Sistema Recomendador Basado En Lecciones Aprendidas Para Cursos de Capstone

Ignacio Contreras
icontreras1@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

Danilo Molina
dnmolina@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

Mónica Stambuk
cstambuk@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

ABSTRACT

La red, Internet y las nuevas tecnologías están revolucionando el orden que conocemos. Las personas pueden elegir lo que quieren estudiar, cómo lo van a estudiar y cuándo. La Pontificia Universidad Católica de Chile construyó una plataforma tecnológica que aporta en este contexto. Específicamente se creó una herramienta llamada Zmartboard que ayuda a los estudiantes a gestionar los proyectos de software que ellos realizan en el curso de Capstone. La plataforma cuenta actualmente con un sistema recomendador que permite a los estudiantes acceder a las lecciones aprendidas que otros estudiantes colocaron en la plataforma, pero no cuenta con ningún tipo de métrica sobre la cual poder evaluar su desempeño. Mediante experimentaciones anteriores se ha observado que las recomendaciones que actualmente el sistema entrega no son consistentes con el tópico consultado, es por ello que a partir de estas deficiencias, este estudio buscan tres objetivos centrales; mejorar el indicador de relevancia para las recomendaciones entregadas por el chatbot, implementar algoritmos de deep learning en la recomendación de lecciones aprendidas, y actualizar el preprocesamiento de las lecciones aprendidas frente a tags html y caracteres especiales. Los principales hallazgos obtenidos fueron que el sistema propuesto aumentó la relevancia percibida por los usuarios en un 17%, en comparación con el sistema anterior. La aceptación de los usuarios mejoró también para la transparencia, novedad y serendipity de las recomendaciones. Las lesiones calificadas con 5 estrellas aumentaron su aparición en un 13%, y aquellas con evaluación negativas disminuyeron a un 10% del total. Y 7 de cada 10 recomendaciones entregadas a los usuarios fueron relevantes para su consulta, mejorando la recomendación en un 27%.

KEYWORDS

Zmartboard, Capstone, recomendación basada en contenido, BERT

ACM Reference Format:
Ignacio Contreras, Danilo Molina, and Mónica Stambuk. 2020. Sistema Recomendador Basado En Lecciones Aprendidas Para Cursos de Capstone. In *RecSys '20: Sistemas recomendadores, Diciembre 15, 2020, Santiago, CL.* ACM, New York, NY, USA, 8 pages.

RecSys '20, Diciembre 15, 2020, Santiago, CL

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

1 INTRODUCCIÓN

El actual escenario mundial y local altamente cambiante, ha provocado que la tecnología tome un lugar prioritario en la manera en que las personas llevan a cabo actividades en diferentes ámbitos de su quehacer. El campo de la educación, no está ajena a estos cambios vertiginosos que ha llevado a replantear la manera en que se enseña, adaptándose a las nuevas necesidades de hacer escuela, y sobre todo a la forma en que los estudiantes y docentes se relacionan con las tecnologías.

La red, Internet y las nuevas tecnologías están revolucionando el orden que conocemos. En Internet podemos encontrar diversos recursos docentes. Las personas pueden elegir lo que quieren estudiar, cómo lo van a estudiar y cuándo. El aprendizaje no tiene por qué darse sólo dentro del aula, y esto invita a reflexionar sobre el papel que ejercerán los nuevos docentes. Desarrollar modelos heterogéneos y colaborativos, pareciera ser la clave, ya que promueven pasar de los libros a las "app", y pasar de las aulas a los móviles. Esto hace que se produzcan cambios relevantes en la manera en que los estudiantes se relacionan, pero sobre todo en los recursos que actualmente disponen las instituciones de educación superior (Vázquez, 2015).

Para dar respuesta de alguna forma a estas nuevas necesidades la Pontificia Universidad Católica de Chile construyó una plataforma tecnológica que aporta en este contexto. Específicamente se creó una herramienta llamada Zmartboard que ayuda a los estudiantes a gestionar los proyectos de software que ellos realizan en el contexto del curso de Capstone, que los estudiantes de ingeniería cursan al finalizar sus carreras, y que tiene como objetivo hacer que los estudiantes se enfrenten a situaciones reales de su quehacer como ingenieros, pero sobre todo poner en movimiento habilidades cognitivas, procedimentales, y actitudinales como liderazgo, gestión de proyectos, resolución de problemas, toma de decisiones técnicas, entre otras.

Zmartboard permite que los estudiantes puedan registrar las lecciones aprendidas durante su estancia en el curso, éstas pueden ser tanto técnicas como experiencias que guardan relación con lo actitudinal, generando una base de conocimiento que apoya su futuro quehacer como profesionales. Hasta ahora Zmartboard ha tenido resultados positivos, en vista a experimentaciones previas, sin embargo, hasta ahora no se ha detonado todo el potencial que está a la base de este conocimiento, y por ende no se ha logrado evidenciar el desarrollo personal que alcanzan los estudiantes cada semestre. Por lo que este trabajo permitirá potenciar fuentes de recomendación que apoyen a esta labor de gestión educativa.

Para alcanzar este objetivo se decidió disponibilizar todo este conocimiento mediante el uso de Asistentes Virtuales Inteligentes

(chatbots) en conjunto con Sistemas de Recomendación, esto con el fin de enriquecer toda la experiencia de Zmartboard a los estudiantes nuevos, y así puedan adquirir esas habilidades que están a la base del conocimiento que se espera los estudiantes adquieran durante su estadía en Capstone.

Existen resultados obtenidos por experimentos anteriores sobre el sistema recomendador utilizado por Zmartboard, basado en Latent Semantic Indexing (LSI). Sobre esto se espera mejorar las métricas obtenidas, a partir de la utilización de técnicas de deep learning, para hacer que las recomendaciones sean útiles, y relevantes para los alumnos (Diaz-Mosquera, Sanabria, Neyem, Parra and Navon, 2018).

La plataforma Zmartboard cuenta actualmente con un sistema recomendador que permite a los estudiantes acceder a las lecciones aprendidas que otros estudiantes colocaron en la plataforma, algunos ejemplos de estas lecciones pueden ser el uso de un framework en particular, la instalación de alguna herramienta de desarrollo o experiencias sobre el uso y aplicación de alguna metodología. Como se ve en la Figura 1, la forma de utilizar este sistema es mediante preguntas a través de un chatbot. Además, se puede utilizar un buscador de la misma plataforma, donde los estudiantes reciben la respuesta más acorde al tipo de pregunta realizada.

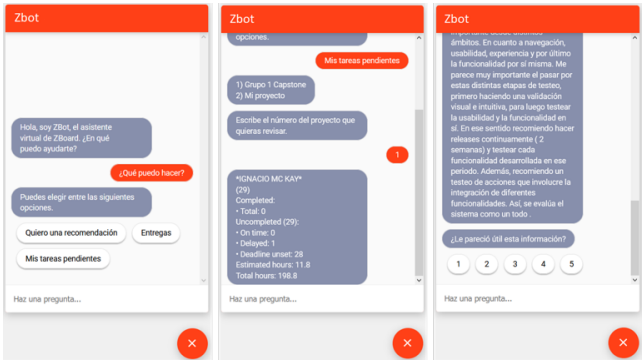

Figura 1: Integración con un asistente virual IA

En este contexto surgen algunos problemas respecto al sistema recomendador, ya que en primer lugar este no cuenta con ningún tipo de métrica sobre el cual poder evaluar su desempeño, por lo que no es posible hacer seguimiento de su desempeño, por otro lado, el sistema recomendador utilizó datos contaminados con tags html, no se utilizó ningún tipo de limpieza de datos ni en el proceso de entrenamiento ni en el proceso de testing. Finalmente, mediante experimentación se ha observado que las recomendaciones que actualmente el sistema entrega no son consistentes con el tópico consultado, por ejemplo, al consultar sobre un framework de desarrollo web llamado React, el sistema recomendador no entrega ninguna recomendación que haga referencia a esta tecnología.

La dinámica del sistema recomendador será similar a la actual, siendo una aplicación REST, que recibe una consulta (cadena de texto) desde la cual se genera una recomendación que tenga sentido con lo que el usuario consulte. Para ello, como primer paso, se definieron métricas que permiten evaluar el sistema recomendador a construir. Una de ellas es el feedback de los usuarios, recomendaciones que el sistema entrega mediante una nota que varía entre 1 y 5, siendo esta última la calificación más alta. De esta manera se puede mejorar su desempeño en base a esa métrica. Por otro lado, se va a reemplazar el actual sistema recomendador por uno mixto basado en contenido (texto) y en recomendación basada en interacciones Item-item. Para la recomendación basada en contenido las características del texto serán obtenidas mediante el uso del algoritmo BERT, que nos proveerá vectores de características de las lecciones aprendidas, sobre las cuales el sistema recomendador podrá recomendar.

2 TRABAJOS RELACIONADOS

Actualmente el avance de las tecnologías emergentes, especialmente las basadas en aprendizaje automático e inteligencia artificial están irrumpiendo en la educación superior, ayudando a docentes, ayudantes, y miembros de las comunidades académicas, a promover el uso de estas tecnologías, como parte del quehacer formativo de sus estudiantes.

Los sistemas recomendadores no son la excepción, han entrado en diferentes ámbitos y áreas del conocimiento, pero específicamente en la educación superior, han permitido resolver algunas problemáticas, pero aún su incorporación ha sido acotada y los trabajos relacionados están en una etapa de diseño, sin embargo, se pueden observar algunos trabajos interesantes, como por ejemplo el uso de sistemas recomendadores para disminuir la deserción de los estudiantes a la universidad (Karypis, 2017). También se ha evidenciado algunos avances en la utilización de sistemas recomendadores para la selección de cursos universitarios, en base a objetivos o preferencias individuales de los estudiantes (Jiang, 2019).

Respecto al uso de sistemas recomendadores en la Pontificia Universidad Católica de Chile, actualmente, existe un trabajo realizado en el curso de Capstone del Departamento de Computación, el cual se enfoca en ayudar a mejorar y resolver una serie de problemáticas propias del curso, donde el objetivo central se encuentra en recomendar lecciones aprendidas de desarrollo software, desde estudiantes veteranos hacia estudiantes primerizos, como un medio de apoyo colaborativo entre estudiantes (Neyem, 2017).

3 ANÁLISIS EXPLORATORIO DE DATOS

En esta ocasión se utilizó un dataset que contiene 1843 lecciones obtenidas desde mayo del 2019 a noviembre de 2020. Las lecciones se generaron a partir de la realización del Proyecto de Especialidad de Computación, de esta manera, al finalizar el curso, los alumnos debieron registrar sus lecciones aprendidas durante el transcurso del semestre. Estos registros involucran diversos tópicos, desde administración de proyectos de software hasta soluciones de bugs. De este modo las lecciones comprenden nombre, título, problema, solución y tags.

Es relevante destacar que tanto el problema como la solución pueden contener elementos html, y que una representación de las lecciones se obtiene a partir de la contabilización de palabras dentro de cada atributo. Esta información se aprecia en la Tabla 1. Además, se pueden retirar las stopwords en español de las lecciones para permitir un entrenamiento enfocado en conceptos menos utilizados. De esta manera, las recomendaciones para el query "docker" resultan valiosas frente a queries como "en".

Tabla 1: Cantidad de palabras por columna en lecciones aprendidas.

Concepto	Nombre	Problema	Solución	Tags
Mean	35.45	273.85	731.36	3.06
SD	18.73	215.64	1237	1.84
Min	2	5	0	0
Max	164	1528	21597	16

Eliminando los stopwords de los atributos de las lecciones aprendidas, se obtiene lo descrito en la Tabla 2, utilizando los datos procesados para el entrenamiento.

Tabla 2: Cantidad de palabras sin stopwords por columna en lecciones aprendidas.

Concepto	Nombre	Problema	Solución	Tags
Mean	26.06	198.35	551.98	3.06
SD	13.79	157.91	995.10	1.84
Min	2	3	0	0
Max	121	1179	17556	16

Combinando los atributos de la lección dentro de uno solo y contabilizando sus palabras luego de extraer las stopwords, se observa la distribución indicada en la Tabla 3.

Tabla 3: Cantidad de palabras sin stopwords concatenadas en lecciones aprendidas.

Cantidad de palabras	Combinación sin *stopwords*
Mean	779.45
SD	1000.77
Min	22
25%	406
50%	686
75%	836
Max	17656

Por otro lado, se extenderá el entrenamiento del modelo utilizando ratings o interacciones entregados por los alumnos. Esto se debe a que, durante el mes de Octubre, se agregaron interacciones entre los alumnos y las lecciones. Cabe señalar que actualmente el curso de Proyecto de Especialidad tiene 93 estudiantes que utilizan la aplicación Zmartboard semanalmente.

Las interacciones son votos que entregan información respecto al rating otorgado a la lección recomendada dependiendo de su utilidad para el alumno. Éstas van de 1 a 5, donde 1 se refiere a una baja valoración y 5 a una alta valoración, por parte del usuario, información que respalda la posición de la recomendación entregada y el query asociado a la recomendación. De esta manera, si un usuario consulta por "react" y le entrega 3 puntos al primer resultado de la recomendación, se guardará la interacción (userId, attempt, rating, query).

En la Tabla 4 y Figura 2 se representan los usuarios con mayor cantidad de interacciones, número de usuarios por cantidad de interacciones realizadas, lecciones aprendidas con mayor cantidad de interacciones, y distribución de interacciones por usuario.

Tabla 4: Cantidad de palabras sin stopwords concatenadas en lecciones aprendidas.

Id de usuario	Cantidad de interacciones
836e51aa-9ccc-4105	26
a2ec5814-23bb-4571	18
429127d8-770e-412b	16
604ffbd8-f3b4-41b5	15
8808bc39-8dd4-4190	14

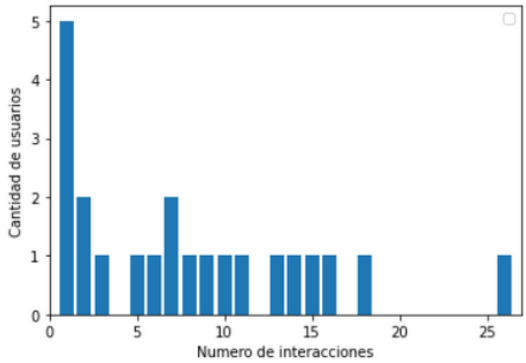

Figura 2: Cantidad de usuarios versus cantidad de interacciones realizadas

4 SISTEMA PROPUESTO

4.1 Procesamiento de datos

El proceso para obtener los vectores de palabras para cada lección aprendida se centró en una arquitectura secuencial. Esta serie de pasos consistieron en: (1) convertir a lowercase, (2) remover tags html de cada atributo de la lección, (3) eliminar tildes y caracteres no ASCII, (4) tokenizar el documento y finalmente (5) remover stopwords en español.

A partir del anterior proceso se obtuvo el input procesado para entrenar el modelo. Es relevante destacar que es importante realizar este mismo proceso sobre las queries al momento de recomendar para evitar discordancia entre caracteres y palabras del modelo. Así, al realizar la consulta "Quiero aprender un tutorial sobre react" se traduce al vector de palabras ["quiero", "aprender", "tutorial", "react"].

4.2 Método de entrenamiento

Se utilizó recomendación basada en contenido (texto), en cuanto al procesamiento del lenguaje natural se utilizó el algoritmo de BERT para la extracción de vectores de características, debido a que el dataset está en español e incorpora palabras en inglés, se

Tabla 5: Aspectos a evaluar por cada lección recomendada.

Aspectos a evaluar	Descripción
Relevancia	Las recomendaciones fueron relevantes.
Transparencia	El origen de las recomendaciones fue claro.
Novedad	Las recomendaciones eran novedosas.
Serendipity	Las recomendaciones llamaron mi atención aunque no estén relacionadas con la consulta.

utilizó el modelo pre-entrenado BERT-Base, Multilingual Cased, que incorpora una serie de idiomas por defecto, entre ellos español y inglés (Wu Dredze, 2019).

BERT (Bidirectional Encoder Representations from Transformers) es una técnica de NLP (Natural Language Processing) desarrollada por Google y publicada en 2018 por Jacob Devlin. Donde Encoder Representations es el sistema de modelado de lenguaje, pre entrenado con datos sin etiquetar y luego se ajustan sus parámetros, Transformer quien define la arquitectura de BERT y Bidirectional se refiere a que utiliza el contexto de en ambos sentidos del texto (izquierdas y derechas) al trabajar con palabras, además es quien define el entrenamiento.

Es importante señalar que, para el entrenamiento del sistema recomendador detallado a continuación, se utilizaron todo el dataset de lecciones aprendidas. Esto debido a que el entrenamiento se realizó por sobre todos los elementos. No es necesario un dataset para testear ya que se realizó evaluación online.

Para entrenar el modelo, se obtienen los embeddings de los los títulos de las lecciones, éstas se almacenan como un arreglo de oraciones, luego de ese paso, se calculan los embeddings de los tags de las lecciones, los cuales al igual que los títulos, se guardan como un arreglo, solo que en este caso, son palabras y no oraciones. Debido a que durante el entrenamiento se obtienen embeddings tanto de palabras como de oraciones, resulta complejo realizar el cálculo de distancia entre embeddings debido a problemas de dimensionalidad. Por esto, el análisis de palabras por separado resulta más complejo ya que por un lado estas oraciones tienen en general un contexto, y por otro, puede existir una diferencia de tamaños entre los embeddings. Por ello se optó por realizar el entrenamiento utilizando la librería Sentence Transformers que utiliza BERT (en su versión multilenguaje) y simple transformer para calcular los embeddings de distinto tamaño entre palabras y oraciones.

Tanto BERT como Simple Sentences utilizan como parte central de su algoritmo la arquitectura Transformer, un modelo que tiene como principal innovación la sustitución de las capas recurrentes, como las LSTMs que se utilizan en problemas comunes de PLN, por las denominadas capas de atención (Vaswani, 2017). En el caso de BERT y de las arquitecturas basadas en Transformer, se utilizan redes neuronales de tipo transformer, que son una clase reciente de redes neuronales secuenciales (Vaswani, 2017), de manera que se utiliza Deep Learning para el apartado de procesamiento de lenguaje natural. No se utiliza LSTM directamente en el modelo actual debido a que el modelo antiguo ya lo contiene.

Estas capas de atención codifican cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto, motivo por el cual a los modelos basados en Transformer se les denomina también Embeddings Contextuales. La arquitectura de Transformer incluye otras innovaciones, como los embeddings posicionales, que permiten al algoritmo conocer la posición relativa de cada palabra del texto (Rae, 2019), (Kitaev, 2020).

4.3 Interacciones

Las interacciones fueron añadidas al modelo luego de obtener los embeddings del corpus. De esta manera, para cada interacción se calculó el score de similitud entre la query y su lección asociada. Luego, este score se ponderó con un valor asociado al rating de esa interacción. Un puntaje menor a 3 se consideró como negativo, y superior o igual a 3 como positivo. Para valores más alejados de 3 se consideraron ponderaciones mayores. De esta manera el score fue beneficiado cuando las recomendaciones fueron relevantes para el usuario.

Una vez obtenido el score ponderado (sp) para cada interacción se calculó el promedio por cada lección. Así, al momento de entregar los Top N lecciones aprendidas, en base a la suma de los scores de los embeddings se pudo agregar un tercer parámetro (sp) con el fin de aumentar la prioridad de la lección cuando se tienen buenas interacciones y baje su relevancia cuando tienen malas recomendaciones.

4.4 Método de recomendación

Una vez realizado el entrenamiento del modelo basado en contenido se procedió a realizar las recomendaciones. De esta manera, dada una consulta en el sistema, ésta se procesa para obtener el vector de palabras y luego se concatenan para realizar el encoding con los embeddings del modelo. Así, se realiza una operación de similitud por coseno del embedding resultante. Este proceso de comparación se hace frente al corpus de las lecciones sin los tags, y también se obtiene un embedding frente al corpus de los tags.

Una vez obtenidos los scores de comparación de ambos corpus se procede a iterar por sobre todas las lecciones y realizar la suma de ambos scores. De esta manera, podemos ordenar utilizando el aporte de cada embedding. Si no se realiza esta suma entonces lecciones que contienen un nombre poco descriptivo pero útiles aparecerán en posiciones menores del ranking de recomendaciones. Por ende, para evitar este problema se identificó que los tags señalaban los conceptos principales de las lecciones, como es su objetivo. Así, aquellas lecciones poco descriptivas pero con tags representativos y acordes a la query pueden aparecer en posiciones útiles del ranking.

Finalmente, cuando ya se procesaron los scores y se procedió a la suma de ellos, se ordena en orden decreciente el total calculado para cada lección aprendida y se entregan al usuario las N (para este caso 10) lecciones con mayor score.

Tabla 6: Calificaciones de los aspectos: media y desviación estándar.

Modelo	Relevancia	Transparencia	Novedad	Serendipity
LSI (Antiguo)	2.81 ± 1.40	2.49 ± 1.25	2.53 ± 1.23	2.92 ± 1.25
Transformer (Nuevo)	3.28 ± 1.35	3.77 ± 1.32	3.34 ± 1.25	3.12 ± 1.28

Algunos ejemplos para entender su funcionamiento, tanto como consulta, recomendaciones y calificación, se pueden apreciar a continucación:

- Quiero aprender sobre react native
  - "Uso de native base y react-native-community en aplicación móvil con react native" (4)
  - "Tutorial React Native + Redux" (5)
  - "Funcionalidades extras en Gifted Chat: paquete de Chat UI" (1)
- Cómo usar git
  - "Diferencia entre rebase y merge (Git)" (5)
  - "Regresa tu rama de git a la versión remota" (5)
  - "URL de la API en front-web" (2)

4.5 Arquitectura del sistema

Para implementar el sistema recomendador y realizar los experimentos se realizó el deployment sobre un servidor ElasticBeanstalk existente y que se utiliza para administrar el modelo antiguo. De esta manera, se aplicó un patrón Strategy sobre la función de entrenamiento y recomendación, como también nuevos endpoints del servidor para poder acceder desde el exterior a la activación del entrenamiento y a la consulta de recomendaciones basadas en una query. Una vez realizado el deploy y entrenamiento se comenzó con la experimentación.

El sistema recomendador fue utilizado a través de la plataforma Zmartboard. Ésta tiene el objetivo de cubrir todas las necesidades básicas de los proyectos ágiles de software. Sus principales funcionalidades están centradas en la "representación de tareas" basadas en la filosofía del tablero Kanban. Además, Zmartboard posee un asistente virtual capaz de responder preguntas de los alumnos, con relación a lecciones aprendidas y al curso.

En particular, el asistente virtual (chatbot) permite hacer de interfaz con el sistema recomendador y cuando un estudiante consulta por un concepto o frase, éste entrega las lecciones recomendadas, la opción de calificarlas con un número de 1 a 5, y la posibilidad de mostrar más resultados. Esta funcionalidad fue utilizada en la experimentación.

4.6 Evaluación de la recomendación

Para poder evaluar la efectividad del modelo entrenado y obtener métricas relevantes frente al modelo existente, en primer lugar, se definieron como métricas Mean Average Precision (MAP) y Normalized Discounted Cumulative Gain (nDCG). Estas métricas permiten comparar el modelo en producción con el entrenado anteriormente. Esto dado que uno de los objetivos centrales de este estudio fue mejorar las recomendaciones que se realizan actualmente, sin crear nuevas.

En relación a las métricas, MAP y nDCG se basan en encontrar la posición de los elementos relevantes de la recomendación. MAP, revisa por todos los usuarios la cantidad de elementos relevantes dentro de sus recomendaciones. Luego, se calcula un promedio por sobre los usuarios. Y nDCG se basa en obtener la posición de los elementos relevantes seleccionados por el usuario para dar cuenta de la efectividad de la recomendación. Por lo que, la precisión se basa en obtener, sobre el total de recomendaciones, el porcentaje de lecciones relevantes seleccionadas por los usuarios.

Además de las métricas relacionadas a la relevancia de las recomendaciones, se buscó obtener información sobre novedad, serendipity y transparencia de las recomendaciones entregadas, aportando a la explicabilidad y satisfacción del usuario frente al sistema recomendador (Kaminskas, 2016).

5 EXPERIMENTO Y ANÁLISIS

5.1 Experimentación

Se realizó un experimento con los alumnos del curso de Proyecto de Especialidad dictado por el Departamento de Ciencias de la Computación de la Pontificia Universidad Católica de Chile. Este curso tiene capacidad para 93 alumnos los que son distribuidos en 9 grupos asignados a proyectos de software de la vida real. Para la realización del experimento, la participación fue voluntaria. El experimento que se llevó a cabo fue una réplica del realizado en el mes de Septiembre con el modelo antiguo.

El experimento tuvo dos objetivos principales. En primer lugar, evaluar el sistema recomendador basado en 1843 lecciones aprendidas que fueron procesadas por el sistema. Estas lecciones fueron provistas por 196 alumnos del curso desde el año 2019. Y por otro lado, en una segunda parte de la experimentación, se desarrolló una prueba de usabilidad sobre las funcionalidades del asistente virtual Dialogflow (chatbot).

Durante el experimento, cada participante debió utilizar el asistente virtual para realizar 4 consultas sobre temas relacionados a los contenidos del curso y/o temáticas de software. Estas consultas debían ser palabras o frases. Para cada consulta, el sistema recomendador entregó 3 lecciones. Luego, debieron realizar 3 consultas más en el buscador de Zmartboard, donde se recomendaron 10 lecciones aprendidas por cada búsqueda. En total cada participante debió calificar 21 lecciones aprendidas con un rating de 1 a 5. Números de rating mayores o iguales a 3 representan relevancia para el usuario y fueron utilizados para medir la precisión de las recomendaciones. Además, luego de calificar las 5 lecciones recomendadas de una consulta, el participante completó una encuesta en la que indicó su percepción sobre la Relevancia, Transparencia, Novedad y *Serendipity*. Cada uno de estos aspectos fueron clasificados con una escala de Likert, lo cual se muestra en la Tabla 5.

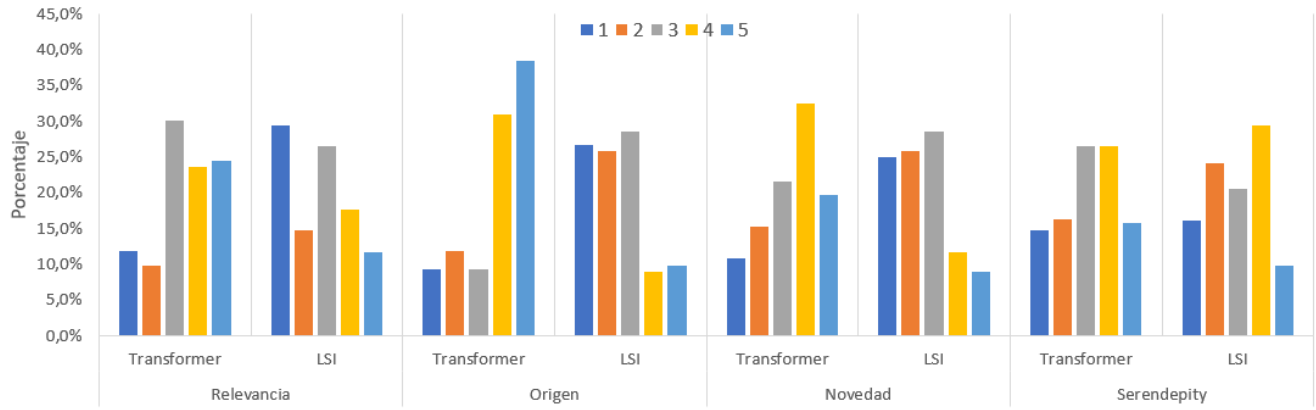

Figura 3: Evaluación de aspectos: Comparación entre modelos Transfomer y LSI

5.2 Resultados

A partir del experimento desarrollado 39 participantes calificaron un total de 216 lecciones aprendidas a través de 704 interacciones.

De esta manera, se calcularon las métricas para el modelo original y el modelo entrenado. En relación a esto, Diaz-Mosquera, Sanbria, Neyem, Parra y Navo (2018), previamente realizaron la experimentación obteniendo resultados para el modelo original.

La Tabla 6 muestra el promedio y desviación estándar para cada modelo, el modelo antiguo basado en LSI y el modelo actual basado en Transformer, con respecto a cada uno de los cuatro aspectos a ser evaluados. El nuevo sistema es capaz de recomendar lecciones aprendidas basadas en un input. La fila con mayor media en cada aspecto resulta ser la fila del modelo propuesto. Cada uno de estos aspectos se evaluó con un número entre 1 y 5, con números mayores más relevantes para el usuario.

Además, en la Figura 3, se presentan los porcentajes de calificación, para cada uno de los cuatro aspectos, en el modelo antiguo LSI y el modelo propuesto Transformer. Como se observa, el modelo propuesto presenta una precisión basada en su relevancia de 78%, equivalente a que 8 de cada 10 recomendaciones sean relevantes para el usuario. Este valor es mayor al del modelo LSI ya que este presenta un 56% de relevancia en sus lecciones aprendidas recomendadas. En adición a lo anterior, con el nuevo modelo, lecciones calificadas con 5 estrellas (muy relevantes para el usuario), aumentaron su aparición un 13%, y aquellas con una evaluación negativa (1), disminuyeron a un 10%.

Al considerar la transparencia de las lecciones recomendadas, como su origen, percibida por los participantes durante el experimento, se observa que se entregaron recomendaciones relacionadas con las consultas y se les agregó el creador de la lección para conocer su autoría. En comparación con LSI, la transparencia mejoró su calificación de un 47% a un 79% de aceptación. Esta misma tendencia se aprecia en la novedad de las recomendaciones y del hallazgo afortunado de ellas (*serendipity*), las cuales mejoraron su aceptación de un 49% a un 73% y de un 59% a un 69%, respectivamente.

En relación a las métricas directamente relacionadas con la relevancia de las recomendaciones se tienen MAP y nDCG. En la Tabla 7, se observa que los valores de las métricas basadas en el modelo propuesto aumentaron con relación al modelo basado en LSI. Se consideró (n=5) debido a la cantidad de calificaciones por consulta. El modelo anterior posee un MAP@5 de 0.56, mientras que el modelo Transformer entregó un MAP@5 de 0.71. Esto indica que 7 de cada 10 recomendaciones entregadas a los usuarios son relevantes para su consulta. Ahora, considerando nDCG@5, se observa que el modelo propuesto permitió entregar las recomendaciones más relevantes para los usuarios en mejores posiciones que el modelo LSI. Esto se basa en la capacidad del modelo de utilizar tanto el contenido, los tags y las interacciones de los usuarios con la lección aprendida.

Tabla 7: Comparación de métricas: MAP@5 y nDCG@5.

Modelo	MAP@5	nDCG@5
LSI (Antiguo)	0.561	0.816
Transformer (Nuevo)	0.714	0.947

A partir de los comentarios de los participantes y considerando el preprocesamiento de las lecciones tanto para el modelo como para la respuesta entregada, se señala que hubo una notoria limpieza debido a que antes, consultas como "tutorial html y de css" entregaban lecciones al azar debido a que éstas están generadas en base a código HTML, mientras que el nuevo modelo retira los tags relacionados a este código, para entregar recomendaciones basadas en su contenido textual. Además, al realizarse un entrenamiento continuo se observó que lecciones con baja calificación eran desplazadas o incluso retiradas de las recomendaciones, lo que favoreció la calificación de otras lecciones. Sin embargo, una de las principales críticas al sistema realizado fue el método de calificación basado en una escala de 1 a 5 junto con una opción de like/dislike. Esto se realizó tanto para aportar al modelo como para visualizar votos sobre una lección. Este hecho perjudica la claridad y transparencia con la que ocurren las recomendaciones debido a que como se señaló, separan la utilidad y la calidad de una lección aprendida.

Se debe relevar que, dada la naturaleza del problema, se determinó que no existe un perfil de usuario para poder utilizar Collaborative Filtering o para tener métricas offline. De esta manera, las evaluaciones que se realizaron se basaron en la experimentación

y en los ratings que entregaron los estudiantes del curso. Además, dadas las características del problema a resolver, no fue posible obtener información detallada del usuario que hace la consulta y tampoco se logró tener suficientes interacciones de los usuarios, como para extraer factores latentes. De esta manera, no existe personalización en los resultados del sistema recomendador.

5.3 Limitaciones

Una de las limitaciones evidenciadas al momento de desarrollar este estudio fue que el algoritmo BERT para el procesamiento de lenguaje natural no soporta en principio el lenguaje español, esto llevó a probar una opción desarrollada por el Departamento de Ciencias de la Computación de la Universidad de Chile BETO (Cañete, Chaperon, Fuentes, Pérez, 2020), modelo de BERT entrenado en español, sin embargo, surgió una segunda dificultad con la implementación de este modelo, esta fue que debido al contexto en que se desarrolla el proyecto, gran parte del vocabulario existente tiene palabras en inglés, como por ejemplo "Testing", "REACT", "FLOW", etc. Dado este escenario se optó por utilizar un modelo pre entrenado llamado "BERT-Base, Multilingual Cased", el cual incorpora diversos idiomas, entre ellos inglés y español, lo cual soluciono en gran medida la problemática inicial de utilizar palabras en dos idiomas.

Una limitación a la implementación fue el tamaño o peso que el algoritmo le da a los conectores y proposiciones para el desarrollo del sistema recomendador, ya que esto influyó en el resultado esperado de la recomendación. Pero esta fue solucionada adoptando dos estrategias al momento de realizar la implementación, estas fueron: se realizó una lematización de las oraciones, además se aplicaron filtros con "Stop Words" durante la fase de limpieza del texto. Y se realizó una ponderación entre el análisis textual de los tags y el de los títulos de las lecciones, tal y como se explicó en el modelo de recomendación.

Finalmente, un error que surgió debido a la utilización de embeddings de palabras, fue que los títulos de las lecciones son oraciones, esto es un problema a la hora de calcular la distancia entre los embeddings, ya que la distancias típicas utilizadas para estos fines no funcionan si existen diferencias de dimensionalidad, una opción para solucionar este problema fue utilizar ceros o unos, para rellenar la dimensión faltante, la otra estrategia fue quitar información a la de mayor funcionalidad para equiparar con la menor, sin embargo ambas estrategias resultaron en un desempeño notablemente bajo. Debido a esta situación se optó por utilizar sentence transformers, que permitió obtener embeddings de oraciones completas y calcular su diferencia con embeddings de distintas magnitudes.

6 CONCLUSIONES

En cuanto a la primera conclusión obtenida al realizar la experimentación, observamos que la utilización de un modelo basado en deep learning como BERT permitió abstraer las relaciones entre las palabras y mejorar de las recomendaciones de lecciones aprendidas en el contexto de un curso de capstone.

Por otro lado, la utilización de interacciones para entrenar el sistema recomendador permitió mejorar las recomendaciones que, aunque estaban relacionadas con el tópico consultado, podrían no ser útiles o ser incompletas. Esto favoreció la aparición de lecciones aprendidas novedosas y relevantes para los estudiantes. También, este feedback continuo permitió que las nuevas lecciones fueran calificadas, evitando que solo se realizarán recomendaciones de las lecciones más populares, lo que mejoró, en gran medida la serendipity de las recomendaciones.

Finalmente, se evidenció que, no solo los repositorios de conocimiento permitieron entregar experiencias significativas a los estudiantes que permitieron fortalecer su desarrollo profesional, y académico bajo el entorno del cursos de capstone. Sino también, dado el contexto de pandemia que se encuentra actualmente en el país, donde el acceso a ayudantes y profesores se vio restringido, tener un apoyo basado en este conocimiento compartido, permite resolver dudas teóricas, y prácticas cuando el estudiante lo necesitaban, promoviendo con ello la resolución de problemas que normalmente se presentan en clases presenciales.

En un futuro y gracias a la generación de interacciones, se podría armar un perfil del usuario que aporte a la recomendación de lecciones, y se podría considerar la utilización de repositorios de conocimientos externos, para apoyar de manera significativa en el desarrollo profesional, y académico de los estudiantes.

7 REFERENCIAS

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. (2017). Attention Is All You Need.

[2] Cañete, J., Chaperon, G., Fuentes, R., Pérez, J. (2020). Spanish Pre-Trained BERT Model and Evaluation Data. PML4DC at ICLR.

[3] Engine. IEEE/ACM 4th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE), pp. 25-29.

[4] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Timothy P. Lillicrap. (2019). Compressive Transformers for Long-Range Sequence Modelling.

[5] Jiang, W., Pardos, Z. (2019). Time Slice Imputation for Personalized Goal-Based Recommendation in Higher Education. In Proceedings of the 13th ACM Conference on Recommender Systems (pp. 506–510). Association for Computing Machinery.

[6] J. D. Diaz-Mosquera, P. Sanabria, A. Neyem, D. Parra and J. Navon. (2018) Enriching Capstone Project-Based Learning Experiences Using a Crowdsourcing Recommender.

[7] Kaminskas, M., Bridge, D. (2016). Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems.

[8] Karypis, G. (2017). Improving Higher Education: Learning Analytics Recommender Systems Research. In Proceedings of the Eleventh ACM Conference on Recommender Systems (pp. 2). Association for Computing Machinery.

[9] Kitaev, N., Kaiser, Ł., Levskaya, A. (2020). Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451.

[10] Neyem, A., Diaz-Mosquera, J., Munoz-Gama, J., Navon, J. (2017). Understanding Student Interactions in Capstone Courses to Improve Learning Experiences. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 423–428). Association for Computing Machinery.

[11] Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya. (2020). Reformer: The Efficient Transformer.

[12] Vázquez, J. A. (2015). Nuevos escenarios y tendencias universitarias. Revista de Investigación Educativa, 33 (1), 13-26. DOI: http://dx.doi.org/10.6018/rie.33.1.211501.

[13] Wu, S., Dredze, M. (2019). Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT. arXiv:1706.03762.