Sistemas de Recomendación: Filtros Colaborativos

Integrantes:
Juliana Ochoa Ramírez - Código: 201910048228
Juan Esteban Torres Marulanda - Código: 201910052228
Andrés Franco Zapata - Código: 201910043228

Minería de datos para grandes volúmenes de información
Maestría en ciencias de los datos y analítica
Universidad EAFIT

Docentes:
Tomas Olarte Hernández, Santiago Cortés, Santiago Hernández

18 de abril de 2020

Resumen

El objetivo del presente trabajo es desarrollar un sistema de recomendaciones para un conjunto de datos masivos a partir de técnicas de filtros colaborativos y el respectivo despliegue en la nube a través de *AWS Educate* apalancando los temas de paralelización a través de la herramienta Apache Spark. Para el desarrollo del proyecto se utilizó la base de datos *"Amazon Customer Reviews"* disponible en https://s3.amazonaws.com/amazon-reviews-pds/readme.html, la cual contiene un alto volumen de datos de las evaluaciones y opiniones de los usuarios de Amazon acerca de diferentes productos disponibles para la venta en la plataforma.

El material y notebooks del proyecto se encuentran en el siguiente repositorio github https://github.com/franco18/TrabajoFinal_ST1803_MINERIA.

Introducción

Los sistemas de recomendación son ampliamente conocidos en aplicaciones comerciales donde se busca predecir la evaluación o preferencia de un usuario por un producto en específico, existiendo en la literatura amplia información con respecto a diferentes técnicas para su construcción. El objeto del presente trabajo busca entender los retos computacionales a los cuales se enfrentan los sistemas de recomendación para su implementación en grandes volúmenes de datos. En este contexto el contenido del trabajo final abarca el siguiente orden:

Comenzamos con una descripción breve del marco teórico de los sistemas de recomendación, seguido de un resumen del análisis descriptivo de la base de datos a la cual nos enfrentamos, posteriormente se detalla el proceso de pre-procesamiento de los datos. Luego se presenta la metodología para el montaje del sistema de recomendación a través de Apache Spark, utilizando la técnica de filtros colaborativos de la categoría model-based, mediante un modelo de factores latentes ejecutando el algoritmo alternating least squares (ALS) para el proceso de optimización. Posteriormente se explica en detalle el proceso de modelación para calibrar los hiperparámetros y encontrar el mejor modelo, luego se realiza una mejora al modelo mediante la extensión de factores latentes incluyendo sesgos.

Por último, se explica la infraestructura en AWS Educate para el despliegue y aprovisionamiento de un Cluster donde se corre el modelo construido previamente, y se finaliza con las conclusiones.

En la figura 1 se presenta un esquema del modelo construido para el sistema de recomendación.

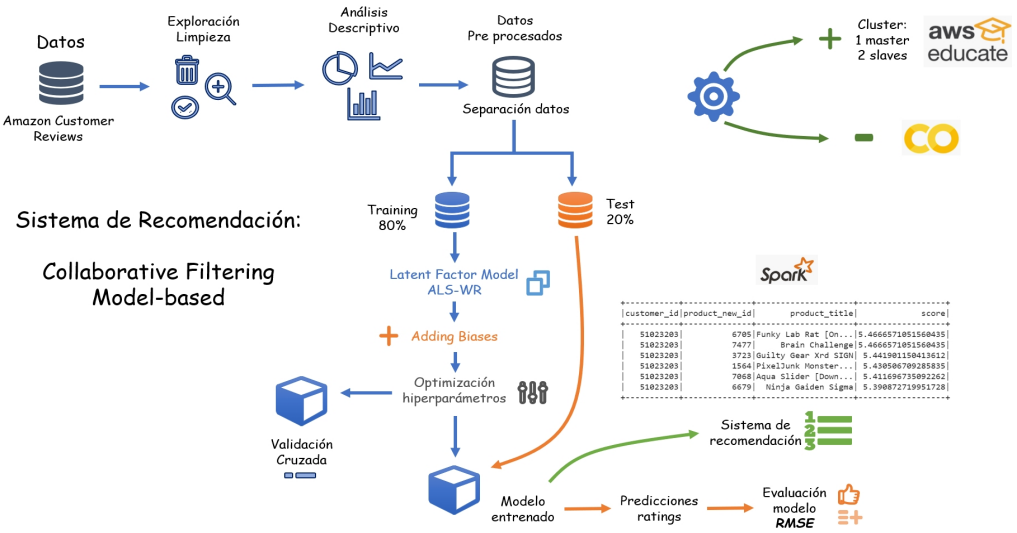


Figura 1: Estructura modelo - Sistema de recomendación

Marco teórico

En la literatura disponible se observa como los sistemas de recomendación se enmarcan principalmente en dos categorías: primero están los *content-based filtering*, los cuales se basan en las características de los productos y la definición de un perfil de las preferencias del usuario (Aggarwal, 2016). En segundo lugar, están los *collaborative filtering*, los cuales solo utilizan información de las calificaciones de los usuarios de los productos, asumiendo que los usuarios que estuvieron de acuerdo en el pasado van a estarlo en el futuro y que tendrán gustos por productos similares a los que les gustaron en el pasado (John S. Breese, 1998).

En la categoría de filtros colaborativos existen dos clases: primero están los *memory-based*, que buscan a través de las similitudes entre usuarios o productos identificar vecindades para dar recomendaciones. Segundo están los *model-based*, los cuales buscan a través de minería de datos y diferentes algoritmos de machine learning predecir las calificaciones de productos que no han sido calificados por los usuarios (Xiaoyuan Su, 2009). Los *model-based* incluyen algoritmos de factores latentes, los cuales realizan reducción de dimensionalidad mediante la factorización de la matriz usuario-producto, a través de métodos de optimización que son resueltos por técnicas iterativas como *gradiente descendente estocástico* y *alternative least aquares (ALS)*, este último muy útil para altos volúmenes de datos gracias a la posibilidad de paralelización del algoritmo (Koren Y, 2009).

Por último, la competición Netflix Prize motivó muchos de los avances en la investigación y desarrollo de nuevos algoritmos de filtros colaborativos (Töscher A, 2008). El presente trabajo se enfoca en el algoritmo *alternative least squares (ALS)* para factores latentes como técnica de filtro colaborativo gracias a la posibilidad de paralelización en *Spark*.

Descripción de los datos

Los datos utilizados para la construcción del modelo de recomendación corresponden a la base de datos *"Amazon Customer Reviews"*, consistente en la información recolectada por Amazon de las evaluaciones y experiencias de los usuarios sobre los productos adquiridos en la plataforma durante 20 años.

Amazon provee una descripción de la información suministrada en la base de datos indicando el contenido de cada columna en relación con los reviews de los usuarios (ver Figura 2), adicionalmente se puede acceder separadamente a la base de datos por categoría de producto.

En promedio cada categoría de producto tiene un tamaño cercano a 1 GB.

Se usaron dos bases de datos para la construcción del modelo de aprendizaje, la primera base de datos es de menor tamaño con el fin de encontrar la estructura correcta de aprendizaje del modelo general, y la segunda para desplegar el modelo con un mayor

volumen de datos en AWS Educate.

```
DATA COLUMNS:
marketplace       - 2 letter country code of the marketplace where the review was written.
customer_id       - Random identifier that can be used to aggregate reviews written by a single author.
review_id         - The unique ID of the review.
product_id        - The unique Product ID the review pertains to. In the multilingual dataset the reviews
                    for the same product in different countries can be grouped by the same product_id.
product_parent    - Random identifier that can be used to aggregate reviews for the same product.
product_title     - Title of the product.
product_category  - Broad product category that can be used to group reviews
                    (also used to group the dataset into coherent parts).
star_rating       - The 1-5 star rating of the review.
helpful_votes     - Number of helpful votes.
total_votes       - Number of total votes the review received.
vine              - Review was written as part of the Vine program.
verified_purchase - The review is on a verified purchase.
review_headline   - The title of the review.
review_body       - The review text.
review_date       - The date the review was written.

DATA FORMAT
Tab ('\t') separated text file, without quote or escape characters.
First line in each file is header; 1 line corresponds to 1 record.
```

Figura 2: Descripción de los datos

- **Base de datos de prototipo a menor escala:** Despliegue en Colab. *amazon_reviews_us_Digital_Video_Games_v*1_00.*tsv*

  Se realizó la primera exploración con la base de datos de los reviews de la categoría *Digital* utilizando Spark para determinar los tipos de datos (ver Figura 3).

```
root
 |-- marketplace: string (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- review_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- product_parent: string (nullable = true)
 |-- product_title: string (nullable = true)
 |-- product_category: string (nullable = true)
 |-- star_rating: string (nullable = true)
 |-- helpful_votes: string (nullable = true)
 |-- total_votes: string (nullable = true)
 |-- vine: string (nullable = true)
 |-- verified_purchase: string (nullable = true)
 |-- review_headline: string (nullable = true)
 |-- review_body: string (nullable = true)
 |-- review_date: string (nullable = true)
```

Figura 3: Digital Reviews Datos cargados en Spark

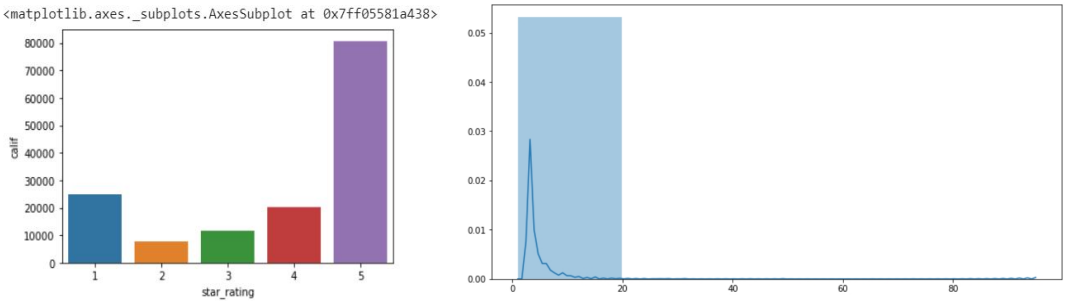
El tamaño de la base de datos estructurada inicial cargada como dataframe es de 160 MB la cual contiene 145.431 filas y 15 columnas (ver Figura 4).

Se evalúa la información por usuario y producto, donde se observa un total de 113.405 clientes y 7.948 productos. La distribución de las calificaciones muestra una concentración importante en la mayor calificación. Adicionalmente como es común en este tipo de información los usuarios califican muy pocos productos, lo cual implica un reto en el modelo considerando los problemas de $low - rank$ y $sparse - data$ (ver figura 5).

customer_id	review_id	product_id	product_title	product_category	star_rating	review_headline	review_body	review_date
21269168	RSH1OZ87OYK92	B013PURRZW	Madden NFL 16 - X...	Digital_Video_Games	2	A slight improvem...	I keep buying mad...	2015-08-31
133437	R1WFOQ3N9BO65I	B00F4CEHNK	Xbox Live Gift Card	Digital_Video_Games	5	Five Stars	Awesome	2015-08-31
45765011	R3YOOS71KM5M9	B00DNHLFQA	Command & Conquer...	Digital_Video_Games	5	Hail to the great...	If you are preppi...	2015-08-31
113118	R3R14UATT3OUFU	B004RMK5QG	Playstation Plus ...	Digital_Video_Games	5	Five Stars	Perfect	2015-08-31
22151364	RV2W9SGDNQA2C	B00G9BNLQE	Saints Row IV - E...	Digital_Video_Games	5	Five Stars	Awesome!	2015-08-31

only showing top 5 rows

Figura 4: Digital Reviews base de datos parcial



(a) Ratings

(b) Items calificados por un usuario

Figura 5: Distribuciones de las calificaciones

- **Base de datos de altos volúmenes:** Despliegue en AWS Educate. *amazon_reviews_us_Sports_v*1_00.*tsv*

  Para trabajar en AWS Educate se utilizó la base de datos de la categoría de deportes, la cual al cargarse como dataframe tiene un tamaño de 5.07 GB con 4.850.360 filas y 15 columnas.

  La base de datos contiene 2.818.178 usuarios y 1.046.129 productos. Igual que la primera base de datos se trata de una matriz dispersa.

Pre-procesamiento de los datos

Previo al entrenamiento de los modelos se procede con el pre-procesamiento de los datos, donde solo se selecciona la información necesaria para correr el modelo de filtros colaborativo a través del aprendizaje de factores latentes.

Los campos requeridos de la base de datos original son: *customer id*, *product id*, *star rating*, *product title*.

Posteriormente se realiza el proceso de limpieza con el fin de corregir y eliminar las evaluaciones que no tienen calificación o se encuentran en otro formato.

Igualmente se procede a transformar los datos de usuario, producto y calificación a tipo entero como lo requiere el modelo ALS.

Luego del pre-procesamiento y limpieza de los datos se procede con la implementación en Colab para la base de datos a menor escala, y en AWS Educate para la base de datos de mayor volumen de datos.

Metodología de investigación

Para el desarrollo del proyecto se implementó una técnica de "Collaborative filtering" para sistemas de recomendación, la cual busca a través del rating dado por los usuarios, identificar patrones similares de ratings que permitan estimar predicciones o similitudes para un determinado usuario.

Se aplicó la categoría de tipo "model-based" utilizando las herramientas equipadas en spark para el aprendizaje de factores latentes como es el algoritmo alternating least squares (ALS) para la construcción de un modelo (Koren Y, 2009) que permita la predicción de los ratings no provistos por los usuarios.

Alternating Least Square (ALS), también conocido como algoritmo de matriz de factores, esta implementada en Apache Spark ML y trata de resolver problemas como son los de escalabilidad y esparcimiento en la matriz de los ratings.

Se asume que cada usuario i está asociado con un vector $q_i \in \mathbb{R}$ y cada usuario u está asociado con un vector $p_u \in \mathbb{R}$. donde los vectores representan ya sea al usuario o al ítem en un subespacio de menor dimensionalidad. Por lo anterior puedo representar la matriz de rating de usuarios para cada ítem como $\hat{r}_{ui} = q_i^T p_u$

Mientras el gradiente descendiente es más fácil y más rápido que ALS, ALS es más favorable en los casos que se pueda usar paralelización dado que el algoritmo computa q_i y p_u independientemente de los otros usuarios e ítems, lo que permite la paralelización. El segundo caso que es muy valioso es cuando se usa los sistemas basados en la data implícita, dado que los datos de entrenamiento no se consideran sparse y usar gradiente descendente no sería práctico.

El error que se busca minimizar es el siguiente:

$$\min_{q*,p*} \sum_{(u,i)\in k} (r_u i - q_i^T p_u) + \lambda(||q_i||^2 + ||p_i||^2) \tag{1}$$

Dentro del modelo existen tres hiperparámetros muy importantes:

- **maxIter:** es el número de iteraciones que realizará el algoritmo.
- **Rank:** es el número de factores latentes que usarán en el modelo.
- **regParam:** es el parámetro de regularización λ, el cual es escalado a través del tamaño de la base de datos y con la aproximación Alternating-Least-Squares with Weighted--Regularization (ALS-WR),

Sin embargo, mucha de la variación observada en los valores de los ratings está afectada

con otros ítems o usuarios, conocido como el sesgo. Por ejemplo, en algunas ocasiones se observa calificaciones más altas de rating de algunos usuarios que de otros, y para algunos ítems altos rating que otros y esto da la percepción a las personas para que los perciban mejor o peor que otros.

Lo anterior denota de qué manera se puede representar el rating estimado por una persona para un ítem, donde se asume que la media μ de todos los ítems y los usuarios, más el sesgo del ítem b_i de los ítems, más sesgo del usuario b_u y por último la interacción usuario ítem lo que compone el rating estimado total $\hat{r}_{ui}$. Como se muestra en la siguiente ecuación:

$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$

Por lo anterior ya la función a optimizar cambia por la siguiente:

$$\min_{q*,p*,b*} \sum_{(u,i)\in k} (r_u i - \mu + b_i + b_u - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) \tag{2}$$

Evaluación y Resultados

1. **Base de datos de prototipo a menor escala:**

   Toda la implementación y resultados se encuentra en el Github. A continuación mostramos un pequeño resumen de la información más relevante.

   Como primera medida se particionó el data set en train (80 %) y test (20 %). Con esta información se corre el modelo con los parámetros por defecto que incluye, rank de 10, máximas iteraciones de 10 y regparam 0.1. Lo anterior nos arrojó un RMSE en training de 0.17, y un RMSE en test de 1.82, evidenciando problemas de sobreajuste y varianza del modelo.

   Para realizar una mejora, se procede a calibrar los hiperparámetros para encontrar el mejor modelo con la partición train y test. Para lo anterior se movió los hiperparámetros de rank de 2 a 10 saltando de a 2 y el regparam de 0.1 a 0.5 en saltos de 0.1; el proceso anterior se demoró 514 segundos. El mejor modelo encontrado fue con un rank de 2 y un factor de regularización de 0.4, con el cual el RMSE en training fue de 0.60 mientras en test fue de 1.54 presentando una mejora con respecto al inicial.

   Con respecto al número de iteraciones encontramos que este hiperparámetro se estabiliza rápidamente (ver figura 6). Trabajamos con 20 iteraciones.

   Tomamos el anterior modelo como base para realizar una mejora, específicamente incluyendo los sesgos de la matriz usuario-ítem, para lo cual se calcula el promedio general de la base de datos, la cual para los rating de *digital video games* fue de 3.85, se calcula el promedio por cada usuario y por cada ítem, posteriormente se corre el algoritmo ALS para encontrar los factores del modelo extendido. Se

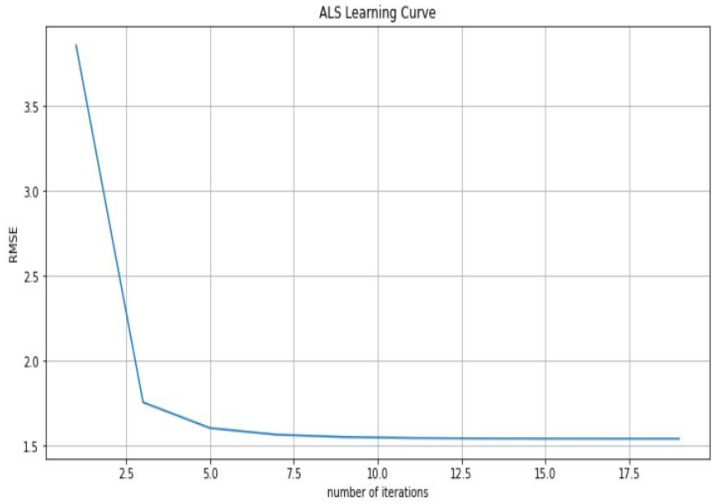


Figura 6: número de iteraciones

corre la misma metodología para optimizar los hiperparámetros y regularizar el modelo. El mejor modelo encontrado tiene 6 factores latentes, un parámetro de regularización de 0.4, un RMSE en training de 0.47 y un RMSE en test de 0.99.

Se obtuvo una mejora de 45 % con respecto al modelo inicial con hiperparámetros sin optimizar, y una mejora de 35 % con respecto al modelo base con hiperparámetros optimizados (ver figura 7).

	RMSE		
	ALS default	**ALS tuning**	**ALS - Biases tuning**
train	0.17	0.6	0.47
test	1.82	1.54	0.99
rank	10	2	6
regParam	0.1	0.4	0.4
maxIter	10	20	20
variación		-15.38%	-35.71%

Figura 7: Resumen resultados

Con el modelo anterior se monta el sistema de recomendación, donde para un usuario me entrega una lista de los productos recomendados (ver figura 8).

customer_id	product_new_id	product_title	score
51023203	6705	Funky Lab Rat [On...	5.4666571051560435
51023203	7477	Brain Challenge	5.4666571051560435
51023203	3723	Guilty Gear Xrd SIGN	5.441901150413612
51023203	1564	PixelJunk Monster...	5.430506709285835
51023203	7068	Aqua Slider [Down...	5.411696735092262
51023203	6679	Ninja Gaiden Sigma	5.390872719951728

Figura 8: Sistema de recomendaciones para un usuario

2. **Base de datos de altos volúmenes:**

   Se corrió el modelo tanto con los parámetros optimizados y agregando la versión extendida con sesgos. Donde se obtuvo una mejora de 45 % con respecto al modelo inicial con hiperparámetros sin utilizar sesgo.(ver figura 9).

	RMSE	
	ALS Tuning	ALS-Biases Tuning
Test	1.5957	0.8414
Rank	2	8
regParam	0.4	0.5
maxIter	20	20
Variacion		-47.3%

Figura 9: Resumen resultados

Con el modelo anterior se monta el sistema de recomendación, donde para un usuario me entrega una lista de los productos recomendados. (ver figura 10)

customer_id	product_new_id	product_title	score
2254854	608102	agm m500 8871 spr...	6.0897112173923045
2254854	853519	Nike Mens Aptare ...	6.06749635766826
2254854	719153	NCAA LSU Tigers N...	6.057541815746644
2254854	928445	McDavid Men's Com...	6.041225670088151
2254854	1010302	Galactic Firebird...	6.033261476267198
2254854	552112	Maxx HD Collegiat...	6.031867383230546

Figura 10: Sistema de recomendaciones para un usuario

Arquitectura

Durante el proceso de implementación y prototipado utilizamos el servicio de google colab utilizando pyspark como herramienta para la construcción del sistema de recomendaciones. Por las limitaciones de procesamiento utilizamos un dataset con los reviews de *digital video games* (145,431 registros).

Luego de construir, entender y estudiar el algoritmo Alternating Least Squares se midió la capacidad que tiene el método y las ventajas de escalabilidad que me ofrece spark como motor de procesamiento distribuido de grandes volúmenes de información, para lograr escalar nuestra implementación con un dataset más grande se utilizó el servicio de infraestructura de AWS Educate, donde se detalla su procesamiento:

Se realizaron las siguientes actividades para el proceso de inicio, configuración y uso de AWS:

1. Para la adecuación del ambiente en nube AWS, creación y configuración de llaves: Key Pair.
   - AWS management console: EC2 dashboard para la creación del cluster desde el cual vamos a trabajar. Launch mode Cluster. S3 folder para el almacenamiento de la información.
   - Configuración de software: EMR-5.29.0 con aplicaciones por defecto Spark 2.4.4, Hadoop 2.8.5, JupyterHub 1.0.0, Hive 2.3.6, Hue 4.4.0, Hue 4.4.0, Zeppelin 0.8.2, HCatalog 2.3.6, Pig 0.17.0 y TensorFlow 1.14.0.
   - Configuración Hardware: Instance type m5.xlarge con 1 master y mínimo 2 core nodes.
   - Acceso y seguridad: Se asignan las llaves y roles por defecto. $EMR\_defaultRole$ y $EMR\_EC2\_DefaultRole$.
   - Se crea un notebook para ejecutar el modelo.
2. Base de datos en S3
   - Se crea el bucket datosrev donde se carga el tsv con los datos necesarios para la construcción del modelo.
   - Se habilitan los permisos de seguridad para poder consumir la información desde cualquiera de los servicios.

La información utilizada durante todo el proceso de escalamiento se almacenó en un bucket de Amazon Simple Storage Service S3 desde donde fue cargada y posteriormente procesada. Se adjunta HTML en el repositorio Github con el notebook ejecutado.

Durante las pruebas en Colab no logramos ejecutar completamente el modelo construido con la base de datos de mayor tamaño *Sports*, mientras en AWS logramos correr la base de datos y también observamos mejor rendimiento sobre la base de datos de menor tamaño *digital video games*, encontrando fundamentos para decir que la estructura es

la adecuada para grandes volúmenes de datos a través del uso de Spark.

Conclusiones y trabajo futuro

La construcción del modelo desde un inicio debe contemplar la escalabilidad, este elemento es determinante para el éxito del modelo en producción.

Si un proyecto busca ser desplegado en altos volúmenes de datos no es estratégico tener un buen prototipo de modelo que no puede ser replicado en datos masivos.

Preferiblemente es mejor trabajar con herramientas que ya optimicen el cómputo en grandes volúmenes de datos. Para nuestro proyecto mientras más queríamos mejorar el modelo requeríamos de mayor computo, lo cual era complejo sin la ayuda de Spark.

La optimización de los hiperparámetros y la regularización jugaron un papel muy importante en el sistema de recomendación final, lo que nos muestra que la paralelización y métodos para hacer eficiente el computo son no negociables en proyectos de altos volúmenes de datos si se quiere obtener una buena generalización.

El método de factores latentes aplicando el algoritmo de ALS-WR nos mostró la importancia de tener en cuenta métodos iterativos que contemplen la paralelización ante problemas de alta dimensionalidad, en nuestro caso se logra dar manejo a los problemas de dimensionalidad y de falta de información de la matriz de ratings (sparse).

Tuvimos mejores avances en la construcción de la metodología cuando trabajamos en un prototipo de menos datos, dado que intentar modificar la estructura tardaba mucho en AWS, por lo cual nos pareció óptimo una vez definimos la arquitectura de aprendizaje escalarlo en AWS con una base de datos más significativa.

Para trabajos futuros consideramos interesante incluir otras técnicas que permitan complementar y validar el sistema de recomendaciones. Específicamente aprovechar otra información de la base de datos como los comentarios escritos, donde algoritmos como locality sensitive minhashing nos permiten igualmente trabajar en altos volúmenes de datos.

Adicionalmente incluir la temporalidad de las evaluaciones puede ser interesante para identificar el comportamiento de las preferencias de los usuarios con el tiempo, ejercicio que también puede ser escalado en altos volúmenes de datos con las técnicas vistas.

Ejecución del plan

Inicialmente en el planteamiento del anteproyecto fuimos muy ambiciosos con los entregables, subestimando el tiempo que nos podría tomar. Por consiguiente, pensábamos que lograríamos combinar dos técnicas para grandes volúmenes de datos incluyendo análisis de textos y procesamiento del lenguaje natural, específicamente con los comentarios de los reviews.

La realidad nos mostró que la construcción y pruebas del modelo en grandes volúmenes de datos toma bastante tiempo, por tanto se tiene que ser muy estratégico para lograr el entregable, por consiguiente renunciamos a la parte de texto y concentramos esfuerzos en mejorar el modelo de factores latentes con sesgos, igualmente recurrimos a Colab para hacer prototipos, dado que en AWS estar configurando el Cluster periódicamente era muy costoso en tiempo, el cual no teníamos dada la cercanía del entregable.

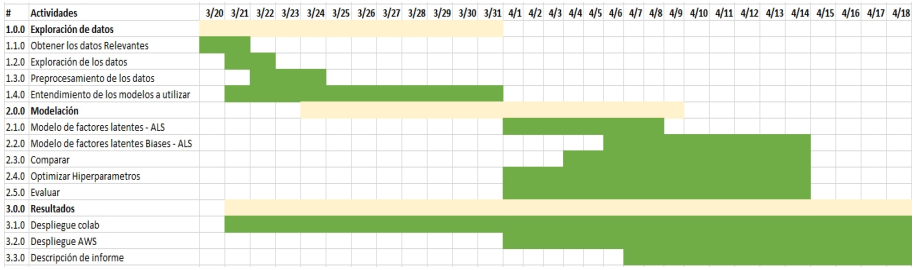


Figura 11: Ejecución proyecto

Implicaciones éticas

Las implicaciones éticas para este caso estarían muy relacionadas con el manejo de los datos que las personas brindan en la plataforma y como se busca que sea algo que las personas sientan y no inducido para dar gusto algún interés en particular o manipular los intereses de los que participan. Acá se busca es como a través de la data encontramos patrones que relacionen el gusto entre las personas (Milano, Taddeo, y Floridi, 2019).

Aspectos legales y comerciales

Es una aplicación que le puede servir a almacenes de cadena para recomendar ofertas según los hábitos de consumo de los clientes para fidelizar al consumidor con nuevos productos y servicios, con el fin de estar en la cotidianidad de los clientes. Es muy importante el tratamiento de los datos y el cumplimiento de la regulación jurídica que establezca cada estado sobre los datos sensibles de clientes, dado que información obtenida sobre las calificaciones que las personas brinden podría exponer la integridad de los clientes.

Referencias

Aggarwal, C. C. (2016). Recommender systems: The textbook. *Springer*.

John S. Breese, D. H. . C. K. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI'98)*.

Koren Y, V. C., Bell R. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*(8), 30-37.

Milano, S., Taddeo, M., y Floridi, L. (2019). Recommender systems and their ethical challenges. *Available at SSRN 3378581*.

Töscher A, J. M. (2008). The bigchaos solution to the netflix prize 2008. *ATT Labs - Research*.

Xiaoyuan Su, T. M. K. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence archive*.