BioMedICUS [Internet]. Available from: https://bitbucket.org/nlpie/biomedicus. Accessed September 20, 2015.
14. UIMA [Internet]. Available from: https://uima.apache.org/. Accessed September 20, 2015.
15. Horvath MM, Rusincovitch SA, Brinson S, Shang HC, Evans S, Ferranti JM. Modular design, application architecture, and usage of a self-service model for enterprise data delivery: the Duke Enterprise Data Unified Content Explorer (DEDUCE). Journal of Biomedical Information. 2014 Dec;52:231-242.
16. Roth C, Rusincovitch SA, Horvath MM, Brinson S, Evans S, Shang HC, Ferranti JM. DEDUCE Clinical Text: An Ontology-based Module to Support Self-Service Clinical Notes Exploration and Cohort Development. AMIA Joint Summits Translational Science Proc. 2013 Mar 18;2013:227.
17. Jiao Y, Wei W, Chapman W. An Information Extraction Based Search System for Clinical Records. Poster, AMIA Annual Symposium. 2013. Available from: http://knowledge.amia.org. Accessed September 5, 2015.
18. Bill R, Melton GB, Gottschalk B, Ring A, Dale JL, Vinals J, Lamblin A-F, Pakhomov SV. Enabling Use of High Performance Computing for Secure Processing of Protected Health Information. American Medical Informatics Association Joint Summits Translational Science Proceedings. 2015: 244-5.