Survival Analysis with Apache Spark and Apache SystemML on Stackexchange

Mateo Álvarez Calvo

June 8, 2017

Contents

1 Introduction & main goals

Many studies have been done over the Stackexchange community [such as], one of the biggest Q&A sites in the world. The present is yet another study over the data of the famous site, but in this case, the study has two particularities, the use of Apache Spark with the library of Apache SystemML for the processing in a parallel environment, and the use of Survival Analysis to analyze the impact of the variables in the time an answer is accepted for each question, the "survival of each question" in the community.

1.1 Main technologies

As one of the biggest Q&A communities, Stackexchange has large amount of data of each interaction. Stackexchange is separated in several communities, regarding different topics. These communities can be small, as [] or really big, as Stackoverflow, the developers community. This particularity makes necessary the use of technologies prepared to process large amounts of data, in the later case.

The purpose of the present study is to analyze this big community, so a distributed processing technology has to be used. For this purpose, Apache Spark, the latest distributed open-source processing technology, has been chosen to parallelize the operations on the data.

Spark ML is the machine learning library of spark, which contains lots of algorithms. It also includes some Survival Analysis algorithms, but just for parametric modeling. This gives an excuse to use the recently adopted by the Apache Foundation SystemML, a machine learning library developed by IBM, which has non, semi and parametric algorithms for survival analysis.

Regarding the development environment, Jupyter Notebook provides a simple and flexible interface for this analysis, and can also be integrated with Spark, allowing the complete development in just one environment.

1.1.1 Apache Spark

Apache Spark is a distributed processing technology developed in Scala by Databricks that represents the next step of Apache Hadoop, including the best parts of it, such as the Hadoop File System, but under a complete new paradigm that allows operations different from the famous map-reduce, using RAM as storage for results rather than writing to disk, lazy and optimized execution of tasks, and special focus on machine learning and SQL-like language, to mention some of the main features.

1.1.2 Apache SystemML

Recently included in the Apache Foundation Incubating program, Apache SystemML is a machine learning library that works over distributed frameworks, Spark or Hadoop, written in Java.

This library provides many distributed implementations of important algorithms as well as a sintaxis to create new algorithms in a distributed mode. It provides a high-level declarative machine learning language, which has two variations, the R-like sintaxis, DML, and the Python-like sintaxis, Py-DML. These self made algorithms pass through a compilation process and are optimized for a distributed environment.

It can be executed in a variety of distributed and non distributed modes, with it's standalone mode, and the integration with Hadoop, and Spark via SystemML context, which allows the interaction through Scala, Python and R.

1.2 The Stackexchange data

1.2.1 Stack Exchange

Stack Exchange is a network of Q&A websites created in 2009 after the great success the creators obtained with *Stack Overflow* community in 2008, a Q&A community website for computer programming.

Each community covers a different topic, from physics to software, and is structured in a reputation award format. Each site has questions, answers and users, all subjected to this reputation award process.

All these communities generate large amounts of data, data thet Stackexchange facilitates every once in a while for data scientists and people in general to download and analyze. The data is available in a torrent file and each package has about 35 - 40 GB of compressed information.

This compressed file has data from different communities for a certain period of time. In this case, the analysis is done over the Scifi community, which is a median size community for science-fiction Q&A.

1.2.2 Scifi community

Scifi is a community in Stack Exchange that focuses on science fiction and fantasy. This community was selected because it has a medium size which is perfect to test the mentioned technologies in a reasonable period of time. Selecting just the data from Scifi community from the big compressed file, it weights around 110 MB in a 7z compressed format. This allows the computation on a local machine for experimentation and then

scale the problem to a bigger community such as *Ask Ubuntu* or *Stack Overflow* when the process is refined and it can be launched remotely in a cluster, as the data structure is common to al other communities. The data is divided in 8 files, regarding different information:

File	Description	Size
Votes.xml	Voting results for each question and more cosas	84,1 MB
Tags.xml	Relational table for tags on each question	169 KB
Users.xml	Users on the net	16,7 MB
PostLinks.xml	links to posts	1,5 MB
Posts.xml	List of all questions	137,3 MB
PostHistory.xml	All interactions of each post	268,6 MB
Comments.xml	List of all comments of each post	66,3 MB
Badges.xml	All users' badges	16,1 MB

Table 1: List of files from the compressed Scifi folder

Further details about the relational database structure is explained below, the objective is to show the variables obtained from the dataset so that the later variable selection is understood.

1.2.3 Votes.xml

This file contains information about votes of the users to each question. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique vote identifier
PostId	Integer	Foreign key that indicates the post that was voted
VoteTypeId	Integer	Type of vote, 1 for Down-vote and 2 for Upvote
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time of votation

Table 2: Votes table

1.2.4 Tags.xml

This file contains all tags and the posts that contains them. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique tag identifier
TagName	Text	Name of the tag
Count	Integer	Number of times used
ExcerptPostId	Integer	
WikiPostId	Integer	

Table 3: Tags table

1.2.5 Users.xml

This file contains information about users. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique user identifier
Reputation	Integer	Reputation level of the user
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	User creation date
DisplayName	Text	Alias to display on question
LastAccessDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Last login date
WebsiteUrl	Text	Site where the user signed up to
Location	Text	Location of the user
AboutMe	Text	Information user provided
Views	Integer	User views count
UpVotes	Integer	User up votes count
DownVotes	Integer	User down votes count
AccountIf	Integer	Unique user identifier

Table 4: Users table

1.2.6 PostLinks.xml

This file contains information about relation between posts. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique post links identifier
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Post links creation date
PostId	Integer	Post unique identifier
RelatedPostId	Integer	Unique identifier of the post related to the PostId
LinkTypeId	Integer	Type of relation between posts

Table 5: Post links table

1.2.7 Posts.xml

This file contains all questions posted along with the accepted answers and other info related to the time and user who posted the question. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique question identifier
PostTypeId	Integer	Type of post codified as integer
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time of question creation in extended format
Score	Integer	Question's score, calculated from the users' votes
ViewCount	Integer	Count of all visualizations of the question
Body	Text	The question itself, in utf8 format
OwnerUserId	Integer	Id of the user who posted the question
LastEditorUserId	Integer	
LastEditDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Last edition date
LastActivityDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Last interaction with the question time
Title	Text	Title of the question
Tags	Text	Tags added to the question
AnswerCount	Integer	Number of answers to the question
CommentCount	Integer	Number of comments to the question posted
FavoriteCount	Integer	Number of times the question has been added to favorite by another user
ClosedDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time the question has been closed
CommunityOwnedDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time

Table 6: Posts table

1.2.8 PostHistory.xml

This file contains information about the interactions with each post. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique interaction identifier
PostHistoryTypeId	Integer	Type of interaction with the post ()
PostId	Integer	Unique identifier of the post this interaction is related to
RevisionGUID	Text	
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time of question creation in extended format
UserId	Integer	Unique identifier of the user that created the interaction with the post
Text	Text	Text the user introduced on the interaction

Table 7: Post history table

1.2.9 Comments.xml

This file contains the comments posted for every question created. The file has the following structure:

Feature	Data type	Description
Id	Integer	Unique comment identifier
PostId	Integer	Unique identifier of the post this comment is related to
Score	Integer	Total score of the comment
Text	Text	Comment text
CreationDate	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time of comment creation in extended format
UserId	Integer	Unique identifier of the user who posted the comment

Table 8: Comments table

1.2.10 Badges.xml

This file contains information about the badges the user has obtained.

Feature	Data type	Description
Id	Integer	Unique Badge identifier
UserId	Integer	Unique identifier of the user who obtained the badge
Name	Text	Name of the badge
Date	Timestamp YYYY-MM-DDTHH:MM:SS.dScSmS	Time the user obtained the badge in extended format
Class	Integer	Type of the badge
TagBased	Boolean	Whether the badge is based on a tag or not

Table 9: Comments table

1.3 Main Objectives

The main objective of this study is to test the scalability and integration of the proposed technologies, Spark, SystemML and Jupyter Notebook in the usecase of Stack Exchange communities, so further data analysis can be performed. This main goal is divided in three major goals:

- Use Spark to make the data cleaning to create a script for further research on the Stackexchange site.
- Verify SystemML integration with Spark for further research and scalability.
- Use SystemML survival analysis algorithms to analyze Stackexchange's data and obtain conclusions on the main variables affecting the time taken by the community to answer each question.

2 Technologies

The downloaded data from Stackexchange for the analysis weights about 40 GB, which is enough amount to consider distributed processing. Going down to the distributed processing frameworks, Apache Spark was chosen

2.1 Apache Spark

Apache Spark is a fast and general-purpose cluster computing system, widely used for data processing. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib and SparkML for machine learning and pipelines, GraphX for graph processing, and Spark Streaming.

This distributed data processing framework was initially developed at the University of California, Berkeley's AMPLab, and donated to the Apache Software Foundation in February 2014, the first release was on May 30th 2014.

Apache Hadoop presented some limitations that Apache Spark tried to solve:

- It is difficult to write most of the algorithms in a MapReduce form.
- It is very slow to write each iteration to disk, which, for example makes difficult to use Hadoop to process streaming.
- Apache Hadoop's support for iterative jobs and Machine Learning restricts it's use for this task.
- Apache Hadoop's SQL tools doesn't work well on complex queries, sometimes it doesn't work at all and other times it is quite slow as it writes on HDD each iteration.

Some solutions Apache Spark provides to these problems are:

- Lazy computation, Spark only executes a set of tasks when a result is required. This gives the opportunity to optimize jobs before executing them.
- In-memory data caching, Spark scans HDD only once to read the input data and then uses RAM as much as it can, which is faster than scanning disk on every step.
-

2.1.1 Spark Data Structure

2.1.2 Apache Spark Structure

Spark has a master-slave architecture and supports various resource managers: standalone, Mesos and YARN. The resource manager will only be in charge of identify the resources. Independently of the resource manager chosen, the Apache Spark architecture doesn't change.

All Spark processes share the same architecture, the driver process takes care of calculating the job's parallelism and calculates every job's stage. Each stage is divided on a series of tasks, which will be sent to the executors, where will be executed. The communication between the Spark application and the resource manager will be done through a SparkContext.

The deployment of Spark has two variants, client and cluster mode. On the client mode, the Driver is launched on the machine the process has been invoked, and it can be inside or outside the cluster. On cluster mode, the cluster manager is assigned to control the Driver process, and the process itself will be launched inside the cluster. In case Mesos is the cluster manager, it will require an additional service.[][][][][][][][]

Apache Spark is formed by the Spark Core API, available in R, SQL, Python, Scala and Java languages, and built up on it four main libraries: Spark SQL + DataFrames, Spark Streaming, Spark MLlib, Spark ML and GraphX, that complements functionality for Spark, specially on the parts Hadoop failed, Machine Learning, SQL and Streaming.

Spark Core API

The core API represents the basic structure of Spark, it can be addressed form any of the supported languages, scala, python, R and java.

Spark SQL + DataFrames

The SQL layer over data is known as Spark SQL

Spark MLlib + Spark ML

Spark has two main Machine Learning libraries, the first one, Spark MLlib, which is the basic library, that includes the main algorithms and is addressed with RDDs, the second one, Spark ML, which uses Spark MLlib but through DataFrames, and includes further functionality, such as Pipelines, a set of operations performed over data, that allows the user to build sequences of actions over Data Frames.

Spark GraphX

GraphX is Spark's graph library, which is used to interact with graphs.

2.2 Apache SystemML

Developed by IBM, Apache SystemML is the [precursor] of Spark ML (MLLib), the machine learning libraries of Spark. SystemML is coded in a self-made programming language, Distributed Machine Learning (DML) and it can be used from Spark or Hadoop in a [submit-like execution or in a interactive execution, the one it has been used in this study]

2.2.1 SystemML Structure

2.3 Reproducible research with Python, Scala and Jupyter

Regarding the selection of the development environment, it was important to use a standardized one so that the analysis could be reproduced by anyone. Jupyter notebook is one of the most commonly used, specially in the education and investigation institutions. Jupyter Notebook is easy to use and configure and it can be easily integrated with Spark. It has interactive interpreters for many languages, including python, ruby, scala, R, etc

To configure the same environment as the one used in this study, some steps have to be followed:

2.3.1 Environment setup

Technology	Version
Jupyter Notebook	4.3.1
Python	3.5
Scala	2.11
Toree kernel	0.2.0.dev1
Spark	2.1
SystemML	0.12.0

Table 10: Technologies and versions used

The first step is to setup Jupyter Notebook, either running it in a Docker container or installing it directly on the machine. The docker image can be obtained entering the following command in a shell: *docker pull jupyter/notebook*. Regarding the other option, installing it, the instructions can be found in the following link: *http://jupyter.readthedocs.io/en/latest/install.*

Once Jupyter Notebook is running, the kernels have to be configured. In this study, Scala and Python were used for the data processing, so both kernels were configured. The python kernel is usually configured, as it comes with the IPython kernel installation, if not, follow the guide[][][][][][]. To install the scala kernel, several options can be considered, as there are several implementations of the scala kernel. The chosen one was Scala Toree.

Apart from the kernels, some dependencies must be installed to do the data processing in python, those dependencies are:

- appnope==0.1.0
- bokeh==0.12.4
- botocore==1.5.43
- findspark==1.1.0
- ipykernel==4.5.2
- ipython==5.3.0
- ipython-genutils==0.1.0
- matplotlib==2.0.0
- notebook==4.3.1
- numpy==1.12.0
- pandas==0.19.2
- py4j==0.10.4
- pyparsing==2.2.0
- python-dateutil==2.6.0
- scipy==0.18.1
- systemml==0.14.0
- toree==0.2.0.dev1
- traitlets==4.3.2

For the python kernel, the SystemML library has to be installed, instructions for the installation can be found in the Apache SystemML's get started documentation: https://systemml.apache.org/systemml.html

The scala Toree kernel was used just for the preprocessing of the data, so the SystemML installation didn't have to be installed for this kernel.

3 Infrastructure and resources

As commented before, selecting an environment to be used is the first step, and it is a essential part.

3.1 Architecture scheme

The whole study has been executed in a standalone model, with a MacBookPro Retina 2015.

3.2 Configuration

3.3 Workflow

The first step is to clean the data and make the feature selection. One the data is clean, it is used as the input for the models training, both Kaplan-Meier model and Cox Proportional Hazards Model.

3.3.1 Data cleaning

The data provided by StackOverflow comes in a reasonably clean xml set of files, which were previously described, on section 1.2.

4 Results

4.1 Used data

4.2 Kaplan-Meier model

4.3 Cox Proportional Hazards Model

5 Conclusions

5.1 Most important results

5.2 Lessons learnt