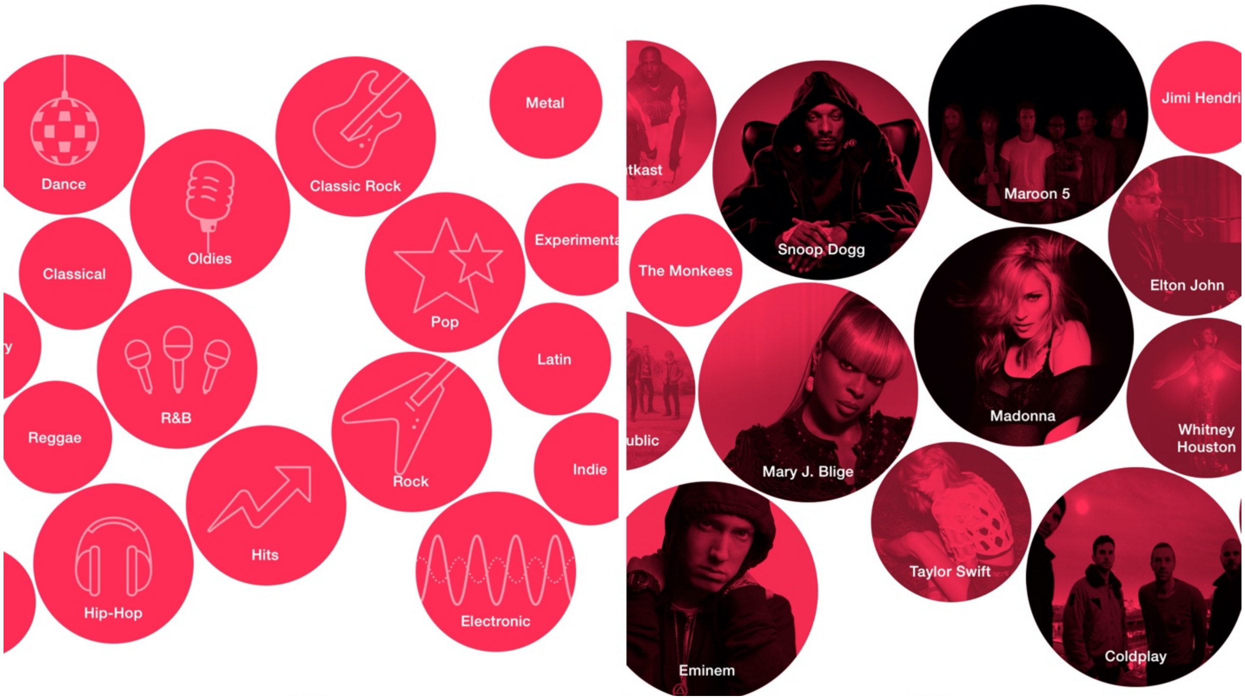
CAPSTONE PROJECT

ADVANCED DATA SCIENCE

To build an artist recommendation system

By Le Mai Minh Phong

Mar 2020



Architectural Components Overview

I. Data Source

1. Definition

Understanding data is one of the most important part when designing any machine learning algorithm. In the notebook, I will use a data set published by Audioscrobbler - a music recommendation system for last.fm. Audioscrobbler is also one of the first internet streaming radio sites, founded in 2002. It provided an open API for “scrobbling” or recording listeners’ plays of artists’ songs. last.fm used this information to build a powerful music recommender engine.

2. Technology choice

- IBM Watson Studio jupiter notebooks, scikit-learn, pandas, matplotlib
- IBM Watson Studio jupiter notebooks, Apache Spark, SparkMLlib, SparkSQL

II. Data Integration

1. Technology Choice

- IBM Watson Studio jupyter notebooks, scikit-learn, pandas, matplotlib
- IBM Watson Studio jupyter notebooks, Apache Spark, SparkMLlib, SparkSQL

2. Apache Spark

Apache Spark is often the primary choice when it comes to cluster grade data processing and machine learning. Apache Spark is often a very flexible choice which also supports writing up integration processes in SQL. But a UI is missing.

- What throughput is required?
  Apache Spark scales linearly, so throughput is just a function of cluster size.
- Which data types must be supported?
  Apache Spark works best with structured data, but binary data is supported as well.
- What source systems must be supported?
  Apache Spark can access a variety of SQL and NoSQL data based as well as file source out of the box. A common data source architecture allows adding capabilities. 3rdparty project add functionality as well.
- What skills are required?
  At least advanced SQL skills are required and some familiarity with either Java, Scala or python.

3. Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

4. IBM Watson Studio

IBM Watson Studio provides tools for data scientists, application developers and subject matter experts to collaboratively and easily work with data to build and train models at scale. It gives you the flexibility to build models where your data resides and deploy anywhere in a hybrid eco-system, so you can operationalize data science faster.

5. Apache SparkSQL

It’s important to notice that data integration is mostly done using ETL tools or plain SQL or a combination of both. ETL tools are very mature technology and an abundance of technologies exist. On the other hand, if streaming analytics is part of the project it is worth to check if one of those technologies fits the requirements since reuse of such a system reduces technology heterogeneity with all its advantages.

III. Data Repository

1. Architectural Decision Guidelines

There exists an extremely huge set of technologies for persisting data. Most of them are relational databases. The second largest group are NoSQL databases and file system (including Cloud Object Store) form the last one. The most important questions to be asked are:

- How does is the impact of storage cost?
- Which data types must be supported?
- How good must point queries (on fixed or dynamic dimensions) be supported?
- How good must range queries (on fixed or dynamic dimensions) be supported?
- How good must full table scans be supported?
- What skills are required?
- What’s the requirement for fault tolerance and backup?
- What are the constant and peak ingestion rates?
- What’s the amount of storage needed?
- How does the growth pattern look like?
- What are the retention policies?

2. Technology choice

- IBM Watson Studio jupyter notebooks, scikit-learn, pandas, matplotlib
- IBM Watson Studio jupyter notebooks, Apache Spark, SparkMLlib, SparkSQL
- Cloud file storage (NoSQL databases: Parquet file storage)

3. Justification

NoSQL databases: the most prominent NoSQL databases like Apache CouchDB, MongoDB, Redis, RethinkDB, ScyllaDB (Cassandra) and InfluxCloud are supported. In this project, I will use Parquet format to store and read the dataset.

- How does is the impact of storage cost? NoSQL databases are usually storage fault-tolerant by default. Therefore, quality requirements on storage is less which brings down storage cost.

- Which data types must be supported? Although NoSQL databases are meant for structured data as well, they usually use JSON as storage format which can be enriched with binary data. Although, lot of binary data attached to JSON document can bring the performance down as well.
- What skills are required? Usually, special query language skills are required for the application developer and if a cloud offering isn't chosen, data base administrator (DBA) skills are needed for the specific database.
- What's the amount of storage needed? RDMBS perform well to around 10-100 TB of data. Cluster setups on NoSQL databases are much more straightforward than on RDBMS. Successful setups with >100 nodes and > 100.000 database reads/writes per second have been reported.
- How does the growth pattern look like? Growth of NoSQL database is not a problem. Volumes can be added during runtime. For shrinking the system might need to be taken offline.

IV. Discovery and Exploration

1. Definition

This component allows for visualization and creation of metrics of data. In various process models, data visualization and exploration are one of the first steps. Similar tasks are also applied in traditional data warehousing and business intelligence. So, for choosing a technology, the following questions should be kept in mind:

- What type of visualizations are needed?
- Are interactive visualizations needed?
- Are coding skills available / required?
- What metrics can be calculated on the data?

2. Technology Choice

- IBM Watson Studio jupyter notebooks, scikit-learn, pandas, matplotlib.
- IBM Watson Studio jupyter notebooks, Apache Spark, SparkMLlib, SparkSQL.

3. Justification

The components mentioned above are all open source and supported in the IBM Cloud. Some of them have overlapping features, some of them have complementary features. This will be made clear by answering the architectural questions

- What type of visualizations are needed? Matplotlib supports the widest range of possible visualizations including bar charts, run charts, histograms, box-plots and scatter plots.
- Are interactive visualizations needed? Whereas matplotlib creates static plots, pixie dust supports interactive ones.
- Are coding skills available / required? Whereas matplotlib needs coding skills. For computing metrics, some code is necessary in Python.
- What metrics can be calculated on the data? Using scikit-learn and pandas, all state-of-the-art metrics are supported.
- Do metrics and visualization need to be shared with business stakeholders? Watson Studio supports sharing of jupyter notebooks, also using a fine-grained user and access management system.

V. Actionable Insights

1. Technology Choice

There exists an abundance of open and closed source technologies (IBM Watson Studio, Jupyter, PySpark, SparkMLlib, SparkSQL …). Here, the most relevant are introduced. Although it holds for other sections as well, decisions made in this section are very prone to change due to the iterative nature of this process model. Therefore, changing or combining multiple technologies is no problem, although decisions let to those changes should be explained and documented.

2. Justification

➢ Python, pandas and scikit-learn

Python is a much cleaner programming language than R and easier to learn therefore. Pandas is the python equivalent to R dataframes supporting relational access to data. Finally, scikit-learn nicely groups all necessary machine learning algorithms together. It's supported in the IBM Cloud via IBM Watson Studio as well.

- What are the available skills regarding programming languages? Python skills are very widely available since python is a clean and easy to learn programming language.
- What are the cost of skills regarding programming languages? Because of python's properties mentioned above, cost of python programming skills is very low
- What are the available skills regarding frameworks? Pandas and scikit-learn are very clean and easy to learn frameworks, therefore skills are widely available.
- What are the cost of skills regarding frameworks? Because of the properties mentioned above, cost of skills is very low.
- Is model interchange required? All scikit-learn models can be (de)serialized. PMML is supported via 3rdparty libraries.
- Is parallel or GPU based training or scoring required? Neither GPU nor scale-out is supported, although scale-up capabilities can be added individually to make use of multiple cores.
- Do algorithms need to be tweaked or new algorithms to be developed? Scikit-learn algorithms are very cleanly implemented. They all stick to the pipelines API making reuse and interchange easy. Linear algebra is handled throughout with the numpy library. So, tweaking and adding algorithms is straightforward

- Python, Apache Spark and SparkMLlib

Although python, pandas and scikit-learn are more widely adopted, the Apache Spark ecosystem is catching up. Especially because of its scaling capabilities. It's supported in the IBM Cloud via IBM Watson Studio as well.

- What are the available skills regarding programming languages? Apache Spark supports python, Java, Scala and R as programming languages.
- What are the cost of skills regarding programming languages? The costs depend on what programming language is used with Python being usually the cheapest.
- What are the available skills regarding frameworks? Apache Spark skills are on high demand and usually not available.
- What are the cost of skills regarding frameworks? Apache Spark skills are on high demand and usually expensive.
- Is model interchange required? All SparkML models can be (de)serialized. PMML is supported via 3rdparty libraries.
- Is parallel or GPU based training or scoring required? All Apache Spark jobs are inherently parallel. But GPU's are only supported through 3rdparty libraries.
- Do algorithms need to be tweaked or new algorithms to be developed? As in Scikit-learn, algorithms are very cleanly implemented. They all stick to the pipelines API making reuse and interchange easy. Linear algebra is handled throughout with built-in Apache Spark libraries. So, tweaking and adding algorithms is straightforward.