UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 4

Adaptación del algoritmo Bayesian Knowledge Tracing para la estimación del conocimiento latente sobre datos educacionales masivos

Trabajo final presentado en opción al título de

Máster en Informática Avanzada

Autor: Ing. Lisset Salazar Gómez

Tutores: Dr.C. Roxana Cañizares González.

MSc. Angel Alberto Vazquez Sánchez.

La Habana, 2019

DEDICATORIA

A mis padres y mi hijo.

AGRADECIMIENTOS

En primer lugar, deseo agradecer Angel por todo su apoyo y colaboración en la maestría, por siempre estar ahí cuando lo necesitaba, por cada hora y cada minuto que me dedicó, muchas gracias por todo.

A mis padres y hermano por todo su apoyo y por cada palabra de aliento, los amo.

A mi niño querido por darme aquellos momentos de desconcentración que tanto necesitaba y proporcionarme la fuerza para seguir adelante, por nosotros.

A mi otra familia Vazquez, Milagro y Luisi.

A mis amigas (que en mi corazón son mis hermanas Yorge y Michi), por estar siempre a mi lado con sus buenos consejos.

A mi tutora Roxana Cañizares González por cada consejo que me proporcionaba, por su tiempo y su colaboración a lo largo de todo el proceso de la tesis.

A todo el claustro de profesores de la maestría.

A mi oponente y miembros del tribunal por todos sus consejos.

Así como a colegas y amigos que no he mencionado, mi más sincero agradecimiento.

A la Universidad y la Revolución por todas las oportunidades que nos brindan para superarnos profesionalmente.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Lisset Salazar Gómez, con carné de identidad 85072626813, soy la autora principal del trabajo final de maestría “Adaptación del algoritmo Bayesian Knowledge Tracing para la estimación del conocimiento latente sobre datos educacionales masivos”, desarrollada como parte de la Maestría en Informática Avanzada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ___ días del mes de _____ del año 2019.

_____

Ing. Lisset Salazar Gómez

_____ _____

Dr.C. Roxana Cañizares González MSc. Angel Alberto Vazquez Sánchez

RESUMEN

Ante la masividad de los datos que se generan en la educación, se han tenido que cambiar los métodos tradicionales para el descubrimiento de conocimientos. Uno de los algoritmos es el Bayesian Knowledge Tracing (BKT) que permite Estimar Conocimiento Latente (ECL). La ECL no es más que la forma de medir el conocimiento de un estudiante sobre habilidades y conceptos específicos, que es evaluada por sus patrones de corrección sobre esas habilidades. Dicho algoritmo está diseñado para ser utilizado en volúmenes de datos pequeños, afectándose su rendimiento ante la presencia de grandes volúmenes de datos. Para dar solución al problema se presentará como resultado la transformación del algoritmo BKT teniendo en cuenta la programación paralela y distribuida; además, se utilizaron herramientas de procesamiento en paralelo como el marco de trabajo Apache Spark en un entorno de minado. Se valida la propuesta de solución mediante pruebas para medir rendimiento y eficacia, usando métricas como speedup, eficiencia, error medio cuadrático del diferencial de probabilidades y error medio cuadrático del diferencial del área bajo la curva ROC; para las pruebas fueron empleadas bases de datos educacionales.

Palabras clave: Estimación del Conocimiento Latente (ECL), Minería de Datos Educacionales (MDE), Rastreo del Conocimiento Bayesiano (BKT).

ABSTRACT

Faced with the massive amount of data generated in education, the traditional methods of discovering knowledge had to be changed. One of the algorithms is the Bayesian Knowledge Tracing (BKT) that allows to Estimate Latent Knowledge (ELK). The ELK is no more than the way to measure a student's knowledge about specific skills and concepts, which is evaluated by their correction patterns about those skills. This algorithm is designed to be used in small volumes of data, affecting their performance in the presence of large volumes of data. In order to solve the problem, the transformation of the BKT algorithm will be presented as a result, taking into account parallel and distributed programming; in addition, parallel processing tools such as the Apache Spark framework were used in a mining environment. The solution proposal is validated through tests to measure performance and effectiveness, using metrics such as speedup, efficiency, mean quadratic error of the probability differential and mean quadratic error of the differential of the area under the ROC curve; educational databases were used for the tests.

Keywords: Latent Knowledge Estimation (ELK), Educational Data Mining (EDM), Bayesian Knowledge Tracking (BKT).

ÍNDICE GENERAL

ÍNDICE FIGURA

Introducción

La proliferación y el constante desarrollo de plataformas educativas con la utilización de internet, han abierto nuevas posibilidades de análisis debido al gran volumen de datos que se generan y almacenan en servidores. Esta masividad de los datos se debe a las trazas que se van originando a partir de las interacciones con las actividades que proponen las plataformas educativas. Esta información almacenada con datos potenciales de múltiples fuentes procedentes de diferentes formatos y a diferentes niveles de granularidad se caracteriza por ser abundante y accesible. Por lo que, se hace necesario un análisis previo que transforme los datos almacenados en conocimiento útil que permita mejorar el proceso de enseñanza- aprendizaje en dichas plataformas educativas.

La aplicación de la minería de datos en la educación es un campo de investigación interdisciplinario emergente, también conocido como minería de datos [1]. Al análisis y exploración de grandes volúmenes de datos en el contexto educativo es llamado Minería de Datos Educacional (MDE)[2], que tiene como objetivo promover nuevos descubrimientos y avances en el terreno educativo mediante el uso de la información almacenadas en las plataformas educativas. Trata de desarrollar métodos para explorar los tipos únicos de datos que provienen de los entornos educativos. Su objetivo es comprender mejor cómo aprenden los estudiantes e identificar los entornos en los que aprenden para mejorar los resultados educativos y comprender y explicar los fenómenos educativos [1]. La MDE es un proceso utilizado para extraer información útil y patrones de una enorme base de datos educativa. Esta información útil y los patrones pueden ser utilizados para predecir el desempeño de los estudiantes, lo que ayudaría a los educadores a proporcionar un enfoque de enseñanza eficaz. Los estudiantes podrían mejorar sus actividades de aprendizaje, permitiendo a la administración mejorar el rendimiento de los sistemas [3].

En [1],[4], [5] y [6] la definen como una disciplina emergente, preocupada por el desarrollo de métodos para explorar los tipos únicos de datos que provienen de entornos educativos y utilizan esos métodos para comprender mejor a los estudiantes y los entornos en los que aprenden. Otros autores reconocidos en el área [7] lo definen como un campo que explota algoritmos estadísticos, de aprendizaje de máquina y de minería de datos sobre los diferentes tipos de datos educativos. MDE busca utilizar estos repositorios de datos para entender mejor a los estudiantes y al aprendizaje, y a desarrollar enfoques computacionales que combinan datos y teoría para transformar la práctica en beneficio para los aprendices. Un análisis realizado en el área, arroja que el número de publicaciones en la MDE ha aumentado exponencialmente desde el 2005, siendo el mayor pico en el 2014 [8], manteniéndose los avances de investigación hasta la fecha. Específicamente se ha dedicado un gran número de las investigaciones hacia la predicción, el cual permite predecir el rendimiento de un estudiante [7].

Dentro de la MDE, se utilizan diferentes técnicas de minería de datos, de ellas están las predictivas que tienen como objetivo predecir comportamiento de un aspecto de los datos [9]. Las técnicas predictivas se basan esencialmente en regresiones y clasificaciones supervisadas. Estas técnicas se han empleado con éxito para crear modelos de predicción del rendimiento de los estudiantes, en el que se han obtenido resultados prometedores que demuestran cómo determinadas características sociológicas, económicas y educativas de los alumnos pueden afectar en el rendimiento académico [10]. Normalmente estas técnicas se aplican para predecir el rendimiento académico de los alumnos. Por ejemplo, establecer modelos predictores sobre el índice de aprobados de un curso, sobre su nota media, o predecir el tiempo que un estudiante tardará en completar una tarea. También pueden utilizarse para predecir si el alumno posee una determinada competencia sobre una habilidad. A esto último es lo que se conoce como Estimación de Conocimiento Latente (ECL), llamado así porque el conocimiento no es una variable directamente observable [2].

El área de la ECL es de particular importancia dentro de la MDE, debido a que aumentar el conocimiento de los estudiantes es la meta primaria de la educación. Por tanto, si el conocimiento puede ser medido, puedes saber dónde los estás haciendo mejor, puedes informar a los instructores (o cualquier otro interesado en el proceso) sobre el mismo y además puedes realizar decisiones pedagógicas automáticas[11] [9].

Inferir el conocimiento de los estudiantes puede ser útil para varios objetivos, por ejemplo, puede ser una entrada significativa para otros tipos de análisis [7], puede ser útil para decidir cuándo avanzar un estudiante en el currículo o intervenir en otras vías [12], y además puede constituir información útil para los instructores [13].

Existen diferentes métodos que permiten ECL que surgen en el ambiente de la MDE [13], entre los que se encuentran Análisis de los Factores de Rendimiento (PFA, *Performance Factors Analysis*), Teoría de Respuesta al Ítem (IRT, *Item Response Theory*) y el Rastreo del Conocimiento Bayesiano (BKT, *Bayesian Knowledge Tracing*). Cada uno de los algoritmos trata la capacidad de estimar el conocimiento latente de diferentes maneras [9] y [14]. En el caso del IRT, predice la respuesta de una persona ante un ítem determinado, en cambio el PFA no es una expresión directa de la cantidad de habilidad latente, excepto por la probabilidad de responder correctamente y el BKT expresa la probabilidad de que el estudiante domine la habilidad latente, y además muestra la probabilidad de que el estudiante responda correctamente la próxima vez que enfrente un problema donde deba aplicar dicha habilidad. Por lo que se diferencian en la manera de ECL, siendo el BKT el único algoritmo que puede determinar la probabilidad de dominio sobre una habilidad.

El BKT determina en qué medida un estudiante conoce una determinada aptitud o habilidad a partir de su rendimiento pasado con esa habilidad. Proporciona un conocimiento sobre habilidades de un

sistema y predice comportamientos futuros sobre dichas habilidades, en pos de mejorar los sistemas de enseñanza- aprendizaje[15]. Esta información resulta de gran utilidad para determinar en qué medida una plataforma educativa cumple con su objetivo, para informar a los profesores o incluso para realizar acciones correctoras pedagógicas de manera automática [2].

Dicho algoritmo está diseñado para ser utilizado en volúmenes de datos pequeños, afectándose su rendimiento ante la presencia de grandes volúmenes de datos [10], [16], [17] teniendo como limitaciones que:

- Los tiempos de cálculo para estimar conocimiento latente son elevados debido al gran cúmulo de información que generan las plataformas.
- El rendimiento del proceso de inferencia de conocimiento estaría condicionado por el volumen de estos datos, por lo que se requeriría de una gran cantidad de recursos de cómputo.
- Al ser el origen de los datos muy diversos, se hace engorrosa la aplicación del algoritmo debido a que hay que adecuar los datos a los parámetros que necesita el algoritmo ya que está pensado para tratar con datos dicotómicos, o sea, resultados absolutos.
- Al estar los datos dispersos en diferentes bases de datos y no tener el mismo formato trae consigo que no se ejecute correctamente el algoritmo, porque por cada formato por separado habrá que estructurar el algoritmo para su utilización.

Por lo que, la utilización de dicho algoritmo en el contexto de grandes volúmenes de datos en la educación constituye un proceso complejo y engorroso porque actualmente si se toma en cuenta que es necesario aplicar nuevos mecanismos que permitan procesar de forma distribuida un algoritmo que no está diseñado para ello.

A partir de esta situación descrita se identifica el siguiente **problema de investigación:**

¿Cómo lograr que el algoritmo Bayesian Knowledge Tracing mejore los tiempos de cálculo manteniendo la eficacia en los resultados obtenidos en datos educacionales masivos?

Se presenta como **objeto de estudio** algoritmos para la estimación del conocimiento latente en datos educacionales masivo y como **campo de acción** algoritmo de estimación del conocimiento latente de forma paralela y distribuida en datos educacionales masivos.

El **objetivo de la investigación** es adaptar el algoritmo Bayesian Knowledge Tracing utilizando técnicas de programación paralela y distribuida para disminuir los tiempos de ejecución manteniendo la eficacia en la estimación del conocimiento latente en datos educacionales masivos.

.

El objetivo general se desglosa en los siguientes objetivos específicos:

1. Establecer los fundamentos teóricos y metodológicos relacionados con la estimación del conocimiento latente y la minería de datos educativos en el contexto de grandes volúmenes de datos.
2. Adaptar el algoritmo Bayesian Knowledge Tracing para estimar el conocimiento latente en presencia de datos educacionales masivos.
3. Validar la propuesta desarrollada utilizando métodos científicos.

Por lo que se plantea la siguiente **hipótesis investigativa**:

Con la adaptación del algoritmo Bayesian Knowledge Tracing al contexto de grandes volúmenes de datos educativos de forma paralela y distribuida, se mejoran los tiempos de cálculo manteniendo la eficacia en los resultados obtenidos.

Operacionalización de las variables dependientes e independientes

Variable independiente:

- Nivel de distribución y paralelismo.

Variables dependientes:

- Tiempo requerido para estimar el conocimiento latente.
- Eficacia de la estimación del conocimiento latente.

Variable independiente	**Dimensión**	**Indicadores**	**Unidades de medida**
Presencia de distribución y paralelismo	Distribución y paralelismo	Presencia de paralelismo y distribución	Valor dicotómico
Variable dependiente	**Dimensión**	**Indicadores**	**Unidades de medida**
Tiempo de ejecución	Ejecución	Cantidad de segundos (Milisegundos)	Valor numérico continuo
		Speedup	Valor numérico continuo
		Eficiencia	Valor numérico continuo

Eficacia de la estimación del conocimiento latente	Eficacia	Conocimiento latente estimado (Valor de probabilidad calculado)	Valor numérico continuo
		AUC[1]	Valor numérico continuo

Métodos de investigación:

Analítico-Sintético: Permite descomponer el problema de investigación en varios elementos; de esta forma fue más sencillo profundizar en la caracterización de cada elemento, para que luego fueran sintetizados en la solución de la propuesta.

Histórico-Lógico: El empleo de este método permite llevar a cabo la caracterización crítica del proceso de estimación del conocimiento latente, además permitió constatar teóricamente cómo han evolucionado los algoritmos para la estimación del conocimiento latente.

Inductivo-deductivo: Se emplea para identificar la técnica con que se desarrollará el algoritmo.

Experimento: Será utilizado para la verificación de la hipótesis.

Aporte y novedad de la investigación:

A partir de la presente investigación **se contará** con un algoritmo que podrá ser empleado por los sistemas e-learning (LMS, MOOCs), que generen grandes volúmenes de datos para ECL de sus estudiantes.

Lo **novedoso** de la investigación está reflejado en el empleo de técnicas de procesamiento de datos propio del Big Data para adaptar un algoritmo secuencial para que se comporte de una forma distribuida y paralela. Otro aspecto que constituye novedad es el empleo de un sistema distribuido sobre HDFS[2] de Hadoop y Spark para la ejecución del algoritmo diseñado. Contará con el pre-procesamiento de los datos y la visualización de los mismos como pasos del descubrimiento de conocimiento en los datos.

[1] AUC, siglas en inglés para Área debajo de la Curva.
[2] HDFS, Hadoop Distributed File System (HDFS) es un sistema de archivos distribuido, escalable y portátil escrito en Java para el framework Hadoop

Capítulo 1. Preliminares

Elevar el conocimiento es unos de los principales objetivos de la educación, por tanto, si se predice el comportamiento del aprendizaje de los estudiantes, permitiría descubrir información relevante para que los profesores ajusten sus métodos para lograr el éxito. En este capítulo se brinda una panorámica sobre la estimación del conocimiento latente, así como los principales algoritmos empleados para dicha actividad. Se realiza una comparación de los algoritmos para estimar conocimiento latente y se define las herramientas a utilizar para adaptar el algoritmo escogido.

1.1 Fundamentos sobre Estimación del Conocimiento Latente

La ECL es una de las tareas predictivas de la MDE. Este tipo de tarea es también conocida como Inferencia del Conocimiento[3] (IC) o como Seguimiento del Conocimiento[4](SC) [18], [19].

En [20] lo definen como la evaluación del conocimiento o habilidades que posee un estudiante. Para [21] en la ECL el conocimiento de los estudiantes sobre habilidades específicas es evaluado mediante sus patrones de exactitud en esas habilidades (y en ocasiones también otras habilidades). Según [19] el SC modela la comprensión del estudiante mediante una colección de componentes de conocimiento lo que permite estimar el conocimiento latente que posee.

Por lo que, se puede concluir según las definiciones dadas por autores que la ECL es: La técnica utilizada para determinar en qué medida un estudiante domina un(os) conocimiento(s) o habilidad(es) determinada(s) en un momento dado.

Los principales objetivos de la inferencia de conocimiento en este contexto son [19]:

- Medir que sabe un estudiante en un momento específico.
- Medir que componentes de conocimientos relevantes un estudiante domina en un momento específico.

Los modelos usados para la ECL en MDE vienen de dos fuentes fundamentales: las tomadas de los acercamientos psicométricos tradicionales y las investigaciones en el modelado de usuario (en la inteligencia artificial) [9].

Existen un amplio rango de algoritmos para la ECL; los dos más usados actualmente son el Bayesian Knowledge Tracing [22] y Performance Factors Analysis [23] (cada uno con sus respectivas variantes y/o actualizaciones de mejora, por ejemplo [23]–[26]) se ha comprobado que ambos tienen rendimientos equiparables en un número de análisis [16].

[3] En inglés Knowledge Inference
[4] En inglés Knowledge Tracing

Otro método que permite estimar el conocimiento que posee un estudiante es conocido por Item Response Theory [27].

1.1.1 Algoritmos para la estimación del conocimiento latente

En este epígrafe se brindará una breve descripción de 3 métodos principales para la ECL. Se describirá los parámetros que utiliza, el método de ajuste para obtener los parámetros y las fórmulas para estimar el conocimiento latente.

1.1.1.1 Modelo de Rastreo del Conocimiento

El modelo de Rastreo o Seguimiento del Conocimiento (SC) es una técnica de modelado de estudiantes ampliamente usada (Figura 1). Está basada en una red Bayesiana dinámica de dos estados, donde el rendimiento del estudiante es la variable observable y el conocimiento de los estudiantes es la variable latente. El modelo toma el rendimiento del estudiante y lo usa para estimar los niveles de conocimiento del estudiante. Como parte del entrenamiento del modelo dos parámetros de rendimiento son estimados: cometer un desliz y adivinar, los cuales median entre el conocimiento del estudiante y el rendimiento del mismo [28].

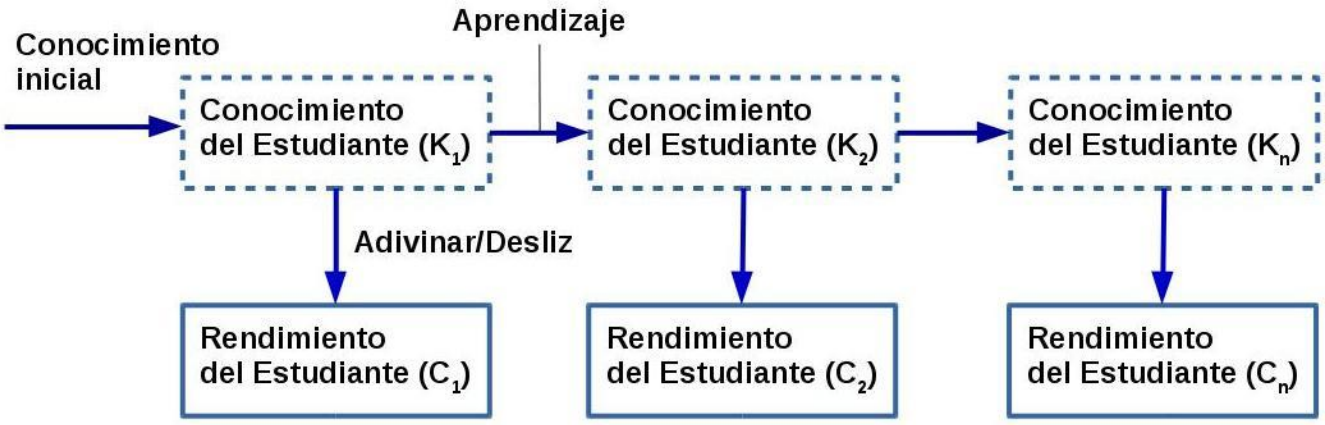


Figura 1 Modelo de Seguimiento del Conocimiento. Fuente: [28].

El algoritmo que implementa este modelo es conocido como Rastreo Bayesiano del Conocimiento (más conocido por Bayesian Knowledge Tracing, o BKT por sus siglas en inglés).

El algoritmo BKT es el método clásico para medir con exactitud las habilidades definidas en un entorno de aprendizaje en línea. Fue propuesto inicialmente por Richard Atkinson en los 60s, pero ha sido más estudiado y articulado por Albert Corbett y John Anderson a partir de 1995 [19].

El objetivo principal es medir en qué medida un estudiante conoce o domina una habilidad o componente de conocimiento (KC, por sus siglas en inglés) en un momento específico, basado en su historial de rendimiento con esa habilidad o KC.

Según [29] este modelo viene en dos formas diferentes. La primera forma en la que aparece BKT es usando un Modelo Oculto de Markov [30] el cual predice la probabilidad de aplicar correctamente una habilidad como la función de un número previo de oportunidades para aplicar esa habilidad y en los parámetros del modelo. La segunda forma es un algoritmo de Seguimiento del Conocimiento (SC) el cuál usa el rendimiento del estudiante en cada oportunidad de aplicar una habilidad y actualiza la probabilidad condicional que el estudiante ha aprendido esa habilidad.

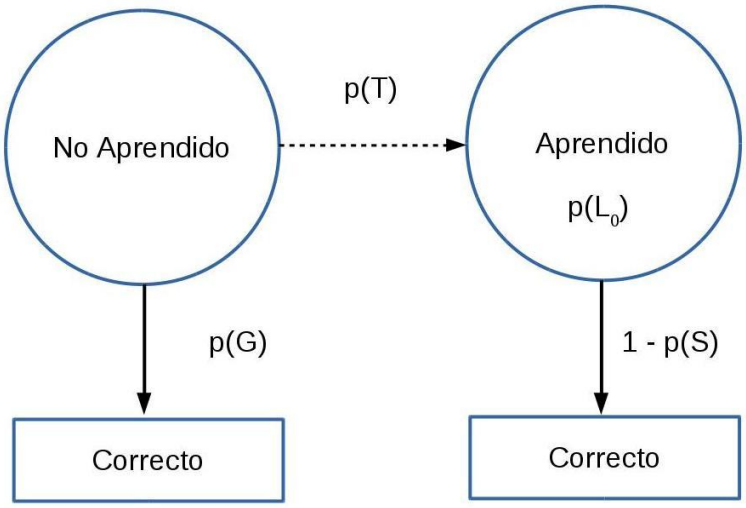


Figura 2 Modelo Oculto de Markov adaptado a BKT. Fuente: [19].

Las asunciones principales para este algoritmo son [19]:

- Cada elemento debe involucrar solamente una característica o habilidad latente.
- Cada habilidad tiene cuatro parámetros. Desde esos parámetros, y el patrón de éxitos y fallos que el estudiante ha tenido en cada habilidad relevante hasta el momento se puede computar el conocimiento latente $P(L_j)$ y la probabilidad de que el estudiante responda correctamente el elemento $P(C_j)$.
- Es modelo de dos estados (cada habilidad está aprendida o no).
- En la solución de problemas el estudiante puede aprender la habilidad en cada oportunidad de aplicarla.
- El estudiante no olvida una habilidad una vez que la ha aprendido.

Modelo Oculto de Markov

El modelo para BKT tiene cuatro parámetros [29]:

- $P(L_0)$ - Es la probabilidad inicial de que el estudiante conoce una habilidad particular.

- $P(G)$ - Es la probabilidad de que adivine (*guess*) correctamente, si el estudiante no conoce la habilidad.
- $P(S)$ - Es la probabilidad de que cometa un desliz (*slip*), si el estudiante conoce la habilidad.
- $P(T)$ - Es la probabilidad de aprender la habilidad si el estudiante no conoce la habilidad. Se asume que esta probabilidad es constante en el tiempo.

Se define el paso j como la j-ésima oportunidad para un estudiante de aplicar una habilidad dada. Sea $P(L_j)$ la probabilidad de que un estudiante domine la habilidad en el paso j. De acuerdo al modelo, $P(L_j)$ puede ser determinado de la previa oportunidad por:

$$P(L_j) = P(L_{j-1}) + P(T)P(1 - P(L_{j-1})) \quad (1)$$

En este modelo la probabilidad de que el estudiante responda correctamente la oportunidad es:

$$P(C_j) = P(G)\left(1 - P(L_j)\right) + \left(1 - P(S)\right)P(L_j) \quad (2)$$

En las ecuaciones (*1*) y (*2*) se define el modelo oculto de Markov, donde $P(L_j)$ es la variable "oculta". Luego de realizar varias transformaciones la expresión (*2*) puede ser reescrita como:

$$P(C_j) = 1 - P(S) - Ae^{-\beta j} \quad (3)$$

Donde $A = (1 - P(S) - P(G))(1 - P(L_0))$ y $\beta = -\log(1 - P(T))$.

Como se puede apreciar en Figura 3 la función que describe $P(C_j)$ es exponencial con tres parámetros, acotada superiormente por $1 - P(S)$.

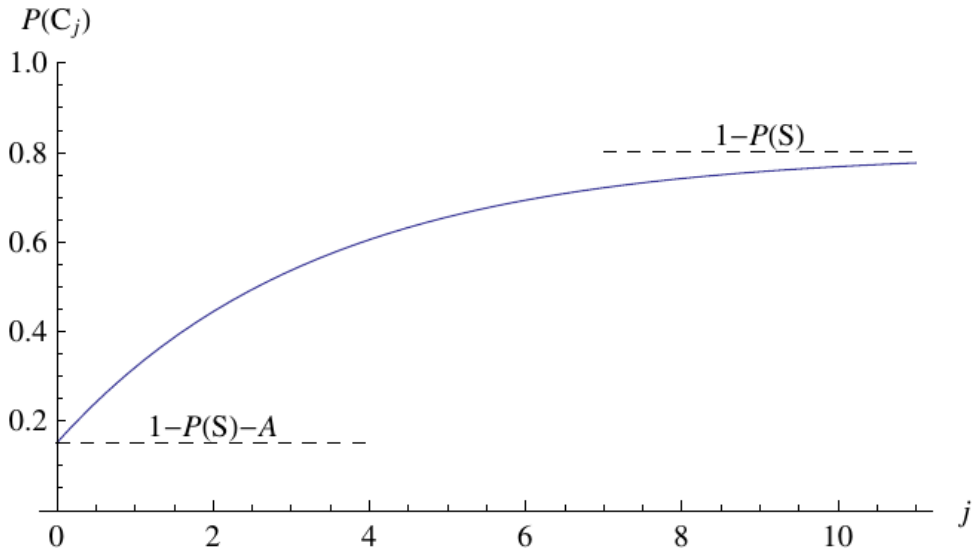


Figura 3 Gráfico de la función $P(C_j)$. Fuente: [29].

Algoritmo BKT

Para predecir $P(L_j)$ para un estudiante individual se puede emplear el algoritmo de SC. En este caso se busca $P(L_j|O_j)$, la probabilidad de que el estudiante ha aprendido la habilidad justamente luego de completar el paso j dado el rendimiento del estudiante O_j en los pasos previos, donde $O_j = \{ o_1, o_2, \ldots, o_j \}$ es el rendimiento del estudiante en las primeras j oportunidades y o_i puede ser correcto o incorrecto [29]. Estas probabilidades condicionales responden a la recurrencia:

$$P(L_{j-1}|O_j) = \frac{P(L_{j-1}|O_{j-1})(1-P(S))}{P(L_{j-1}|O_{j-1})(1-P(S))+[1-P(L_{j-1}|O_{j-1})]P(G)}, O_j = correcto \quad (4)$$

$$P(L_{j-1}|O_j) = \frac{P(L_{j-1}|O_{j-1})P(S)}{P(L_{j-1}|O_{j-1})P(S)+[1-P(L_{j-1}|O_{j-1})](1-P(G))}, O_j = incorrecto \quad (5)$$

$$P(L_j|O_j) = P(L_{j-1}|O_j) + [1 - P(L_{j-1}|O_j)]P(T) \quad (6)$$

Existen una variedad de algoritmos que permiten realizar el ajuste de los parámetros necesarios para realizar la estimación del conocimiento latente del algoritmo BKT. Entre estos algoritmos se encuentran:

- Maximizar la Expectación (EM, por sus siglas en inglés)[31],[29].
- Fuerza Bruta (BF, por sus siglas en inglés) [25], [32].
- Probabilidad Empírica (EP, por sus siglas en inglés) [33].
- Ajuste Aleatorio (RF, por sus siglas en inglés).
- Recocido Simulado (SAF, por sus siglas en inglés)[34].
- Baum-Welch (BW, por sus siglas en inglés)[35].

En **EM** se procesa el rendimiento de un estudiante como una pieza de evidencia con un orden temporal, y usa esta evidencia para la etapa de expectativa donde se calcula la probabilidad esperada; entonces se computa los parámetros que maximizan esta probabilidad y se ejecutan estos dos pasos de forma iterativa hasta que finalmente encuentra el mejor ajuste para estos parámetros. Este método no garantiza encontrar un global, dado que puede quedar en un máximo local [28].

En el enfoque por **BF**, por otro lado, intenta minimizar la Suma del Error Cuadrado (SSE). Como originalmente los parámetros para BKT son continuos, no existe forma de componer un espacio de búsqueda finito, lo que es necesario para realizar una búsqueda exhaustiva. Para mitigar esta dificultad se limita el espacio de búsqueda considerando solamente dos lugares decimales de precisión, y de esta forma se reduce el espacio de búsqueda de infinito a 99^4 conjuntos de parámetros posibles para cada habilidad. En el algoritmo cada parámetro comienza en 0.01 y se incrementa en 0.01 en cada

iteración. Finalmente para cada iteración se encuentra el conjunto de parámetros que resulte en la menor SSE [28].

En **EP** se ajustan los parámetros al estimar el punto más probable en que cada alumno aprendió la habilidad, lo que evita problemas presentes en otros métodos de ajuste como son: múltiples conjuntos de parámetros muy diferentes que predicen los datos igualmente bien (identificar), mínimos locales, parámetros degenerados y costo computacional durante la adaptación. Este método logra una precisión predictiva similar mientras evita los problemas anteriores. Es un proceso de dos pasos que implica anotar los datos de rendimiento con conocimiento y luego usar esa información para calcular los parámetros de BKT [33].

El algoritmo **SAF** es una variación del algoritmo de Fuerza Bruta propuesto por Baker en [25], en esta versión se implementa el ajuste a través del algoritmo de Recocido Simulado. En este algoritmo los parámetros son inicializados con valores aleatorios. Un valor inicial de RMSE[5] es calculado al usar ese conjunto de parámetros para predecir el rendimiento del estudiante vía BKT. Para cada paso en el proceso, uno al azar de estos valores se incrementa o disminuye en una cantidad aleatoria entre 0 y 0.05, y se calcula un nuevo valor de RMSE [34].

En el método **BW** es una variación del método EM, que permite estimar los parámetros de un modelo que hacen máxima la probabilidad de una secuencia de observables. Del método se calcula 3 matrices fundamentales, una que es la distribución de probabilidad de la transición del estado, el símbolo de la distribución – observación de la probabilidad del estado y la distribución del estado inicial. A partir de los cálculos entonces se obtiene los 4 parámetros fundamentales del algoritmo BKT [35].

El **RF** consiste en otorgar de forma aleatoria un valor entre 0 y 1 a cada uno de las cuatro variables que componen el modelo BKT.

Para decidir cuál de los algoritmos es el más adecuado para efectuar el ajuste de parámetros, se realizó un experimento donde se ejecutaron estos algoritmos para diferentes conjuntos de datos extraídos de "LearnLab DataShop[6]". Se calculó el tiempo medio para el ajuste, así como el área debajo de la curva ROC [36].

En dicho experimento se utilizaron 4 bases de datos que fueron Chinese_tonestudy, FractionStudy2012, Handwriting2/Examples Spring 2007 y Geometry Area (1996-97), donde se obtuvo los siguientes resultados:

---

[5] RMSE – Raíz del error medio cuadrático.

6 https://pslcdatashop.web.cmu.edu, repositorio de datos para investigadores en la ciencia de aprendizaje.

Como se puede observar en la Tabla 1, los mejores algoritmos respecto al AUC son el de Fuerza Bruta y el de Probabilidades Empíricas, sin embargo la diferencia en cuanto tiempo de ejecución entre estos algoritmos es evidente.

Tabla 1 Resultados del experimento sobre ajuste de parámetros.

Método	AUC	Tiempo
Random Fitting	0,418312214	1,2368
Simulated Annealing Fitting	0,922519486	86,181
Empirical Probabilities	**0,999978001**	**1,4012**
Baum Welch	3,71065E-05	1,5892
Brute Force	1	308,0394
Expectation Maximization	0,809759778	1,5706

Como se puede apreciar en la tabla anterior, el algoritmo Probabilidades Empíricas muestra los mejores resultados para los experimentos realizados. Es el mejor respecto al AUC y el tiempo de ejecución es solamente superado por el algoritmo de Ajuste Aleatorio, pero este último, muestra peores resultados respecto al estadístico utilizado.

Por tanto, el método que se utilizará para realizar el ajuste de parámetros al algoritmo BKT es Probabilidades Empíricas por dar los mejores resultados en cuanto al AUC y tiempo de respuesta.

1.1.1.2 Análisis de los factores de rendimiento

Otro método que permite ECL es el análisis de los factores de aprendizaje. Este modelo constituye una alternativa a BKT, trabajando en dirección de algunas limitaciones de este algoritmo; pero no alcanza todas las buenas características de BKT. Fue propuesto en 2009 por Pavlik, Cen y Koedinger en [37]. Surge a partir de la reconfiguración del algoritmo de Análisis de los factores de aprendizaje (LFA, por sus siglas en inglés) y es mayormente conocido por el nombre en inglés *Performance Factors Analysis* (PFA) [28].

Mide cuanta habilidad latente tiene un estudiante mientras está aprendiendo. Pero la expresa en términos de exactitud la próxima vez que la habilidad es encontrada. No es una expresión directa de la cantidad de habilidad latente, excepto por esta probabilidad de responder correctamente [19].

Este modelo ha sido más ampliamente investigado como una solución al problema de múltiples rendimientos en los componentes de conocimiento. Modelos como este han sido empleados frecuentemente para predecir la duración de cada acción, dado que produce un valor estimado real de la fuerza para cada KC [37].

Este algoritmo puede ser usado en contextos donde [19]:

- Es necesario estimar el conocimiento de un estudiante en un tópico X.
- Donde los estudiantes pueden aprender en cada elemento (debido a la ayuda, a la retroalimentación, etc.).
- Cuando existen una secuencia de elementos que son puntuados de forma dicotómica (por ejemplo, el estudiante adquiere una nota de 0 o 1 en cada elemento).

Este modelo asume que el "rendimiento" es indicativo del aprendizaje de los estudiantes por dos razones. La primera, respuestas correctas son un fuerte indicativo de que la fortaleza actual ya es alta, por tanto, dar una respuesta correcta va a ayudar al modelo a incrementar la fuerza estimada. Y en segundo lugar, las respuestas correctas puede llevar a mayor aprendizaje que las respuestas incorrectas [37]. La sensibilidad a las respuestas incorrectas permite actuar como indicador y estimar el aprendizaje en el sentido inverso.

Los parámetros principales de este método son [19], [37] :

- Cada elemento puede tener múltiples habilidades latentes o componentes de conocimiento (representado por KC).
- Cada habilidad tiene un índice de aprendizaje exitoso γ y un índice de fallo en el aprendizaje ρ.
- El parámetro β indica la dificultad para los KCs.

A partir de estos parámetros, y del número de éxitos y fallos que el estudiante ha tenido en cada habilidad relevante hasta el momento, se puede computar la probabilidad $P(m)$ de que el estudiante responderá correctamente el ítem:

$$m(i, j \epsilon KCs, s, f) = \beta + \sum_{j \in KCs} \gamma_j s_{i,j} + \rho_j f_{i,j} \quad (7)$$

$$P(m) = \frac{1}{1 + e^{-m}} \quad (8)$$

Donde s es el seguimiento de éxitos anteriores para el KC para el estudiante, y f es el seguimiento de los fallos anteriores para el KC.

El parámetro β se propone en [37] para tres elementos diferentes:

- Ítem.
- Tipos de Ítem.
- Habilidades.

Los métodos usados para ajustar los parámetros son EM y fuerza bruta [28]. Pero a diferencia de BKT la familia de modelos logísticos de descomposición del aprendizaje está basado en la forma de la regresión logística estándar, por lo que se asegura que el procedimiento ajuste alcance un máximo global, por tanto, existe un único ajuste óptimo de los parámetros.

1.1.1.3 Teoría de respuesta al ítem

La Teoría de Respuesta al Ítem (más comúnmente conocido por *Item Response Theory*, IRT) es una de las teorías de prueba que pueden ser aplicada para evaluar datos para describir como estimar, inferir y predecir una característica, rasgo o habilidad particular que una persona pueda hacer a partir de las respuestas a los ítems de un examen [27].

Los modelos IRT involucran una clase de modelos matemáticos que pueden ser usados para predecir las respuestas de una persona para cada ítem de un examen mediante el uso de las características de la persona y las características de los ítems. Las características de la persona refleja uno o más constructos latentes siendo medidos por un instrumento (examen, encuesta o escala), y el nivel o estatus particular de la persona en cada constructo siendo medido es asumido para afectar el desempeño de la prueba [27].

El objetivo principal de IRT es saber en qué medida una persona posee alguna característica latente. Y es típicamente usada para medir el conocimiento actual de un estudiante sobre un tópico determinado [19].

Para predecir o modelar las respuestas a los ítems, los modelos IRT consisten en uno o más parámetros latentes de la persona y uno o más parámetros del ítem. Estos parámetros, en conjunción con una función matemática, como la distribución normal o función logística (caso general de la función sigmoide), puede ser usada para modelar la probabilidad de una respuesta u_j al ítem j. Esta respuesta puede ser, por ejemplo, *correcta* o *incorrecta*, o la aceptación de una opción particular de respuesta. Una expresión general para la clase de modelos IRT es la siguiente:

$$P_j(U_j|\omega_j, \theta) \tag{9}$$

Donde θ es un vector de características de la persona reflejando diferentes atributos siendo medidos, y ω_j es un vector de parámetros de rasgos característicos para el ítem j que se asume que afecta el rendimiento.

El modelo IRT para un ítem puntuado de forma dicotómica (0,1; incorrecto, correcto; No, Si) es mostrado típicamente usando una curva de respuesta al ítem o *curva característica del ítem* (ICC, por sus siglas en inglés). En la Figura 4 se muestra una ICC para un ítem puntuado de forma dicotómica, donde $P(U_j = 1|\omega_j, \theta)$ es la función de respuesta del ítem con una habilidad latente asumida para

determinar el rendimiento del ítem. En la Figura 4, el eje x corresponde con la habilidad latente (θ) siendo medida, y el eje y se corresponde con la probabilidad de contestar correctamente el ítem. Como se puede apreciar la función matemática refleja una relación monótona creciente entre la probabilidad de respuesta y el estatus o nivel de la habilidad siendo medida [27].

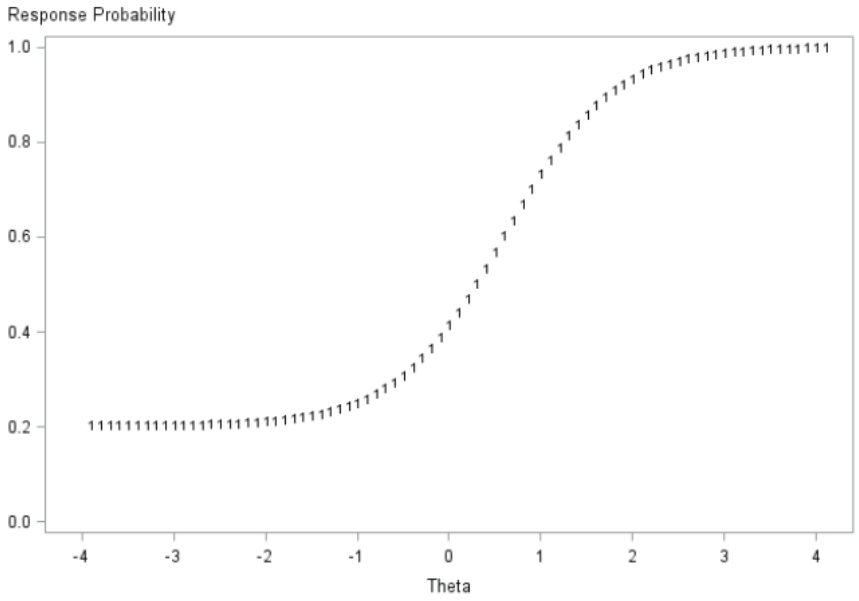


Figura 4 Ejemplo de una ICC para un ítem puntuado de forma dicotómica. Fuente: [27].

En la ICC, el conjunto de parámetros del ítem (ω_j) para el modelo específico de IRT determina la forma de la función. El parámetro pendiente *(slope)* para ICC es usado para describir cuán rápido la probabilidad de respuesta cambia (*sensibilidad*) como una función de cambios en el rasgo latente θ (Figura 5). Un segundo parámetro es el umbral (*threshold*), está en la misma escala que la habilidad latente θ y describe donde la función de respuesta está centrada en la escala de θ (Figura 6). La asíntota inferior (*lower asymptote*) puede también ser incluida para modelar la probabilidad de la respuesta de los estudiantes con bajo θ. En el contexto educacional, estos tres parámetros (pendiente, umbral y asíntota inferior) son a menudo referidos como parámetros de discriminación, dificultad y adivinación [27].

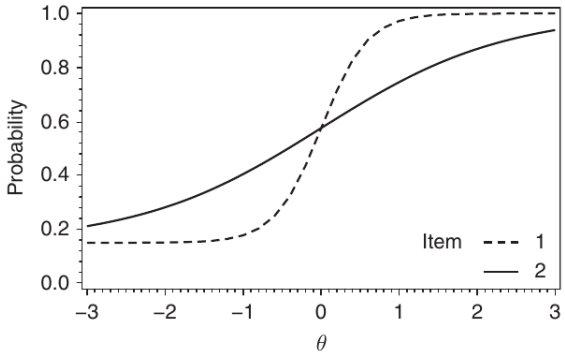


Figura 5 Ítems con diferentes parámetro a. El ítem 1 es más discriminante que 2. Fuente: [38].

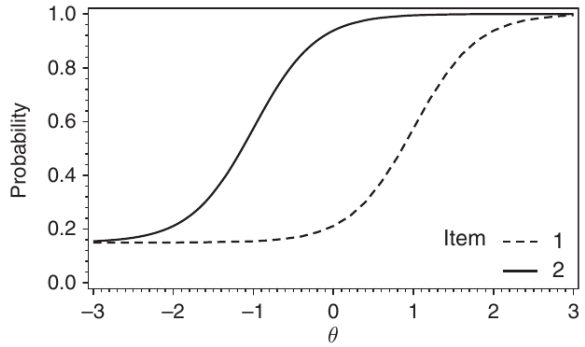


Figura 6 Ítems con diferentes parámetros b. El ítem 1 es más difícil que el 2. Fuente: [38].

Para respuestas puntuadas de forma dicotómica generalmente se utilizan tres modelos IRT: Modelos de 1, 2 y 3 parámetros. En la forma general para estos tres modelos, la probabilidad de que un estudiante reciba una respuesta (1 = correcta como contraria a 0 = Incorrecta) para el ítem j-ésimo es comúnmente modelado por una función logística:

$$P_j(\theta) = P_j\left(U_j \middle| \omega_j, \theta\right) = \Psi\left(z_j\right) = c_j + (1 - c_j)\frac{e^{z_j}}{1 + e^{z_j}} \quad (10)$$

Donde $\Psi(z_j)$ es la función logística acumulativa y $z_j = Da_j(\theta - b_j)$ es una desviación logística; D es la constante de escalado (comúnmente se utiliza los valores 1 y 1.702 de manera indistinta); θ referencia al parámetro de la persona (la habilidad examinada); y ω_j referencia el vector de posibles parámetros de ítem para el ítem j, y es la asíntota inferior para el parámetro j [27], [38].

El modelo de 3 parámetros (3PL) incluye los tres parámetros del ítem (a_j, b_j y c_j) y se aplica la fórmula expresada en (*10*) para calcular la probabilidad de responder correctamente el ítem [19], [27], [38].

El ajuste de parámetros para este algoritmo puede ser logrado mediante la Maximización de la Expectación [19].

1.1.2 Comparación entre los diferentes algoritmos

Para la evaluación de los algoritmos se decide basarse en los criterios siguientes atendiendo a sus características [9], [14], [17]:

- Capacidad de estimar el conocimiento latente.
- Cantidad de habilidades por ítem.
- Métodos de ajuste de parámetros del modelo.
- Métricas para la medición de resultados.

En la Tabla 2 se muestra un resumen comparativo de los tres algoritmos estudiados respecto a los criterios escogidos. Como se puede apreciar todos estos algoritmos comparten características comunes como son el uso de métricas similares como Área debajo de la curva (AUC), Error Cuadrático Medio (RMSE).

Un elemento que los diferencia concretamente es la capacidad de estimación del conocimiento latente, siendo únicamente el algoritmo BKT quien calcula un estimado de la habilidad latente, así como la probabilidad de responder correctamente al próximo ítem con la presencia de dicha habilidad; el resto solamente calcula un estimado de la posibilidad de que se conteste correctamente el ítem asumiendo que es una medida de la posesión de la habilidad asociada con ese ítem.

Por otro lado, el algoritmo BKT posee un método de ajuste de parámetros (Probabilidad Empírica) que en [33] demostró tener probabilidad predictiva similar a EM, pero que evita los modelos degenerados y trabaja considerablemente más rápido que el EM.

Tabla 2 Comparativa entre los diferentes algoritmos: Fuente: Elaboración propia.

Criterio	**Bayesian Knowledge Tracing**	**Performance Factors Analysis**	**Item Response Theory**
Capacidad de estimar el conocimiento latente	Expresa la probabilidad de que el estudiante domine la habilidad latente, y además muestra la probabilidad de que el estudiante responda correctamente la próxima vez que enfrente un problema donde deba aplicar dicha habilidad.	No es una expresión directa de la cantidad de habilidad latente, excepto por la probabilidad de responder correctamente.	Predice la respuesta de una persona ante un ítem determinado.
Cantidad de habilidades por ítem	Una habilidad por ítem.	Múltiples habilidades por ítem.	Una habilidad por ítem.
Métodos de ajuste de parámetros del modelo	• Maximización de la expectación. • Fuerza Bruta. • Probabilidad Empírica. • Ajuste Aleatorio. • Recocido Simulado. • Baum-Welch.	• Maximización de la expectación. • Fuerza Bruta.	• Maximización de la expectación. • Estimadores bayesianos.
Métricas para la medición de resultados	• Media del error absoluto (MAE). • Error Cuadrático Medio (RMSE). • Área debajo de la Curva (AUC).	• Error Cuadrático Medio (RMSE). • Área debajo de la curva (AUC).	• Curva Característica del Ítem. • Error Cuadrático Medio (RMSE). • Área debajo de la curva (AUC).
Observaciones		Existe un único ajuste óptimo de los parámetros[28].	No es utilizado a menudo en el aprendizaje en línea donde el conocimiento de los estudiantes está cambiando a medida que es evaluado[19].

En [39] se muestra que no existen diferencias significativas respecto a la exactitud de las predicciones entre los modelos BKT y PFA.

Por otro lado, como se muestra en las observaciones del cuadro comparativo el algoritmo IRT no es utilizado a menudo en el aprendizaje en línea [19], el cual constituye una de las tendencias actuales en los sistemas e-learning.

Por lo todo lo anterior, se evidencia que el algoritmo BKT es el único algoritmo que mide en sí, la probabilidad de que un estudiante domina una habilidad.

1.1.3 Características del algoritmo Bayesian Knowledge Tracing

Como se detalló en el epígrafe 1.1.1 el algoritmo BKT está regido por la expresiones (4) y *(5)*. Estas expresiones en su forma funcional se pueden apreciar en la Figura 7. En la misma la línea puntuada representa la frontera entre soluciones crecientes y decrecientes. Desde que la curva "step j correct" está por encima de la línea puntuada, la ecuación (4) provoca que la secuencia converge al punto fijo en 1. De igual manera, desde que la curva "step j incorrect" está por debajo de la línea, la ecuación *(5)* causa que la secuencia $\{P(L_j|O_j)\}$ converge al punto fijo menor.

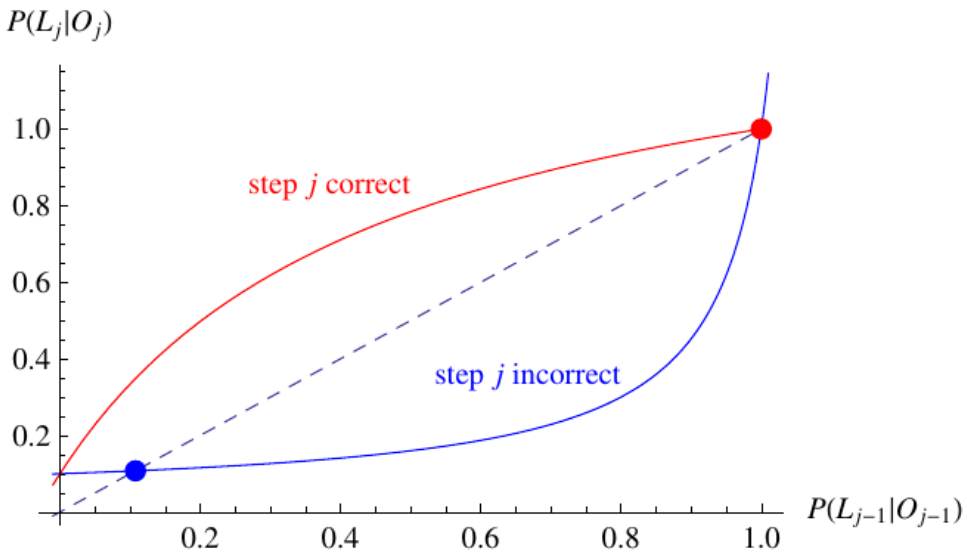


Figura 7 Gráfico de la relación de recurrencia para el algoritmo BKT, de las ecuaciones (4) y (5). Los parámetros de este modelo son P(S)=0.5, P(G)=0.3, P(T)=0.1 y P(L0) = 0.36. Fuente: [29].

Esta relación recurrencia (ecuaciones (4) y *(5)*) no puede ser resuelta analíticamente, se puede aprender mucho de sus soluciones mediante la realización de un análisis de punto fijo [29]. El objetivo de un análisis de punto fijo es determinar la calidad del comportamiento de la secuencia $\{P(L_j|O_j)\}_{j=0}^{n}$ como una función de j. Si $P(L_j|O_j) > P(L_{j-1}|O_{j-1})$, entonces se puede decir que crece con j. Por otro lado, si $P(L_j|O_j) < P(L_{j-1}|O_{j-1})$, entonces se puede decir que decrece con j. Por lo que, la frontera

entre soluciones crecientes y decrecientes es $P(L_j|O_j) = P(L_{j-1}|O_{j-1})$, mostrada como la línea punteada en la Figura 7. Un punto fijo es un valor de $P(L_j|O_j)$ tal que la relación de recurrencia obedece $P(L_j|O_j) = P(L_{j-1}|O_{j-1})$.

En [29] también se definen dos tipos de puntos fijos :

- *Punto fijo estable*: Si $P(L_j|O_j)$ está cerca del punto fijo, entonces $P(L_j|O_j)$ converge al punto fijo mientras j se incrementa.
- *Punto fijo inestable*: Si $P(L_j|O_j)$ está cerca del punto fijo, entonces $P(L_j|O_j)$ se aleja del punto fijo al incrementarse j.

Aplicando estas ideas en las ecuaciones (4) y *(5)*. En la ecuación (4) se encuentra un punto fijo estable en 1 y un punto fijo inestable en:

$$-\frac{P(G)P(T)}{1-P(G)-P(S)} \tag{11}$$

Similarmente, la ecuación *(5)* tiene un punto fijo inestable en 1 y un punto fijo estable en:

$$\frac{(1-P(G))P(T)}{1-P(S)-P(G)} \tag{12}$$

Para que $P(L_j|O_j)$ permanezca en el intervalo [0,1] para cualquier valor inicial de $P(L_0) \in [0,1]$ y cualquier secuencia de pasos O_j correctos/incorrectos, necesitamos que el punto fijo (*12*) esté en el intervalo [0,1] y el punto inestable (*11*) se mantenga negativo. Esto nos da las siguientes restricciones en los valores permitidos para el modelo de los parámetros [23]:

$$P(G) + P(S) < 1 \tag{13}$$

$$0 < P(T) < 1 - \frac{P(S)}{1-P(G)} \tag{14}$$

Otras restricciones propuestas para este modelo son:

- $P(S) < 0.5$ y $P(G) < 0.5$ [25]
- $P(S) < 0.1$ y $P(G) < 0.3$ [22]

La idea conceptual detrás de este algoritmo es:

- Dominar una habilidad generalmente conlleva a un rendimiento correcto.
- Un buen rendimiento implica que un estudiante domina la habilidad relevante.

Por tanto, mediante la búsqueda de dónde el estudiante muestra un buen rendimiento se puede inferir que domina la habilidad [19].

1.2 Descubrimiento de Conocimiento en Bases de Datos

La minería de datos es una disciplina que ha surgido principalmente por el crecimiento de grandes bases de datos. El estímulo motivador básico detrás de la minería de datos es encontrar información de valor para la obtención de conocimiento, pero esta información está oculta dentro de la masa de datos sin interés y que aún no ha sido descubierta. Es decir, se busca información sorprendente, novedosa, inesperada o valiosa, y el objetivo es extraer esta información.

La minería de datos es la extracción, o "minería", del conocimiento a partir de una gran cantidad de datos. Por lo tanto, es un campo interdisciplinario que emplea el uso de herramientas de análisis de modelos estadísticos, algoritmos matemáticos y métodos de aprendizaje de máquinas para descubrir patrones y relaciones válidos y desconocidos en grandes conjuntos de datos. En [40], se definen como "un paso en el proceso de descubrimiento de conocimiento conocido como KDD, que consiste en aplicar análisis de datos y algoritmos de descubrimiento. Si bien los términos minería de datos y descubrimiento de conocimiento en bases de datos son usados como sinónimos, el término KDD describe el proceso completo de extracción de conocimiento a partir de los datos. Mientras que MD, se refiere exclusivamente a una fase del descubrimiento de un proceso general KDD [41].

El KDD es un proceso iterativo de extracción de información de los datos que consta de los siguientes pasos (Figura 8) [42]:

- **Limpieza de datos:** Este paso comienza con la carga de los datos que se necesita procesar. A partir de ahí, incluye eliminar ruido y datos inconsistentes. Incluye limpieza, transformación, integración y reducción de datos. Este paso es de gran importancia porque mejora la calidad de los datos obtenidos, a la vez que disminuye el tiempo requerido por el algoritmo que se aplique posteriormente.
- **Integración de datos:** En este paso se pueden combinar múltiples fuentes de datos.
- **Selección y transformación de los datos:** Los datos relevantes para el analista son recuperados de la base de datos y de ahí, se transfieren o se consolidan en una estructura apropiada para la minería al realizar operaciones de resumen o agregación.
- **Minería de datos:** Este es el paso fundamental del proceso, donde se aplican algoritmos para extraer patrones de datos.
- **Evaluación de patrones:** Se utiliza para identificar los patrones realmente interesantes que representan el conocimiento basado en algunas medidas de interés.
- **Obtención del conocimiento:** En este paso se utilizan técnicas de visualización y representación del conocimiento para representar el conocimiento minado al usuario.

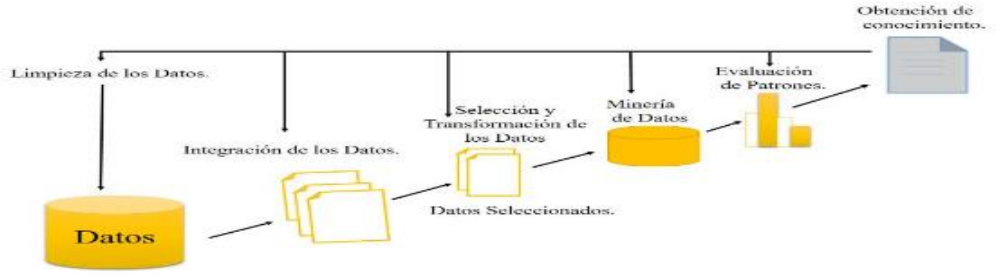


Figura 8 Pasos del proceso KDD. Fuente: [42].

Por lo que, la minería de datos integra numerosas técnicas de análisis de datos y extracción de modelos, capaz de obtener patrones, describir tendencias y regularidades. Además, permite predecir comportamientos y explorar información computarizada, proveniente de fuentes heterogéneas y de grandes volúmenes de datos, permitiendo a los individuos y a las organizaciones comprender y modelar de una manera más eficiente y precisa el contexto en que deben actuar y tomar decisiones inteligentes [41]. Este proceso es utilizado en diferentes ramas de la ciencia, como por ejemplo en la educación.

1.3 Minería de Datos Educacional

La aplicación de la minería de datos con el avance de las tecnologías abarca numerosos campos como el comercio, la bioinformática, la educación, entre otros. En el área de la educación ha incrementado el interés en utilizar la minería de datos en el estudio educacional, centrándose en el desarrollo de métodos de descubrimiento que utilicen los datos de plataformas educacionales y en el uso de esos métodos para comprender mejor a los estudiantes y el entorno en el que aprenden. Esta área de la minería de datos es conocida como Minería de Datos Educacional (MDE). Es definida como la aplicación de técnicas de minería de datos a este tipo específico de conjunto de datos que provienen de la educación [1].

La MDE se puede modelar como la combinación de tres áreas principales informática, educación y estadística como se muestra en la siguiente Figura 9. La intersección de estas tres áreas también forma otras subáreas estrechamente relacionadas con la MDE, tales como la educación basada en computadora, el aprendizaje automático (*Machine Learning, ML*) y el análisis de aprendizaje (*Learning Analytics*, LA).

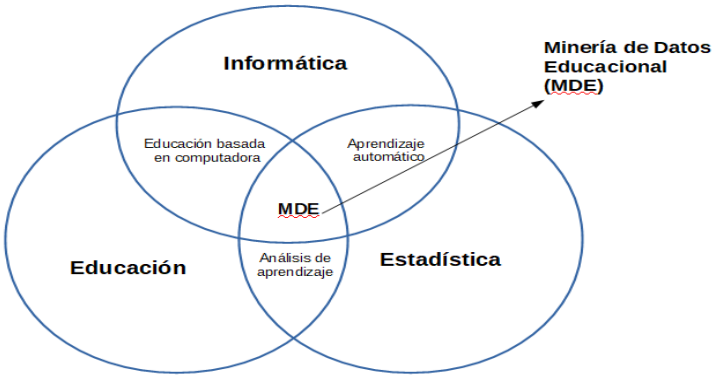


Figura 9 Principales áreas de la Minería de Datos Educativos. Fuente: [4].

Constantemente usuarios tales como: instructores, estudiantes, administradores de cursos entre otros, interactúen con sistemas educativos informatizados como son las aulas inteligentes, sistemas *e-learning*, LMS[7] (*Learning Management System*) y MOOCs[8], los cuales generan un gran cúmulo de información, como son los componentes de información, el rendimiento de usuarios y sus transacciones, registrados en bases de datos educacionales. La creación de repositorios públicos de datos educacionales ha creado una base que hace posible la utilización de minería de datos educacionales. Los datos de estos repositorios son totalmente válidos (ya que son datos reales sobre el rendimiento y aprendizaje de estudiantes reales, en ambientes educacionales, tomados en tareas de aprendizaje), y cada vez más fácilmente accesibles para su utilización. A partir de la generación de estas bases de datos son aplicadas las técnicas de minería de datos, las cuales funcionan a través de algoritmos de inteligencia artificial, haciendo posible la identificación de patrones de comportamiento para la estimación del conocimiento latente. La Figura 10 muestra un esquema general de lo antes descrito [43].

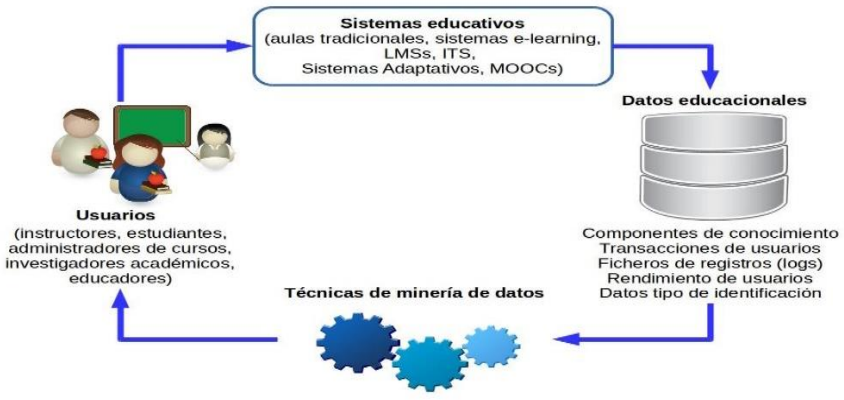


Figura 10. Proceso para la aplicación de la minería de datos en el contexto educativo. Fuente: [43].

[7] LMS: Sistema de Gestión de Aprendizaje, es un término global para un sistema informático desarrollado específicamente para la gestión de cursos en línea.
[8] MOOCs: Cursos masivos abiertos en línea.

De esta manera en la MDE, se propone un proceso iterativo de descubrimiento de conocimiento en los datos que provienen de entornos educacionales. Autores como Sachin y Vijay [44], Romero y Ventura en [1], [4] o Ballestero *et al*.[15] proponen un proceso de aplicar la MDE cercana al de KDD (Knowledge discovery in databases) u otros procesos de aplicación de minería de datos Figura 11. Este proceso comienza con la recolección o elección de los datos que se van a estudiar de la base de datos entorno educativo. Los datos brutos obtenidos requieren limpieza y pre-procesamiento (fusión de datos heterogéneos, tratamiento de valores perdidos e incorrectos), conversión de los datos a un formulario apropiado, selección de características, etc.). Una vez pre-procesados los datos, se aplica el método/técnica de minería de datos en la educación. Finalmente, el último paso es la interpretación y la evaluación de los resultados obtenidos [45].

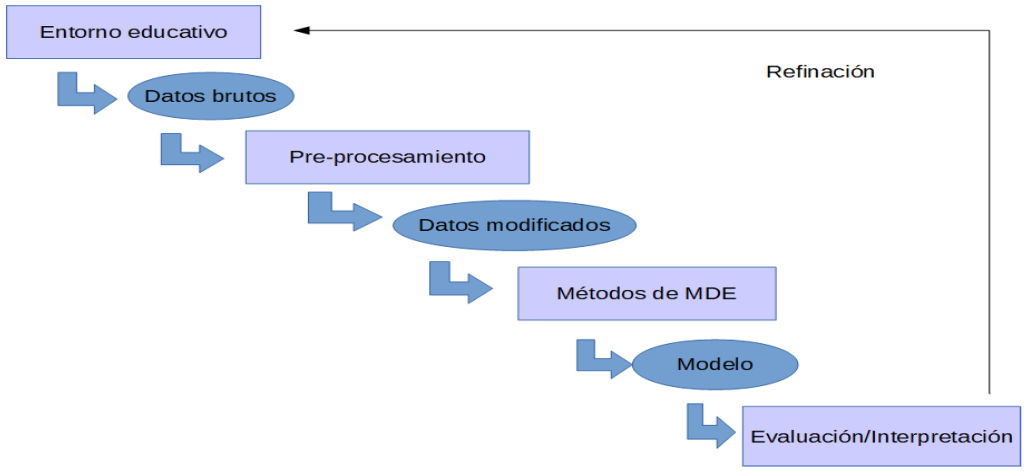


Figura 11 El proceso de aplicación de la minería de datos en la minería de datos educativos. Fuente: [45].

Este proceso es muy similar a los pasos que se plantean en KDD, pero tienen una particularidad y es que trabaja con datos que se generan en entornos educativos. Esto propicia que se empleen técnicas de minería de datos propias para este contexto. Por lo que las técnicas de minería de datos que se puede emplear son: predicción (aquí se incluye la estimación del conocimiento latente), agrupamiento, minería de relaciones, inferencia a través de modelos, y destilación de datos para la interpretación por parte de un ser humano. Las tres primeras categorías son universales para distintos tipos de minería de datos (aunque en algunos casos con distintos nombres), pero con datos que se obtienen del contexto educacional. Las categorías cuarta y quinta consiguen una particular importancia dentro de la minería de datos educacionales [46].

Pasos como el pre-procesamiento es de gran importancia en el ámbito educativo, es importante adquirir conjuntos de datos adecuados y hacer un esfuerzo adicional para recopilar y preparar los datos a fin de incluir toda la información potencialmente útil. Las tareas u operaciones realizadas en un proceso de pre-procesamiento pueden reducirse a dos familias principales de técnicas: Técnicas de detección

para encontrar imperfecciones en los conjuntos de datos y técnicas de transformación orientadas a obtener conjuntos de datos más manejables.

El pre-procesamiento de datos educativos en general es muy similar a la tarea de pre-procesamiento en otros dominios. Sin embargo, es importante señalar que con los datos educativos tiene ciertas características que lo diferencian del pretratamiento de datos en otros ámbitos específicos, como el hecho de que [45]:

- Los sistemas educativos proporcionan una gran cantidad de información generada por los estudiantes diariamente a partir de diferentes fuentes de información, aunque no todos los estudiantes completan todas las actividades, ejercicios, etc. En consecuencia, a menudo faltan datos y éstos son incompletos.
- Por lo general, hay un gran número de atributos disponibles sobre los estudiantes y en muchos casos en diferentes niveles de granularidad. Por lo tanto, es necesario utilizar selección de atributos y tareas de filtrado para seleccionar los más representativos atributos e instancias que pueden ayudar a abordar un problema educativo específico.

Otro paso importante es la visualización de los datos para una mayor comprensión de los mismos, y así permitir a las instituciones de enseñanza tomar mejores decisiones, proporcionar una planificación más avanzada en dirección a los estudiantes, predecir tendencias futuras y comportamientos individuales con mayor precisión.

De manera general, los procesos que expresa KDD en la MDE son de gran relevancia para aplicarse en soluciones que utilicen datos que se generan en los entornos educativos.

1.4 Minería de datos en entornos distribuido

El aumento de información en las grandes bases de datos, ha traído consigo que cambie la forma de manejo de los datos haciendo uso de la minería de datos, por lo que, la minería con grandes cantidades de datos se ha convertido en un área de investigación activa. Es difícil utilizar las metodologías tradicionales y las herramientas de software de minería de datos para que una sola computadora personal pueda manejar eficientemente conjuntos de datos muy grandes, ya que requiere costos computacionales muy elevados. En el enfoque tradicional de minería de datos, los datos suelen estar centralizados y se elige un algoritmo específico para procesar y analizar los datos bajo una única plataforma informática, que trae consigo que el procesamiento sea más lento e ineficiente. Por lo que, es necesario utilizar entornos informáticos más potentes para procesar y analizar eficazmente los datos de gran tamaño.

Las plataformas de computación en paralelo y distribuida se considera una de la mejor solución para la minería de datos de gran tamaño. El concepto de computación en paralelo se basa en dividir un

problema grande en problemas más pequeños y cada uno de ellos es llevado a cabo por un solo procesador individualmente. Además, estos procesos se realizan simultáneamente de forma distribuida y paralela [47].

La computación distribuida puede referirse al uso de sistemas distribuidos para resolver problemas computacionales. En particular, un problema se divide en muchas tareas, cada una de las cuales es resuelta por uno o más ordenadores (o procesadores) que funcionan simultáneamente en paralelo. Además, cada uno puede comunicarse entre sí mediante el paso de mensajes.

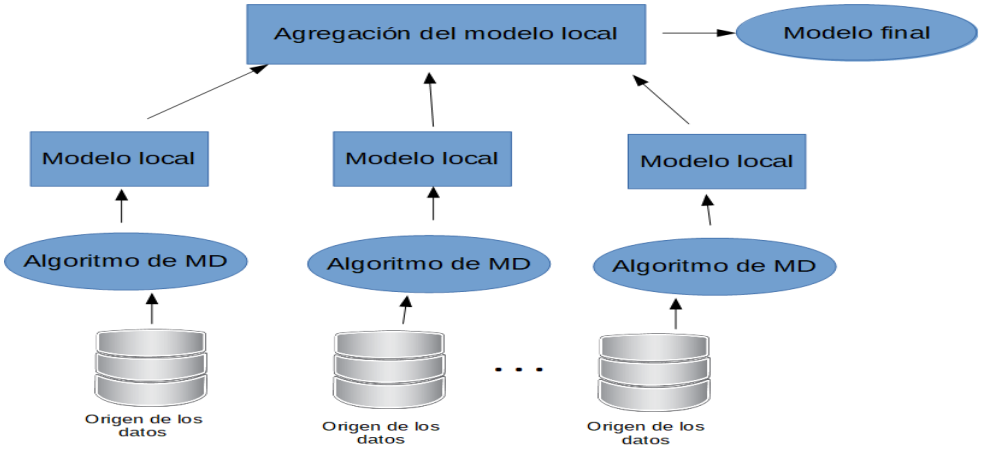


Figura 12 Ejemplo de marco de trabajo de la minería de datos distribuida. Fuente: [47].

En términos generales, el objetivo de la minería de datos en un entorno distribuido se ocupa fundamentalmente de la aplicación del procedimiento clásico de la minería de datos en un entorno informático distribuido que intenta aprovechar al máximo recursos disponibles ya sean redes de comunicación, unidades de cómputo y bases de datos. La Figura 12 muestra un marco de trabajo de minería de datos distribuida, donde las diferentes fuentes de datos pueden ser homogéneas y/o heterogéneas. Cada el algoritmo de minería de datos maneja su correspondiente fuente de datos bajo una sola computación que conduce a un modelo local. Luego, estos modelos locales se agregan con el fin de generar el modelo final [47].

1.4.1 Entorno de minado distribuido

En la facultad 4 de la Universidad de las Ciencias Informáticas se ha desarrollado un modelo capaz de estimar conocimiento latente del estudiante en datos educacionales masivos. Dicho modelo será aplicado teniendo en cuenta el contexto de la educación cubana y los proyectos que se realizan con el fin de la informatización de nuestro país. La estructura que sigue dicho modelo, se muestra en la Figura 13 que se muestra a continuación[48].

Figura 13 Modelo para la estimación de conocimiento latente en datos educacionales masivos. Fuente: [49].

El modelo está compuesto por cuatro módulos, dando inicio a través de las diversas fuentes de información educacionales. Los datos educacionales almacenados son agrupados para formar un gran conjunto, el cual es tratado en el primer proceso que conforma el modelo de la Figura 13, que describe el proceso de gestión de conjunto de datos masivos. Posteriormente los datos obtenidos, se trasladan hacia el segundo proceso, denominado plataforma para el minado de datos masivos, la cual interactúa directamente con los algoritmos de minería de datos a utilizarse, en la que los resultados del procesamiento son trasladados al sistema de visualización. De esta manera a través de técnicas predictivas es obtenida la estimación de conocimiento.

La plataforma para el minado de datos masivos, ocupa gran relevancia dentro del modelo. Esta plataforma, es la encargada de recibir los datos del proceso anterior y realizar el agrupamiento de dichos datos. Posteriormente, se aplican dentro de la plataforma los algoritmos, en este caso el BKT, asegurando a su vez que los mismos se ejecuten de forma eficiente teniendo en cuenta el hardware disponible. Finalmente, es obtenida la estimación del conocimiento, la cual es visualizada en el siguiente proceso del modelo descrito anteriormente. Debido a la importancia que ocupa la plataforma para el minado de datos masivos dentro del modelo desarrollado y la importancia que se le confiere a la velocidad de procesamiento de los datos dentro de la misma, se genera la necesidad de la presente investigación, de conformar un entorno distribuido para contribuir al uso de técnicas de minería de datos distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos.

En la propuesta de solución [48] se ejecutará el algoritmo adaptado BKT. La plataforma de minado de datos educacionales masivos está compuesta por una arquitectura que tiene en cuenta la minería de datos distribuida, la cual se convierte en un punto clave de la investigación. A partir de la misma crear un entorno que propicie la aplicación de diversos algoritmos que utilicen minería de datos educacional. La aplicación de la Minería de Datos Distribuida (MDD) ofrece diversas ventajas, como es el caso de utilizar una estrategia descentralizada haciendo posible que el sistema de trabajo sea completamente

escalable. De manera tal que se distribuya la carga de trabajo entre diversas estaciones de forma heterogénea. Como valor agregado, esta estrategia posibilita la reducción de costos que presupone la utilización de grandes servidores dedicados a la extracción y procesamiento de los datos.

A partir de la utilización de la minería de datos distribuida en el desarrollo del entorno, se estableció una arquitectura que es capaz de sostener las bases por la cual se aplica esta técnica en la propuesta de solución. Dicha arquitectura sienta sus bases en los elementos básicos que compone todo enfoque basado en la MDD y está compuesta por cuatro niveles (ver Figura 14) [48].

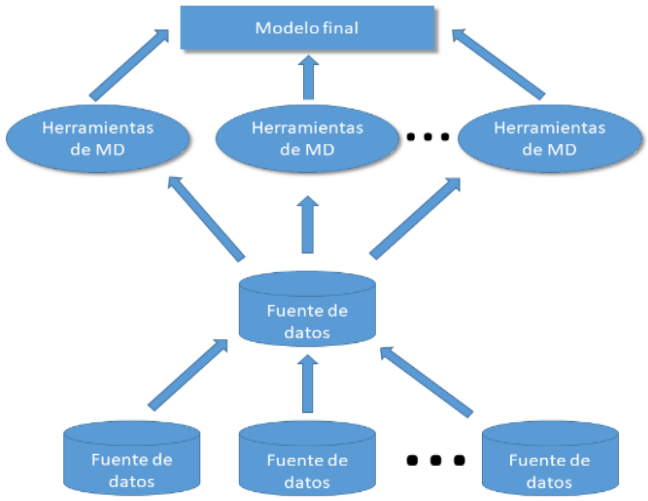


Figura 14 Arquitectura generada para el entorno distribuido para el minado de datos en ambientes educacionales masivos. Fuente: [48].

Quedando de esta forma la propuesta del entorno distribuido para que el algoritmo adaptado se ejecute siguiendo el paradigma de la programación paralela y distribuida.

1.4.2 Programación distribuida y paralela

Con la utilización de MDD, se ha tenido que utilizar técnicas de la programación distribuida y paralela. Existen varias definiciones de programación distribuida y programación paralela.

Según [50] un sistema distribuido es una colección de computadoras independientes que parecen a sus usuarios como un sistema coherente único. Otra definición de sistemas distribuido es la brindada en [51] donde se define sistemas distribuidos como una colección de diversas unidades de cómputo poco acopladas que están geográficamente distribuidas y conectadas por una red de comunicación. Por otro lado en [52] se plantea que un sistema distribuido es una colección de entidades independientes que cooperan para resolver un problema que no puede ser resuelto individualmente. De igual manera en [53] se define un sistema distribuido como la arquitectura de sistema que hace a una colección de computadoras, estaciones de trabajo, o servidores heterogéneos actuar y comportarse como un sistema de computación único.

Un sistema distribuido se puede caracterizar como una colección de procesadores mayormente autónomos que se comunican a través de una red de comunicación y que tienen las siguientes características [52]:

- *Sin reloj físico común*: Esta es una suposición importante porque introduce el elemento de "distribución" en el sistema y da lugar a la asincronía inherente entre los procesadores.
- *Sin memoria compartida:* Esta es una característica clave que requiere pasar mensajes para la comunicación. Esta característica implica la ausencia del reloj físico común.
- *Separación geográfica*: Cuanto más separados geográficamente están los procesadores, más representativo es de un sistema distribuido.
- *Autonomía y heterogeneidad:* Los procesadores están “débilmente acoplados” en que tienen diferentes velocidades y cada uno puede estar ejecutando un sistema operativo diferente. Por lo general, no forman parte de un sistema dedicado, sino que cooperan entre sí ofreciendo servicios o resolviendo un problema en forma conjunta.

Un sistema distribuido típico es mostrado en la Figura 15. Cada computadora tiene una unidad de procesamiento de memoria y estas están conectadas mediante una red de comunicaciones. En la Figura 16 se muestra las relaciones de los componentes de software que se ejecutan en cada una de las computadoras y utiliza el sistema operativo local y la pila de protocolos de red para su funcionamiento. El software distribuido también se denomina *middleware*. Una ejecución distribuida es la ejecución de procesos en todo el sistema distribuido para lograr de manera colaborativa un objetivo común.

El *middleware* es el software distribuido que impulsa el sistema distribuido, al tiempo que proporciona transparencia de la heterogeneidad en el nivel de la plataforma.

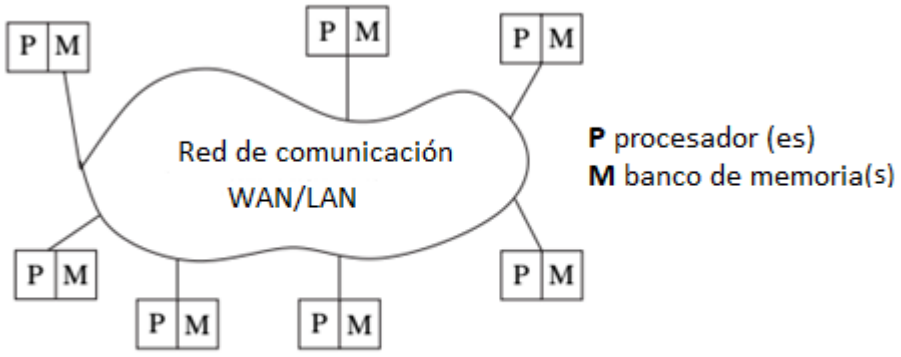


Figura 15 Un sistema distribuido conecta procesadores mediante un red de comunicaciones. Fuente: [52].

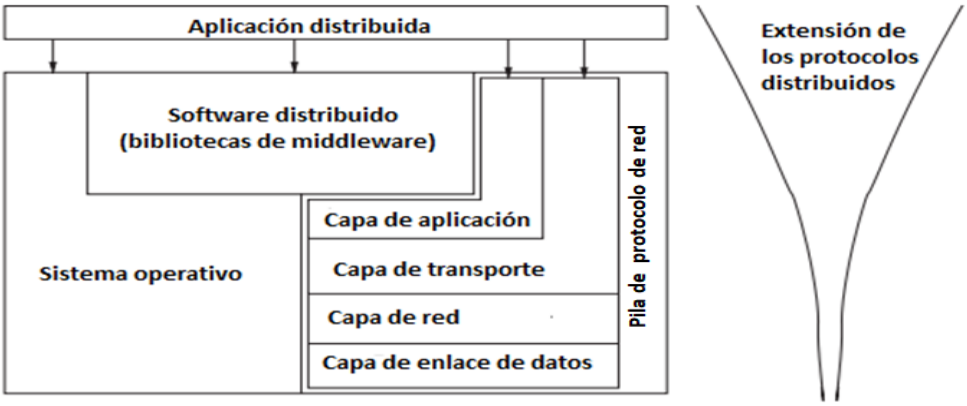


Figura 16 Interacción de los componentes de software en cada procesador. Fuente: [52].

El desarrollo de algoritmos es un componente crítico en la solución de problemas usando computadoras. Un *algoritmo secuencial* es esencialmente una secuencia de pasos básicos para resolver un problema dado usando una computadora serial. Similarmente, un *algoritmo paralelo* es una sucesión que nos dice cómo resolver un problema dado usando múltiples procesadores [54].

La programación paralela se enfoca en el mapeo de algoritmos para trabajar en plataformas de computación paralela de propósito general [55].

Los tres objetivos principales de la programación paralela son los siguientes [56]:

1. Rendimiento.
2. Productividad.
3. Generalidad.

El nirvana de entornos de programación paralelos, uno que ofrece *rendimiento*, *productividad* y *generalidad* de clase mundial, simplemente aún no existe. Hasta que aparezca un nirvana tal, será necesario hacer concesiones de ingeniería entre el rendimiento, la productividad, y la generalidad [56].

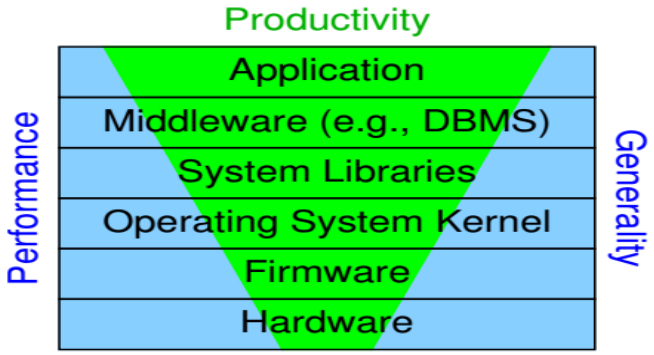


Figura 17 Capas de Software y Rendimiento, Productividad y Generalidad. Fuente: [56]

Una de estas concesiones se muestra en la Figura 17 donde se muestra como la productividad se vuelve cada vez más importante en las capas superiores de la pila del sistema, mientras que el

rendimiento y la generalidad se vuelven cada vez más importantes en las capas menores de la pila del sistema [56].

Es importante tener en cuenta que el paralelismo no es más que una forma de mejorar el rendimiento. Otros enfoques conocidos para esto incluyen los siguientes, en un orden de dificultad creciente [56]:

1. Ejecutar múltiples instancias de una aplicación secuencial.
2. Hacer que la aplicación utilice software paralelo existe.
3. Aplicar optimización de rendimiento a la aplicación serial.

Con el fin de alcanzar un equilibrio entre los objetivos de la programación paralela y distribuida se hace uso de marcos de trabajo que faciliten dicho objetivo. En el siguiente epígrafe se dará detalles del marco de trabajo Spark.

1.4.3 Detalles del marco de trabajo Spark

El procesamiento de datos propuesto en el entorno minado se realizará a través del marco de trabajo Spark [48]. Por lo que, para la adaptación del algoritmo BKT se utilizará esta herramienta potente para el procesamiento de grandes conjuntos de datos. Tiene un amplio ecosistema de software que se integra fácilmente a otras herramientas como Hadoop. Está concebido como un sistema de computación en clústeres de uso general, y se ha especializado hasta convertirse en una de las plataformas de soporte para al análisis de datos. Dentro de sus principales características específica a un alto nivel las funciones necesarias para obtener y transformar los datos, y se encarga que dichas funciones se ejecuten paralelamente y de manera más eficiente posible de acuerdo a como estén almacenados los datos en los clústeres. Spark aprovecha al máximo los recursos del clúster: almacenamiento, procesamiento y memoria. El mismo minimiza la latencia entre etapas de procesamiento. Mediante el concepto de dataset distribuidos (RDD) que incluyen especificación en lenguaje funcional y los datos obtenidos en cada partición de los datos totales en los que se trabaja en cada etapa del grafo de operaciones (DAG), facilita la recuperación de fallos en forma total o parcial [57].

La forma en que Spark puede generar tareas complejas de análisis que involucran múltiples fuentes de datos, posibilita especificar sucesivas transformaciones y uniones sobre los datos, esto hace que este tipo de tareas complejas puedan especificarse y ejecutarse de una sola vez en un clúster de Spark, utilizando no solamente las capacidades de almacenamiento persistente y procesamiento de los nodos sino también un recurso básico como la memoria para almacenar datos a medida que se van procesando. Esto conlleva que la velocidad con que se realizan estas tareas de múltiples transformaciones sea muy buena en Spark.

Spark posee un conjunto de bibliotecas y herramientas agrupada por componente en forma independiente, y dada la amplitud y flexibilidad de las mismas conforma un amplio ecosistema de software.

Spark Core constituye el núcleo de Spark y consiste en el motor de ejecución de procesos de Spark, además de sus utilidades como el Spark Shell en Scala y las API en Scala, Java y Python. Spark Core está presente en toda implementación de Spark [58].

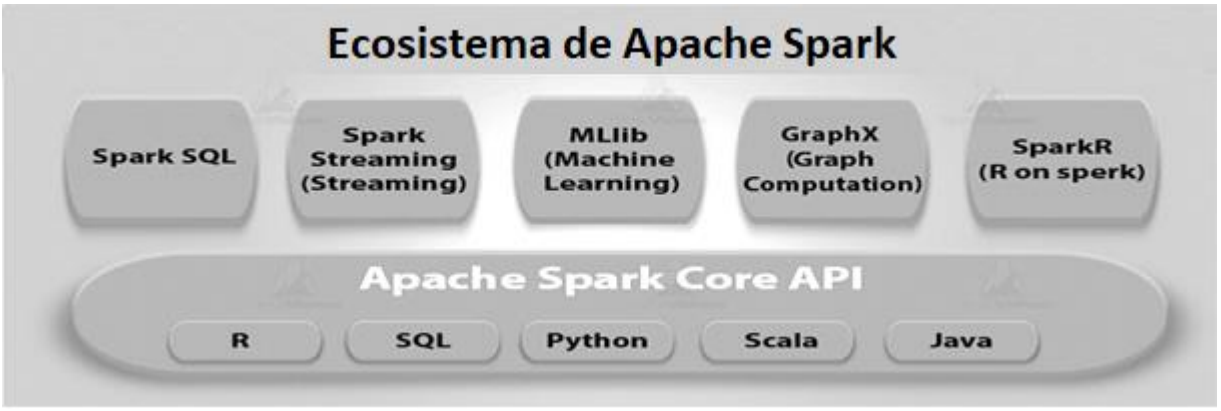


Figura 18 Ecosistema de Apache Spark. Fuente: [58].

Spark SQL es un conjunto de biblioteca que se ejecutan sobre Spark y que proveen lo necesario para realizar consultas SQL sobre grandes volúmenes de datos no estructurados. Spark SQL es la herramienta más rápidamente adoptada por analistas de datos y científicos de datos para la realización de consultas adhoc para resolver preguntas del dominio específico donde se cuenta con Spark para proveer soporte a estas tareas.

Spark MLlib provee bibliotecas con algoritmos de Machine Learning implementados en Spark y permite ejecutar análisis de regresiones y otros tipos de procesamiento de datos iterativos en un clúster Spark. Esto es muy utilizando en la denominada ciencia de datos para identificar patrones en los datos de forma automatizada entre otras tareas que se pueden realizar en Spark.

GraphX y Spark R son herramientas específicas de Spark que permiten, la primera analizar datos de relaciones que se pueden representar como grafos, donde los procesamientos de datos convencionales no pueden ser ejecutados y la segunda provee una interface con el lenguaje de análisis estadístico R muy utilizando en el ambiente de la ciencia de datos donde Spark se está volviendo cada vez más popular.

Almacenamiento de datos: Spark utiliza el sistema de archivos HDFS para el almacenamiento de datos. Funciona con cualquier fuente de datos compatible con Hadoop, incluidos HDFS, HBase, Cassandra, Mesos y Kurbenete [58].

Administración de recursos: Spark se puede implementar como un servidor independiente o en un marco de computación distribuida como Mesos o YARN. Las operaciones de acción evalúan y devuelven un nuevo valor. Cuando se invoca una función de acción en un objeto RDD, todas las consultas de procesamiento de datos se calculan en ese momento y se devuelve el valor resultante.

Algunas de las operaciones de acción son: reduce, collect, count, first, take, countByKey, join and foreach.

1.5 Visualización de los datos

Tanto en el ámbito científico o didáctico, la información basada en datos, numérica o estadística suele resultar difícil de asimilar. Los diferentes tipos de gráficos y diagramas contribuyen a facilitar su interpretación de una manera mucho más rápida y visual. Se representa la relación entre los elementos de un conjunto, haciendo destacar los patrones y tendencias que aportan información relevante. Incorporar estos elementos gráficos resulta muy útil, ya que aportan información de valor para tomar futuras decisiones o mejorar procesos.

La visualización clásica de la minería de datos educacionales es la curva de aprendizaje, que representa el número de oportunidades de practicar una habilidad en el eje X, y muestra el rendimiento (como el porcentaje de aciertos o tiempo tomado para responder) en el eje Y como muestra la Figura 19 [2], [59]. Para este tipo de visualización es recomendable utilizar un gráfico de dispersión.

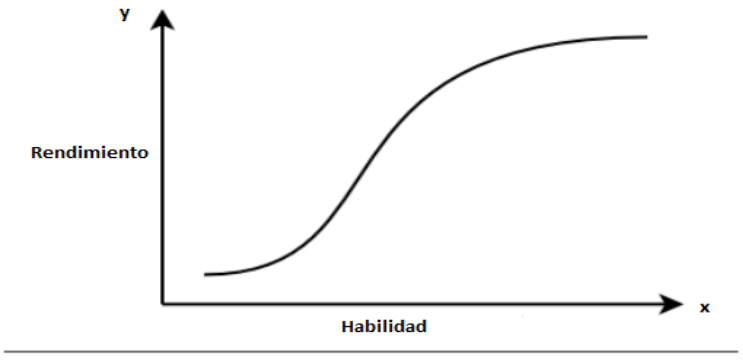


Figura 19 Curva de aprendizaje. Fuente: [59].

Otras de las visualizaciones que se realiza es mostrar el comportamiento de las habilidades de los estudiantes a través de un gráfico de radar (ver Figura 20) [19].

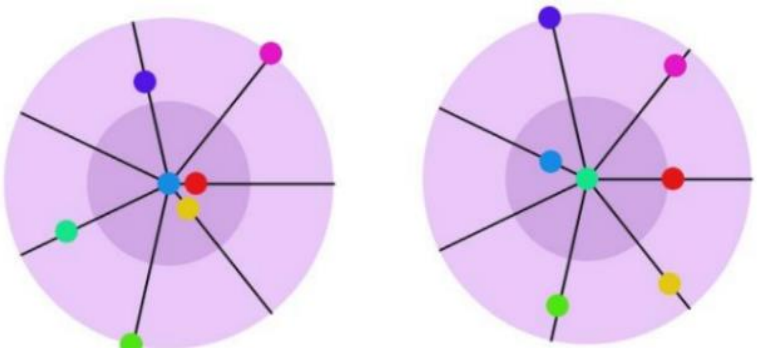


Figura 20 Gráfico de radar. Fuente: [19]

Por otra parte, se utiliza gráfico de burbuja para mostrar el tamaño de incidencia sobre una habilidad por varios estudiantes, este tipo de gráfico recoge varios parámetros que inciden directamente sobre las habilidades de los estudiantes [19].

Para la visualización de los datos utilizando el lenguaje de programación Java se pueden usar las bibliotecas como JFreechart[9] y JavaFX Charts[10] [60], [61] las mismas proveen gráficos como:

- **Gráfico de burbuja:** Un gráfico de burbujas es un gráfico de múltiples variables que supone un cruce entre un diagrama de dispersión y un gráfico de área proporcional. Como un diagrama de dispersión, las burbujas utilizan un sistema de coordenadas cartesianas para trazar puntos a lo largo de una cuadrícula en los ejes X e Y con variables independientes, pero a diferencia de un diagrama de dispersión, a cada punto se le asigna una etiqueta o categoría que puede estar asociada con una leyenda. Cada punto representado a continuación representa una tercera variable por la zona de su círculo. Los colores también se pueden utilizar para distinguir entre categorías o ser utilizados para representar una variable de datos adicional. El tiempo puede mostrarse como variable sobre uno de los ejes o por la animación de las variables de datos cambiantes [62],[61].
- **Gráfico de dispersión:** Los diagramas de dispersión utilizan una colección de puntos colocados mediante coordenadas cartesianas para mostrar los valores de dos variables. Al mostrar una variable en cada eje, se puede detectar si existe una relación o correlación entre las dos variables [62], [61].

En el caso de JFreeChart además tiene el gráfico de radar:

- **Gráfico radial o araña:** Los gráficos radiales son una manera de comparar múltiples variables cuantitativas. Esto los hace útiles para ver qué variables tienen valores similares o si hay valores extremos entre cada variable. Los gráficos radiales son útiles también para ver de qué variables

[9] http://www.jfree.org/jfreechart/index.html
[10] https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/charts.htm

están resultando altas o bajas dentro de un conjunto de datos, lo que es ideal para la visualización del rendimiento [61].

Para la representación de los gráficos se hará a través de la biblioteca de JFreeChart que es una biblioteca de gráficos 100% de Java gratuita, que facilita a los desarrolladores la visualización de gráficos de calidad profesional en sus aplicaciones. El amplio conjunto de funciones de JFreeChart incluye: una API consistente y bien documentada, que soporta una amplia gama de tipos de gráficos. Tiene un diseño flexible que es fácil de ampliar y se dirige tanto a aplicaciones del lado del servidor como del lado del cliente. Soporte para muchos tipos de salida, incluyendo componentes Swing y JavaFX, archivos de imagen (incluyendo PNG y JPEG), y formatos de archivos de gráficos vectoriales (incluyendo PDF, EPS y SVG). JFreeChart es de código abierto o, más específicamente, software libre [61].

JFreeChart es compatible con una serie de gráficas diferentes, incluyendo cuadros combinados. Dentro de los tipos de gráficos que se pueden obtener están [61]:

- Gráficos XY (línea, spline y dispersión). Es posible usar un eje del tiempo.
- Gráfico circular.
- Diagrama de Gantt.
- Gráficos de barras (horizontales y verticales, apiladas e independientes). También tiene incorporado un dibujador de histogramas.
- Valor individual (termómetro, brújula, indicador de velocidad) que luego se pueden colocar sobre el mapa.
- Varias gráficas específicas (tabla de viento, gráfica polar, burbujas de diferentes tamaños, etc.).

Dibuja automáticamente las escalas de los ejes y leyendas. Con el ratón informático se puede hacer zoom en la interfaz de la gráfica automáticamente y cambiar algunos ajustes a través del menú local. Las tablas existentes pueden actualizarse fácilmente a través de los oyentes (listeners) que la biblioteca tiene en sus colecciones de datos [61].

1.6 Conclusiones del capítulo

- Con la caracterización de los diferentes algoritmos para estimar conocimiento latente, se pudo evidenciar que el método BKT permite determinar la probabilidad de que un estudiante domine la habilidad latente. Definiéndose los elementos esenciales del algoritmo secuencial.
- Con el experimento realizado con los métodos para ajustar parámetros que utiliza el algoritmo BKT, se determinó utilizar para el algoritmo el de Probabilidad Empírica, por arrojar los mejores valores en el cálculo del AUC y el tiempo de respuesta.

- El análisis de KDD permitió definir pasos esenciales para la transformación del algoritmo BKT como son la limpieza de los datos y la visualización, además del paso fundamental que es la utilización de un método de minería de datos en el contexto educativo.
- Se definió la herramienta Apache Spark para el procesamiento de datos masivos debido a los mecanismos de tolerancia a fallo y su velocidad de procesamiento, también por integrarse con el sistema de archivo que proporciona Hadoop.
- La visualización de los datos mostrará la información de manera intuitiva para que sean comprendidos correctamente. Para ellos, se determinó representar la información de las habilidades mediante un gráfico de burbuja, los datos de habilidades por estudiante a través de un gráfico radial y, por último, la curva de aprendizaje del estudiante mediante un gráfico de dispersión haciendo uso de la biblioteca JFreeChart.

Capítulo 2. Propuesta de solución

En el presente capítulo se realiza la propuesta de un algoritmo que adapte al *Bayesian Knowledge Tracing* para ser utilizado en el contexto de grandes volúmenes de datos. Esta propuesta está sustentada en la modificación de varias de las características del algoritmo adaptado, en el uso de métodos de programación distribuida y paralela, específicamente en la utilización del marco de trabajo Apache Spark. Se formaliza el problema a resolver y se describe el algoritmo que permite la solución a dicho problema.

2.1 Modelación del problema

Para modelar el problema se asumen las siguientes premisas:

- Cada estudiante e_k posee una cantidad n de habilidades o componentes (KC) de conocimiento que él debe dominar.
- Se denota la cantidad de estudiantes a analizar como l. El conjunto de estudiantes es denotado por $E = \{e_k | 1 \leq k \leq l\}$.
- Las habilidades o componentes de conocimiento que debe dominar un estudiante estarán denotadas por L. El conjunto de habilidades o KC que posee un estudiante e_k estará denotado por $A_k = \{L_{k,i} | 1 \leq i \leq n\}$.
- La probabilidad del dominio de dicha habilidad o KC i para un estudiante k es representada por $P(L_{k,i})$.
- Cada problema a resolver está relacionado directamente a una habilidad.
- Cada estudiante k realiza m intentos de resolver problemas que tributen a la habilidad i. La secuencia $O_{k,i} = \{o_1, o_2, \ldots, o_m\}$ denota las respuestas del estudiante k a las preguntas sobre la habilidad i. En cada intento para resolver un problema, la respuesta puede ser correcta o incorrecta, por lo que $o_i \in \{0,1\}$.
- Existe la posibilidad de que el estudiante adivine la respuesta (denotado por $P(G)$) si no domina la habilidad y de que se equivoque (denotado por $P(S)$) si conoce la respuesta.
- Sea $T_{k,i}(O_{k,i})$ el tiempo requerido para calcular $P(L_{k,i})$.
- El tiempo total requerido para calcular la probabilidad de todas las habilidades de todos los estudiantes estaría dada por la expresión:

$$TT = \sum_{e_k \in E} \sum_{i \in A_k} T_{k,i}(O_{k,i}) \qquad (15)$$

Una vez establecido lo anterior se puede plantear como **formulación del problema:**

¿Cómo obtener un método para calcular la probabilidad de todas las habilidades de los estudiantes en un tiempo TT' tal que se cumpla que TT' < TT?

2.2 Estructura general de la propuesta

Como bien se hacía referencia en el capítulo 1, la propuesta de solución se ejecutará sobre un entorno minado propuesto en la tesis de investigación [48] como parte del grupo de investigación Minería de Datos Educativos Masivos de la facultad 4. En [49] se propone un modelo para ECL basada en el empleo de la programación distribuida y paralela a través del marco de trabajo Apache Spark para adaptar el algoritmo BKT al contexto de grandes volúmenes de información (Figura 21).

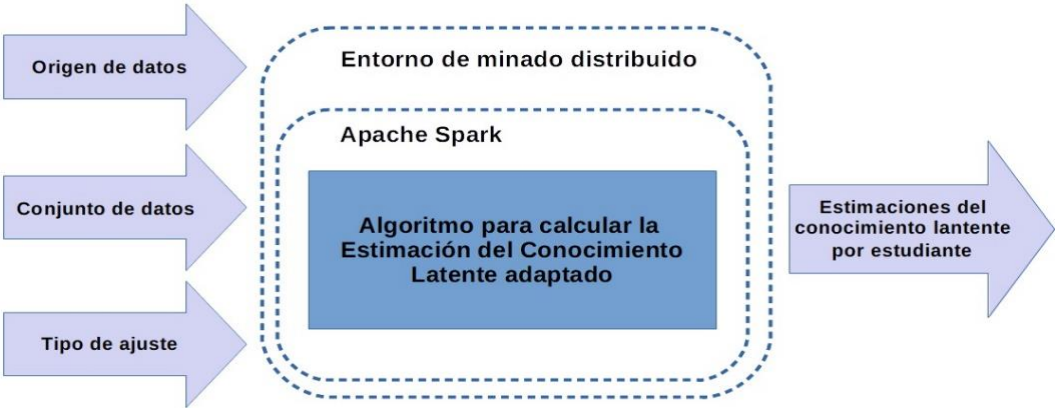


Figura 21 Esquema general para ejecutar los algoritmos. Fuente: [49].

El algoritmo propuesto posee la siguiente estructura general:

- El algoritmo propuesto debe integrarse en un *entorno de minado distribuido* y estará basado en el marco de trabajo Apache Spark.
- Para su funcionamiento debe recibir un conjunto de datos que detalla los intentos a cada habilidad de cada estudiante.
- Al aplicar una adaptación del algoritmo BKT se debe obtener como resultado un mapeo con la probabilidad de dominio de cada habilidad a cada estudiante.

2.3 Estructura del algoritmo propuesto

El algoritmo estará compuesto por 5 pasos fundamentales siguiendo el proceso de KDD en la MDE, pero adaptado a la necesidad de cálculo del algoritmo. De forma general el algoritmo parte de la modificación del BKT adaptándolo a las condiciones propias del problema planteado. Todos los pasos del algoritmo sufren modificaciones respecto al algoritmo original. En la Figura 22 se esbozan estos pasos que posteriormente serán definidos.

2.3.1 Elementos principales del algoritmo propuesto

El algoritmo propuesto utiliza los siguientes elementos para su funcionamiento:

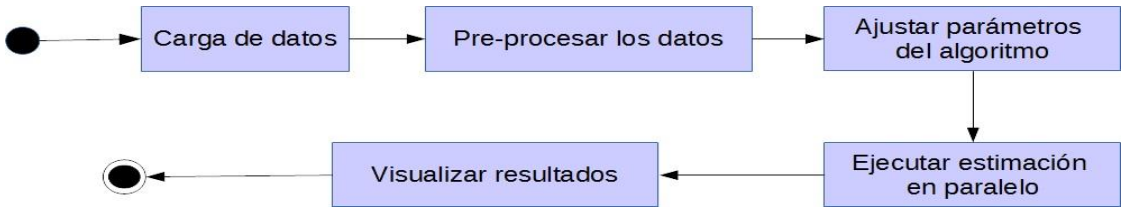


Figura 22 Pasos lógicos para la construcción del algoritmo. Fuente: Elaboración propia.

1. Carga de datos: En este paso se realiza la carga de datos desde las diferentes fuentes que proporcionan datos.
2. Pre-procesar los datos: Se encarga de modificar el conjunto de datos para que estos estén listos para la ejecución del algoritmo propuesto.
3. Ajustar parámetros del algoritmo. El algoritmo propuesto utiliza un grupo de parámetros para su funcionamiento que necesitan ser ajustados antes de poder aplicarlos.
4. Ejecutar estimación en paralelo: Se realiza la estimación para todos los estudiantes en el conjunto de datos de forma paralela.
5. Visualizar resultados: Se visualizan y salvan los resultados obtenidos.

2.3.2 Precondiciones

Para el correcto funcionamiento del algoritmo propuesto es necesario tener definido los siguientes aspectos:

1. Conjunto de estudiantes a los que se le realizará la estimación de su conocimiento.
2. Conjunto de datos con las acciones realizadas por los estudiantes para cada habilidad latente a ser estimada.
3. Parámetros definidos para la plataforma de minado a ser utilizada.
4. Tipo de ajuste de parámetros.

2.3.3 Entradas y salidas del algoritmo propuesto

Las **entradas** del algoritmo propuesto son el conjunto de estudiantes a ser procesado, un conjunto de habilidades por estudiantes a ser estimada y un conjunto de acciones realizada por cada estudiante en cada habilidad a ser estimada. Otra entrada debe ser cuál será el origen de datos (HDFS y archivos CSV o TSV). Por último, como realizar el procesamiento para el ajuste de parámetros.

Por su parte la **salida** del algoritmo será un conjunto que representa un mapeo de habilidad-estimación (grado de dominio de la habilidad) por cada estudiante.

2.4 Definición formal del algoritmo

En el presente epígrafe se definirán formalmente los pasos que componen el algoritmo propuesto. Se considera que se estará utilizando el marco de trabajo (framework) Apache Spark. Por lo que el algoritmo será descrito tomando en cuenta consideraciones propias de este framework.

2.4.1 Paso 1 Carga de datos

Para comenzar el algoritmo es necesario cargar desde una de las fuentes de datos posibles (Figura 23) disponibles para Spark SQL.

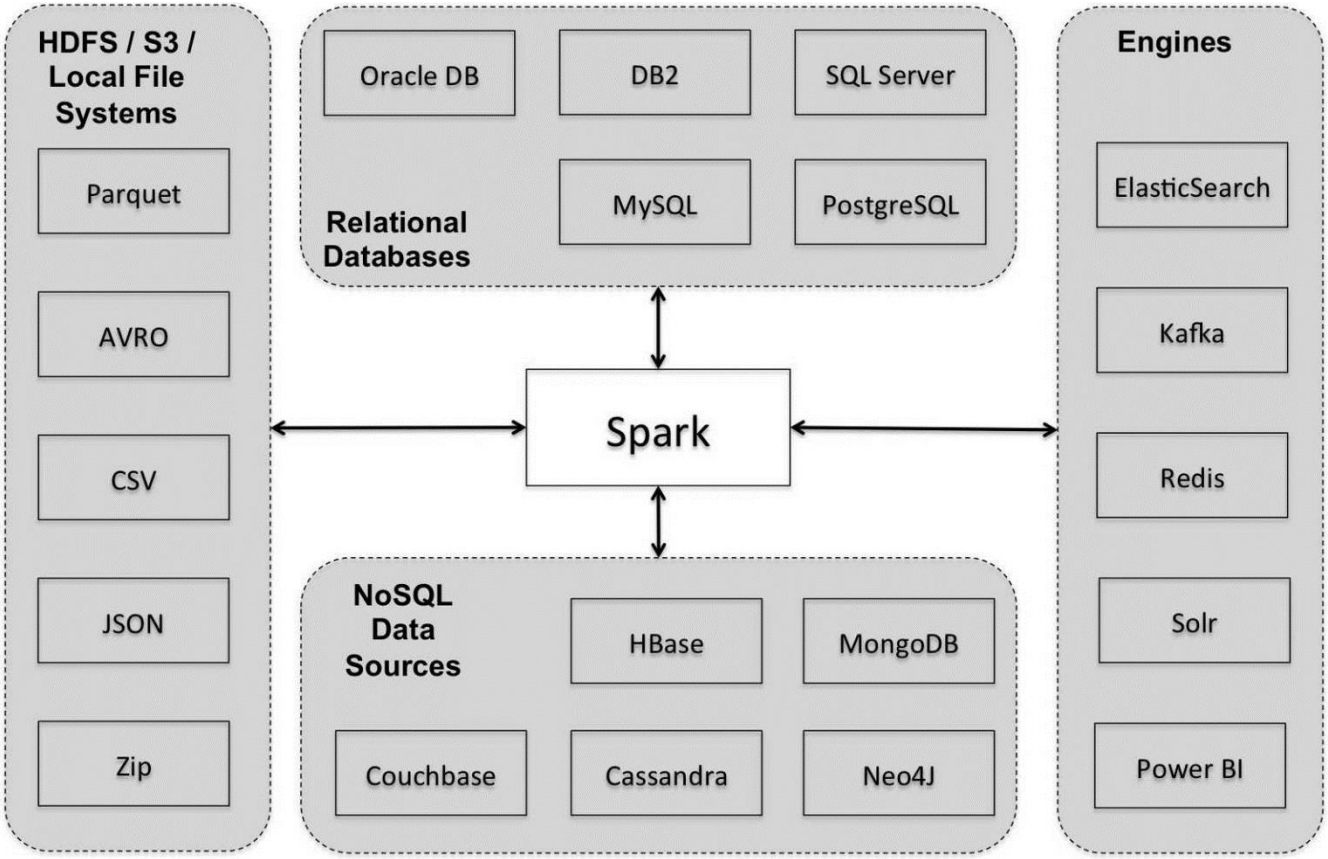


Figura 23 Origen de datos para Spark. Fuentes: [63]

En este caso, se tiene como opciones de orígenes de datos a:

- Archivos CSV o TSV.
- Hadoop Distributed File System (HDFS).

Por tanto, este procedimiento recibe como entrada el tipo de origen de los datos y los datos necesarios para cargarlo y consta de los siguientes pasos (ver Figura 24):

1. Configuración del SparkSession a través del objeto SparkConf.
2. A partir del SparkSession y del origen de datos, se obtiene el conjunto de datos (dataset) a utilizar.
3. El dataset obtenido es devuelto a la ejecución central del algoritmo.

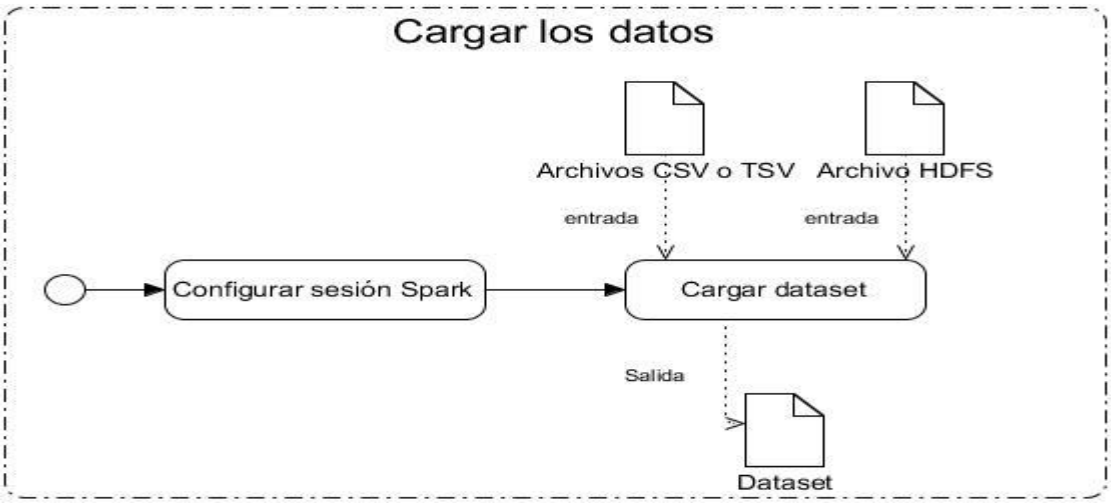


Figura 24 Paso cargar los datos. Fuente: Elaboración propia.

En la Tabla 3 se muestra cómo quedaría en código los pasos anteriores para el caso en que se cargan los datos desde un archivo CSV desde el sistema de archivo.

Tabla 3 Pasos para la carga del conjunto de datos.

Entrada	Origen de datos (HDFS o CSV).
Salida	Conjunto de datos
1. Creación de la sesión de Spark. 2. Configurar parámetros de entrada con el origen de datos. 3. Invocar al método para la carga de dato según el tipo de origen de datos. 4. Asignarle al algoritmo BKT adaptado el conjunto de dato obtenido.	

2.4.2 Paso 2 Pre-procesar los datos

Para el correcto funcionamiento del algoritmo es necesario que el conjunto de datos posea la información referente al estudiante, el problema, la habilidad y el valor observado de respuesta a dicho problema. Por tanto, es preciso transformar el conjunto de datos obtenido en el paso anterior para obtener uno que tenga la estructura adecuada (Tabla 4). Para ello se seguirán una series de pasos para el pre-procesamiento de los datos (ver Figura 25).

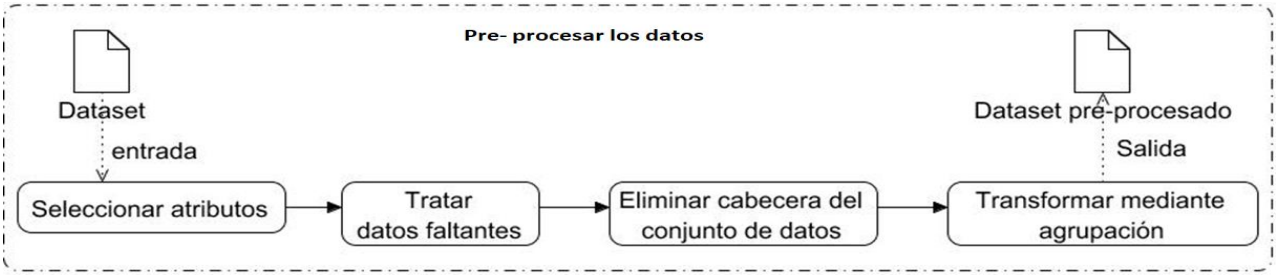


Figura 25 Pre-procesamiento de los datos. Fuente: Elaboración propia.

Este proceso de transformación el conjunto de datos debe tener el siguiente formato[64]:

Tabla 4 Estructura conjunto de datos para BKT.

Observación	ID Estudiante	Problema	Habilidad
id_observación	id_estudiante	id_problema	id_habilidad

Donde

- id_observación – Observación de la respuesta (correcto, incorrecto).
- id_estudiante – Identificación del estudiante.
- id_problema – Concatenación de jerarquía del problema – nombre del problema – nombre del paso.
- id_habilidad – Habilidad o componente de conocimiento a medir.

Para lograr esto es preciso realizar transformaciones al conjunto de datos con el fin de dejar el mismo con solamente estas columnas. El conjunto de transformaciones que se le pueden realizar a un dataset son:

Tabla 5 Pre-procesamiento de datos.

Entrada	Conjunto de datos cargado
Salida	Conjunto de datos pre-procesado

1. Seleccionar atributos necesarios:
   Si la cantidad de atributos en el dataset es mayor o igual que 4 entonces
   - Comprobar que tiene las columnas que son necesarias (Tabla 5)
   - Seleccionar del conjunto de datos las 4 columnas a utilizar.

   Sino
   - Retornar fallo (el conjunto de datos no es adecuado).

2. Tratamiento de datos faltantes:
   Filtrar del conjunto de datos original aquellas tuplas que tengan todos sus datos disponibles; eliminando aquellas donde falten datos en alguna de sus columnas.
3. Eliminar cabecera del conjunto de datos:
   Eliminar de conjunto de datos la primera tupla que posee las cabeceras del conjunto de datos.
4. Transformación del conjunto de datos:
   Agrupar los elementos usando como llave el par (estudiante, habilidad) y concatenando las observaciones del estudiante para esa habilidad de modo que quede una secuencia ([estudiante,habilidad]->secuencia observaciones)
5. Retornar conjunto de datos.

Una vez terminado el pre-procesamiento de los datos, estos están listos para realizar el ajuste de los parámetros necesarios para el cálculo de la estimación del conocimiento de cada estudiante.

2.4.3 Paso 3 Ajustar parámetros del algoritmo

Para este proceso se hará el ajuste de parámetro con el par estudiante habilidad, donde se ejecutará en paralelo el proceso (ver Figura 26).

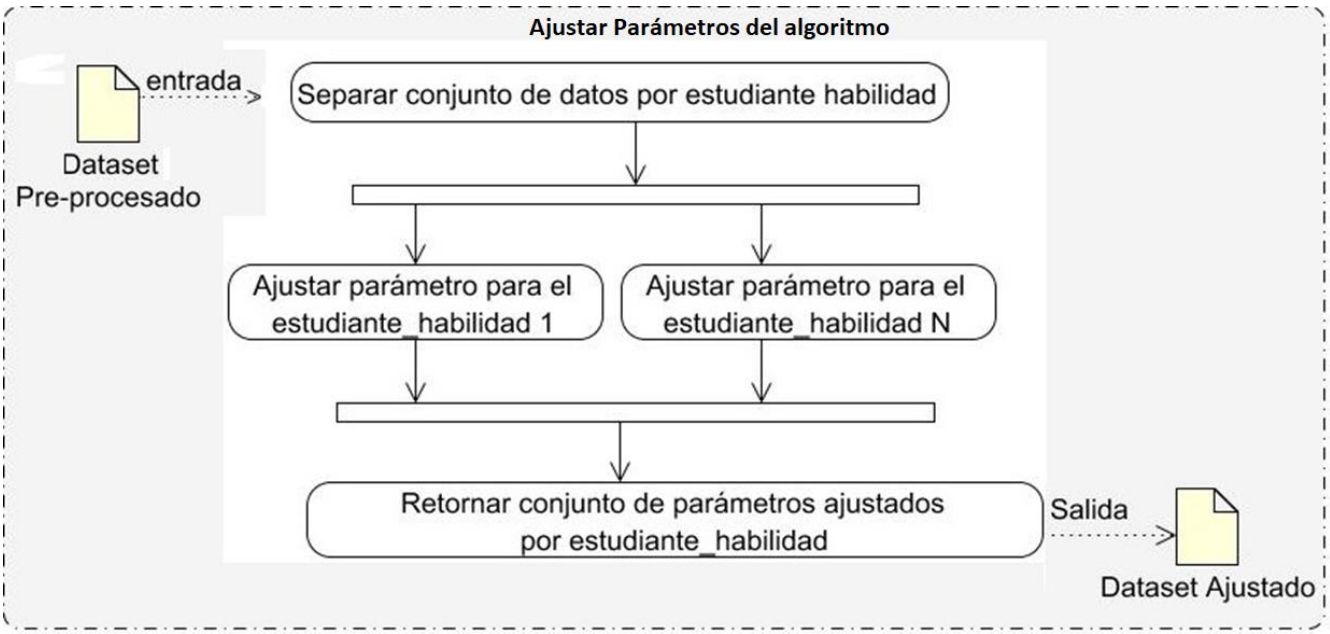


Figura 26 Ajuste de parámetro en paralelo. Fuente: Elaboración propia.

Este algoritmo es un proceso de dos pasos que involucra anotar los datos del rendimiento con conocimiento y usar esta información para computar los parámetros de BKT [33].

Anotar el conocimiento: el primer paso de este algoritmo es anotar los datos de rendimiento para cada estudiante en cada habilidad y estimar cuando el estudiante ha aprendido la habilidad. Se asume que solamente existen dos estados del conocimiento, aprendido (1) y no aprendido (0), y no se permite el olvidar (o sea un estado de aprendido no puede estar seguido por un estado de olvidado) [33].

Es usada una heurística simple para determinar cuándo un estudiante aprende una habilidad: se escoge la secuencia de conocimiento que mejor se ajuste a su rendimiento. Se considera que un estado de aprendido coincide con una respuesta correcta y un estado de no aprendido coincide con una respuesta incorrecta [33].

Por ejemplo, si el rendimiento de un estudiante es la secuencia (correct, incorrect, incorrect, correct, correct), entonces la secuencia de conocimiento que mejor coincide con esta es (unknown, unknown, unknown, known, known), dado que esta coincide con cuatro de los cinco elementos de rendimiento, y es mejor que cualquier otra posible secuencia [33].

En la Figura 27 se muestra un ejemplo de dicha anotación. Este es un ejemplo de algunos casos donde dos secuencias de conocimiento coinciden con la secuencia del rendimiento del estudiante, en estos casos como tomamos el estado de aprendido como 1 y el estado de no aprendido como 0, se realiza un promedio de estas dos secuencias de aprendizaje (0.5, 0.5, 1, 1, 1) [33].

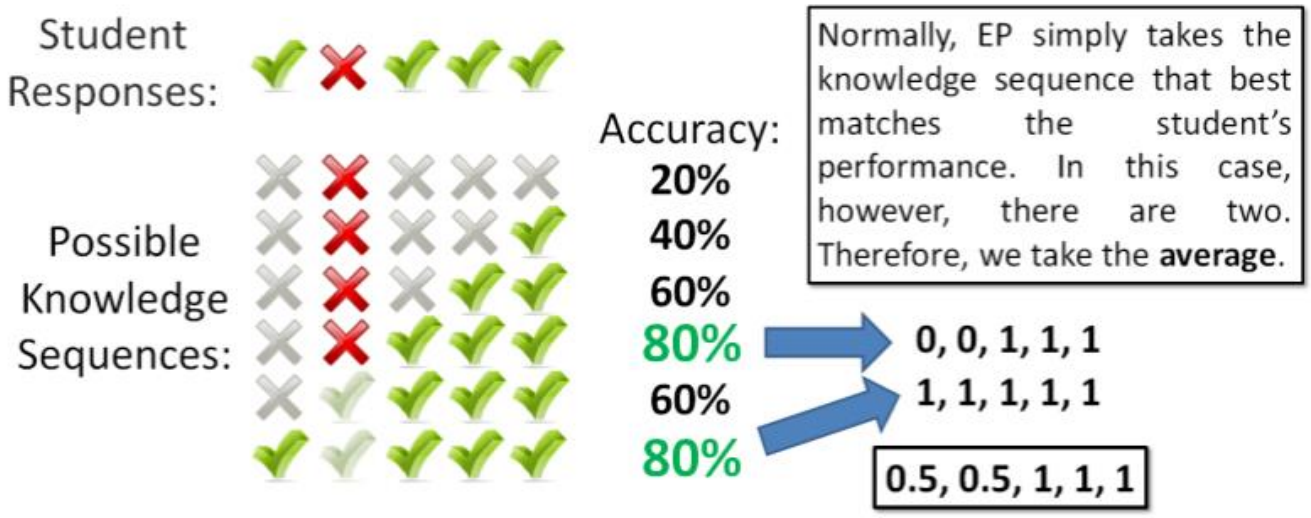


Figura 27 Ejemplo de anotación de conocimiento. Fuente:[33]

Calcular las probabilidades: Usando el conocimiento estimado, es posible calcular los cuatro parámetros del BKT empíricamente para cada habilidad de los datos.

El primero de estos parámetros es $P(L_0)$, la probabilidad de que el estudiante conociera la habilidad antes de interactuar con el sistema. Se puede calcular empíricamente tomando el promedio del valor del conocimiento del estudiante en la primera oportunidad [33].

$$P(L_0) = \frac{\sum K_0}{|K_0|} \tag{15}$$

Usando K_i y C_i como conocimiento y correctitud en el problema i respectivamente, las siguientes ecuaciones son usadas para computar los otros tres parámetros de BKT [33].

$$P(T) = \frac{\sum_{i\neq 0}(1 - K_{i-1})K_i}{\sum_{i\neq 0}(1 - k_{i-1})} \tag{16}$$

$$P(G) = \frac{\sum_i C_i(1 - K_i)}{\sum_i(1 - K\_i)} \tag{17}$$

$$P(S) = \frac{\sum_i(1 - C_i)K_i}{\sum_i K_i} \tag{18}$$

A continuación, se muestra el seudocódigo de la propuesta para este método adaptada al entorno donde se desarrolla la propuesta.

Tabla 6 Código de implementación del método de ajuste Probabilidades Empíricas.

Entrada	Conjunto de datos pre-procesados
Salida	Conjunto de datos de parámetros ajustados

1. Crear acumulador para los parámetros a calcular
2. Registrar el acumulador en la sesión de Spark.
3. Para cada tupla (par estudiante-habilidad) hacer (de forma distribuida y paralela)
    a. Obtener los datos del par estudiante-habilidad (estudiante, habilidad, secuencia de observaciones)
    b. Crear un objeto de la clase del método de ajuste (EmpiricalProbabilitiesFitting)
    c. Ejecutar el método de ajuste
    d. Acumular los datos resultantes del ajuste
4. Crear un conjunto de datos a partir de los datos acumulados
5. Retornar el conjunto de datos obtenido.

Como resultado de este proceso se obtendrán los parámetros del algoritmo BKT ajustados para cada estudiante-habilidad presente en el conjunto de datos. En la siguiente tabla se muestra el método que permite este ajuste y se obtiene como resultado los parámetros ajustados por estudiante- habilidad.

Tabla 7 Método que permite el ajuste de parámetros.

Entrada	Secuencia de observaciones del estudiante para la habilidad
Salida	Valores de los parámetros ajustados.

1. Anotar el conocimiento (Tabla 8) obteniendo los valores de K
2. Calcular el valor de $P(L_0)$ a partir de:
$$P(L_0) = \frac{\sum K_0}{|K_0|}$$
3. Calcular el valor $P(T)$, $P(G)$ y $P(S)$ a partir de:
$$P(T) = \frac{\sum_{i \neq 0}(1 - K_{i-1})K_i}{\sum_{i \neq 0}(1 - k_{i-1})}$$
$$P(G) = \frac{\sum_i C_i(1 - K_i)}{\sum_i (1 - K\_i)}$$
$$P(S) = \frac{\sum_i (1 - C_i)K_i}{\sum_i K_i}$$
4. Retornar el valor calculado para los parámetros.

A continuación, se muestra el seudocódigo del método para anotar el conocimiento.

Tabla 8 Método para anotar el conocimiento.

Entrada	Secuencia de observaciones del estudiante para la habilidad
Salida	Valores de la secuencia de anotación del conocimiento

1. Generar secuencias de anotaciones
2. Determinar las secuencias de anotaciones que más se ajustan a las secuencias de observaciones.
3. Si hay más de una secuencia de anotaciones que se ajusta a las observaciones entonces:
    a. Promediar dichas anotaciones para obtener solo una

    Sino
    b. Obtener la anotación que más se ajusta a la secuencia de observaciones
4. Retornar las anotaciones obtenidas.

2.4.4 Paso 4 Ejecutar estimación en paralelo

Una vez terminado el paso anterior del algoritmo, por cada par estudiante-habilidad se posee cuales parámetros se deben usar para calcular el conocimiento latente que posee el estudiante para cada habilidad. En el siguiente gráfico se muestra cómo se comportará el flujo de los datos en este método.

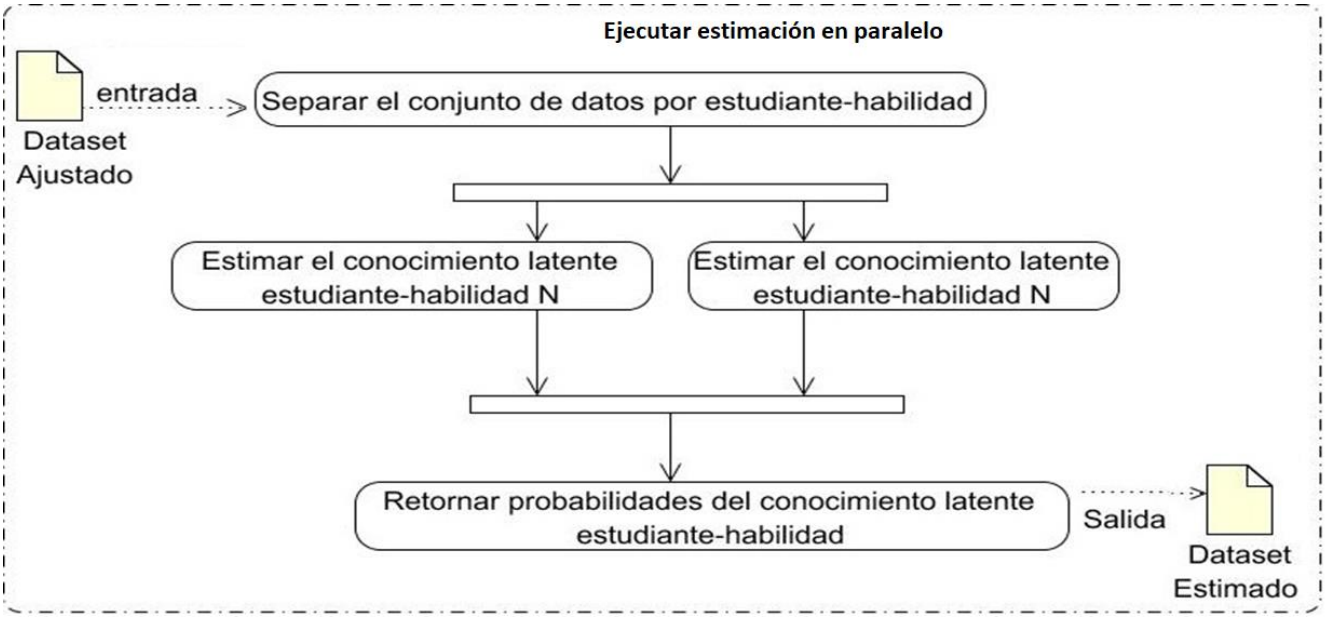


Figura 28 Estimación del conocimiento por estudiante. Fuente: Elaboración propia.

Este paso del algoritmo recibe un conjunto de datos con las transacciones de todos los estudiantes-habilidades (estudiante, habilidad, secuencia de observaciones, parámetros con sus valores ajustados). De forma paralela y distribuida se calcula para cada par estudiante-habilidad cuál es el valor de la probabilidad de que se domine dicha habilidad por el estudiante.

Tabla 9 Cálculo de la probabilidad de dominio de las habilidades.

Entrada	Conjunto de datos con los parámetros ajustados.
Salida	Conjunto de datos con los valores de probabilidad de dominio de las habilidades.
1. Crear un acumulador para los valores de probabilidad a calcular. 2. Registrar el acumulador en la sesión de Spark. 3. Para cada par estudiante-habilidad hacer (de forma paralela y distribuida) a. Obtener los datos de la tupla actual (estudiante, habilidad, secuencia de observaciones, parámetros ajustados) b. Invocar el algoritmo BKT para los parámetros ajustados y secuencia de observaciones actuales. c. Acumular el valor resultante de la ejecución del algoritmo BKT. 4. Crear un conjunto de datos a partir de los valores acumulados. 5. Retornar el conjunto de datos obtenido.	

En la Tabla 10 se muestra el seudocódigo del cálculo de la probabilidad siguiendo el algoritmo BKT.

Tabla 10 Seudocódigo del algoritmo BKT

Entrada	Secuencia de observaciones Valores de los parámetros del algoritmo ($P(L_0)$, $P(T)$, $P(G)$ y $P((S))$
Salida	Probabilidad de dominio de la habilidad

1. Crear arreglo de las probabilidades de dominio de la habilidad L
2. Para cada observación hacer
   a. Si observación actual es correcta hacer

$$P(L_{j-1}|O_j) = \frac{P(L_{j-1}|O_{j-1})(1 - P(S))}{P(L_{j-1}|O_{j-1})(1 - P(S)) + [1 - P(L_{j-1}|O_{j-1})]P(G)}$$

   Sino

$$P(L_{j-1}|O_j) = \frac{P(L_{j-1}|O_{j-1})P(S)}{P(L_{j-1}|O_{j-1})P(S) + [1 - P(L_{j-1}|O_{j-1})](1 - P(G))}$$

   b. Calcular

$$P(L_j|O_j) = P(L_{j-1}|O_j) + [1 - P(L_{j-1}|O_j)]P(T)$$

3. Retornar el último valor de probabilidad calculado.

Una vez terminado este paso se tiene un conjunto de datos con llave estudiante-habilidad donde se posee el valor de la probabilidad de que el estudiante domine dicha habilidad.

2.4.5 Paso 5 Visualizar resultados

Este paso se encarga de visualizar los resultados derivados del paso anterior. La información se representará en diferentes tipos de gráficos representando conocimientos diversos. Para la visualización se utilizará la biblioteca gráfica JFreeChart[11].

Unos de los gráficos representados es el de habilidades a través de un gráfico de burbuja. En el mismo se visualizará por habilidad la cantidad de estudiantes, la cantidad de problemas y el promedio de dominio de la habilidad para todos los estudiantes (radio de la burbuja).

Para poder realizar los gráficos es necesario procesar la información obtenida para adecuarla a su representación visual. En el caso del gráfico de habilidades se obtendrán los parámetros habilidad, cantidad de problemas, cantidad de estudiante y el promedio de habilidad (ver Figura 29, solamente se representan en la imagen los primeros 20 resultados). En la Tabla 11 se presenta el seudocódigo para obtener esta información.

Tabla 11 Seudocódigo para obtener los datos para visualizar gráfico de burbuja.

Entrada	Conjunto de datos de las probabilidades de dominio por par estudiante-habilidad.
Salida	Conjunto de datos listo para visualizar (Figura 29)

1. Agrupar el conjunto de datos por la llave *habilidad* agregando un conteo de los estudiantes, y el promedio de las probabilidades para esa habilidad.
2. Agrupar el conjunto de datos inicial (salida del algoritmo en Tabla 3) por la llave *habilidad* agregando un conteo de la cantidad de problemas.
3. Juntar (*join*) los conjuntos de datos obtenidos en los pasos 1 y 2, por la llave *habilidad*

[11] http://www.jfree.org/jfreechart/

4. Retornar conjunto de datos resultante.

habilidad	count_problem	count_estudiante	avg_prob
ALT:TRIANGLE-SIDE	194	31	0.8336255768469941
ALT:PARALLELOGRAM...	806	45	0.9799971295481011
ALT:CIRCLE-RADIUS	293	35	0.9181172627394458
ALT:PARALLELOGRAM...	186	33	0.9393258763650231
ALT:PENTAGON-AREA	184	35	0.9040856106273961
ALT:CIRCLE-AREA	563	39	0.9708077508660576
ALT:TRIANGLE-AREA	378	37	0.9699360642820161
ALT:COMPOSE-BY-AD...	656	44	0.9445138639237194
ALT:CIRCLE-CIRCUM...	271	35	0.9777442011276605
ALT:CIRCLE-DIAMETER	264	35	0.8790610871881608
ALT:TRAPEZOID-AREA	150	37	0.7791070824432661
ALT:PENTAGON-SIDE	361	34	0.9083701625477026
ALT:TRAPEZOID-BASE	147	35	0.7918728262895418
ALT:COMPOSE-BY-MU...	500	34	0.8614983992246082
ALT:TRAPEZOID-HEIGHT	151	35	0.8370217666886987

Figura 29 Datos para representar en el gráfico de burbuja. Fuente: Elaboración propia.

Una vez obtenido los datos a representar, entonces se visualiza el gráfico de burbuja siguiendo el procedimiento seguido en la Tabla 12. En el mismo se muestra en cada burbuja la etiqueta que representa la información sobre la cantidad de problema, la cantidad de estudiantes y el promedio de probabilidad (ver Figura 30).

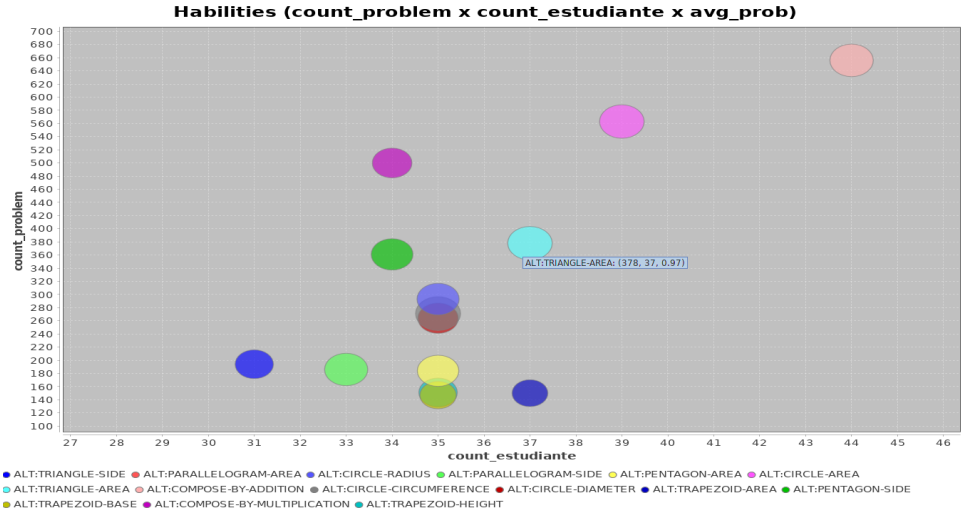


Figura 30 Ejemplo de un gráfico de burbuja de las habilidades. Fuente: Elaboración propia.

Tabla 12 Seudocódigo para obtener el gráfico a representar.

Entrada	Conjunto de datos a visualizar
Salida	Gráfico de JFreeChart
1. Crear conjunto de datos para el gráfico 2. Recolectar los datos presentes en el conjunto de datos. 3. Para cada tupla del conjunto de datos para el gráfico hacer a. Obtener valores de la tupla actual (habilidad, cantidad de estudiantes, cantidad de problemas, promedio de la probabilidad de dominio para cada habilidad). b. Añadir los valores de la tupla a conjunto de datos del gráfico. 4. Crear gráfico de JFreeChart a partir del conjunto de datos para el gráfico. 5. Retornar el gráfico	

Otras de las gráficas a visualizar es las probabilidades de dominio del estudiante por cada habilidad. La misma se muestra a través de un gráfico de radar. Para obtener el gráfico se siguen procedimientos similares a los vistos para el gráfico anterior (Tabla 12). En este caso será de tipo *radar* quedando de la siguiente manera (Figura 31):

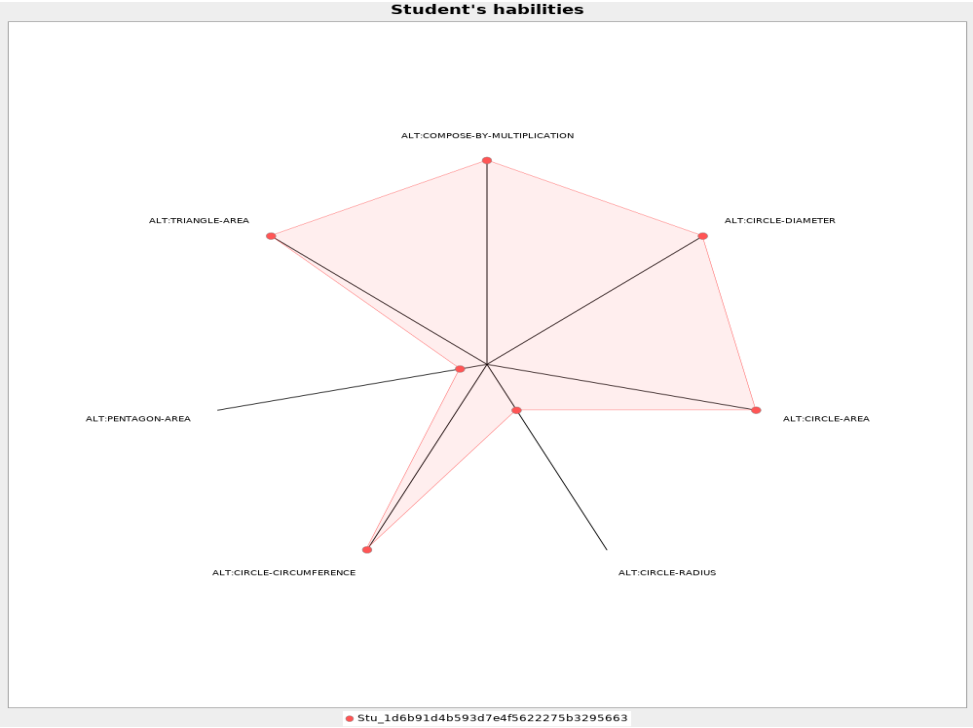


Figura 31 Gráfico de radar por habilidades del estudiante. Fuente: Elaboración propia.

Para obtener los datos de este gráfico se sigue el siguiente procedimiento:

Tabla 13 Seudocódigo para obtener los datos para el gráfico de radar.

Entrada	Conjunto de datos de las probabilidades de dominio por par estudiante-habilidad. El identificador del estudiante del que se va a representar la información.
Salida	Conjunto de datos de las probabilidades por estudiante-habilidad.
1. Filtrar las tuplas del conjunto de datos por aquellas que tengan identificador del estudiante igual al identificador recibido por parámetros. 2. Retornar conjunto de datos resultante.	

De igual manera otra gráfica que se representa es la curva de aprendizaje, que muestra el comportamiento del estudiante sobre las habilidades. En este caso los datos que se visualizan son el estudiante y las curvas de aprendizaje de cada habilidad(ver Figura 32). El procedimiento para obtener el gráfico final es similar al visto en Tabla 11. Para obtener los datos se debe seguir el siguiente procedimiento:

Tabla 14 Seudocódigo para obtener los datos para el gráfico de las curvas de aprendizaje.

Entrada	Identificador del estudiante Conjunto de datos de parámetros ajustados

Salida	Conjunto de datos con datos del gráfico

1. Filtrar el conjunto de datos de entrada por el campo estudiante igual al identificador de la entrada.
2. Crear un acumulador para obtener la secuencia de probabilidades de las habilidades por cada paso.
3. Registrar el acumulador en la sesión de Spark.
4. Para cada tupla del conjunto de datos (paso 2) hacer (de forma paralela y distribuida)
    a. Obtener los valores de la tupla actual (estudiante, habilidad, secuencia de observaciones, parámetros ajustados)
    b. Crear objeto de la clase BKT con los valores de parámetros y observaciones.
    c. Obtener a partir del objeto de BKT la secuencia de probabilidades de correctitud para cada valor de la secuencia de observaciones.
    d. Añadir los valores obtenidos al acumulador
5. Recuperar los valores del acumulador
6. Crear un conjunto de datos a partir de los valores acumulados.
7. Retornar el conjunto de datos.

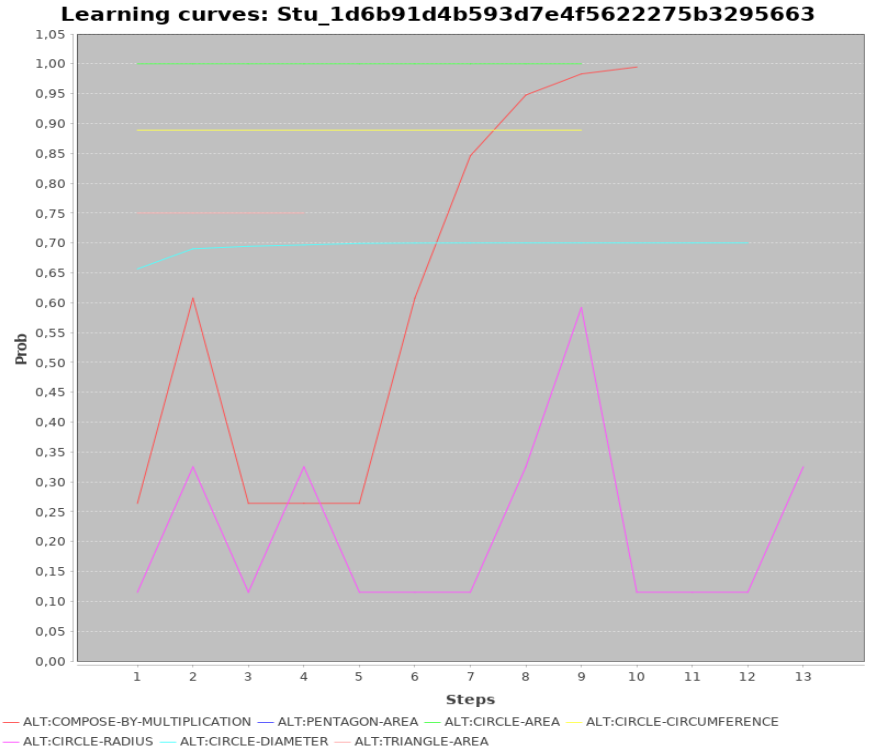


Figura 32 Gráfico de curva de aprendizaje. Fuente: Elaboración propia.

2.5 Conclusiones del capítulo

Para la propuesta de solución se tuvo en cuenta los pasos que define la MDE para el descubrimiento de conocimiento para la correcta adaptación del algoritmo al contexto de datos masivos en la educación, siendo los pasos carga de datos, pre-procesamiento, ajuste de parámetros, ejecutar algoritmo BKT y visualización de los resultados.

La utilización del marco de trabajo Apache Spark facilitó la tarea de distribuir el trabajo de cálculo en varias estaciones de trabajo para que el proceso se ejecutara en paralelo.

Con la propuesta de solución se obtiene una transformación del algoritmo BKT para que pueda ser utilizado en entornos distribuidos y pueda ejecutarse en paralelo, obteniéndose la probabilidad de dominio de las habilidades por estudiante.

Capítulo 3. Validación de la propuesta

En este capítulo se realizan las validaciones del Rastreo del Conocimiento Bayesiano Adaptado sobre datos educacionales masivos. Se realiza un experimento sobre un entorno minado, calculando el speedup y eficiencia del nuevo algoritmo; comprobando así que el algoritmo propuesto cumple con las mejoras propuestas. Además, se realiza otro experimento calculándose el AUC para ambos algoritmos (secuencial y paralelo) y el diferencial entre los valores de probabilidad obtenidos; ambas métricas para comprobar la eficacia en la solución propuesta.

3.1 Solución propuesta

La adaptación del algoritmo BKT para estimar conocimiento latente sobre datos educacionales masivos se puede acceder mediante una petición por el protocolo HTTP. Desde el clúster master se realiza la ejecución del algoritmo donde se carga el dataset (como parte del primer paso del algoritmo). Una vez cargado el dataset, se realiza el procesamiento en paralelo en el entorno distribuido definido, pasando por cada uno de los pasos siguiente del algoritmo (pre-procesamiento, ajuste de parámetros, estimar BKT y visualizar los resultados), teniendo como resultado el cálculo de la probabilidad de dominio de los estudiantes por cada habilidad. Este aporte práctico permite que las plataformas educativas donde se genera grandes volúmenes de datos que se pueda estimar conocimiento latente, para predecir el rendimiento de los estudiantes sobre dichas habilidades, y poder tomar decisiones sobre el desempeño del estudiante.

A continuación, se presenta una vista donde se muestra el algoritmo integrado en la capa de servicio REST, ver Figura 33:

Figura 33 Vista general de la propuesta de solución. Fuente: Elaboración propia.

En la Figura 34 y Figura 35 se muestra la salida del paso de pre-procesamiento.

Figura 34 Salida del paso pre-procesamiento en la plataforma de servicio. Fuente: Elaboración propia.

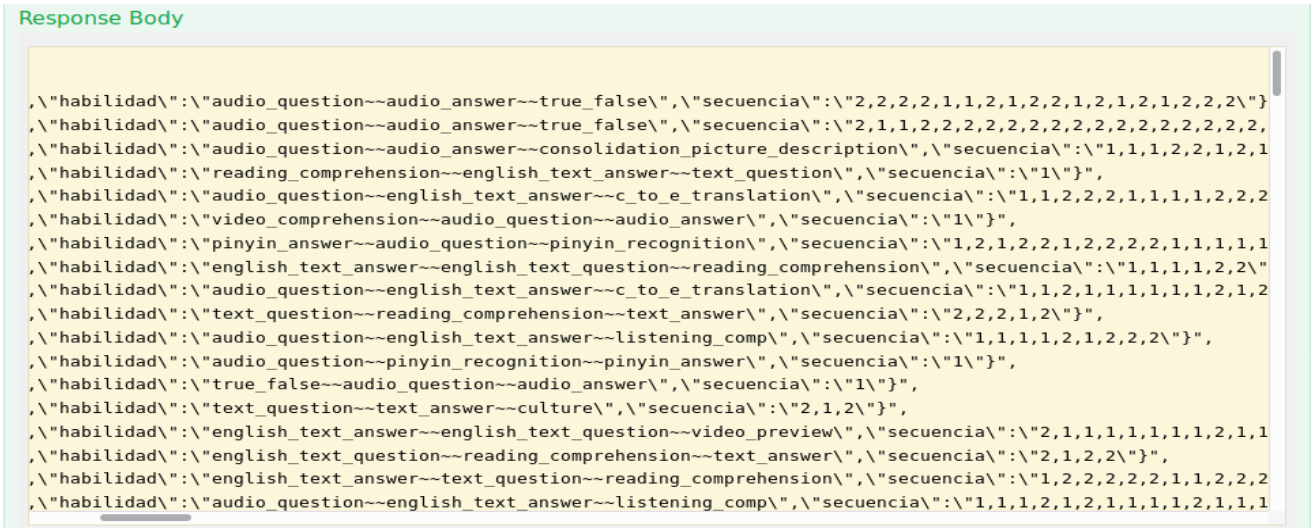


Figura 35 Salida del paso pre-procesamiento en la plataforma de servicio. Fuente: Elaboración propia.

Unos de los resultados del método es obtener la probabilidad de dominio de las habilidades por estudiantes, en la Figura 36 se muestra el resultado del cálculo utilizando el algoritmo en el entorno minado (se muestra los primeros 20 resultados). Al final de la imagen se muestra el tiempo requerido para calcular estas probabilidades (milisegundos).

```
19/06/21 16:58:51 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
19/06/21 16:58:51 INFO Utils: Successfully started service 'SparkUI' on port 4042.
19/06/21 16:58:51 INFO SparkUI: Bound SparkUI to 10.35.30.100, and started at http://10.35.30.100:4042
19/06/21 16:58:51 INFO SparkContext: Added JAR file:/opt/infoobjects/dataset/bkta2.0-1.0-SNAPSHOT.jar at spark://10.35.30.100:40733/jars/b
kta2.0-1.0-SNAPSHOT.jar with timestamp 1561150731529
19/06/21 16:58:51 INFO Executor: Starting executor ID driver on host localhost
19/06/21 16:58:51 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 36025.
19/06/21 16:58:51 INFO NettyBlockTransferService: Server created on 10.35.30.100:36025
19/06/21 16:58:51 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
19/06/21 16:58:51 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.35.30.100, 36025, None)
19/06/21 16:58:51 INFO BlockManagerMasterEndpoint: Registering block manager 10.35.30.100:36025 with 366.3 MB RAM, BlockManagerId(driver,
10.35.30.100, 36025, None)
19/06/21 16:58:51 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.35.30.100, 36025, None)
19/06/21 16:58:51 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.35.30.100, 36025, None)
19/06/21 16:58:51 INFO SharedState: Warehouse path is 'file:/opt/infoobjects/spark/bin/spark-warehouse/'.
+------------------+------------------+-----------------+
|          estudiante|          habilidad|      probabilidad|
+------------------+------------------+-----------------+
|Stu_7983017826033...|Equivalent-Fracti...|              1.0|
|Stu_798fd2c225022...|          Percent-Of|              0.1|
|Stu_7b162ffe9cec8...|    Ordering-Numbers|              1.0|
|Stu_7b162ffe9cec8...|Prime-Number~~Div...|              1.0|
|Stu_7bd89113f43d9...|         Probability|              0.0|
|Stu_7bea4bcdc0070...|Fraction-Multipli...|              1.0|
|Stu_7c06a262b0be3...|          Percent-Of|0.3828571428571428|
|Stu_7cba16873d5b3...|Inducing-Function...|0.8846153846153846|
|Stu_7d42655499008...|Graph-Shape~~Qual...|              1.0|
|Stu_7d47e4e7664b9...|Least-Common-Mult...|              1.0|
|Stu_7d54c7453212e...|Percent-Of~~Discount|            0.875|
|Stu_7e655a920cd0d...|     Adding-Decimals|              0.1|
|Stu_7e8f54e709184...|Pattern-Finding~~...|              0.1|
|Stu_7ee812f03ba32...|Inducing-Function...|              0.1|
|Stu_7efd826b8728f...|Fraction-Multipli...|              0.1|
|Stu_7efd826b8728f...|           Perimeter|            0.875|
|Stu_7f46fedbb6353...|         Measurement|              0.0|
|Stu_7f876207cc5d7...|Evaluating-Functions|0.9999999930036916|
|Stu_7faa2664a6a0e...|Probability~~Prop...|              1.0|
|Stu_7fb519e91f2ba...|Ordering-Numbers~...|              0.1|
+------------------+------------------+-----------------+
only showing top 20 rows

TIME: 14292
```

Figura 36 Salida del paso ejecutar estimación en el entorno de minado. Fuente: Elaboración propia.

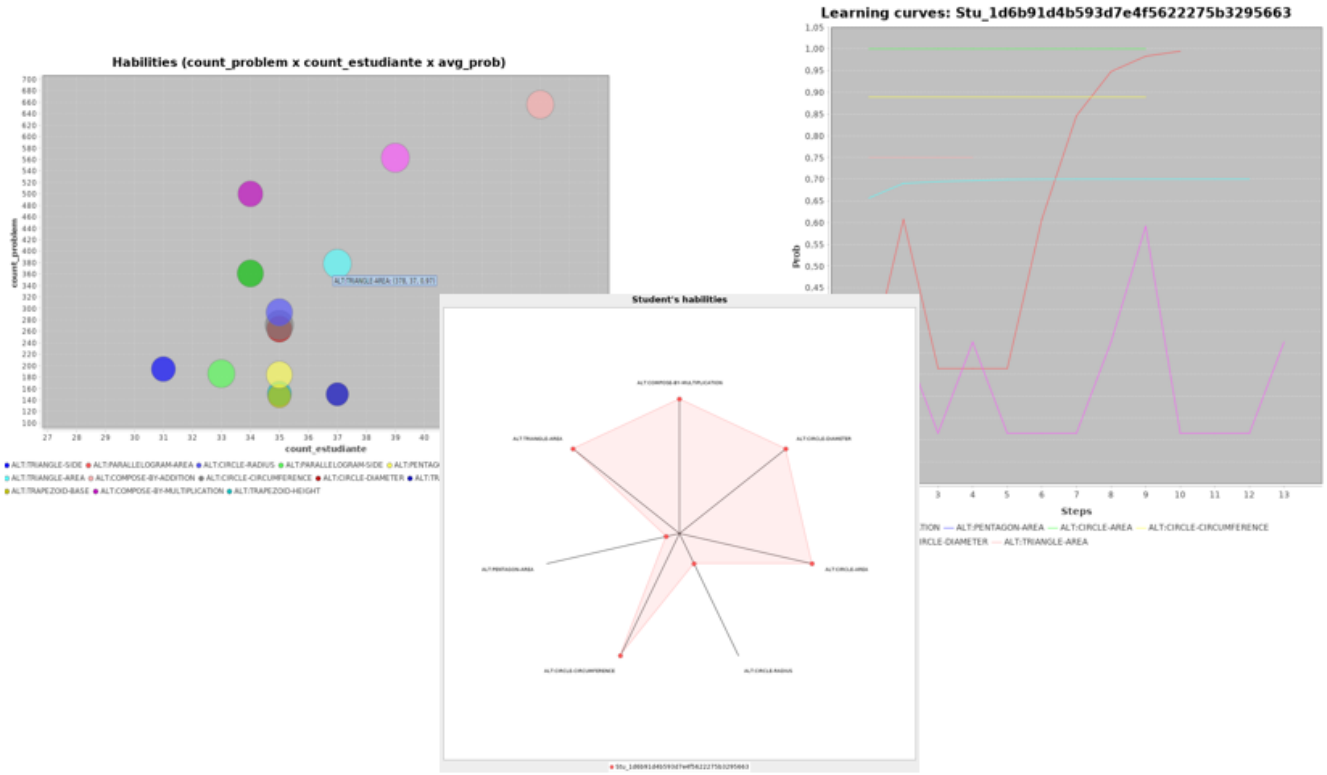


Figura 37 Ejemplo de visualización de resultado. Fuente: Elaboración propia.

En la Figura 37 se muestra ejemplos de gráficos obtenidos durante la ejecución del algoritmo adaptado.

3.2 Valoración del costo computacional del algoritmo propuesto

El área de procesamiento paralelo se ha convertido en la clave dentro de las ciencias de la computación, debido al creciente interés por el desarrollo de soluciones a problemas con muy alta demanda computacional y de almacenamiento. El desafío se centra en cómo aprovechar las prestaciones obtenidas a partir de la evolución de las arquitecturas físicas.

Cuando se paraleliza un algoritmo debe tenerse en cuenta medir el rendimiento computacional, donde dicho proceso sea eficiente y rápido, procesándose la mayor cantidad de datos posibles en un menor tiempo. Dentro de estas métricas esta calcular la velocidad de procesamiento (speedup) y eficacia. Para la propuesta de solución se realizó un estudio de los diferentes enfoques que permiten calcular el rendimiento del algoritmo, dentro de los que se encuentra la Ley de Amdalh y Ley de Gustafson. Estos enfoques definen modelos matemáticos que permiten calcular el incremento de velocidad de los sistemas de información, donde una parte de estos puede ser mejorada o paralelizable, para poder ser ejecutada por más de una unidad de procesamiento de manera simultánea [65].

3.2.1 Ley de Amdahl

El enfoque propuesto por Gene M. Amdahl conocido como Ley Amdahl, distingue entre las partes de un programa que son paralelizable α_{par} y las que son intrínsecamente secuenciales que nunca se puede paralelizar denotado por α_{seq}, de manera que se cumple que: $\alpha_{par} + \alpha_{seq} = 1$ [65].

El tiempo que demora en ejecutarse el algoritmo paralelo con P procesadores sería el siguiente:

$$t_p = \alpha_{par} \frac{t_1}{P} + \alpha_{seq} t_1 \quad (19)$$

Asumiendo que $t_{seq} = t_1$, representa el tiempo que demora el algoritmo secuencial en ejecutarse. Por lo que el speedup para P procesadores se calcula:

$$speedup\ (P) = \frac{t_1}{t_P} \quad (20)$$

Donde t_P representa el tiempo de ejecución del algoritmo paralelo con P procesadores.

Si el número de elemento de procesamiento tiende a infinito $(P \to \infty)$, el speedup es siempre superior dado por la proporción $\alpha_{seq} = 1 - \alpha_{par}$ de código que no puede ser paralelizado. Por lo que, si el límite del $speedud\ (P)$ cuando P tiende a infinito nunca pasará a la razón $\frac{1}{\alpha_{sec}}$ quedando:

$$\lim_{P \to \infty} speedup(P) = \frac{1}{1 - \alpha_{par}} \quad (21)$$

El incremento de velocidad de un programa utilizando múltiples procesadores en computación distribuida está limitada por la fracción secuencial que no es paralelizada, ya que estas pueden formar cuello de botella que influye en el speedup cuando se incrementa el número de procesadores en la solución.

3.2.2 Ley de Gustafson

Por otro lado, la Ley de Gustafson nos brinda otro punto de vista de cálculo del speedup. Este enfoque propuesto no asume que el tamaño de los datos es una constante, sino que depende del número de procesadores que tenga disponible. En la medida que aumentan los datos se incrementa el número de procesadores para mantener un tiempo de ejecución constante.[65]

Para el cálculo del speedup entonces se utiliza:

$$S(P) = \alpha_{seq} + P\alpha_{par} \quad (22)$$

De esta forma a medida que crece el tamaño del problema para un α_{seq} constante si aumenta el número de procesadores aumenta la ganancia de velocidad $S(P)$.

A partir de que las mediciones se realizarán tomando datos de tamaño fijo se aplicará el enfoque que brinda la ley de Amdhal y Gustafson. El primero permitirá medir el Speedup conociendo los tiempos de

ejecución a posteriori del algoritmo secuencial y su solución paralelizada. Con el Speedup obtenido se procederá a calcular la eficiencia de la solución y qué tanto mejora el tiempo de esta en la medida que aumentan los procesadores como propone Gustafson.

3.3 Validación del algoritmo propuesto

Una vez seleccionada las métricas a utilizar, se realiza la validación experimental del mismo. Para ello, primero se realiza un muestro de diferentes conjuntos de datos que permitirán ejecutar el algoritmo propuesto. Se describe el entorno de prueba, se ejecuta el algoritmo para los diferentes datasets, y finalmente se realiza un análisis de los resultados obtenidos.

3.3.1 Muestreo

Para el experimento se utilizarán diferentes conjuntos de datos públicos pertenecientes al repositorio de base de datos educacionales PSLC Datashop, disponible en http://pslcdatashop.org[66]. Los datos estarán organizados de forma tal que en cada columna se tenga la siguiente información:

- First attemp – Valor del resultado del primer intento del estudiante (1 – intento correcto, 0 – intento incorrecto).
- Anon Student Id – Identificador del estudiante.
- Concatenación de los campos Problem Hierarchy – Problem Name – Step Name. Identifica el problema donde fue aplicada la habilidad.
- Knowledge Component – Habilidad a estimar.

Los conjuntos de datos cuentan con las siguientes características:

Tabla 15 Características de los datasets.

Conjunto de datos	Número de estudiantes	Número de componentes de conocimiento	Número de problemas	Cantidad de filas
OSU, Honors Physics: Mechanics, Fall 2011 (dataset 1)	314	154	44	323,912
USNA Physics Fall 2006 (dataset 2)	66	250	251	345,536
ElemChinese (dataset 3)	221	22	868	812,329
Assistments Math 2006-2007 (5046 Students) (dataset 4)	5,046	318	1872	1,451,003

La elección de estos datasets fue de forma dirigida tomando en cuenta el tamaño y cantidad de filas que poseen. Estos conjuntos de datos no son de gran volumen, pero su elección se debe a que para poder realizar la comparación entre los algoritmos secuencial y paralelo fue necesario utilizar aquellos que pudieran ser ejecutados por el algoritmo secuencial. Como se puede apreciar fueron elegidos 4 datasets públicos en esta base de datos para comprobar el rendimiento y eficacia del algoritmo adaptado.

3.3.2 Diseño de los experimentos

Para la realización de las pruebas se desplegó la propuesta de solución en un clúster de computadoras usando la herramienta Apache Spark en su versión 2.2.0. Se aplicó el modo Standalone de despliegue, donde se utilizan las funciones nativas de la herramienta. El clúster estaba conformado por una estación de trabajo que actuó como máster y dos estaciones de trabajo utilizadas como nodos trabajadores. Se debe tener en cuenta que las pruebas fueron realizadas en un entorno no dedicado de nodos que pertenecen a la misma subred.

Para ejecutar el algoritmo secuencial se utilizó una computadora con las siguientes características de hardware y software.

Tabla 16 Especificaciones de hardware y software para ejecutar el algoritmo secuencial.

Estación de trabajo usada para ejecutar el algoritmo secuencial		
Tipo de procesador	**Cantidad de núcleos**	**Memoria principal**
Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz	8	8 Gb DDR3 1600

Tabla 17 Especificaciones de software y sistema operativo.

Sistema operativo
Ubuntu 18.04 LTS 64 bits, versión del núcleo: 4.15.0-43-generic
Software necesarios instalados
Java OpenJDK versión "8" Update 192
Spark 2.2.0

Para las pruebas del algoritmo adaptado se utilizó un entorno de minado que consiste de las siguientes características:

Tabla 18 Especificaciones de hardware del clúster de computadoras empleado para las pruebas.

Estación de trabajo máster		
Tipo de procesador	**Cantidad de núcleo**	**Memoria principal**
Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz	8	8 Gb DDR3 1600

Estaciones de trabajo usadas como nodos trabajadores		
Tipo de procesador	**Cantidad de núcleo**	**Memoria principal**
Intel(R) Celeron(R) CPU G1830 @ 2.80GHz	2	4 Gb DDR3
Intel(R) Celeron(R) CPU G1830 @ 2.80GHz	2	4 Gb DDR3
Características de la red de datos		
100 Mb/s		

Tabla 19 Especificaciones de software y sistema operativo del entorno distribuido.

Sistema operativo
Ubuntu 18.04 LTS 64 bits, versión del núcleo: 4.15.0-43-generic
Software necesarios instalados
Java OpenJDK versión "8" Update 192
Spark 2.2.0

Para analizar si el algoritmo cumple con las expectativas de mejora requeridos, se ejecutará el mismo para los diferentes conjuntos de datos en el entorno de minado y se analizará principalmente dos aspectos: primeramente, que el algoritmo adaptado mejora los tiempos de ejecución y en segundo lugar que la eficacia en el algoritmo se mantiene sin cambios significativos respecto al algoritmo original.

En la prueba de los algoritmos secuencial y paralelo se utilizarán 4 dataset de diferentes tamaños. Para el algoritmo secuencial se harán de 10 ejecuciones por cada dataset recogiéndose el tiempo de ejecución y obteniendo la probabilidad de dominio de la habilidad por cada estudiante y el AUC. Lo mismos se hará con el algoritmo paralelo, pero utilizando el entorno minado, con 10 ejecuciones por cada dataset, recogiendo el tiempo de ejecución, la probabilidad de dominio de la habilidad por cada habilidad y el AUC. A partir de esos valores entonces se procederá al cálculo de speedup y eficiencia. Para la eficacia, se utilizará el Error Cuadrático Medio (ECM), para verificar que no exista diferencia significativa entre los resultados arrojados por el algoritmo secuencial y el paralelo.

3.3.3 Análisis de los resultados

Luego de realizadas las diferentes pruebas se obtienen los siguientes resultados:

Tiempo de ejecución

Para el análisis de los resultados y comprobación del mismo se plantea como hipótesis:

H_0: El tiempo de ejecución del algoritmo paralelo es mejor que el algoritmo secuencial.

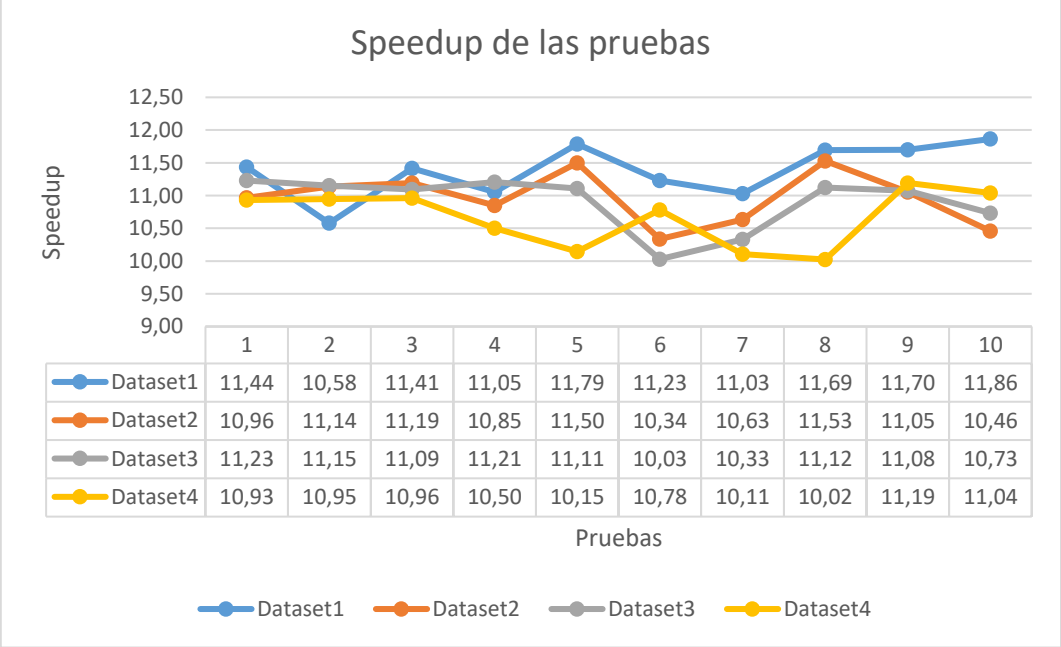
H_1: El tiempo de ejecución del algoritmo paralelo es igual o peor que el algoritmo secuencial.

Para los conjuntos de datos en la Tabla 15, se puede medir el tiempo que demora la ejecución del algoritmo en paralelo utilizando el entorno minado descrito anteriormente. Para cada dataset se realiza 10 ejecuciones, donde el comportamiento del speedup se encuentra en los rangos entre 10 y 12. Estos valores de aceleración indican que el algoritmo adaptado muestra buenos niveles de aceleración respecto al secuencial para este entorno (que tiene un total de 12 núcleos) (Figura 38).

Las eficiencias de las pruebas ejecutadas para cada dataset oscilan entre 84 y 99 porciento. Lo que muestra un buen aprovechamiento de la capacidad del entorno (Figura 39).

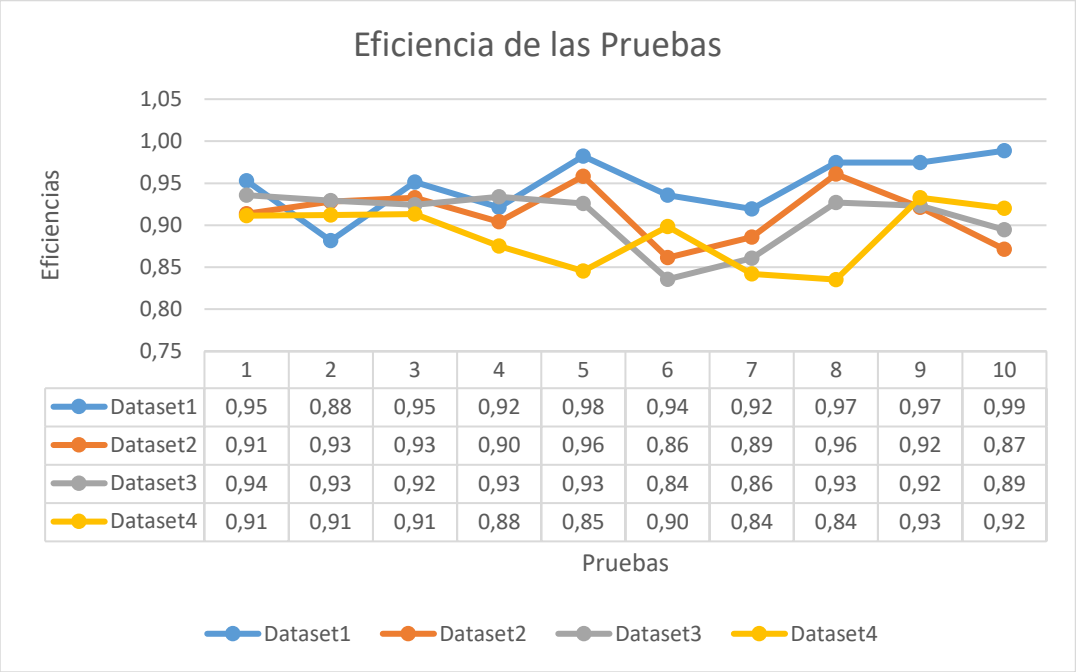
En las siguientes figuras (Figura 40 y Figura 41) se muestran los resúmenes de speedup y eficiencia para los datasets utilizados. En estas gráficas se muestran los casos mínimos, máximos y promedios de ambas métricas. Como se muestra en las figuras el speedup y eficiencia disminuye ligeramente al aumentar el tamaño de los conjuntos de datos.

Se puede concluir que no se rechaza la hipótesis nula H_0, porque en las pruebas realizadas para el algoritmo adaptado con los conjuntos de datos seleccionados en el entorno de minado configurado, los valores obtenidos de aceleración y eficiencia permiten afirmar que el algoritmo adaptado presenta una mejora importante de tiempo de ejecución respecto al algoritmo secuencial.



	1	2	3	4	5	6	7	8	9	10
Dataset1	11,44	10,58	11,41	11,05	11,79	11,23	11,03	11,69	11,70	11,86
Dataset2	10,96	11,14	11,19	10,85	11,50	10,34	10,63	11,53	11,05	10,46
Dataset3	11,23	11,15	11,09	11,21	11,11	10,03	10,33	11,12	11,08	10,73
Dataset4	10,93	10,95	10,96	10,50	10,15	10,78	10,11	10,02	11,19	11,04

Figura 38 Comportamiento del speedup por cada ejecución por dataset. Fuente: Elaboración propia.



	1	2	3	4	5	6	7	8	9	10
Dataset1	0,95	0,88	0,95	0,92	0,98	0,94	0,92	0,97	0,97	0,99
Dataset2	0,91	0,93	0,93	0,90	0,96	0,86	0,89	0,96	0,92	0,87
Dataset3	0,94	0,93	0,92	0,93	0,93	0,84	0,86	0,93	0,92	0,89
Dataset4	0,91	0,91	0,91	0,88	0,85	0,90	0,84	0,84	0,93	0,92

Figura 39 Eficiencia de las pruebas por dataset. Fuente: Elaboración propia.

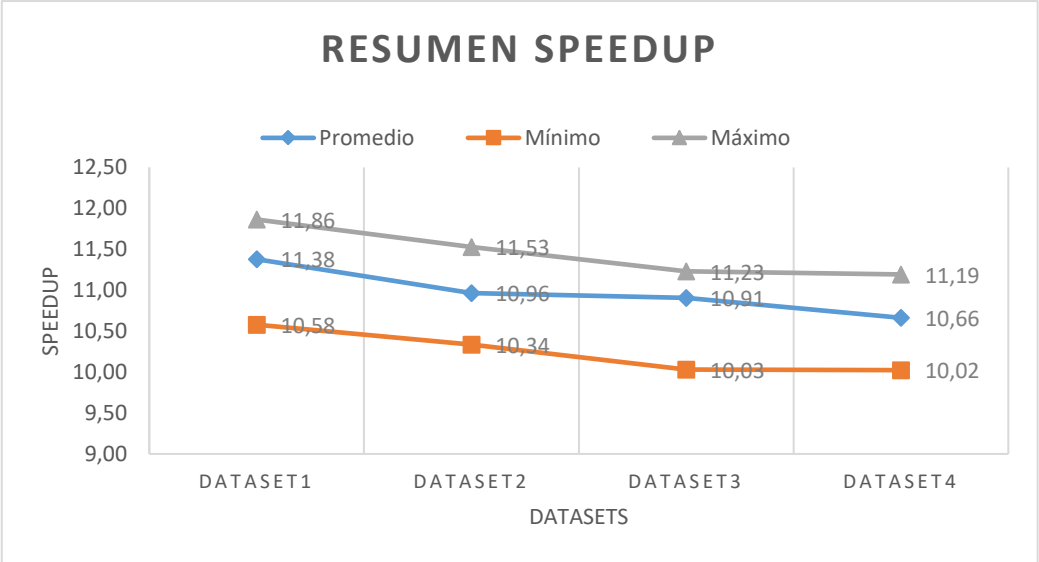


Figura 40 Resumen del speedup. Fuente: Elaboración propia.

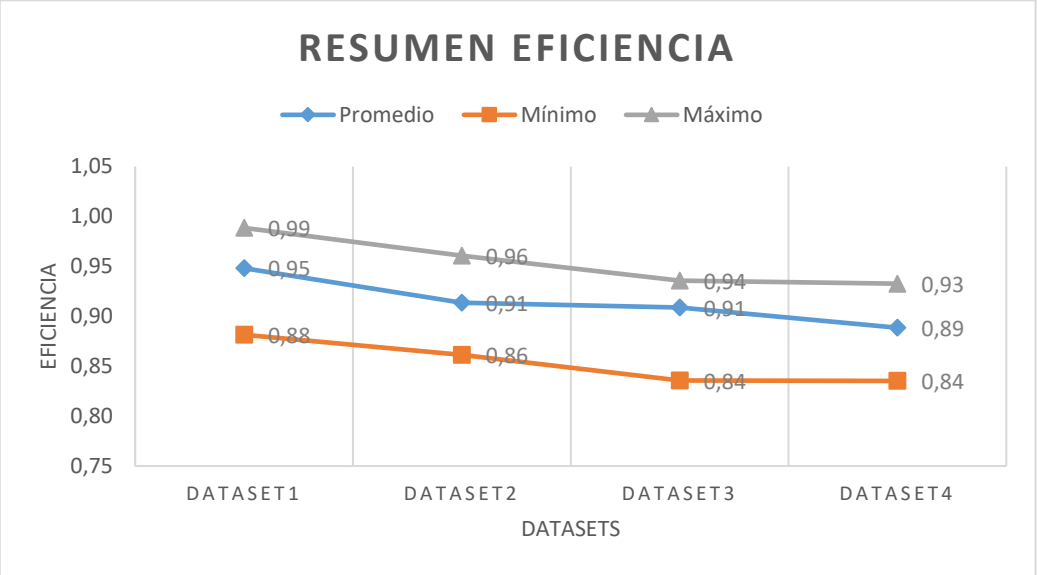


Figura 41 Resumen de la eficiencia. Fuente: Elaboración propia.

Eficacia

Para comprobar la eficacia del algoritmo adaptado se tomará en cuenta dos métricas a medir que son el Error Cuadrático Medio (ECM) de los valores calculados de la probabilidad de dominio de las habilidades y la ECM del área bajo la curva ROC (AUC). Ambas métricas serán medidas inicialmente por el algoritmo secuencial y luego por el paralelo.

Para calcular la ECM de la probabilidad de dominio de las habilidades se realiza el experimento ejecutando para un dataset el algoritmo en secuencial y luego en paralelo. Ambas ejecuciones dan como resultado la lista de estudiantes, habilidad y la probabilidad de dominio (Figura 36). Luego se haya el ECM para esa prueba tomando las probabilidades resultantes de cada ejecución.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(ProbSecuencial_i - ProbParalelo_i)^2 \quad (23)$$

Donde n es la cantidad de tuplas de los datasets resultantes de la ejecución del algoritmo (secuencial y paralelo), $ProbSecuencial_i$ es la probabilidad de dominio obtenida por el algoritmo secuencial para la tupla i, y $ProbParalelo_i$ es la probabilidad de dominio de la habilidad obtenida por el algoritmo paralelo en la tupla i.

En la Tabla 20 se muestra los cálculo de este indicador para las pruebas realizadas a los diferentes conjuntos de datos, en la misma se muestra los valores de ECM para cada prueba en cada conjunto de datos.

Tabla 20 Resumen del cálculo de ECM de las probabilidades resultantes de las pruebas realizadas.

Ejecuciones	Dataset1	Dataset2	Dataset3	Dataset4
1	0,0003	0,0012	0,00263	0,000788
2	0,0017	0,00086	0,00075	0,000665
3	0,00087	0,00045	0,00066	0,000222
4	0,00024	0,00099	0,000143	0,000115
5	0,00083	0,00154	0,000551	0,00552
6	0,00036	0,00241	0,00233	0,00261
7	7,50E-05	0,00017	0,00472	0,0047
8	0,00057	0,000264	0,000887	0,000444
9	0,00071	0,00167	0,000333	0,000768
10	0,0016	0,00201	0,00402	0,00986
AVG	0,0007255	0,0012564	0,0007646	0,0005452
MAX	0,0017	0,00241	0,00263	0,00081
MIN	0,000075	0,000264	0,000143	0,000115
STDEV	0,00055218	0,00066867	0,00069362	0,00024338

Para comprobar que para los diferentes conjuntos de datos no hubo diferencias significativas se realizó una verificación de varianza usando una prueba de Levene.

Tabla 21 Prueba de verificación de varianza.

	Prueba	*Valor-P*
Levene's	2,15177	0,110739

Los estadísticos mostrados en esta tabla evalúan la hipótesis nula de que las desviaciones estándar dentro de cada una de las 4 columnas de dataset son iguales. De particular interés es el valor-P. Puesto que el valor-P es mayor o igual que 0,05, no existe una diferencia estadísticamente significativa entre las desviaciones estándar, con un nivel del 95,0% de confianza.

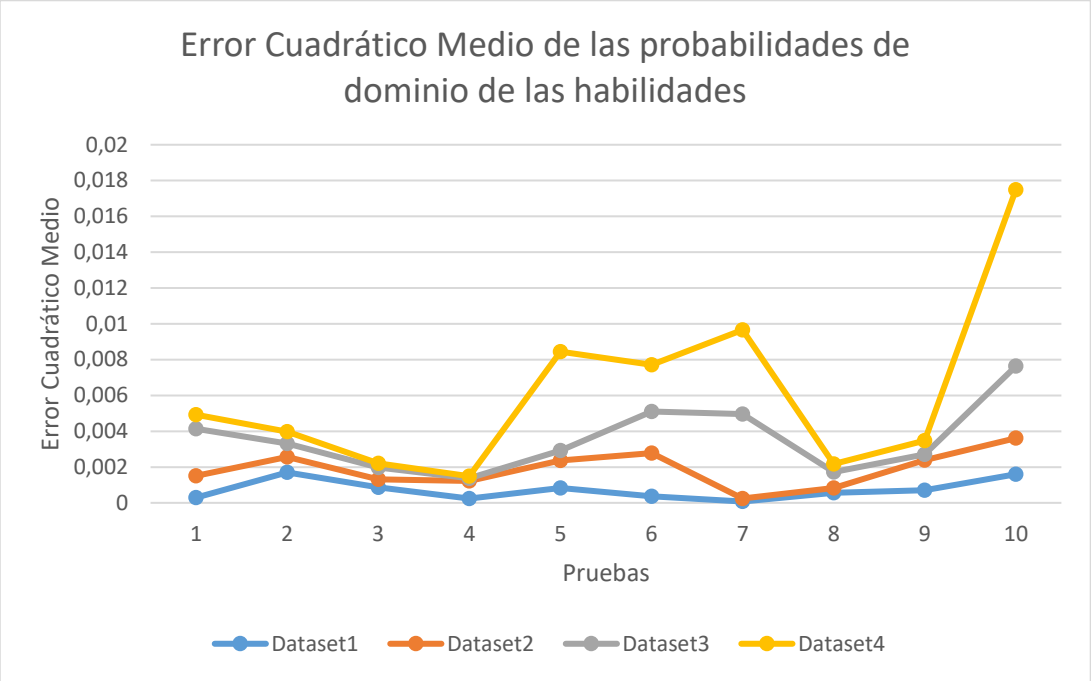


Figura 42 Valores de ECM de las probabilidades para las pruebas realizadas. Fuente: Elaboración propia.

También se realizó una prueba de Análisis de la Varianza con las medias de los resultados de las probabilidades de dominio de las habilidades y AUC entre el algoritmo secuencial y el paralelo. Se plantea como hipótesis:

$H_0: \mu_1 = \mu_2 = \mu_3 = \mu_4$ (las medias no difieren significativamente entre los resultados de la probabilidad de los dataset)

$H_1: \mu_i \neq \mu_j$ para alguna $i \neq j$ (las medias difieren significativamente entre los resultados de la probabilidad de los dataset)

Dado el estadígrafo de prueba ANOVA se muestra los siguientes resultados:

Tabla 22 Tabla ANOVA de la probabilidad de dominio de la habilidad.

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	0,0000189618	3	0,00000632058	1,83	0,1592
Intra grupos	0,000124361	36	0,00000345446		
Total (Corr.)	0,000143322	39			

Una vez obtenido el estadígrafo se calcula la región crítica de la siguiente manera:

$$F > F_{\alpha;k-1;n-k}$$

$$F > F_{0.05;3;36}$$

Donde F es la *Razón-F* que se encuentra en la Tabla 22 y $F_{\alpha;k-1;n-k}$ región crítica.

Buscando en la tabla de Fisher [67] el valor de la región crítica, queda de la siguiente forma $F > 2.84$, siendo la *Razón-F* igual a 1.83, entonces como:

Decisión estadística:

Como el estadígrafo de prueba no cae en la región crítica no rechazamos la hipótesis nula con un nivel de significación de 0.05.

Puesto que el valor-P de la razón-F es mayor o igual que 0,05, no existe una diferencia estadísticamente significativa entre las medias de las 4 variables con un nivel del 95,0% de confianza.

Decisión práctica:

No difieren significativamente las medias de las muestras, lo que significa que se mantiene los resultados entre los dataset.

Luego se puede concluir que no existen diferencias significativas para los resultados obtenidos para cada conjunto de datos. En cada una de las pruebas realizadas el valor del diferencial es menor que 0.005 por lo que la eficacia respecto al cálculo de las probabilidades de dominio se mantiene para el algoritmo paralelo.

Para computar el ECM para el AUC se sigue un procedimiento parecido al realizado para el de las probabilidades de dominio. En este caso lo que varía es que previamente se debe computar el AUC para cada estudiante, obteniéndose un conjunto de datos con los estudiantes y la métrica del AUC para cada estudiante(Figura 43). Este cálculo se realiza con los resultados del algoritmo de forma secuencial y con los datos del algoritmo de forma paralela. Luego se sigue un procedimiento similar calculando el ECM, según la siguiente fórmula:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(AUCSecuencial_i - AUCParalelo_i)^2 \quad (24)$$

```
+----------------------------------+------------------+
|estudiante2                       |auc               |
+----------------------------------+------------------+
|Stu_4e8a5be7f2663d2ccccfb59c684d5452|0.812824314306894 |
|Stu_1afbee9e3e83bb801b589108fb46028a|0.6458333333333334|
|Stu_98e7261dfe23e96d8982adfd203aa086|0.5625            |
|Stu_21b2fcd580ae27d8b9a7a067d10825d0|0.65              |
|Stu_87d1548f2cd78b7b8e334802016e700e|0.8065004659832246|
|Stu_2ebe6a7530ff11f2c0b9b807faf0a0a3|0.890993265993266 |
|Stu_e225ceb425b9c40773ca12f5905514aa|1.0               |
|Stu_ad3610752c4af1c3cac6638ef588e02b|0.789264705882353 |
|Stu_c451386562a85a0cc50c36d6df6db731|0.7468833908707326|
|Stu_e242582bd5729de82485e42d69e22109|0.8398214285714286|
|Stu_d7f18a5fa205a889b0c5b0b56a7127d3|0.7864130434782609|
|Stu_e49290be25bd7ce478394c7becc6c70c|0.5472222222222223|
|Stu_e0e22d49d6bb0a1fe71e19d8d6022lae|0.6880706025562995|
|Stu_1b394ff128c045b7d4ad4f6e83933b72|0.8508928571428571|
|Stu_b008b6015fa6d5a8fa6a5ec1ce0405aa|0.8125            |
|Stu_f01773c1237a0acdf4170202bf5b5c32|0.8743346447581578|
|Stu_8150b92d145a08727bc5a1e77eade01d|0.75              |
|Stu_d670d7de62ee1a748b67448da3c53d8b|0.8               |
|Stu_02ee1b3f31a6f6a7f4b8012298b2395e|0.8533611981887844|
+----------------------------------+------------------+
```

Figura 43 Ejemplo de conjunto de datos con la métrica de AUC para cada estudiante. Fuente: Elaboración propia.

Luego de ejecutadas las 10 pruebas y obtenidos los valores de AUC para los resultados obtenidos, se obtuvo los siguientes valores del ECM para las pruebas realizadas.

Tabla 23 Resumen del cálculo de ECM para el AUC de las pruebas realizadas.

Ejecuciones	Dataset1	Dataset2	Dataset3	Dataset4
1	0,00147	0,00358	0,0001254	0,001525
2	0,00245	0,0017121	0,002548	0,00036589
3	0,00548	0,0002589	0,003659	0,00058793
4	0,000555	0,0003647	0,0006895	0,002547
5	0,0021565	0,006524	0,002587	0,0036589
6	0,00145	0,003648	0,0036547	0,0006698
7	0,0003365	0,0002413	0,0003487	0,0014755
8	0,000141	0,0008974	0,00453	0,0025478
9	0,00112	0,003698	0,000585	0,0036985
10	0,000998	0,006598	0,001486	0,00289678
AVG	0,0016157	0,00275224	0,00202133	0,00199731
MAX	0,00548	0,006598	0,00453	0,0036985
MIN	0,000141	0,0002413	0,0001254	0,00036589
STDEV	0,00154681	0,00245272	0,00159012	0,00124609

Estos valores pueden apreciarse en la siguiente gráfica:

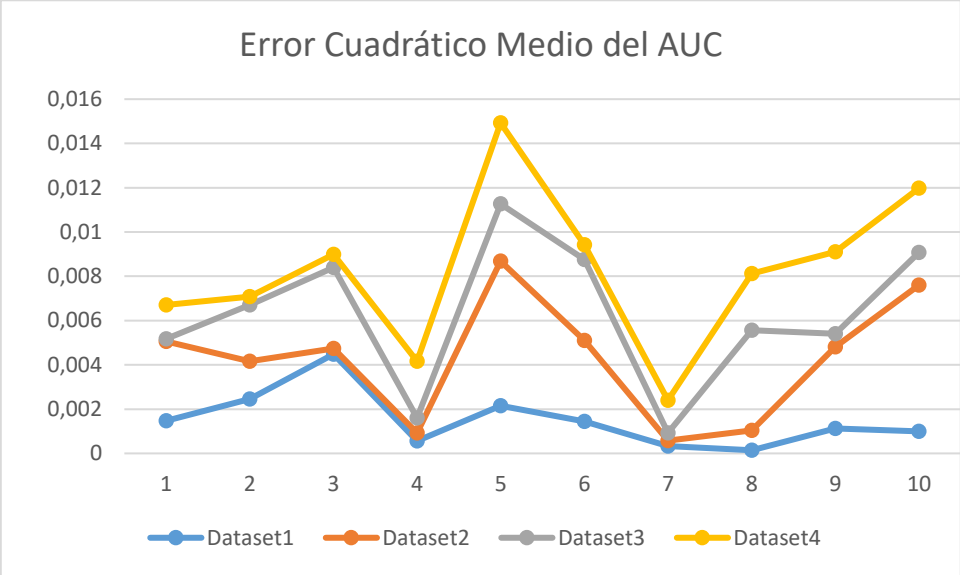


Figura 44 Valores del ECM para el AUC de las pruebas realizadas. Fuente: Elaboración propia.

Nuevamente, para comprobar que para los diferentes conjuntos de datos no hubo diferencias significativas se realizó una verificación de varianza usando una prueba de Levene.

Tabla 24 Prueba de verificación de varianza para el AUC.

	Prueba	*Valor-P*
Levene's	2,72638	0,0583378

Los estadísticos mostrados en esta tabla evalúan la hipótesis nula de que las desviaciones estándar dentro de cada una de las 4 columnas son iguales. Puesto que el valor-P es mayor o igual que 0,05, no existe una diferencia estadísticamente significativa entre las desviaciones estándar, con un nivel del 95,0% de confianza.

El Análisis de la Varianza donde se evalúa la hipótesis nula de las medias de los resultados del AUC entre el algoritmo secuencial y el paralelo, dio como resultado el estadígrafo de prueba ANOVA:

Tabla 25 Tabla ANOVA de AUC

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	0,00000676653	3	0,00000225551	0,72	0,5452
Intra grupos	0,000112407	36	0,00000312241		
Total (Corr.)	0,000119173	39			

Una vez obtenido el estadígrafo se calcula la región crítica de la siguiente manera:

$$F > F_{\alpha;k-1;n-k}$$

$$F > F_{0.05;3;36}$$

Donde F es la Razón-F que se encuentra en la Tabla 25 y $F_{\alpha;k-1;n-k}$ región crítica.

Buscando en la tabla de Fisher [67] el valor de la región crítica, queda de la siguiente forma $F > 2.84$, siendo la Razón-F igual a 0.72, entonces como:

Decisión estadística:
El estadígrafo de prueba no cae en la región crítica no rechazamos la hipótesis nula con un nivel de significación de 0.05.

Puesto que el valor-P de la razón-F es mayor o igual que 0,05, no existe una diferencia estadísticamente significativa entre las medias de las 4 variables con un nivel del 95,0% de confianza.

Decisión práctica:
No difieren significativamente las medias de las muestras del AUC, lo que significa que se mantiene los resultados entre los dataset.

Nuevamente se comprueba que no existen diferencias significativas entre los resultados para los diferentes conjuntos de datos. Lo que demuestra que la eficacia de usar el algoritmo paralelo es similar a la obtenida al usar el algoritmo secuencial. Además, como se aprecia en la tabla las diferencias entre la métrica de AUC para ambos algoritmos (secuencial y paralelo) nunca es mayor que 0,006598 lo que es un indicador de que la métrica se comporta de forma similar para ambos casos.

Luego de realizado todas las pruebas se puede afirmar que los tiempos de ejecución son mejores para el algoritmo paralelo y la eficacia se mantiene con valores similares para las métricas utilizadas.

3.4 Conclusiones del capítulo

Con el desarrollo de la propuesta se obtiene una adaptación del algoritmo BKT para estimar conocimiento latente sobre datos educacionales masivos mediante marco de trabajo Spark.

Para la prueba de ejecución del algoritmo en paralelo se configuró un entorno minado que consta de un clúster master y dos esclavos, en el mismo se ejecutaron 4 dataset de diferentes tamaños.

Se comprobó el tiempo de ejecución del comportamiento de los algoritmos secuencial y paralelo calculándose el speedup y la eficiencia, evidenciándose mejoras de algoritmo paralelo con respecto al secuencial.

La utilización de las métricas de ECM de los valores calculados de la probabilidad de dominio de las habilidades y el ECM del área bajo la curva ROC, permitió comprobar la eficacia del algoritmo adaptado, mostrándose que la eficacia se mantiene con valores similares entre los resultados de los algoritmos.

Con la comparación de los resultados de las ejecuciones de los diferentes dataset de la probabilidad de dominio de las habilidades y el AUC utilizando las pruebas de Levene y Análisis de la Varianza, se demostró que no hay diferencia significativa entre los resultados para los diferentes conjuntos de datos.

Conclusiones generales

- Con el análisis del método BKT se pudo caracterizar el proceso de estimación de conocimiento latente del algoritmo, tomando como premisas utilizar los pasos que propone la MDE para adaptar el algoritmo utilizando el marco de trabajo Apache Spark y como método de ajuste del parámetro el de probabilidad empírica.
- El algoritmo BKT adaptado cuenta con 5 pasos fundamentales cargar de los datos, pre-procesamiento, ajuste de parámetros, ejecutar el algoritmo y la visualización, el cual se ejecuta sobre un entorno minado distribuido sobre el marco de trabajo Apache Spark de forma paralela, obteniéndose las probabilidades por habilidades de cada estudiante a partir de la secuencia de observaciones por cada habilidad.
- Con la ejecución de los algoritmos sobre los 4 dataset de diferentes tamaños se pudo afirmar que el algoritmo adaptado presenta una mejora importante de tiempo de ejecución respecto a la aceleración y eficiencia con el algoritmo secuencial. Así mismo, la eficacia mantiene valores similares para las métricas del ECM utilizadas entre las probabilidades de dominio de las habilidades y el AUC.

Recomendaciones

- Crear un mecanismo de adquisición de datos a partir de diferentes sistemas e-learning para ECL de estudiantes en tiempo real.
- Adicionar la capacidad de adquirir datos desde otras fuentes adicionales (base de datos no relacionales, etc.)
- Incluir biblioteca gráfica interactiva para mejorar la representación visual de los resultados.

Glosario de términos

Apache Spark: Un sistema de computación en clúster de propósito general y orientado a la velocidad. Proporciona APIs en Java, Scala, Python y R.

Bayesian Knowledge Tracing: Es un algoritmo utilizado en muchos sistemas inteligentes de tutoría para modelar el dominio de cada alumno del conocimiento que está siendo tutelado.

Bayesian Knowledge Tracing Adaptado: Adaptación del algoritmo Bayesian Knowledge Tracing para que se ejecute en un entorno minado distribuido en forma paralela.

Big Data (Datos masisvos): Conjuntos de datos extremadamente grandes que pueden ser analizados computacionalmente para revelar patrones, tendencias y asociaciones, especialmente en relación con el comportamiento humano y las interacciones.

Estimación del Conocimiento Latente (ECL): La técnica utilizada para determinar en qué medida un estudiante domina un(os) conocimiento(s) o habilidad(es) determinada(s) en un momento dado.

Hadoop: Es un marco de software de código abierto para almacenar datos y ejecutar aplicaciones en clusters de hardware de productos básicos. Proporciona almacenamiento masivo para cualquier tipo de datos, una enorme potencia de procesamiento y la capacidad de gestionar tareas o trabajos simultáneos prácticamente ilimitados.

HDFS: Es un subproyecto del proyecto Apache Hadoop. Este proyecto de la Fundación de Software Apache está diseñado para proporcionar un sistema de archivos tolerante a fallos diseñado para ejecutarse en hardware básico.

Minería de Datos Educativos (MDE): Hace referencia a las técnicas, herramientas y la investigación diseñados para extraer automáticamente el significado de grandes repositorios de datos generados por las actividades de aprendizaje en los centros educativos.

Resilient Distributed Datasets (RDD): Representa una colección inmutable y particionada de elementos sobre los que se puede operar de forma paralela.

Receiver Operating Characteristic (ROC): Una curva ROC es una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario según se varia el umbral de discriminación.

Referencias bibliográficas

[1] C. Romero and S. Ventura, “Data mining in education,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 3, no. 1, pp. 12–27, 2013.

[2] M. del R. Martínez Torres, D. Gutiérrez Reina, S. L. Toral, and F. Barrero, “Metodologías de análisis de los big data en las plataformas educativas,” pp. 79–83, 2014.

[3] A. M. Shahiri, W. Husain, and others, “A review on predicting student’s performance using data mining techniques,” *Procedia Comput. Sci.*, vol. 72, pp. 414–422, 2015.

[4] C. Romero and S. Ventura, “Educational data mining: A review of the state of the art,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 40, no. 6, pp. 601–618, 2010.

[5] R. S. J. D. Baker and K. Yacef, “The state of educational data mining in 2009: A review and future visions,” *JEDM-Journal Educ. Data Min.*, vol. 1, no. 1, pp. 3–17, 2009.

[6] C. Romero, S. Ventura, M. Pechenizkiy, and R. Sj. Baker, *Handbook of educational data mining*. CRC press, 2010.

[7] C. Romero and S. Ventura, “Educational data mining: a review of the state of the art,” *Syst. Man, Cybern. Part C Appl. Rev. IEEE Trans.*, vol. 40, no. 6, pp. 601–618, 2010.

[8] J. A. Aristizábal, “Diseño y aportes de un modelo para minería de datos educativos en aulas de educación media de carácter presencial,” Tesis de doctorado). Universidad Santo Tomás, Bogotá, Colombia. Retrieved~, 2017.

[9] J. A. Larusson and B. White, *Learning Analytics*. Springer, 2014.

[10] C. M. Vera, C. R. Morales, and S. V. Soto, “Predicción del fracaso escolar mediante técnicas de minería de datos,” *Rev. Iberoam. Tecnol. del/da Aprendizaje/Aprendizagem*, vol. 109, 2012.

[11] R. S. Baker and A. T. Corbett, “Assessment of robust learning with educational data mining,” *Res. Pract. Assess.*, vol. 9, 2014.

[12] I. Roll, V. Aleven, B. M. McLaren, and K. R. Koedinger, “Can Help Seeking Be Tutored? Searching for the Secret Sauce of Metacognitive Tutoring.,” in *AIED*, 2007, vol. 2007, pp. 203–210.

[13] M. Feng and N. T. Heffernan, “Towards live informing and automatic analyzing of student learning: Reporting in assistment system,” *J. Interact. Learn. Res.*, vol. 18, no. 2, pp. 207–230, 2007.

[14] R. S. Baker, *Big Data and Education*, 2nd ed. New York, NY: Teachers College, Columbia University, 2015.

[15] A. B. Román, D. Sánchez-Guzmán, and R. García, “Minería de datos educativa: Una herramienta para la investigación de patrones de aprendizaje sobre un contexto educativo,” *Lat. Am. J. Phys. Educ. Vol*, vol. 7, no. 4, p. 662, 2014.

[16] Z. A. Pardos, S. M. Gowda, R. Sj. Baker, and N. T. Heffernan, “The sum is greater than the parts: ensembling models of student knowledge in educational software,” *ACM SIGKDD Explor. Newsl.*, vol. 13, no. 2, pp. 37–44, 2012.

[17] Z. A. Pardos, Y. Bergner, D. T. Seaton, and D. E. Pritchard, “Adapting Bayesian Knowledge Tracing to a Massive Open Online Course in edX.,” in *EDM*, 2013, pp. 137–144.

[18] R. S. . B. P. Berland Matthew; Baker, “Educational Data Mining and Learning Analytics: Applications to Constructionist Research,” *Technol. Knowl. Learn.*, vol. 19, no. 1–2, 2014.

[19] R. S. Baker, *Big Data and Education*, 2nd ed. New York, NY: Teachers College, Columbia University, 2015.

[20] C. M. Steiner, M. D. Kickmeier-Rust, and D. Albert, “Learning Analytics and Educational Data Mining: An Overview of Recent Techniques,” *Learn. Anal. Serious Games*, p. 6, 2014.

[21] R. S. Baker and P. S. Inventado, “Educational data mining and learning analytics,” in *Learning Analytics*, Springer, 2014, pp. 61–75.

[22] A. T. C. J. R. Anderson, “Knowledge tracing: Modeling the acquisition of procedural knowledge,” *User Model. User-adapt. Interact.*, vol. 4, no. 4, 1994.

[23] J. E. Beck, K. Chang, J. Mostow, and A. Corbett, “Does help help? Introducing the Bayesian Evaluation and Assessment methodology,” in *Intelligent Tutoring Systems*, 2008, pp. 383–394.

[24] Z. A. Pardos and N. T. Heffernan, “Modeling individualization in a bayesian networks implementation of knowledge tracing,” in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 255–266.

[25] R. S. J. d Baker, A. T. Corbett, and V. Aleven, “More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing,” in *Intelligent Tutoring Systems*, 2008, pp. 406–415.

[26] R. S. J. D. Baker, A. B. Goldstein, and N. T. Heffernan, “Detecting learning moment-by-moment,” *Int. J. Artif. Intell. Educ.*, vol. 21, no. 1–2, pp. 5–25, 2011.

[27] X. Z. Clement A. Stone, *Bayesian Analysis of Item Response Theory Models Using SAS*. SAS Institute, 2015.

[28] Y. Gong, J. E. Beck, and N. T. Heffernan, “How to construct more accurate student models: Comparing and optimizing knowledge tracing and performance factor analysis,” in *International Journal of Artificial Intelligence in Education*, 2011, vol. 21, no. 1–2, pp. 27–46.

[29] B. van De Sande, “Properties of the bayesian knowledge tracing model,” *JEDM-Journal Educ. Data Min.*, vol. 5, no. 2, pp. 1–10, 2013.

[30] T. Doleck, R. B. Basnet, E. Poitras, and S. Lajoie, “Towards examining learner behaviors in a medical intelligent tutoring system: A Hidden Markov Model approach,” in *Advance Computing Conference (IACC), 2015 IEEE International*, 2015, pp. 329–332.

[31] F. Dellaert, “The Expectation Maximization Algorithm,” 2002.

[32] M. V Yudelson, K. R. Koedinger, and G. J. Gordon, “Individualized bayesian knowledge tracing models,” in *Artificial intelligence in education*, 2013, pp. 171–180.

[33] W. J. Hawkins, N. T. Heffernan, and R. S. J. D. Baker, “Learning bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8474 LNCS, Springer, Cham, 2014, pp. 150–155.

[34] W. L. Miller, R. S. Baker, and L. M. Rossi, “Unifying Computer-Based Assessment Across Conceptual Instruction, Problem-Solving, and Digital Games,” *Technol. Knowl. Learn.*, vol. 19, no. 1, pp. 165–181, Jul. 2014.

[35] D. Shen, “Some mathematics for HMM,” *Massachusetts Inst. Technol.*, 2008.

[36] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240.

[37] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger, “Performance Factors Analysis--A New Alternative to Knowledge Tracing.,” *Online Submiss.*, 2009.

[38] C. DeMars, *Item Response Theory*. Oxford University Press, USA, 2010.

[39] Y. Gong, J. E. Beck, and N. T. Heffernan, “Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures,” in *Intelligent tutoring systems*, 2010, pp. 35–44.

[40] H.-P. Kriegel, K. M. Borgwardt, P. Kröger, A. Pryakhin, M. Schubert, and A. Zimek, “Future trends

in data mining," *Data Min. Knowl. Discov.*, vol. 15, no. 1, pp. 87–97, 2007.

[41] J. C. Riquelme Santos, R. Ruiz, and K. Gilbert, "Minería de datos: Conceptos y tendencias," *Intel. Artif. Rev. Iberoam. Intel. Artif. 10 (29), 11-18.*, 2006.

[42] Y. T. A. Lara, B. A. O. Zepahua, L. Rodríguez-Mazahua, G. Alor-Hernández, and H. M. Contreras, "Propuesta de arquitectura para un módulo de inteligencia de negocios basado en minería de datos.," *Res. Comput. Sci.*, vol. 128, pp. 47–56, 2016.

[43] S. Aghabozorgi, H. Mahroeian, A. Dutt, T. Y. Wah, and T. Herawan, "An Approachable Analytical Study on Big Educational Data Mining," in *Computational Science and Its Applications--ICCSA 2014*, Springer, 2014, pp. 721–737.

[44] R. B. Sachin and M. S. Vijay, "A survey and future vision of data mining in educational field," in *2012 Second International Conference on Advanced Computing & Communication Technologies*, 2012, pp. 96–100.

[45] A. Peña-Ayala, *Educational data mining: applications and trends*, vol. 524. Springer, 2013.

[46] Á. Jiménez Galindo and H. Álvarez García, "Minería de Datos en la Educación," *2016*, 2016. .

[47] C.-F. Tsai, W.-C. Lin, and S.-W. Ke, "Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies," *J. Syst. Softw.*, vol. 122, pp. 83–92, 2016.

[48] N. Baldoquin Alonso, "Entorno distribuido para el minado de datos en ambientes educacionales masivos.," Universidad de las Ciencias Informática, 2018.

[49] A. A. Vazquez Sánchez, "Proyecto de investigación grupo GITAE," La Habana, 2016.

[50] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[51] S. Jain and S. Jain, "A Survey for Different Classifications of Distributed Systems Scheduling Using Markov Chain Model," 2017.

[52] A. D. Kshemkalyani and M. Singhal, *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2011.

[53] S. Hariri and M. Parashar, *Tools and environments for parallel and distributed computing*, vol. 34. John Wiley & Sons, 2004.

[54] V. K. A. G. Ananth Grama George Karypis, *Introduction to parallel computing*, 2nd ed. Addison Wesley, 2003.

[55] F. Gebali, *Algorithms and parallel computing*, vol. 84. John Wiley & Sons, 2011.

[56] P. E. McKenney, “Is parallel programming hard, and, if so, what can you do about it?(v2017. 01.02 a),” *arXiv Prepr. arXiv1701.00854*, 2017.

[57] N. Pentreath, *Machine learning with spark*. Packt Publishing Ltd, 2015.

[58] The Apache Software Foundation, “Apache Spark,” *Apache Software Foundation*. [Online]. Available: http://spark.apache.org/.

[59] Á. J. Galindo and H. García, “Minería de Datos en la Educación,” *Univ. Carlos III*, pp. 1–8, 2010.

[60] Oracle, “Oracle Java Documentation,” *Oracle*, 2014. [Online]. Available: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/charts.htm. [Accessed: 29-Jun-2019].

[61] S. O. JFreeChart, “JFree Chart,” *2005-2017*, 2017. [Online]. Available: http://www.jfree.org/jfreechart/index.html.

[62] Y. Pérez Peraza, “Big Data y la visualización en el ámbito educativo,” 2017.

[63] A. Sarkar, *Learning Spark SQL*. Birmingham, UK: Packt Publishing, 2017.

[64] M. Yudelson, “Fitting BKT at Scale - Academic page of Michael (Mikhail) Yudelson.” [Online]. Available: https://sites.google.com/site/myudelson/projects/fitbktatscale. [Accessed: 16-Apr-2018].

[65] F. Nielsen, *Introduction to HPC with MPI for Data Science*. Springer, 2016.

[66] J. F. García and M. V Carriegos, “On parallel computation of centrality measures of graphs,” *J. Supercomput.*, pp. 1–19, 2018.

[67] J. E. Freund, I. R. Miller, and R. Johnson, “Probabilidad y estadística para ingenieros,” in *Tomo 2*, La Habana: Editorial Félix Varela, 2006, pp. 307–624.