International Journal of Scientific Research and Engineering Development-– Volume 4 Issue 3, May -June 2021

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Machine Learning in JAVA: Overview

Shubham Jain*, Priya Darshini R**
*(*Computer Science Engineering, RV College of Engineering, Bengaluru*
Email: jshubh12.99@gmail.com)
** (*Computer Science Engineering, RV College of Engineering, Bengaluru*
Email: priyatej27@gmail.com)

----------**********----------

Abstract:

Machine Learning has brought many advancements in the technology recently which we are grateful to. Computers that learn from their mistakes should someday reduce the need for much of this sophisticated programming. Data analysis is a critical issue for businesses since it gives significant business insights that help them operate better. Enterprise Big Data processing apps and platforms are typically found in the Java environment. As a result, it's critical to look into how Java-based platforms for data analytics and machine learning work and what features they offer. Deeplearning4j and DJL are two of the most extensively used machine learning libraries in the Java world, according to the report. For storing massive amounts of data for data analytic tasks, data lakes are a frequent practise. Hadoop has proven to be the most popular platform for creating data lakes. This paper surveys the different libraries available in Java for Machine Learning focusing on integrating machine learning in Spring Boot Application.

Keywords **—Java Spring Boot, MVC, Deep Java Library, Microservices, Machine Learning**

----------**********----------

I. INTRODUCTION

This document is a template. An electronic copy can bedownloaded from the conference website. For questions onpaper guidelines, please contact the conference publicationscommittee as indicated on the conference website. Information about final paper submission is available from the conference website.

Despite being independent fields, Spring Boot and Machine Learning can work together to tackle certain challenges. Spring Boot is the most popular and frequently used open-source framework for developing microservices, making the deployment of distributed systems easier. Machine Learning, on the other hand, is a method of extracting knowledge from data and then delivering insights and predictions.

Although more data is generally better in machine learning, it does not always cope with high-velocity data or data in a variety of formats. When there is a big amount of extremely similar data in which some prediction or insight is required, Machine Learning works best.

Despite its widespread popularity, there are few options for easily integrating it with machine learning (ML) on a local level. Existing solutions rarely meet the needs of customised applications and developing customised solutions is both time-consuming and expensive. This paper will show how Java users can use spring boot starter for Deep Java Library (DJL) to integrate ML into their applications.

II. DJL – DEEP JAVA LIBRARY

Deep Java Library (DJL) is a high-level, framework-agnostic Java API for machine learning that is open source. It is intended for Java developers to be simple to use and get started with. DJL is a native Java development environment that works just like any other Java library. DJL provides a convenient abstraction layer for using the most popular AI/ML frameworks. However, it is not just a convenience on top of the existing libraries (some of which provide Java API / bindings). With DJL API, you are getting a uniform and consistent layer that can interact with all of these frameworks, allowing you to swap out the framework of your choice without any impact to the client code.This unique feature, in combination with a fairly rich model zoo repository (a repository with pre-trained models), can enable ML engineers to find optimal models for the task at hand regardless of the underlying model implementation.

III. JAVA SPRING

For more productive application development, the Spring Framework is a significant open source Java/J2EE application development framework. It is the most widely used Java framework, accounting for 30% of all usage. Spring Framework characteristics allow for quick development of everything from simple Web apps to big business systems. The following are the main notions on which the Spring Framework is built:

- Java Persistence API (JPI)
- Dependency Injection (DI)
- Inversion of Control (IoC)
- Aspect Oriented Programming (AOP)

IoC is a general notion in which flow control is inverted: instead of the programmer managing the flow of a programme, it is controlled by other sources (framework, services, and other components). DI is a pattern in the form of IoC, as described in, which is now a crucial idea in Spring and the quickly growing Google Guice. AOP is built on a philosophy that increases both code modularity and organisation. JPA is responsible for translating object state to database columns and determining how to query across objects.

IV. SPRING BOOT

Spring Boot is a Java-based open source framework for developing microservices. Pivotal Team created it, and it's used to create stand-alone and production-ready spring apps. This chapter provides an overview of Spring Boot and familiarizes you with its fundamental ideas. Spring Boot is a good platform for Java developers to create a stand-alone, production-ready spring application that can be run straight away.

Dependency Management

Although the DJL library is platform-specific, it does give means to hunt for the relevant dependencies based on the target operating system automatically. Even after you've made your decision, you can alter the underlying engine and target operating system architecture by altering your Maven (or Gradle) dependency without affecting your code.

Using PyTorch as an underlying engine, the starter dependency is:

```
<dependency>
 <groupId>ai.djl.spring</groupId>
 <artifactId>djl-spring-boot-starter-pytorch-auto</artifactId>
 <version>${djl.starter.version}</version> <!-- e.g. 0.2 and above -->
</dependency>
```

Fig 1. Starter dependency using Pytorch

Spring auto-configuration

After your Spring Boot application's dependencies are set up correctly, the following step is to configure your beans and wire them up for injection. Configuring DJL-related beans and making them available in the Spring application environment is very simple, but correct scoping

necessitates intrinsic knowledge of the library as well as the quirks of specific classes—Some beans should be scoped per request/thread, while others should be thread-safe. The DJL Spring Boot starter includes an auto-configuration feature to help with this setting.

V. ARCHITECTURE DIAGRAM

On top of current Deep Learning frameworks, DJL is constructed using native Java ideas. It gives customers access to the most cutting-edge Deep Learning breakthroughs as well as the capacity to work with cutting-edge hardware. The straightforward APIs take away the complexities of constructing Deep Learning models, making them simple to understand and implement. Users can start implementing Deep learning into their Java applications right now with model-bundled zoo's set of pre-trained models.

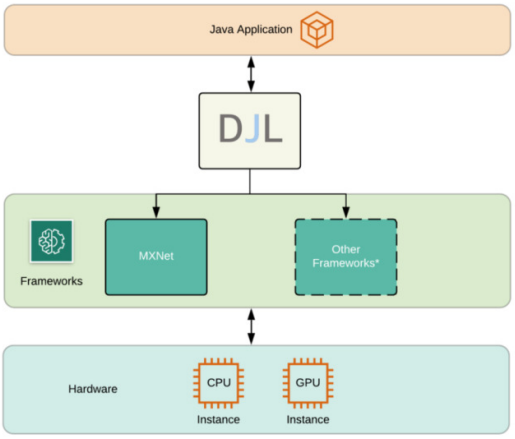

Fig 2. System architecture for Spring Boot JAVA application

VI. DJL SPRING BOOT STARTER

The Deep Java Library (DJL) is a Java library that was created to assist Java developers in learning about deep learning. This is a Spring Boot starter project that enables Spring Boot developers to begin using DJL for inference.

The starter includes auto-configuration and dependency management.

Starter Dependencies

The starter sets up dependencies in one of two ways: platform-specific or automated.

MXNet Configuration

```
<dependency>
    <groupId>ai.djl.spring</groupId>
    <artifactId>djl-spring-boot-starter-mxnet-osx-x86_64</artifactId>
    <version>${djl.starter.version}</version>
</dependency>
```

Fig 3. Configuration code for MXNet

MXNet Auto Configuration

```
<dependency>
    <groupId>ai.djl.spring</groupId>
    <artifactId>djl-spring-boot-starter-mxnet-auto</artifactId>
    <version>${djl.starter.version}</version>
</dependency>
```

Fig 4. Auto Configuration code for MXNet

Pytorch Configuration

```
<dependency>
    <groupId>ai.djl.spring</groupId>
    <artifactId>djl-spring-boot-starter-pytorch-auto</artifactId>
    <version>${djl.starter.version}</version>
</dependency>
```

Fig 5. Configuration code for Pytorch

VII. JAVA-ML

Java-ML (Java Machine Learning Library) is a Java framework/API for software engineers, programmers, and scientists. Techniques for data pre-processing, feature selection, classification, and clustering can all be found in the huge array of machine learning and data mining algorithms. It is simple compared to other clustering algorithms and

enables for quick implementation of any new algorithm. Although there is no graphical user interface, algorithms of the same type share a common interface.

In terms of substance, Java-ML differs from the other libraries in that it focuses on machine learning. Java-ML includes a large number of similarity-based algorithms as well as cutting-edge feature selection algorithms. The enormous number of similarity functions allows for a wide range of clustering and instance-based learning approaches, while feature selection approaches are well-suited for dealing with high-dimensional domains like those found in bioinformatics and biomedical applications.

Clustering
K-means-like (7)
Self organizing maps
Density based clustering (3)
Markov chain clustering
Cobweb
Cluster evaluation measures (15)

Classification
SVM (2)
Instance based learning (4)
Tree based methods (2)
Random Forests
Bagging

Feature selection
Entropy based methods (4)
Stepwise addition/removal (2)
SVM_RFE
Random forests
Ensemble feature selection

Data filters
Discretization
Normalization (2)
Missing values (3)
Instance manipulation (11)

Distance measures
Similarity measures (6)
Distance metrics (11)
Correlation measures (2)

Utilities
Cross-validation/evaluation
Data loading (ARFF and CSV)
Weka bridges (2)

Fig 6. Overview of the algorithms in Java-ML

VIII. WEKA

Last but not least, the open-source Weka machine learning package for Java is likely the most well-known and popular. A comprehensive graphical user interface, command-line interface, and Java API are all included in the general-purpose library. It's completely free, portable, and simple to use. WEKA has been widely accepted by numerous scientists in diverse fields of competence since its inception in 1992. Some of WEKA's primary traits can be found in some of the reasons for its popularity: Data pre-processing, Classification, Clustering, Attribute Selection, Data Visualization.

Despite all of its features, WEKA has proven extremely popular because of its intuitive and simple-to-use user interface. As a result, machine learning methods are now accessible to practically anyone who wishes to classify data, regardless of whether or not they are a coder.

Fig 7. WEKA GUI Chooser Window

IX. APACHE MAHOUT

Apache Mahout(TM) is a distributed algebraic framework with a mathematically descriptive Scala Language for mathematics, statisticians, and data scientists to efficiently construct their own methods. Apache Spark is the suggested distributed back-end out of box, however it may be expanded to work with other distributed back-ends.

Scala DSL Support for Multiple Distributed Backends that is Mathematically Expressive (including Apache Spark)Native Solvers for CPU/GPU/CUDA Acceleration Modular Native Solvers for CPU/GPU/CUDA Acceleration

Apache Mahout is a machine learning library that is scalable. Mahout is a classification, collaborative filtering, and clustering tool that employs the MapReduce model. Mahout makes use of Apache Hadoop to handle numerous operations in parallel. Mahout offers recommendation methods such as collaborative filtering, in addition to classification and clustering, enabling the scalability of fast constructing your model.

X. MALLET

Machine Learning for Language Toolkit is referred to as Mallet. It's one of the few specialist natural language processing toolkits available. It has topic modelling, document categorization, clustering, and information extraction capabilities. We can use Mallet to process textual texts using machine learning models.

MALLET is a Java-based software that may be used to do statistical natural language processing, document classification, clustering, topic modelling, information extraction, and other machine learning tasks on text.

MALLET comes with advanced document categorization capabilities, such as fast routines for transforming text to "features," a large number of algorithms, and more. Includes code for measuring the performance of classifiers using a variety of regularly used metrics.

It provides tools for sequence tagging for applications like named-entity extraction from text, in addition to categorization. Hidden Markov Models, Maximum Entropy Markov Models, and Conditional Random Fields are examples of algorithms. For finite state transducers, these approaches are implemented in an extensible framework.

Topic models come particularly handy when dealing with vast amounts of unlabelled material. Latent Dirichlet Allocation, Pachinko Allocation, and Hierarchical LDA are all included in the MALLET topic modelling toolbox as efficient, sampling-based implementations.

Many of MALLET's algorithms rely on numerical optimization. Among many other optimizations, MALLET contains an efficient implementation of Limited Memory BFGS.

MALLET offers procedures for converting text documents into numerical representations that may later be handled effectively, in addition to complex Machine Learning applications. This is accomplished via a flexible system of "pipes" that handle operations like tokenizing texts, deleting stop words, and transforming sequences into count vectors.

XI. CONCLUSIONS AND FUTURE WORK

Machine Learning has been utilised to solve complicated analytical issues with great success. Because of the variety of data and scenarios, each problem is approached from the ground up, resulting in costly solutions that take an inordinate amount of time to implement. The majority of machine learning research and development is done in Python, which is sometimes inconvenient because most enterprise applications and platforms/frameworks for Machine Learning processing are built on the Java ecosystem and Java environment. It is impossible to implement a system that can solve any machine learning challenge. However, the research presented shows that it is possible to create an easily extendable platform that handles data management and does distributed machine learning over it. This job will be the focus of our future research efforts because it will considerably lower the costs and work required to construct future machine learning applications, as well as make machine learning tools more widely adopted and deployed in a variety of software projects.

REFERENCES

[1] Svetoslav Zhelev *and* Anna Rozeva , "Data analytics and machine learning with Java", AIP Conference Proceedings 2048, 060020 (2018) https://doi.org/10.1063/1.5082135

[2] Patterson, J., Gibson, A. Deep Learning: A Practitioner's Approach, O'Reilly Media, ISBN: 9781491924570, August 2017

[3] Tomcy, J., Pankaj, M., Data Lake for Enterprises: Lambda Architecture for building enterprise data systems, Packt Publishing Ltd., ISBN: 1787281345, May 2017

[4] Zhelev, S., Rozeva, A. Big data processing in the cloud - challenges and platforms, AIP Conference Proceedings, DOI: https://doi.org/10.1063/1.5014007, Dec 2017

[5] Ryza, S., Laserson, U., Owen, S., Wills, J. Advanced Analytics with Spark, O'Reilly Media, ISBN: 978-1-491-97295-3, June 2017

[6] Ashish Singh Bhatia and Bostjan Kaluza. 2018. *'Machine Learning in Java'*: Helpful techniques to design, build, and deploy powerful machine learning applications in Java, 2nd Edition. Packt Publishing.

[7] L. Butgereit, "Big Data and Machine Learning for Forestalling Customer Churn Using Hybrid Software," *2020 Conference on Information Communications Technology and Society (ICTAS)*, 2020, pp. 1-4, doi: 10.1109/ICTAS47918.2020.233972.

[8] H. Arndt, "The Java Data Mining Package - A Data Processing Library for Java," *2009 33rd Annual IEEE International Computer Software and Applications Conference*, 2009, pp. 620-621, doi: 10.1109/COMPSAC.2009.88.

[9] H. Arndt, A. Naegele, and M. Bundschus, "Java Data Mining Package (JDMP)," 2009, http://www.jdmp.org

[10] H. Arndt, A. Naegele, and M. Bundschus, "Towards a next generation matrix library for Java," COMPSAC: International Computer Software and Applications Conference, 2009

[11] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2009). The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11, Issue 1.

[13] Abeel, T.; de Peer, Y. V. &Saeys, Y. Java-ML: A Machine Learning Library, Journal of Machine Learning Research, 2009, 10, 931-934

[14] K. Guntupally, R. Devarakonda and K. Kehoe, "Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5328-5329, doi: 10.1109/BigData.2018.8621924.

