A Big-Data analytics system for combing patient information from disparate data sources – Interim Report

1 INTRODUCTION

This document has been written to support the second proof of concept (PoC) which seeks to build natural language processing (NLP) and speech to Text (STT) capabilities into the Big Data Health analytics platform developed for the previous PoC. It also moves on to address commercial scalability of the Big Data Health analytics platform as whole.

2 STRUCTURE

This document addresses the deliverable described in the introduction in the following manner.

Section	Synopsis
Section 3: Terminology	Defines the terms used in this document.
Section 4: Why?	The reason for pursuing any capability is that it can offer clinical physician, healthcare professional or health care analyst assistance in doing their job. Whether that's identifying improvements, defining successful patterns of behavior or identifying risks. The role of NLP and STT capabilities in that pursuit.
Section 5: Recap: Big Data Analytics Platform	Software Architecture of the Big Data Analytics Platform
Section 6: Overview of Stanford NLP	Description of Stanford NLP
Section 7: Speech-To-Text Solutions	A summary of speech to text solutions investigated during the course if this PoC
Section 8: Amazon Elastic Cloud to Amazon Elastic Map Reduce	Moving from Amazon Elastic Cloud to Amazon Elastic Map Reduce
Section 9: Conclusion	Closing remarks

3 TERMINOLOGY

3.1 NATURAL LANGUAGE PROCESSING

“Natural language processing is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation.”

3.2 SPEECH TO TEXT

“Speech recognition (SR) is the inter-disciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. This field also goes by the names of "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT).”

3.3 BIG DATA ANALYTICS PLATFORM

A comprehensive technology framework assembled from best in class open source technology, for processing, analysis and visualization of health care big data. This architecture has been presented at the 2016 IEEE International Conference on Big Data, in a paper entitled, “A Novel Big-Data Processing Framework for Healthcare Applications”.

3.4 CLOUD COMPUTING

Cloud computing is a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in either privately owned, or third-party data centers that may be located far from the user–ranging in distance from across a city to across the world.

3.5 Amazon Elastic Cloud

Amazon Elastic Compute Cloud (EC2) forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the hour for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

3.6 Amazon Elastic Map Reduce

Amazon EMR is a web service that provides a Highly Scalable Apache Spark infrastructure capable of high performance computing and analytics. Ref - https://aws.amazon.com/documentation/emr

3.7 Amazon Kinesis Firehose

Amazon Kinesis Firehose is a real-time is a cloud based technology stack for loading streaming data into Amazon Web Services such as Amazon Elastic Cloud and Amazon Elastic Map Reduce.

Ref - https://aws.amazon.com/kinesis/firehose

4 WHY?

4.1 IT'S TRUE TODAY AS IT WAS YESTERDAY

The cornerstone of clinical practice from the beginning of time is the interaction between doctor and patient. This means the doctor and patient talking with one another. There are a host of other factors at work in the modern clinical environment, but it still comes back to talking.

This can be in an 8 minute, conversation between a general practitioner and a patient, all the way to a 90 minute, assessment of a patient with complex clinical needs by a specialist clinician.

That interaction provides the basis of a diagnosis and subsequent management. It could be the difference between life and death. Being able to analyze these interactions may tip the balance.

The means by we could do this is the development of Speech to Text (to capture the conversation) and Natural Language Processing (to analyze the conversation).

4.2 LEVERAGING WHAT'S THERE

From the personal assistance on your smartphone to the automated switch boards that use word recognition software to direct your call, the technology exists to convert speech into action.

Although certainly not a Turing machine, provided you offer a contextualized response, you will get an appropriate answer. For example, if you called your internet provider you could ask, "My internet is too slow! Can you please help?"Hopefully you would be routed to the appropriately skilled help desk technician.

4.3 SIGNIFICANT CHALLENGES

The ability to use NLP and STT to gain advantages for the clinical physician in actual clinical practice is a far from trivial problem. It is way beyond the scope of this PoC, however the development of the tools to go in pursuit of such a gain is a first start. Hence this PoC is about using established NLP and STT components and integrating them within the Big Data Analytics Platform developed in PoC1.

5 RECAP: BIG DATA ANALYTICS PLATFORM

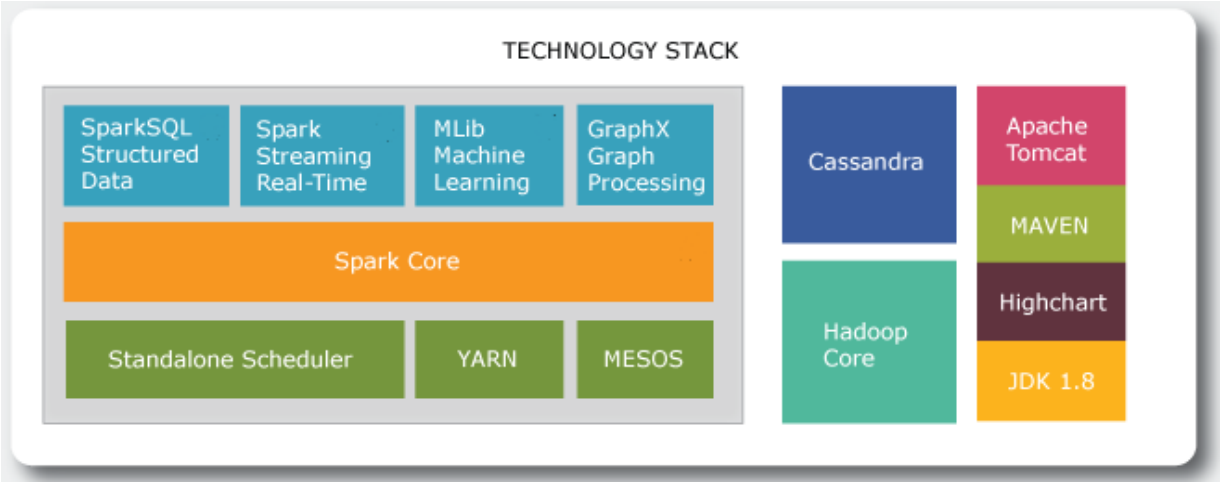
5.1 RECAP

In this section, we take a look at the Big Data in a Box platform developed during PoC1. Usually NLP and STT components are integrated into systems where datasets and data complexity are both small and manageable. As experienced in current biomedical datasets, data volumes can run into terabytes and above, complexity becomes significant. For these reasons our approach from the outset is to integrate NLP and STT into our big data technology stack, as described in this section.

The core data processing element of our technology stack centers around Apache Spark. The reasons for are as follows.

- Lightning Fast Processing
- Support for Sophisticated Analytics
- Real Time Stream Processing
- Integration with both NoSQL and RDBMS
- Ability to Integrate with Hadoop
- Active and Expanding Community

5.2 HIGH LEVEL COMPONENT DIAGRAM



Ref: Apurba Technology – Original Content

The diagram above describes the technology stack that will form the backbone of the biomedical big data analytics platform.

Spark Framework	Component	Description
	Spark Core	Contains basic Spark functionality. Sparks fundamental programming abstraction, Resilient Distributed Data Set (RDD) represents a collection of items spread across parallel computing nodes. Spark provided an API for creating and managing RDDs. This API also takes care of parallel processing and management of RDDs. Applicable here as this would allow handling of large scale datasets requiring NLP and STT
	SparkSQL Structured Data	Package for handling structured data. Querying via SQL as well as Apache Hive. SparkSQL supports interface with RDBMS, NoSQL Databases, HIVE Tables, Parquet and JSON
	Spark Streaming Real-Time	Supports streaming of live data. Spark makes use of RDD creation, processing and management subsystem Allows leveraging real time Speech To Text.
	MLib Machine Learning	Common library containing machine learning algorithms including classification, regression, clustering etc.
	GraphX Graph Processing	Framework for manipulating graphs
	Standalone Scheduler	Cluster Management Services
	YARN	
	MESOS	

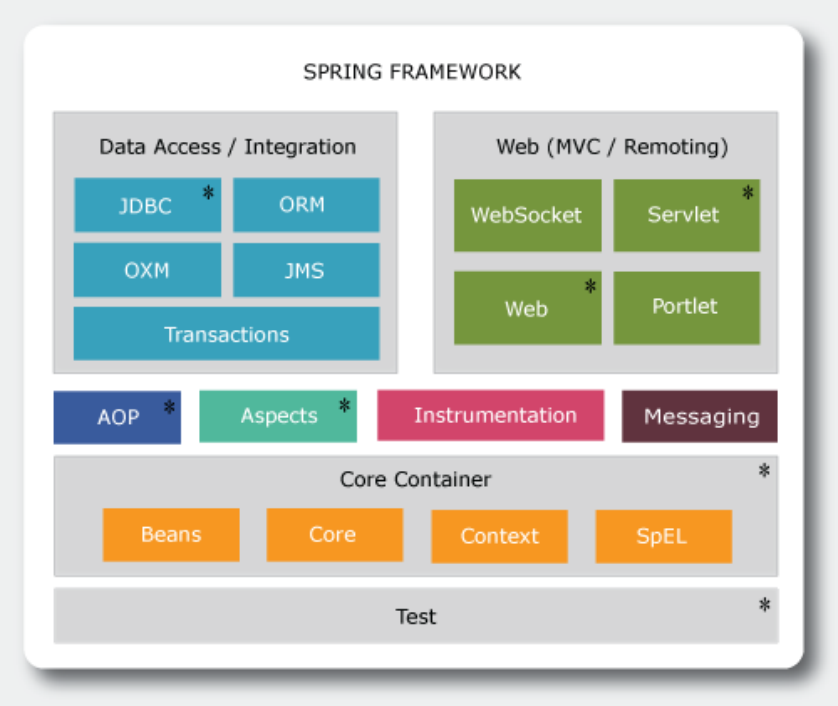
Supporting Technology Stack	Component	Description
	Hadoop Core	Hadoops file system management services. Called directly from within Spark Core
	Cassandra	Industry leading NoSQL database, providing these benefits: High Performance, Elastic Scalability, Open-Source, High Availability and Fault Tolerance, Column Orientated, Tunable Consistency, Schema Free
	JDK 1.8	Java SDK – Programming Language
	MAVEN	Utilized to manage code dependencies
	Apache Tomcat	Web Application Server
	Highcharts	Java Script based charting library for data visualization

5.3 LEVERAGING THE SPRING FRAMEWORK

We are incorporating NLP and STT components into the Big Data Analytics Platform through the use of the Spring Framework.

Spring is a Java based framework that encapsulates the design principle of loosely couple strongly aligned (LCSA). Both the NLP and STT technology sets have been chosen because they offer Java SDK'

The Spring Framework is a set of extensions to the Java programming language that allows development of enterprise applications. Spring is modular in nature and provides the advantage of using only the modules that you want.



Elements of Spring Infrastructure envisioned for use in this PoC based on content from Ref: Spring Tutorials

A comprehensive description of the Spring Architecture can be found here. However, for the purposes of putting forward an architecture for this Healthcare Analytics PoC, the elements of the Spring framework will be discussed in context of the PoC. For this PoC, only the modules marked with * will be used.

5.4 Core Container

The Core Container will be leveraged to develop NLP and STT aware components. These components in turn will come with their own libraries supporting the NLP or STT operation being leveraged,

Springs Core Container consists of the functional elements entitled, Beans, Core, Context and SpEL. These are to be used in the PoC in the following manner.

- Core module encapsulates support for IoC and Dependency Injection design patterns. These design patterns will be used heavily to create an eco-system of interchangeable classes.
- Bean Factory Pattern will be used to generate classes for establishing connection to Apache Spark, Cassandra and other services
- Context builds on top of Core and Bean modules to provide access to any component in the application based on what is called the Application Context. Using this we can have multiple interfaces within our Enterprise Application deployment, each with access to a specific subsystem of classes based on role
- SpEL – allows us to map relationships between objects. This is a very useful feature in situations where you are dealing with an ecosystem of classes

5.5 Data Access/Integration

Spring Data Access/Integration provides support for JDBC (database connection), ORM (Object Relationship Mapping), OXM (Object – XML representation), JMS (Java Messaging Services) and Transactional support. For this PoC we will be using JDBC only. The advantage of Spring is that this does not exclude the utilization of the other components in this layer if needed at a future date.

5.6 Web Layer

Spring Web Layer will be utilized to develop the user interfaces for the Health Analytics platform. This layer consists of the following elements Web, Web-MVC, Web-Socket, and Web-Portlet. Within the context of this ecosystem:

- Web – will provide support for file-upload and utilization of the IoC Container
- Web-MVC. We will use this to develop a web application that will leverage the model-view-controller design pattern
- Web-Socket. Provide communication between the client layer and to the services layer (i.e business logic – leveraging Apache Spark and Cassandra)
- Web Portlets – won't be used for this PoC. It's for development of portlets that leverage MVC design pattern

5.7 Enterprise Architecture Diagram

Development of an Enterprise Architecture (EA) is to take into account these objectives:

- Realize Technology Stack to Software Architecture
- Facilitate Technology Evolution
- Robust
- Maintainable
- Performant

The EA for the PoC is to be expressed as a three tier ecosystem:

- Presentation Services Layer
- Business Process Services Layer
- Data Access Services Layer

The elements of the technology stack will fall into one of these three service layers. The Natural Language Processor (CoreNLP) and Speech-To-Text (Web Speech) fall within the Business Process Services Layer.

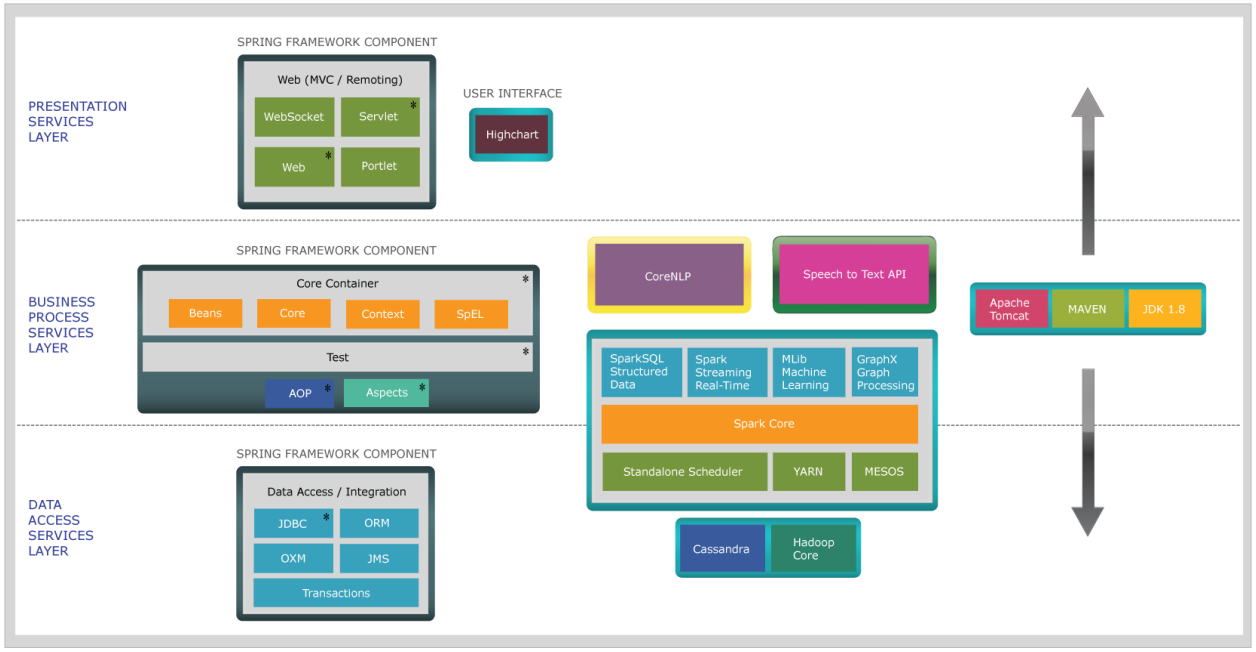


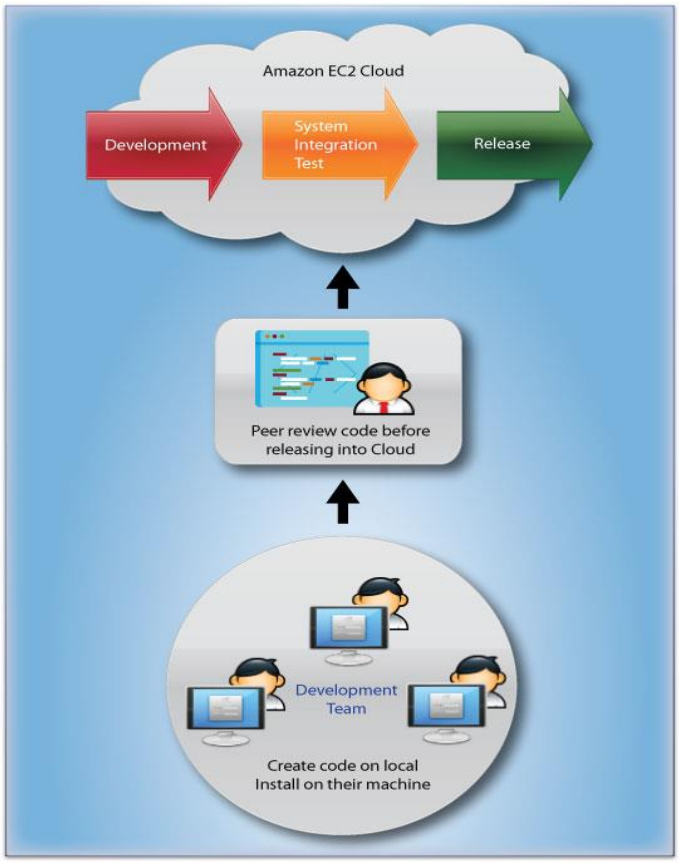
Figure 4.0: Enterprise Architecture Utilizing the PoC Technology Stack – Ref: Apurba Original Content

5.8 DEPLOYMENT CYCLE

The technology stack used for this PoC has the significant advantage of being scalable. It can be installed on Laptop or Mac so that developers can build locally, peer review before uploading to the cloud. In the Amazon Cloud there will be three environments:

- Development
    - Code will undergo build and unit testing
- System Integration Test
    - Application will undergo regression testing.
- Release
    - Stable releases will be deployed to this environment for evaluation

The diagram below depicts this lifecycle.



Ref: Apurba Technology – Original Content

5.8.1 Source code and document management

The system GitHub will be used by the developers as a repository for all source code management. All code as part of the PoC will be documented fully. This documentation will be in the form of both comments and formal implementation documents.

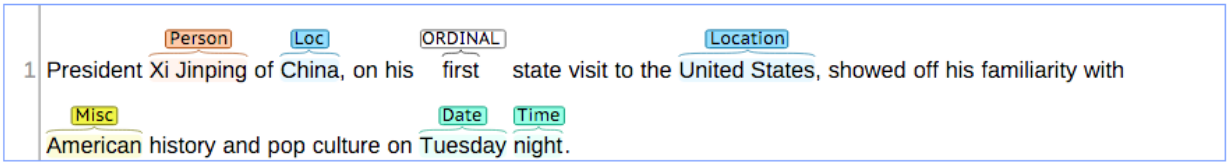
6 OVERVIEW OF STANFORD NLP

Stanford CoreNLP provides a set of natural language analysis tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get quotes people said, etc. Stanford CoreNLP is very suitable for:

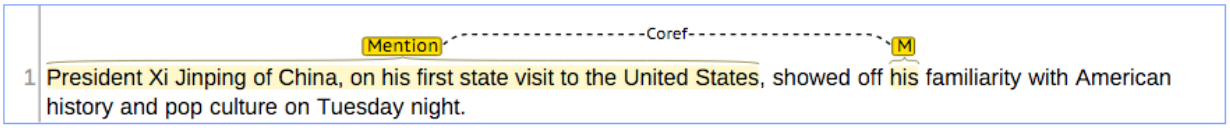
- An integrated toolkit with a good range of grammatical analysis tools
- Fast, reliable analysis of arbitrary texts
- The overall highest quality text analytics
- Support for a number of major (human) languages
- Available interfaces for most major modern programming languages
- Ability to run as a simple web service

Stanford CoreNLP's goal is to make it very easy to apply a bunch of linguistic analysis tools to a piece of text. A tool pipeline can be run on a piece of plain text with just two lines of code. CoreNLP is designed to be highly flexible and extensible. With a single option we can change which tools should be enabled and which should be disabled. Stanford CoreNLP integrates many of Stanford's NLP tools, including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, sentiment analysis, bootstrapped pattern learning, and the open information extraction tools. Moreover, an annotator pipeline can include additional custom or third-party annotators. CoreNLP's analyses provide the foundational building blocks for higher-level and domain-specific text understanding applications.

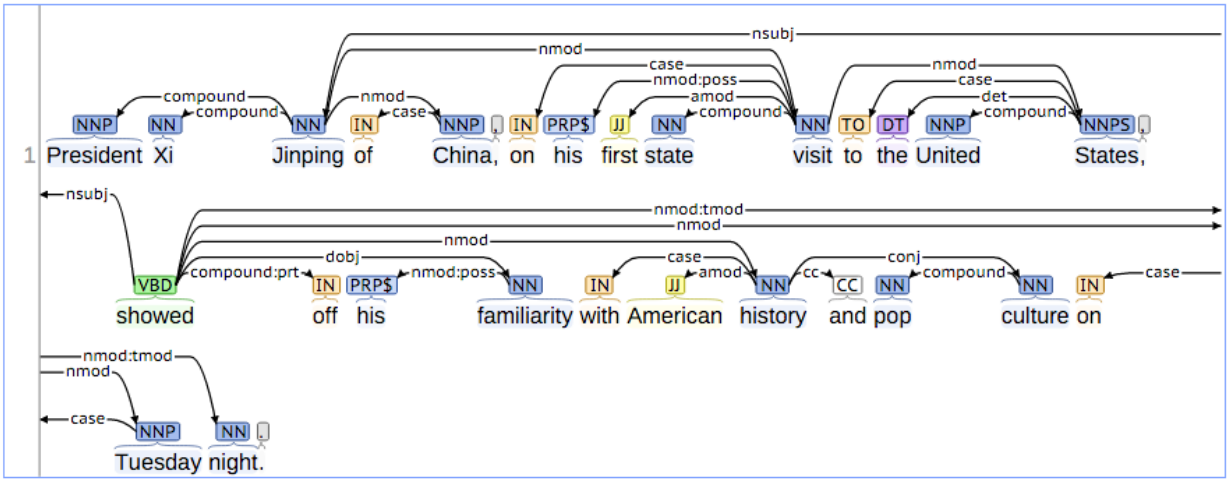
Named Entity Recognition:



Coreference:



Basic Dependencies:



Stanford CoreNLP is written in Java; current releases require Java 1.8+.

We can use Stanford CoreNLP from the command-line, via its Java programmatic API, via third party APIs for most major modern programming languages, or via a service. It works on Linux, OS X, and Windows.

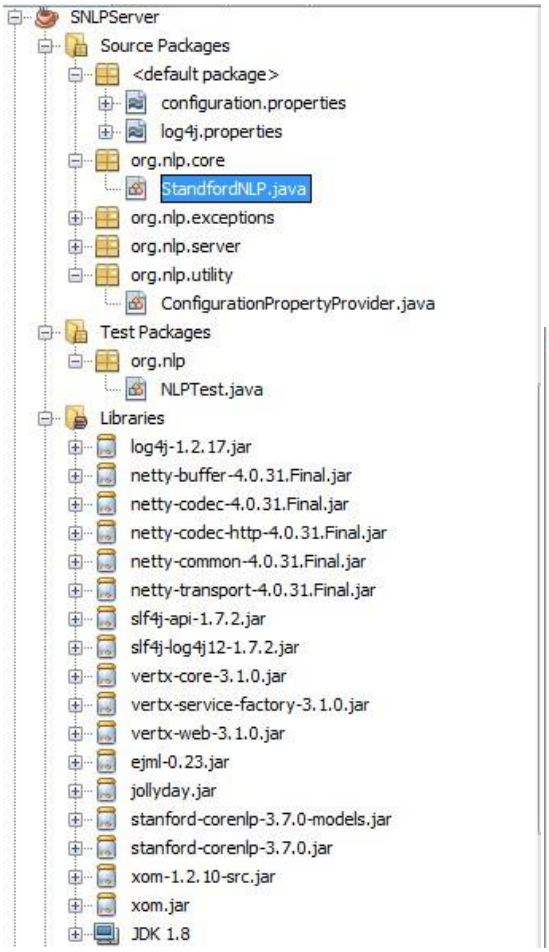
Stanford CoreNLP is licensed under the GNU General Public License (v3 or later; in general Stanford NLP code is GPL v2+, but CoreNLP uses several Apache-licensed libraries, and so the composite is v3+). Note that the license is the full GPL, which allows many free uses, but not its use in proprietary software, which is distributed to others. For distributors of proprietary software, commercial licensing is available from Stanford.

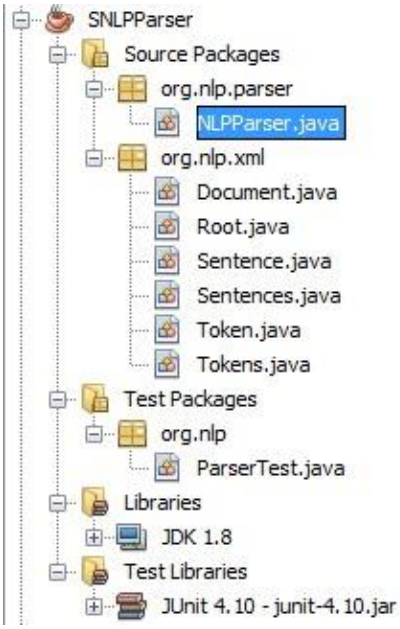
6.1 IMPLEMENTATION

As mentioned, CoreNLP requires the current releases of Java 1.8+. We built an application consisting of a services engine (SNLP Server) and a satellite subsystem (SNLP Parser) which leverages the output of the SNLP server.

The result is providing an ecosystem of components for Natural Language Processing that can be called from anywhere in our Big Data Health Analytics Platform.

The screenshots below are taken from one of our development environments





7 SPEECH TO TEXT SOLUTIONS

7.1 WHY WEB SPEECH API?

The Web Speech API was selected as it was deemed the most suitable for integrating into the Big Data Analytics Platform. We took the decision to leverage an established web service. This web service is called using JavaScript. It fits into the business service layer in the Enterprise Software Architecture.

This provides an additional over a software development kit type solution (i.e Sphinx), in that we do not need to deploy further Java libraries in our environment.

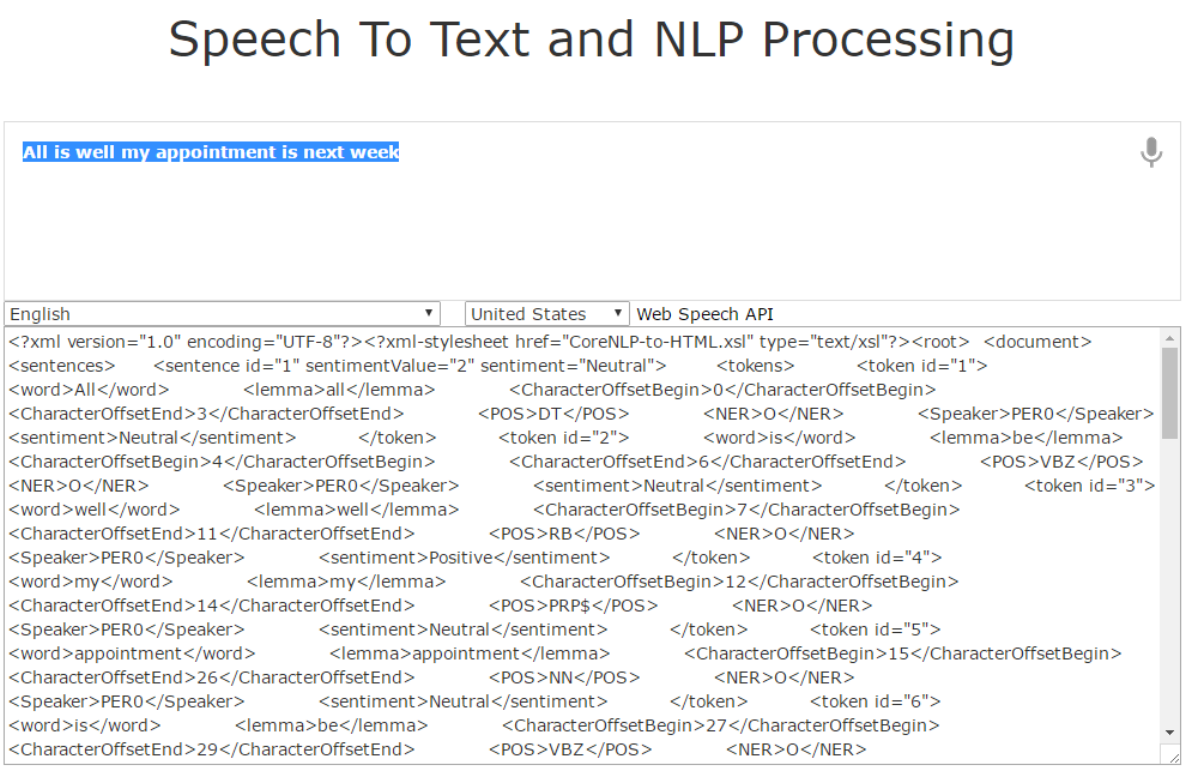
This new open source service allows the application to handle multiple languages simply and easily without complex application development.

7.1.1 Implementation

The figure below shows the two components Web Speech API and CoreNLP used together.

First the user talks into the microphone. This calls the Web Speech API in real-time and then when finished the speech is analyzed using CoreNLP. The CoreNLP application returns a graph encountered in NLP with a full semantic analysis. This happens within seconds.

It should be noted that this demonstrator is running as a cloud based service on a single node. Performance across multi -node is expected to be even faster.



7.2 OTHER CONTENDERS

Before settling on Web Speech API the engineering team assessed other leading technologies.

7.2.1 CMU Sphinx - Sphinx 4

Popular, open source speech to text, text to speech library developed in java.

Features:

- Utilized as a separate offline application or embedded within desktop of mobile app.
- Customizable dictionary
- Customizable grammar

Best Uses:

- Building Application requiring voice command activation.

Limitation:

- Maintaining grammar repositories for word recognition
- Server side library would make maintaining components built using this technology challenging.

7.2.2 Kaldi

Speech to Text software written in C++

Best Uses:

- Good for developing applications for Windows platforms

Limitation

- C++ applications require significant scaffolding to run in Big Data environments. Will require a dedicated windows server and development of .NET adapters to talk with our Apache Spark/Linux based Big Data Engine.

7.2.3 Google Cloud Speech API:

Significant investment by Google and supports over 80 languages, giving the scope of any commercially developed products, international appeal.

Best Uses:

- For Commercial implementations using the Health Care Big Data Analytics Platform

Limitations:

- Currently in Beta release. There is no guarantee that it will go into Alpha in the near future.

8 AMAZON ELASTIC CLOUD (EC2) TO AMAZON ELASTIC MAP REDUCE (EMR)

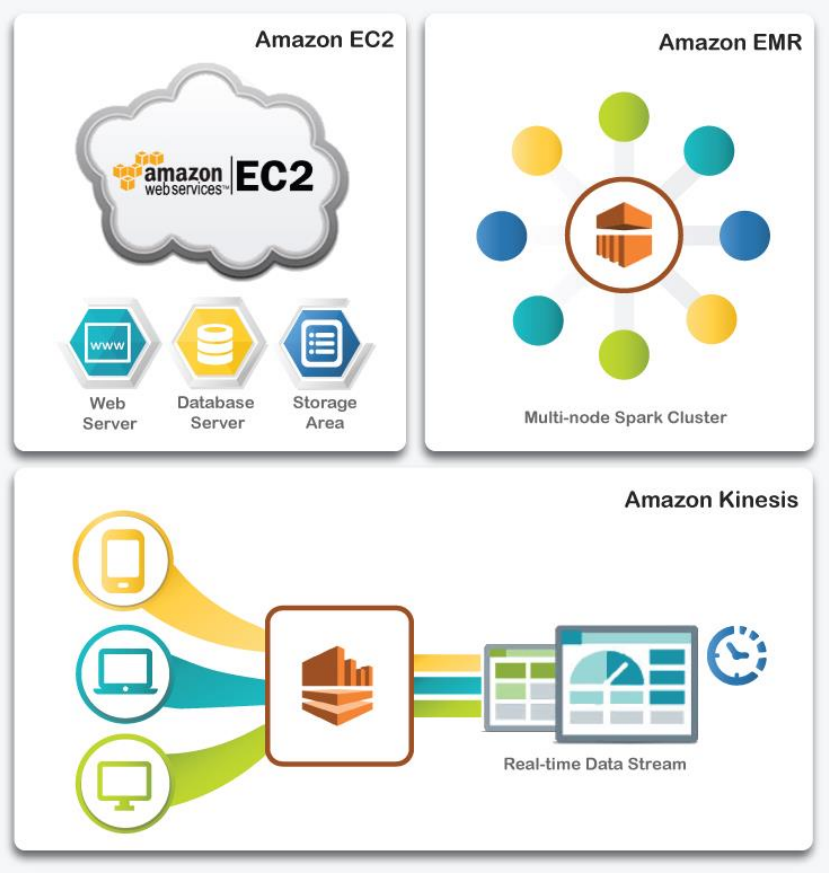
An additional deliverable in this PoC was an investigation into how the Health Care Big Data Analytics Platform may be scaled up to meet actual commercial demand.

In reality, any commercially orientated solution needs to be scalable. This does not mean to be scalable to one humongous size, but to allow itself to be scale to meet the demands of the environment.
This is by no means a trivial task and prior to the advent of Cloud Computing this would have been nearly impossible to do without extensive monitoring and maintenance.

Before Cloud Computing scaling a solution may have been constrained by not being able to physically house more hardware. Or resources had to be provisioned up front to meet peak usage. That means in periods of low usage, expensive computing resources would run idle and form a costly overhead.

In the world of Cloud Computing resources are provisioned on demand. Logically we wish to leverage this key advantage for the Health Care Big Data Analytics Platform.

In the previous PoC, Amazon Elastic Cloud was used to provision a single server to host our prototype. For a commercial scale application we propose the following architecture operating with the Amazon Cloud ecosystem: consisting of Amazon Elastic Cloud(EC2), Amazon Elastic Map Reduce (EMR) and Amazon Kinesis Firehose.



So what do these mini ecosystems do?

Amazon EC2 – is used to host the web application server, database and storage areas.

Amazon EMR used to provide the infrastructure to host Apache Spark capable of handling large amounts of data.

Amazon Kinesis Firehose allows us to handle high throughput real time data. This takes care of a facet of health care data that has been observed before namely, that it's high velocity real-time data from a wide variety of sources, i.e. medical systems, live or streamed audio/video data, twitter feeds or Internet of Things.

9 CONCLUSION

In this document the technology that has been used to deliver Natural Language Processing and Speech to Text capabilities are presented.

The mechanisms by which these components are to be introduced into the Health-Care Big-Data-Analytics-In-a-Box developed in the previous PoC have been discussed.

Furthermore, the commercial scaling of Health-Care Big-Data-Analytics-In-a-Box has been presented.

As previously all the technologies presented have seen rigorous use in other projects with comparative demands on performance, flexibility and ease of use.

The final report will include descriptions and results of use of this platform and how information from Speech and Text can be processed and combined within this platform to deliver robust results.