UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CATEGORIZADOR Y RECOMENDADOR DE ASIGNATURAS DE UN PLAN DE ESTUDIO ACADÉMICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

DIEGO ANDRÉS SANDOVAL LEIVA

PROFESORES GUÍA:
Willy Maikowski Correa
Andrés Abeliuk Kimelman

MIEMBROS DE LA COMISIÓN:
Juan Alvarez Rubio
Eric Tanter

SANTIAGO DE CHILE
2022

CATEGORIZADOR Y RECOMENDADOR DE ASIGNATURAS DE UN PLAN DE ESTUDIO ACADÉMICO

Durante la formación universitaria y ante los procesos de especialización, el estudiante debe tomar distintas decisiones relevantes como, por ejemplo, qué ramos inscribir. La interrogante es cómo completar su plan de estudio y, a la vez, que los cursos pertenecientes a dicho plan estén relacionados con sus intereses de especialización.

Responder lo anterior permitirá al estudiante tener claridad para escoger qué ramos cursar en los próximos semestres académicos. Ante esta problemática, nace la necesidad de guiar al estudiante en la toma de decisiones para que pueda realizar los procesos de su formación de manera informada y con el firme propósito de su proyección ante una formación de un profesional especializado.

En este contexto, en la presente memoria se realiza un trabajo en colaboración del Centro Tecnológico Ucampus de la Universidad de Chile para construir herramientas que ayuden al estudiante en la toma de decisiones, en particular para la inscripción académica. El objetivo del trabajo realizado consistió en crear un sistema que permite al estudiante categorizar todos los ramos pertenecientes a un tópico o área de estudio de su interés, y un sistema de recomendación a partir de un filtro colaborativo basado en usuarios. El propósito es entregar dos herramientas para poder clarificar la toma de decisiones que permita formar profesionales íntegros que le atribuyan un mayor valor a su carrera.

Los sistemas han sido construidos tomando en consideración la información y recursos suministrados por Ucampus. De esta manera se abordó el problema de categorización a partir de una metodología no supervisada, comparando cada una de las opciones existentes para resolver el problema, optando por el algoritmo Top2Vec. Con respecto al sistema de recomendación se ha confirmado el uso del filtro colaborativo y, adicionalmente, se ha realizado un estudio completo de su funcionamiento. Posteriormente, se ha construido un servicio API Rest que pone a disposición ambos sistemas para ser consumidos por una plataforma Web.

Con el propósito de validar dichos sistemas, se ha creado una interfaz que permite a un potencial usuario interactuar con ellos. Tras construir el sitio Web y hacerlo disponible de manera pública, se ha hecho la validación con estudiantes de distintos departamentos de la Facultad de Ciencias Físicas y Matemáticas, guiados mediante un formulario en el cual han registrado sus apreciaciones.

Finalmente, tras realizar las validaciones, se han constatado los resultados obteniendo en su mayoría excelentes apreciaciones, logrando de esta manera cumplir con los objetivos propuestos. Sin embargo, ha quedado como trabajo futuro extender los sistemas construidos en la plataforma de Ucampus y, con tal de responder a aquellos usuarios encuestados que no han quedado satisfechos con los resultados, refinar los pipelines diseñados.

Hay una fuerza motriz más poderosa que el vapor,
la electricidad y la energía atómica: la voluntad.

Albert Einstein

Agradecimientos

A mi Madre, mi Padre, mi Hermana y el resto de mi familia, por apoyarme, soportarme y cuidarme desde el día 0, y que pese a tomar muchas decisiones impulsivas, siempre estuvieron allí para respaldarme aunque no siempre estuvieran de acuerdo.

A mis mascotas que siempre han sido parte de mi familia, mis gatas; Luna por haberme escogido como su humano favorito, a Mili por hacerme reír con sus payasadas, a Pabli que pese a ser gritona y a veces pesada, tiene un lado tierno que adoro, y en especial a mi Sofi, quien me acompaña desde algún otro lugar.

A mis amigos de la U y de vida, Ricardo, Pablo, Polo, Bryan, Joaco y Neira que hicieron que mis días en la Universidad, hayan sido más amenos y disfrútables.

A mi grupo de amigos "Passione" por sacarme de la rutina en cada junta, asado o lo que se organizara para compartir, en especial a Panchito que se hizo uno de mis mejores amigos.

Al Departamento de Computación y a cada uno de sus docentes que me formaron y guiaron durante mi carrera, en especial a Willy Maikowski y Andrés Abeliuk, mis profesores guías por su disposición, sus consejos, su flexibilidad y todo su apoyo.

Finalmente, quisiera agradecer a Ucampus por haberme brindado esta oportunidad y creer en mis capacidades.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto y oportunidad a abordar

A lo largo de la historia de la ciencia, han surgido distintas especializaciones y áreas de estudio, esto es debido al progreso de la humanidad, al nacimiento de nuevas tecnologías y de recursos propios de la evolución del ser humano. En las Ciencias de la Computación, por ejemplo, hoy en día podemos encontrar diferentes campos de especialización como el Desarrollo Web, la Inteligencia Artificial, la Ciencia de Datos, la Computación Gráfica, etc. De igual manera, este patrón se repite en otras ciencias y carreras, como las humanidades, la salud, las artes, el deporte, etc.

Debido a la diversidad de la ciencia, la elección de una especialidad o una rama de estudio dentro de una carrera es una problemática frecuente que suele generar muchas dudas en los estudiantes a la hora de tomar decisiones en la elección de sus ramos o áreas de investigación en las que se dedicarán durante su carrera universitaria y posteriormente en sus vidas como profesionales. Un estudio hecho por Michael Borchert en el 2002 [1] enfatiza la importancia de contar con una planificación profesional eficaz. Además, en este se descubrió que los estudiantes que reciben ayuda para explorar carreras y planificar programas de estudio relacionados con sus intereses profesionales tienen más probabilidades de ver la universidad como algo significativo, lo que permite al estudiante valorizar su formación y usarla como base para desarrollarse como profesional.

En el caso de la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile, los estudiantes suelen tomar los ramos que les corresponden a sus mallas de carrera. Sin embargo, a la hora de escoger una especialización, estos suelen solicitar información a sus pares o indagar por su propia cuenta, y al lograr obtener una respuesta no siempre demuestra el verdadero interés de la persona. Esta situación es recurrente y hace que muchas personas terminen estudiando algo en lo que no tienen afinidad. Para constatar la situación de los estudiantes con respecto a su formación e inscripción de ramos se ha realizado una encuesta a 18 estudiantes pertenecientes a distintos departamentos de la facultad cuyo propósito es dimensionar cuantitativamente la problemática descrita.

Entre las preguntas a destacar en la encuesta se encuentran: “¿Te ha sido fácil hallar aquellos ramos que pertenecen a un mismo área de estudio?”. Esta pregunta ha sido planteada en el contexto descrito previamente, asociado a las diversas áreas pertenecientes a una ingeniería

o ciencia. Un 50,5 % de los encuestados responde con un nivel de acuerdo a esta pregunta. Posteriormente, en linea con la pregunta anterior se preguntó si "Durante tu carrera universitaria, ¿te ha sido fácil escoger los cursos electivos en la inscripción académica?", obteniendo un 50 % de los encuestados con un nivel de desacuerdo. Dado el bajo porcentaje en ambas preguntas, la problemática no podría resultar significativa. Sin embargo, esto tiene una razón, y es debido a que existe una herramienta que permite al usuario tener ayuda al momento de tomar los ramos, estas son las referencias o recomendaciones de otros estudiantes, y ha sido constatado con la pregunta, "¿Has solicitado referencias de otros estudiantes al momento de inscribir un curso?", un 94,1 % de los encuestados afirma haber solicitado una referencia de uno o más cursos más de una vez.

Posteriormente, se ha indagado sobre el futuro y la proyección del estudiante mediante preguntas como: "¿Crees tener claro a qué especialidad o rubro dentro de tu carrera quisieras dedicarte una vez egresado?". Un 47,1 % de los estudiantes encuestados afirma no estar del todo claro. Por otro lado, se ha preguntado: "¿Crees que herramientas que te ayuden al momento de inscribir ramos te permitirían a proyectarte mejor como estudiante y/o un profesional?", con el fin de ratificar la opinión de los estudiantes con la observación obtenida en el estudio de Borchert [1] con respecto a la formación de los estudiantes que reciben ayuda, obteniéndose un 100 % de respuesta con un grado positivo de acuerdo.

Finalmente, tras constatar con el grupo de estudiantes la problemática e identificar los recursos utilizados en su formación universitaria, nace la oportunidad de abordarla mediante la creación de herramientas basadas en la recomendación y categorización de ramos. Ésta última con el fin de reducir la incertidumbre del estudiante para identificar la especialidad de su carrera, y la primera para fomentar el recurso ya existente potenciado a través de una herramienta o software que permita automatizar o facilitar la solicitud de referencias de cursos a otros estudiantes.

1.1.1. Centro Tecnológico Ucampus

Ucampus es un Centro Tecnológico que desarrolla plataformas de apoyo a la gestión, automatizando los procesos de Instituciones de Educación Superior, que trabaja con diversas universidades, entre ellas la Universidad de Chile. Junto con suministrar la plataforma virtual U-Cursos para la gestión de ramos, horarios, tareas, asistencia, etc., también suministra la plataforma que lleva por nombre Ucampus, cuyas principales prestaciones son la inscripción académica, los boletines, el catálogo de cursos por semestre, el historial de notas y ramos cursados, certificado del estudiante, entre otras funcionalidades. Ucampus presenta herramientas para inscribir y conocer los cursos, por lo que las respuestas reflejan la opinión de los estudiantes con respecto a el uso de la plataforma. Por lo demás, Ucampus, no presenta ninguna opción de recomendación y categorización de cursos.

1.1.2. Trabajo a realizar

De acuerdo a lo anterior, el trabajo realizado por el estudiante en la presente memoria, consistió en desarrollar una prueba de concepto, creando dos nuevas herramientas que permiten al estudiante presentarle un perfil de recomendaciones a partir de sus ramos cursados

y una clasificación a partir de los ramos pertenecientes al catálogo de cursos, entregando una herramienta para poder clarificar la toma de decisiones y formar profesionales íntegros que le atribuyan un mayor valor a su carrera. Esto mediante dos sistemas distintos, el primero un sistema de categorización que permite al estudiante clasificar todos los ramos pertenecientes a un tópico o área de estudio de su interés, y el segundo, un sistema de recomendación, a partir de un filtro colaborativo basado en usuarios, que permite al estudiante obtener como recomendaciones todos aquellos cursos que han cursado usuarios con características similares. Junto con crear ambos sistemas, se decidió crear una interfaz Web que permite interactuar a los estudiantes pertenecientes a la FCFM, siendo estos los potenciales usuarios a los que ayudarán ambas herramientas.

El sistema de categorización permitirá al estudiante identificar aquellos ramos que pertenezcan a un área de interés, facilitando la inscripción de cursos. Al tener claridad de qué cursos inscribir, el estudiante logrará especializarse de mejor manera adquiriendo mayor conocimiento del tópico en el que desea desempeñarse como profesional, o bien, en el que desea hacer mayor énfasis en sus estudios. De esta manera, el estudiante podrá consolidarse en las áreas que estime conveniente, automatizando el proceso de búsqueda e investigación para encontrar ramos de su interés.

El sistema de recomendación permitirá al estudiante automatizar la solicitud de referencias de cursos a sus compañeros, debido a que permite hallar los ramos cursados por otros estudiantes que presentan un perfil similar, es decir, aquellos que han cursado los mismos ramos y que además poseen notas parecidas. De esta manera, el estudiante obtendrá recomendaciones de otros compañeros con perfiles similares, sin la necesidad de conocerlos y solicitarles sus recomendaciones. Este sistema utiliza como base un filtro colaborativo basado en usuarios, que es el mismo principio que utilizan sistemas de recomendación sofisticados, como los desarrollados por Google o Netflix.

1.2. Objetivos del trabajo

Objetivo General

El objetivo de este trabajo consiste en crear nuevas herramientas que permitan orientar a los estudiantes en la toma de decisiones, al momento de realizar la selección de los ramos a cursar durante su carrera, a partir de factores como la oportunidad, el entorno y la personalidad del estudiante. Principalmente, se debe implementar una interfaz que muestre al estudiante opciones para obtener retroalimentación visual de su perfil académico, compañeros con perfiles similares y recomendaciones para la inscripción de ramos.

Objetivos Específicos

1. Identificar las distintas áreas de estudio asociados a un curso para poder categorizarlos a partir de sus nombres, planes o programas de estudio.

2. Establecer un perfil de estudiante a partir de los ramos cursados, asociados a las áreas de estudio, para crear un recomendador de los cursos de interés que se pueden inscribir.

3. A partir del perfil del estudiante, identificar aquellos ramos no cursados, que pertenecen a sus principales áreas de interés, para tener las referencias en el recomendador al momento de realizar la inscripción académica.

4. Establecer una conexión entre los perfiles de los estudiantes, identificando el perfil propio del estudiante y el de los compañeros, relacionando a aquellos estudiantes que presentan aptitudes similares.

Capítulo 2

Marco Teórico

Para entender el análisis y la posterior implementación de la presente memoria, deben ser conocidos conceptos, métodos, y tecnologías que han sido utilizadas. En este marco teórico se explica el problema de clasificación automática de texto del procesamiento del lenguaje natural, y se describen los principales métodos para abordar el problema.

Posteriormente, se estudia el sistema de recomendación a partir de la técnica de filtro colaborativo basado en usuarios, y se explica el principio matemático por el cuál está diseñado.

Finalmente, se describen las tecnologías y conceptos usados para construir la interfaz Web que ha sido utilizada como herramienta para la validación del presente trabajo.

2.1. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (NLP, por sus siglas en inglés) es una rama de la inteligencia artificial que ayuda a las computadoras a entender, interpretar y manipular el lenguaje humano. Uno de los problemas más destacados en este campo de estudio, corresponde a la clasificación automática de textos en categorías preexistentes cuyo propósito es que a partir de conjuntos de datos o textos no etiquetados, se deben detectar los tópicos recurrentes y crear las categorías.

Para entender mejor los dos casos de clasificación, se debe comprender bien la concepción del *Machine Learning* o Aprendizaje Automático (ML, por sus siglas en inglés), que consiste básicamente en automatizar, mediante distintos algoritmos, la identificación de tendencias que pueden ser abstraídas a partir de los datos. Existen diferentes tipos de implementación de *Machine Learning* y pueden clasificarse en dos grandes categorías, el aprendizaje supervisado y el no supervisado.

El aprendizaje supervisado, es el paradigma más popular del *Machine Learning*, en este los algoritmos trabajan a partir de datos "etiquetados". Primero se debe contar con un conjunto de datos clasificados manualmente, llamado conjunto de entrenamiento. Posteriormente, se deben extraer las características del conjunto, intentando encontrar una función que, a partir de variables de entrada, les asigne la etiqueta de salida adecuada, corroborando que el resultado coincida con la clasificación manual del conjunto de entrenamiento. Dentro de los ejemplos más clásicos de este paradigma, podemos hallar el de clasificación de correos de

tipo “spam”, que utilizan empresas como *Google* o *Microsoft*.

En el aprendizaje no supervisado no se disponen de datos “etiquetados”, por lo que no existe un conjunto de entrenamiento que permita entrenar un algoritmo o hallar una función que verifique la correctitud de las clasificaciones. En su lugar, sólo se conocen los datos de entrada. Por lo que, sólo se puede describir la estructura de los datos, para intentar encontrar algún tipo de patrón que permita simplificar la organización de los datos. El ejemplo más destacado para este tipo de *Machine Learning*, es un sistema de recomendación basado en el usuario, como los que utilizan plataformas como *Youtube* o *Netflix*. En estos sistemas, a partir de los datos del historial del usuario se intenta predecir que busca y que sugerencias se le puede mostrar.

2.2. Topic Modeling

Topic Modeling o modelamiento de tópicos (en español), es una tarea del *Machine learning* no supervisado, propia del procesamiento del lenguaje natural, de tipo estadístico, que tiene como objetivo automatizar el análisis de texto para determinar un *cluster*[1] de palabras para un conjunto de documentos, y así, identificar estructuras semánticas relacionadas en un corpus[2], es decir, documentos identificados por tópicos. Existen diversas técnicas para abordar este problema, entre los más populares están los modelos probabilísticos, Asignación Latente de Dirichlet [2] (LDA siglas en inglés) y el Análisis semántico probabilístico latente (PLSA) [3]. Por otro lado existen técnicas que buscan resolver esta tarea basadas en *word embedding* [4], como Word2Vec [5], Doc2Vec [5] y Top2Vec [6].

2.2.1. Asignación Latente de Dirichlet (ALD)

La asignación latente de Dirichlet (ALD), es un algoritmo propio del procesamiento del lenguaje natural, basado en un modelo estadístico generativo que se utiliza a menudo para modelar temas o tópicos. El algoritmo asigna temas a los documentos y genera distribuciones de temas a partir de las palabras pertenecientes a un conjunto de textos, de esta manera, se proporcionar una forma de identificar automáticamente el contenido de cada documento.

2.2.2. Word embedding

Word Embedding es una representación del vocabulario de un corpus, que permite hallar las palabras que poseen similitud semántica y sintáctica. La idea principal del *embedding* es crear representaciones vectoriales densas y de baja dimensionalidad de las palabras a partir de su contexto.

Por ejemplo, considere los siguientes nombres de ramos: “Bases de Datos” y “Minería de Datos”. Si bien, ambos ramos poseen un enfoque distinto, ambos comparten el análisis y uso de sistemas de bases de datos. Dejando el conocimiento de lado, es evidente que existe una relación entre ambos sólo analizando el nombre. Sin embargo, existen ramos cuyos nombres

[1] La clusterización aplicado al contexto del presente trabajo es la categorización de la información de un curso para generar agrupaciones relevantes y así definir que cursos pertenecen a un cluster.

[2] Un corpus lingüístico o corpus es un conjunto amplio y estructurado de ejemplos reales de palabras pertenecientes a la lengua, como por ejemplo, un documento.

pueden resultar por ejemplo, totalmente distintos, pero que aún así pertenecen a la misma área de estudio o que comparten tópicos en su programa. Para estos casos es importante utilizar todos los recursos descriptivos que poseen los ramos, como su programa, nombre y requisitos. De esta manera, se podrá obtener una correlación a partir de las palabras y el contexto que constituyen dicha información. Entre los modelos de *Word Embedding* más populares, destacan, *GloVe*, *FastText* y *Word2vec*.

Word2vec es una de las técnicas más populares para construir *Word embedding*. El algoritmo *Word2vec* utiliza distintos modelos de redes neuronales poco profundas para aprender las relaciones entre palabras a partir de un corpus. Esta técnica nos permite utilizar dos modelos de arquitecturas para producir una representación distribuida de palabras, *Skip-gram* y *Continuous bag-of-words* (CBOW).

Word2Vec toma como entrada un gran corpus de texto y produce un espacio vectorial, típicamente de varios cientos de dimensiones, y a cada palabra única en el corpus se le asigna un vector correspondiente en el espacio. Los vectores de palabras se colocan en el espacio vectorial de manera que las palabras que comparten contextos comunes estarán a menor distancia unas con otras. Luego lo que sigue es extender esta técnica para cada uno de los nombres de los ramos, y así encontrar aquellos que presentan cercanía, para esto, lo que se suele hacer es promediar los vectores asociados a las palabras pertenecientes a un título, en este caso, el nombre del curso, obteniendo así, un vector único asociado al nombre. De esta manera ramos con nombres similares como lo son "Bases de datos" y "Minería de datos", estarían cercanos en el espacio vectorial en comparación a ramos como "Teatro" y "Análisis y Diseño de Algoritmos".

Cabe destacar que el uso de esta técnica como otras de *Word embedding*, son capaces de utilizar modelos preentrenados que poseen representaciones densas de palabras. De este modo se facilita la clasificación de las palabras para un conjunto de entrada, a partir de los vectores que describen el modelo. Un ejemplo de esto son los modelos de *Spanish Word Embeddings* [7] entrenados a partir de diferentes técnicas y corpus.

2.2.3. Doc2Vec

Como se mencionó anteriormente, utilizar *Word Embedding* permite encontrar las palabras que poseen similitud semántica y sintáctica. Los algoritmos como *Word2vec* son a menudo preentrenados a partir de un corpus que identifican estadísticas de coocurrencia, lo que puede generar un gran problema, esto es debido a que al usar modelos preentrenados, la técnica de *Word Embedding* se aplica sin tomar en cuenta el contexto, por ejemplo, palabras con polisemia como puede ser "bolsa", que se puede ver en frases como, "la bolsa de basura es grande" y "la Bolsa de Santiago es grande", tendrá exactamente el mismo vector asociado en el espacio vectorial, pese a tener un significado distinto en ambas frases. Posteriormente al extender la técnica a cada una de las palabras de las frases se obtendrán resultados extraños relacionando "basura" con "Santiago", debido a que se tendrá una cercanía en el espacio vectorial. La solución a este problema es entrenar representaciones contextuales sobre el corpus, es decir, identificar todas aquellas palabras que poseen polisemia y distintos usos gramaticales, para asignarle una representación vectorial a partir de sus contextos. Existen diversos algoritmos dedicados a *Contextual Embedding*, entre los más populares están *ElMo*, *Flair* y *BERT*.

2.2.4. BERT

Representaciones de codificador bidireccional de Transformer (*BERT* [8], en sus siglas en inglés), es un algoritmo creado por Google en el año 2018. El objetivo de este algoritmo es el de interpretar nuestro lenguaje de búsqueda de un modo más natural, mediante PNL (Programación neuro lingüística).

La manera en que el algoritmo de *BERT* logra interpretar el lenguaje de búsqueda, es utilizando la bidireccionalidad, esto consiste en analizar una misma frase en dos direcciones, tanto de inicio a fin como viceversa. Esta bidireccionalidad, le permite al algoritmo identificar las palabras que rodean a la palabra clave, permitiendo entender en profundidad el contexto y el significado semántico de cada frase. Al igual que *Word2vec*, *BERT* permite cargar modelos preentrenados. *BETO* [9] es un modelo de lenguaje pre-entrenado exclusivamente con datos en español, este modelo cuenta con mejores resultados en comparación a otros modelos basados en *BERT*, previamente entrenados en corpus multilingües.

2.2.5. Top2Vec

Top2Vec, es un algoritmo para el modelado de tópicos y búsqueda semántica. Detecta automáticamente los tópicos presentes en el texto y genera vectores asociados a dichos tópicos, documentos y palabras, integrados de forma conjunta. Cuando el algoritmo es entrenado para un conjunto de documentos, este permite obtener la cantidad de tópicos detectados, buscar tópicos a través de palabras clave, buscar documentos a partir de un tópico y encontrar documentos similares, como también palabras similares.

Básicamente este algoritmo permite asociar un tópico subyacente para un conjunto de documentos semánticamente similares. El primer paso es crear una incrustación conjunta de vectores de documentos y palabras. Una vez que los documentos y las palabras están incrustados en un espacio vectorial, el objetivo del algoritmo es encontrar grupos densos de documentos y luego identificar qué palabras atrajeron esos documentos juntos. Cada área densa es un tópico y las palabras que juntaron los documentos en un área densa son las palabras del tópico.

2.2.6. Clustering

Clustering o clusterización es una de las técnicas de la ciencia de datos más importante en el aprendizaje de máquinas, es usado para encontrar patrones o grupos dentro de un conjunto amplio de datos. Por lo demás, esta técnica tiene una acción fundamental en el aprendizaje automatizado debido a que permite a los algoritmos entrenar y conocer de forma adecuada los datos con los que se desarrollan sus actividades.

2.2.6.1. K-Means Clustering

K-Means [10] es un algoritmo de agrupamiento, cuyo objetivo es particionar un conjunto de n elementos en k grupos distintos, en donde cada elemento pertenece al grupo cuyo valor medio es más cercano. Las agrupaciones del conjunto de datos suelen ser ilustradas mediante una partición del espacio de datos en celdas de Voronoi.

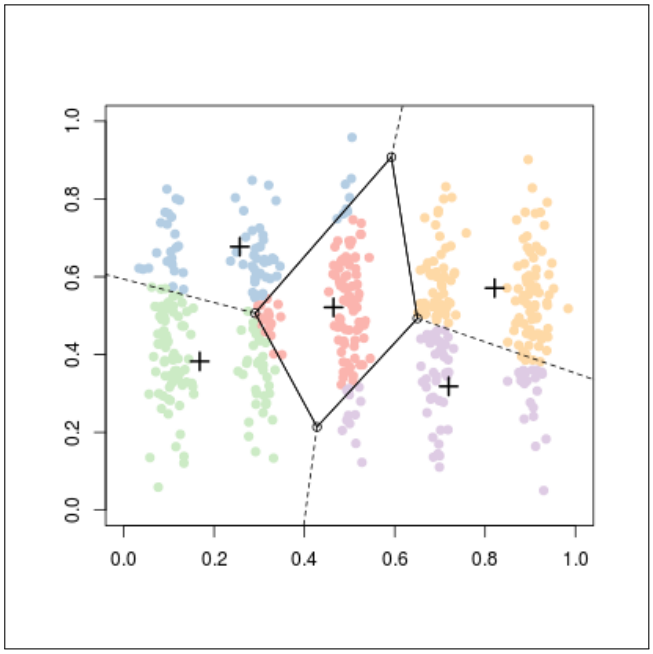

Figura 2.1: Ejemplo de visualización de agrupaciones de K-Means mediante celdas de Voronoi [11]

2.2.6.2. HDBSCAN Clustering

Hierarchical Density-Based Spatial Clustering of Applications [12] (HDBSCAN, en sus siglas en inglés) es la agrupación espacial jerárquica basada en la densidad de aplicaciones con ruido, es usado en el aprendizaje no supervisado para hallar clusteres o regiones densas de un conjunto de datos. El agrupamiento jerárquico permite a un usuario explorar la estructura del cluster dentro de sus datos en múltiples niveles de resolución, sin la necesidad de obligarlo a hacer una suposición con respecto al número de clusteres.

2.2.7. Técnicas de reducción de dimensionalidad

Existen técnicas que permiten reducir representaciones vectoriales de múltiples dimensiones a espacios bidimensionales o tridimensionales. Esto permite mejorar la visualización de datos y sus interpretaciones.

2.2.7.1. t-SNE: T-distributed Stochastic Neighbor Embedding

T-distributed Stochastic Neighbor Embedding [13] (*t-SNE*) es una técnica no lineal no supervisada que permite visualizar datos de alta dimensión, al asignarle a cada punto de un espacio multidimensional una ubicación en un mapa bidimensional o tridimensional. La técnica es una variación de *Stochastic Neighbor Embedding* [14] (*SNE*), sin embargo, es mucho más fácil de optimizar y produce visualizaciones significativamente mejores al reducir la tendencia a agrupar puntos en el centro del mapa.

El uso de *t-SNE*, permite visualizar en un espacio bidimensional, los títulos de los distintos cursos, obteniendo de esta manera mejorar en los resultados de las clasificaciones al poder distinguir la cercanía entre un curso y otro a partir de la distancia euclidiana. Además, permite observar los distintos grupos o *clusters* relacionados a los cursos que pertenecen a un área de estudio.

2.2.7.2. UMAP: Uniform Manifold Approximation and Projection

Uniform Manifold Approximation and Projection [15] (UMAP), es un algoritmo de reducción de dimensionalidad que puede ser usado para una visualización similar a la de t-SNE, como también para la reducción de dimensionalidad no lineal. Está basado teóricamente en la geometría de Riemannian y la topología algebraica, que ha sido desarollado por McInnes y Healy. Este algoritmo permite construir representaciones vectoriales reducidas a partir de vectores en múltiples dimensiones.

2.3. Sistemas de recomendación

Los sistemas de recomendación son algoritmos diseñados para sugerir elementos relevantes para el usuario, como podrían ser productos, películas, libros e inclusive cursos académicos. Existen diferentes técnicas en los que están basados estos sistemas, entre las más populares se reconocen aquellas qué están basadas en el usuario y las que están basadas en el ítem o elemento.

2.3.1. User-Based Colaborative Filtering

User-Based Colaborative Filtering (o filtro colaborativo basado en usuario en español), es una técnica usada principalmente por sistemas recomendadores que consta de un proceso de filtrado de información o modelos, que usan métodos que implican la colaboración entre múltiples agentes y conjuntos de datos. Además, el filtrado colaborativo permite hacer predicciones automáticas sobre los intereses de un estudiante, mediante la recopilación de datos obtenidos a partir de la construcción de su perfil. El Coeficiente de correlación de Pearson forma parte del algoritmo para generar las predicciones hechas por el sistema de recomendación.

2.3.1.0.1. Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson, en estadística es una medida de dependencia lineal entre dos variables aleatorias cuantitativas. Considerando al estudiante y a los ramos como dos variables aleatorias, que por lo demás son cuantitativas, es posible utilizar este coeficiente para hallar la correlación entre ellas. Esta medida nos entrega información acerca de la intensidad y la dirección de la relación entre las variables, es decir, corresponde a un índice que mide el grado de covariación entre las variables relacionadas linealmente.

El coeficiente de correlación de Pearson cuando es aplicado a una muestra, es definido por r_{xy} y es referido como "coeficiente de correlación muestral de Pearson". Dados n pares de datos $\{(x_i, y_i)\}_{i=1}^{n}$, se define el coeficiente de correlación muestral de Pearson como

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{2.1}$$

donde

- n es el tamaño de la muestra.
- x_i, y_i, son puntos muestrales individuales indexados con i.
- $\bar{x}$ denota la media muestral definida por $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (análogamente para $\bar{y}$).

Por otro lado, dados los valores muestrales de dos variables aleatorias $X(x_1, \ldots, x_n)$ e $Y(y_1, \ldots, y_n)$, considerados como vectores pertenecientes a un espacio de n dimensiones, entonces pueden construirse vectores centrados como

$$X(x_1 - \bar{x}, \ldots, x_n - \bar{x}) \quad \text{e} \quad Y(y_1 - \bar{y}, \ldots, y_n - \bar{y}). \tag{2.2}$$

El coseno del ángulo alfa entre estos dos vectores es dado por la siguiente fórmula

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

(2.3)

Por tanto, el coeficiente de correlación muestral de Pearson puede ser expresado como el $cos(\alpha)$.

$$r_{xy} = cos(\alpha) \tag{2.4}$$

Luego, la similitud del coseno no es más que el cálculo entre dos vectores pertenecientes al mismo espacio, por lo que el coeficiente de correlación muestral de Pearson es homologo a la similitud del coseno. Por tanto, es posible utilizar librerías que obtengan esta similitud para utilizar coeficiente de correlación muestral de Pearson como técnica para hallar la correlación entre los estudiantes y los ramos.

2.4. Herramientas y conceptos para la validación.

Para la creación del instrumento de validación del presente trabajo, se han utilizado tecnologías para desarrollar y exponer los resultados. Así también, para entender el uso de dichas tecnologías es importante manejar el contexto en el que se desarrollan, estudiando los conceptos mencionados descritos en la validación e implementación del sistema.

2.4.1. Flask

Flask es un framework de carácter minimalista escrito en el lenguaje Python que permite crear diversos tipos de aplicación, entre ellas, plataformas Web, servicios de APIs, desarrollo de back-end, entre otras opciones, de manera rápida y usando un mínimo de líneas de código. A pesar de su carácter minimalista, este framework tiene la cualidad de ser escalable según la complejidad de la Web que se desea crear. Por lo que es una base sólida para desarrollar una plataforma, en particular el back-end. Está basado en la especificación Web Server Gateway Interface (WSGI) de Werkzeug y utiliza como motor de templates Jinja2, además cuenta con

licencia Berkeley Software Distribution (BSD), una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License.

2.4.2. Flask-Restful

Flask-Restful es una extensión para el framework Flask, que permite construir servicios REST APIs de manera rápida y sencilla. Esta extensión conserva el carácter minimalista de Flask, junto con priorizar las buenas prácticas que conlleva este carácter, permitiendo de esta manera, integrar un servicio API REST en simples pasos a una aplicación ya existente.

2.4.3. Intercambio de recursos de origen cruzado (CORS)

El intercambio de recursos de origen cruzado o CORS (Cross-origin resource sharing, en sus siglas en inglés) es un mecanismo que permite realizar peticiones de recursos restringidos a dominios distintos de aquel dominio origen que realiza la petición.

2.4.4. Secure Sockets Layer (SSL)

El Nivel de conectores Seguros (Secure Sockets Layer o SSL, en inglés) es el protocolo para navegadores web y servidores que permite la autenticación, encriptación y desencriptación de datos enviados a través de Internet. Este protocolo suministra un canal seguro y cifrado, entre dos dispositivos que operan a través de Internet o de una red local. Es distribuido mediante certificados y es requerido por los browsers para garantizar la conexión del usuario con la página Web que se desea visitar, si aparecen las letras HTTPS al principio de la dirección (URL) del sitio web, dicho sitio está protegido por un certificado SSL.

2.4.5. Localtunnel

Localtunnel es un servicio gratuito que permite compartir un servicio Web alojado de manera local, sin lidiar con problemas de DNS o configuraciones de firewall. Este servicio asigna una dirección URL única y pública que actuará como proxy para todas las peticiones entrantes a la máquina local que está corriendo el servicio. Además, la principal característica de este servicio es otorgar certificados SSL gratuitos, por lo que permite garantizar la seguridad al usuario que ingresa mediante la dirección URL generada.

2.4.6. Escala de Likert

La Escala de Likert es una escala de calificación que se utiliza para consultar a un usuario o persona acerca de su nivel de acuerdo o desacuerdo con una afirmación o declaración. Se suele usar para medir reacciones, actitudes, valoración de servicios, probabilidad de hacer una acción futura y medir los comportamientos de una persona. En el presente trabajo, ha sido utilizada para validar los sistemas desarrollados con los estudiantes.

Para realizar la medición de la escala, se divide el grado de acuerdo en cinco niveles, de acuerdo, muy de acuerdo, en desacuerdo, muy en desacuerdo e indiferente, sin embargo, con propósito de forzar al usuario para emitir un juicio ante la pregunta, se ha optado por quitar el grado de indiferencia en la presente memoria. Por lo demás es difícil obtener retroalimentación ante una respuesta neutra, lo que hace más difícil tomar acciones ante la pregunta realizada.

Capítulo 3

Situación Actual

3.1. Plataforma Ucampus

Ucampus actualmente cuenta con herramientas que permiten al estudiante recibir ayuda en su formación y en la toma de decisiones durante su carrera. La primera consta de una vista en la plataforma que muestra el catálogo de cursos visto en la Figura 3.1, en el cual el estudiante puede descubrir los cursos impartidos durante el semestre y semestres previos.

El catálogo despliega la información descriptiva de cada curso, tales como, su nombre, requisitos, equivalencias y créditos, además muestra las secciones disponibles en el semestre junto a sus horarios, los profesores que imparten el curso, la cantidad de cupos disponibles para estudiantes y aquellos que están ocupados. Además, cuenta con filtros de búsqueda por semestre y departamento, esto permite al estudiante evaluar la continuidad de los ramos impartidos entre un semestre y otro, sin embargo, existen cursos que sólo son impartidos en semestres de primavera u otoño, esta cualidad puede ser inferida en el mismo catálogo, sin embargo, el estudiante debe confirmar la disponibilidad del curso para semestres próximos.

Es importante destacar, que al comienzo de cada semestre y previo a la inscripción académica, la mayoría de cursos a impartir durante el semestre, están publicados en el catálogo, por lo que el estudiante puede examinar la información dispuesta para cada curso.

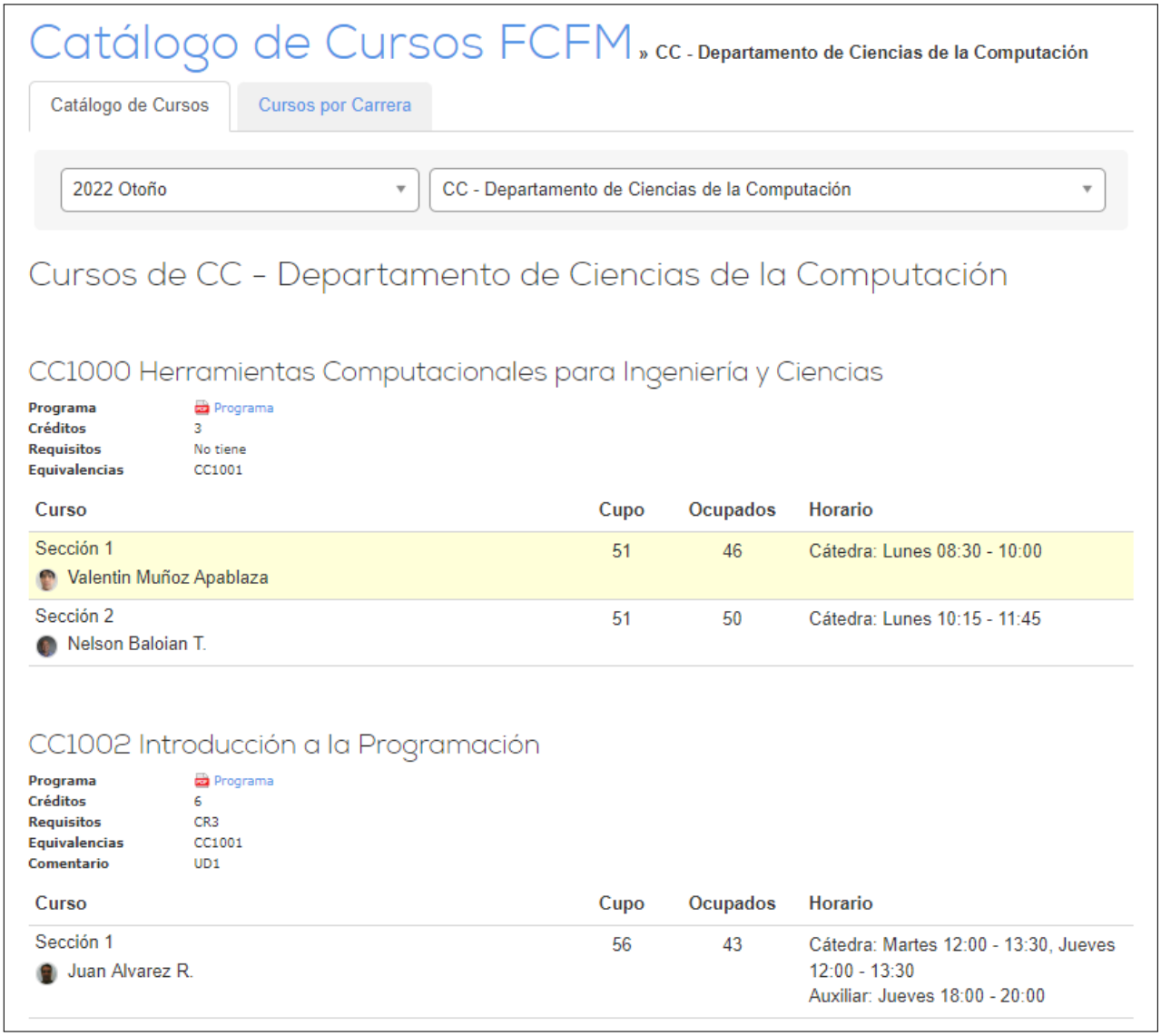

Figura 3.1: Catálogo de cursos de Ucampus.

Otras herramientas con las que cuenta la plataforma Ucampus y permiten orientar al estudiante con el propósito de formación deseado, se encuentran en la sección de boletines, esta sección está subdividida por otras 6, Historial, Antecedentes, Inscripción Académica, Notas, Posiciones y Recuento de Créditos. Entre estas subsecciones se destacan la de Notas y la de Recuento de Créditos.

En la sección de Notas se puede apreciar un gráfico de aptitudes (Figura 3.2) basado en los ramos cursados en los distintos departamentos de la facultad a la cual pertenece la carrera del estudiante. Este gráfico permite tener una visión del rendimiento que tiene el estudiante con respecto a un departamento, sin embargo, no refleja el verdadero interés o perfil del estudiante referente a una especialidad. Si se analiza con detención cada uno de los vértices del grafo, sólo se puede identificar el promedio del estudiante asociado a un área de estudio, y no así, la cantidad de ramos asociados a esta, por lo que el gráfico realmente no muestra los intereses del estudiante.

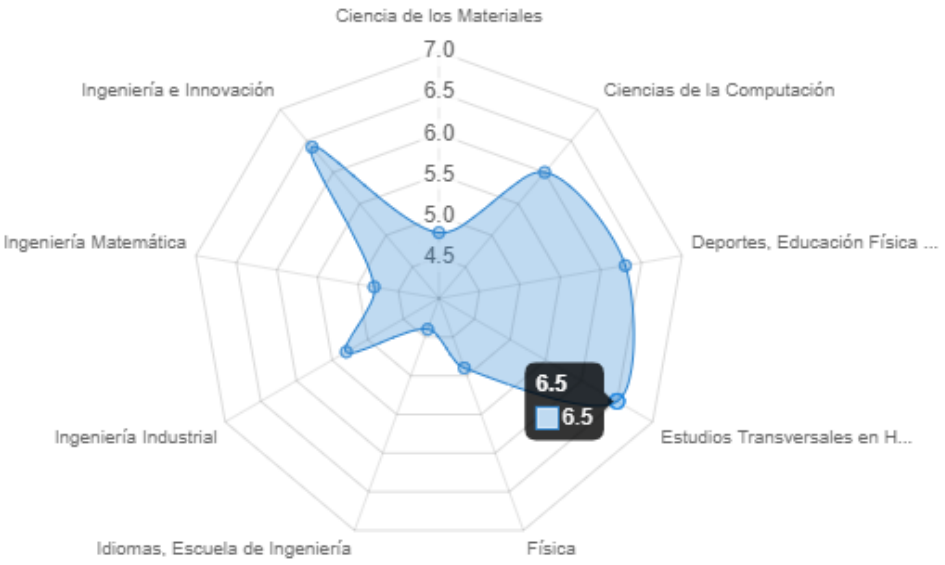

Figura 3.2: Gráfico de aptitudes de Ucampus.

Por otro lado, existe la sección de Recuento de Créditos, que despliega un resumen de todos los créditos aprobados y los que quedan por aprobar pertenecientes a los planes de estudio que cursa el estudiante. Esta herramienta permite al estudiante identificar con facilidad mediante un "checklist" (Figura 3.3) que ramos le restan por aprobar para completar su plan de estudio. Sin embargo, el recuento no permite profundizar en los intereses y la especialización del estudiante, ya que sólo despliega los planes creados e impuestos por la escuela para obtener el título o un grado universitario.

	Ramo		Créditos	Nota	Semestre
	Ingeniería Civil en Computación v3	*Todo 2 de 3*	326	5.7	
✓	Subplan: Ingenieria Computacion v3, Especialidad		76	*	
✓	Subplan: Licenciatura en Ciencias de la Ingeniería, Mención Computación v3		247	*	
	Subplan: Ingenieria Computacion v3, Titulación		3	*	
	Ingenieria Computacion v3, Especialidad	*Todo 8 de 8*	76	6.6	
✓	Subplan: CC5601/CC5602/IN5502		6	*	
✓	Subplan: IN3301/IN4301		6	*	
✓	Subplan: Ingeniería Electivos Computación		33	*	
✓	Subplan: Ingenieria Integral Computación		6	*	
✓	CC5402 Proyecto de Software		12	6.9	2021 Otoño
✓	CC5901 Práctica Profesional II		7	6.7	2020 Primavera
✓	CC5401 Ingeniería de Software II		6	6.5	2020 Primavera
✓	EI2090 Examen de Suficiencia en Inglés II		0	T	2021 Otoño

Figura 3.3: Recuento de créditos de Ucampus.

La inscripción académica dispuesta por Ucampus cuenta con distintos tipos de filtros, entre ellos, el filtro por cursos que no pertenecen al plan de estudio y los cursos que no cumplen los requisitos necesarios para ser inscritos. Si bien, se disponen los cursos pertenecientes al plan de estudio, el estudiante suele inscribir aquellos ramos que debe cursar obligatoriamente debido a que forman parte de la malla de su carrera. De la misma manera, el estudiante debe cursar ramos electivos para completar su formación integral.

¿Cómo decide el estudiante qué cursos electivos debes inscribir? Si la planificación de la carrera se hiciera de manera eficiente, los estudiantes estarían siguiendo un plan de carrera con una toma de decisiones informada, en lugar de hacerlo por mera casualidad. Por otro lado, algunos estudiantes no comienzan a explorar sus posibilidades profesionales "reales" hasta después de haberse titulado. El éxito profesional de los estudiantes se puede lograr mejor si se les entrega la orientación adecuada para elegir el curso correcto en la universidad, idóneo a la personalidad, su entorno, la capacidad y su intelecto.

Finalmente, se debe mencionar que, para la factibilidad del presente tema de memoria, Ucampus ha dispuesto mediante un acuerdo de confidencialidad la información necesaria para realizar los sistemas propuestos. Además ha brindado un API, que permite consultar la información del catálogo de cursos, junto con entregar conjunto de datos, como historiales de estudiantes de manera anonimizada, por tanto, es posible afirmar que se pueden llevar a cabo nuevas funcionalidades a partir de los datos, que permitan orientar y ayudar al estudiante en la toma de decisiones y planificación de su carrera.

3.2. Experiencia de estudiantes

Como se mencionó previamente en el primer capítulo, se realizó una encuesta a 18 estudiantes pertenecientes a distintos departamentos de la facultad (Figura 3.4), en la cual se buscaba obtener información acerca de la inscripción de ramos y su formación durante la carrera. Es importante destacar, que los estudiantes encuestados se encuentran cursando sus últimos años de carrera, esto debido a que cuentan con más experiencia que estudiantes de primeros años, han interactuado más veces con la plataforma de Ucampus y poseen mayor conocimiento de los cursos pertenecientes a sus planes de estudios. Por lo demás, han cursado más electivos por lo que poseen mayor claridad de las áreas de estudio que son de sus intereses. La encuesta permite dimensionar cuantitativamente las problemáticas relativas a la planificación del estudiante en su carrera, como también sus experiencias usando las herramientas dispuestas en la plataforma de Ucampus.

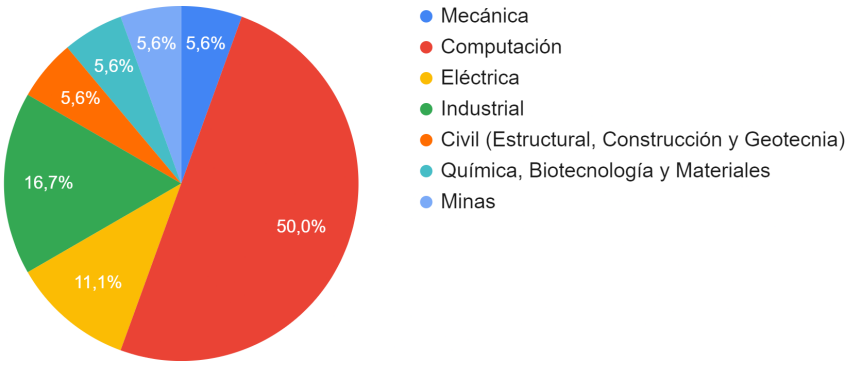

Figura 3.4: Gráfico de porcentaje de los estudiantes encuestados y sus respectivas especialidades.

La primera pregunta (Figura 3.5) tiene como objetivo indagar sobre la dificultad asociada a la inscripción de cursos, durante la inscripción académica. Esto permite identificar la necesidad de los estudiantes para recibir ayuda o mejores recursos para tomar decisiones en su formación, inscribiendo cursos electivos con conocimiento. Es importante reiterar la diversidad de las carreras a los cuales pertenecen los estudiantes encuestados, debido a que permite generalizar la problemática expuesta en múltiples especialidades.

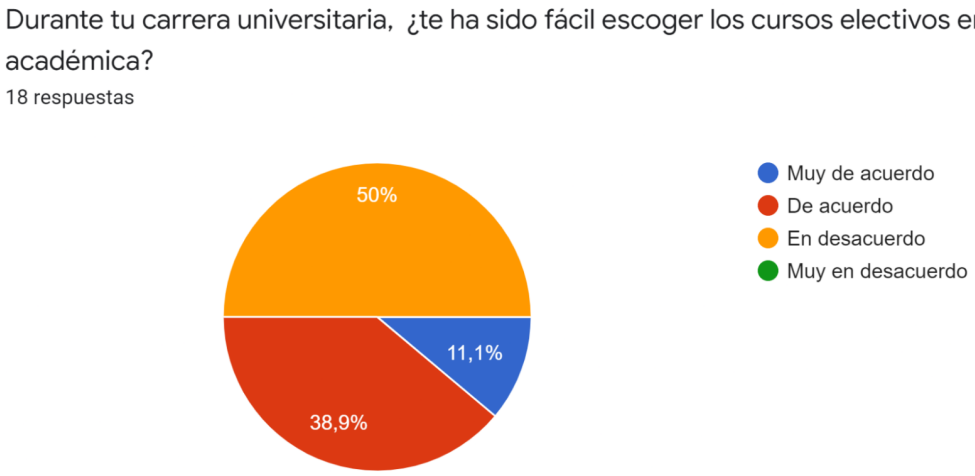

Figura 3.5: Pregunta 1, experiencia estudiante: “Durante tu carrera universitaria, ¿te ha sido fácil escoger los cursos electivos en la inscripción académica?”.

La segunda pregunta (Figura 3.6) tiene como objetivo constatar si el estudiante ha podido identificar aquellos ramos que pertenecen a una rama o área de estudio de su especialidad, esto con el fin de evidenciar la dificultad del estudiante para relacionar los ramos ante no tener un conocimiento previo de los mismos. El tener conocimiento de las áreas de estudio pertenecientes a la carrera permite al estudiante poseer una especialización detallada, despertando sus intereses formativos y profesionales desde los primeros años de carrera.

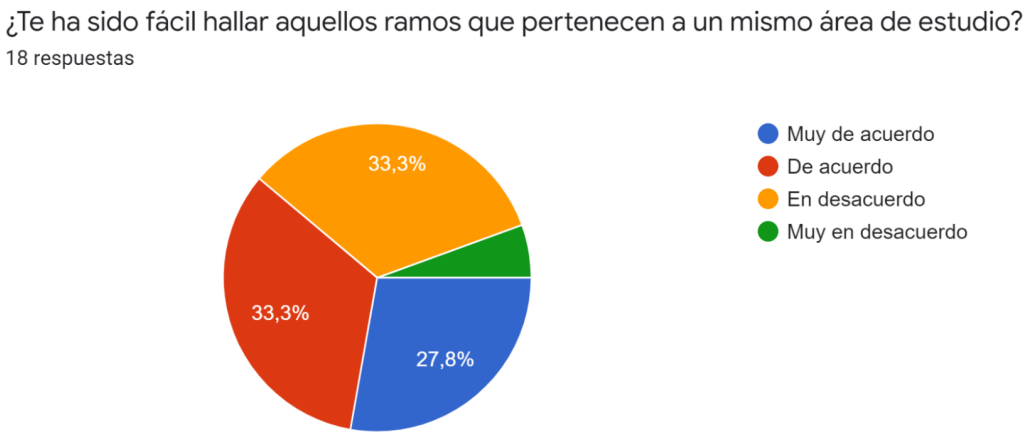

Figura 3.6: Pregunta 2, experiencia estudiante: "¿Te ha sido fácil hallar aquellos ramos que pertenecen a un mismo área de estudio?".

La tercera pregunta (Figura 3.7) permite cuantificar y evidenciar lo común que es la solicitud de referencias o recomendaciones entre los estudiantes, esto con el fin de ratificar un sistema de recomendación como posible funcionalidad u herramienta para desarrollar.

Figura 3.7: Pregunta 3, experiencia estudiante: "¿Has solicitado referencias de otros estudiantes al momento de inscribir un curso?".

La cuarta pregunta (Figura 3.8) evidencia que pese a la especialización que ha obtenido el estudiante en su carrera, la mayoría de ellos no tiene clara su proyección laboral ni el rubro al cual les gustaría dedicarse. En el estudio "Factores que afectan en la carrera de estudiantes universitarios" [16] realizado por Nancy T. Pascual, director de la Universidad del Sistema Ryzal, se ratifica la importancia de los ramos cursados con respecto al mundo laboral. El primer antecedente a considerar es que los estudiantes graduados que han cursado ramos inadaptados a su capacidad, intereses y habilidades, tienen mayor dificultad al momento de aplicar a algún trabajo, en comparación a aquellos que si han tomado ramos acordes a sus

capacidades e intereses. Esto se debe a que su habilidad no se adapta a los cursos que han tomado, por lo que no liberarán su máximo potencial. Los posibles cursos elegidos que no se adaptan a los estudiantes también puede conducir a su incapacidad para calificar a las competencias que necesitan las empresas.

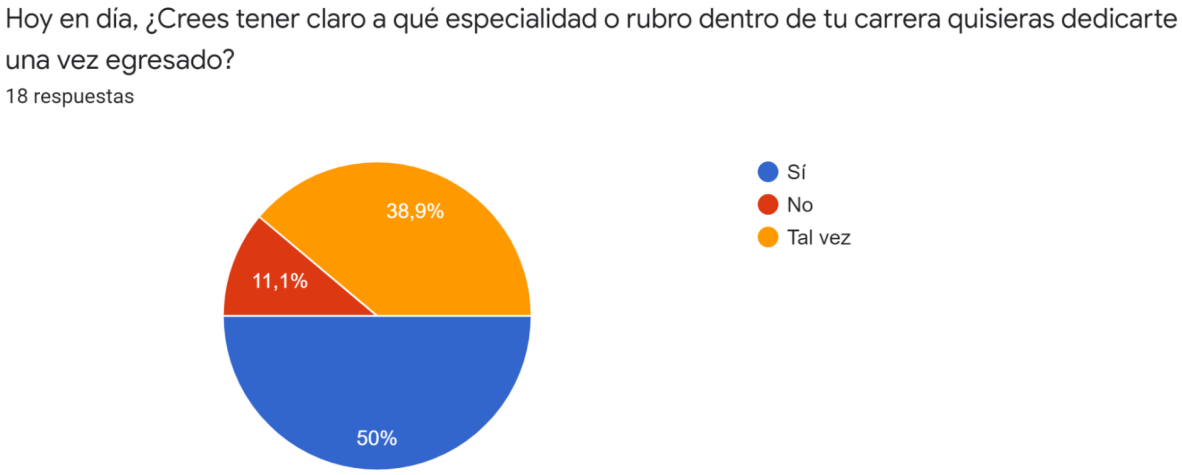

Figura 3.8: Pregunta 4, experiencia estudiante: "Hoy en día, ¿Crees tener claro a qué especialidad o rubro dentro de tu carrera quisieras dedicarte una vez egresado?".

La quinta y última pregunta (Figura 3.9) de experiencia del estudiante, tiene como objetivo evidenciar y verificar la necesidad de crear nuevas herramientas que permitan brindar ayuda al estudiante, como también dar origen a la oportunidad a abordar en el presente trabajo.

Figura 3.9: Pregunta 5, experiencia estudiante: "Durante tu carrera universitaria, ¿te ha sido fácil escoger los cursos electivos en la inscripción académica?".

Finalmente, queda evidenciado que al menos de la mitad de los estudiantes encuestados han tenido problemas a la hora de encontrar cursos afines y estarían dispuestos a usar una herramienta que les permitan facilitar la inscripción de ramos, identificar áreas de estudio y mejorar la toma de decisiones con el fin de obtener una mejor proyección como estudiante o profesional.

3.3. Estado del Arte

Las problemáticas asociadas a la toma de decisiones de un estudiante universitario durante su formación académica, en instancias de inscripciones académicas, la selección de una especialidad e inclusive la clarificación de sus propios intereses, son problemas recurrentes y que pueden afectar a universitarios en cuyas universidades tienen un sistema de inscripción académico libre y sujeto a las decisiones del estudiante. En el contexto internacional, son varias las Universidades que han lidiado con las problemáticas planteadas, algunas han desarrollado sistemas que permiten ayudar al estudiante en sus decisiones a partir de sus intereses, sin embargo, son sistemas de búsqueda basados en filtros, cuyos cursos están sujetos a una etiqueta, que es asociada en la creación del curso.

Un ejemplo de un sistema de búsqueda de cursos creado para ayudar a los estudiantes y mejorar su experiencia formativa, es el de la Universidad de Harvard que ha utilizado un producto de Oracle (PeopleSoft Enterprise PRTL Interaction Hub), basado a grandes rasgos en consultas óptimas a bases de datos, usando expresiones regulares para búsquedas densas. Para entender mejor el sistema descrito, se describirá a continuación la interfaz (Figura 3.10) con la que cuenta Harvard para buscar cursos.

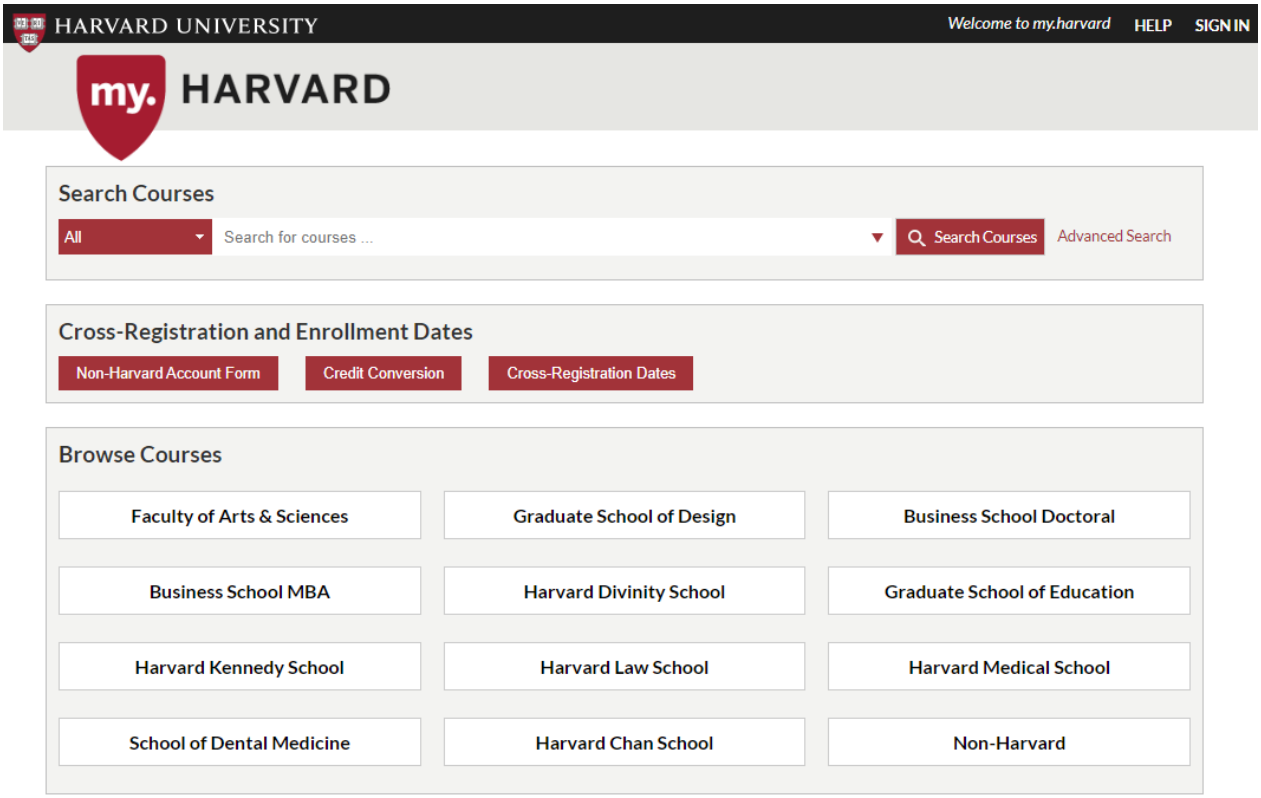

Figura 3.10: Sistema de búsqueda de cursos del sitio my.harvard de la Universidad de Harvard.

La interfaz es accedida mediante las credenciales del estudiante, por lo que debe estar previamente identificado. Al acceder se le ofrece al estudiante dos sistemas de búsqueda, uno por cursos y otro a través de una búsqueda avanzada. El primero (Figura 3.11), permite completar un campo de texto con el curso o los cursos que se desean buscar, y al presionar "Search Courses" realizar la búsqueda.

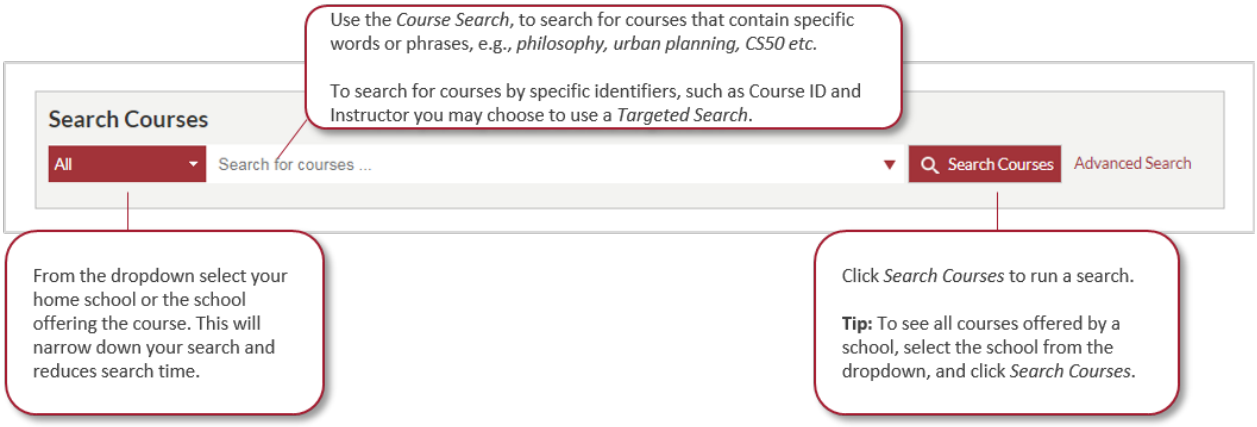

Figura 3.11: Búsqueda de cursos por nombre del sitio my.harvard de la Universidad de Harvard.

El segundo sistema de búsqueda avanzada (Figura 3.13), es accedido mediante la misma vista, seleccionando "Advanced Search". Este sistema, permite especificar la búsqueda completando criterios, para refinar los resultados de los cursos buscados. Adicionalmente, es posible ingresar aún más atributos (Figura 3.12) descriptivos, estos permiten categorizar la búsqueda de acuerdo al departamento o área al cual pertenecen los cursos.

Ambos sistemas están basados en consultas a la base de datos que ofrece un catálogo de cursos con a lo mínimo 33 atributos descriptivos (considerando los campos dispuestos en la interfaz) por cada uno, junto a un programa de contenidos o documentación del mismo, permitiendo construir consultas óptimas que permitan hacer *match* de las palabras, expresiones o atributos con los de los cursos.

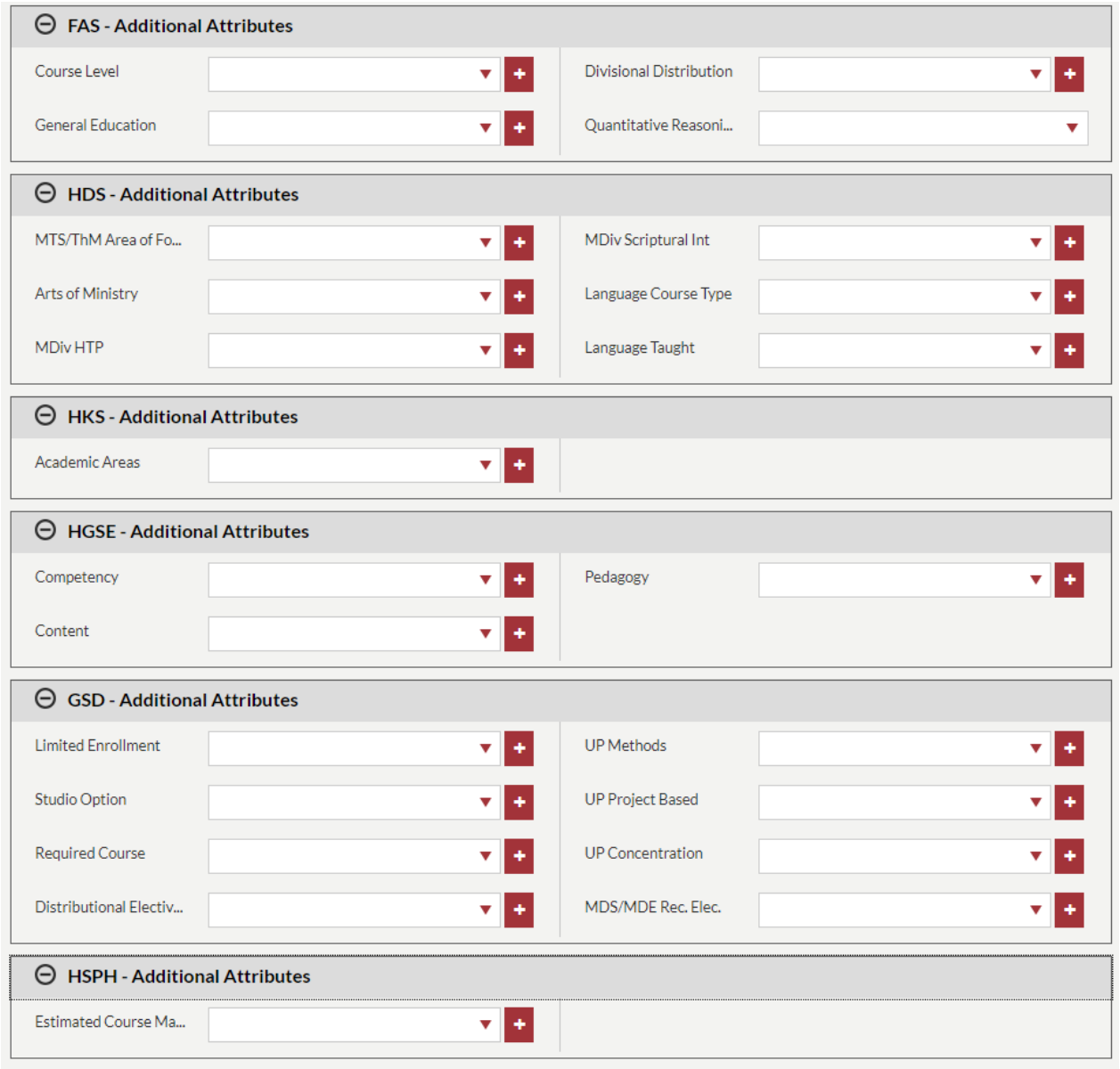

Figura 3.12: Atributos de búsqueda avanzada de cursos por nombre del sitio my.harvard de la Universidad de Harvard.

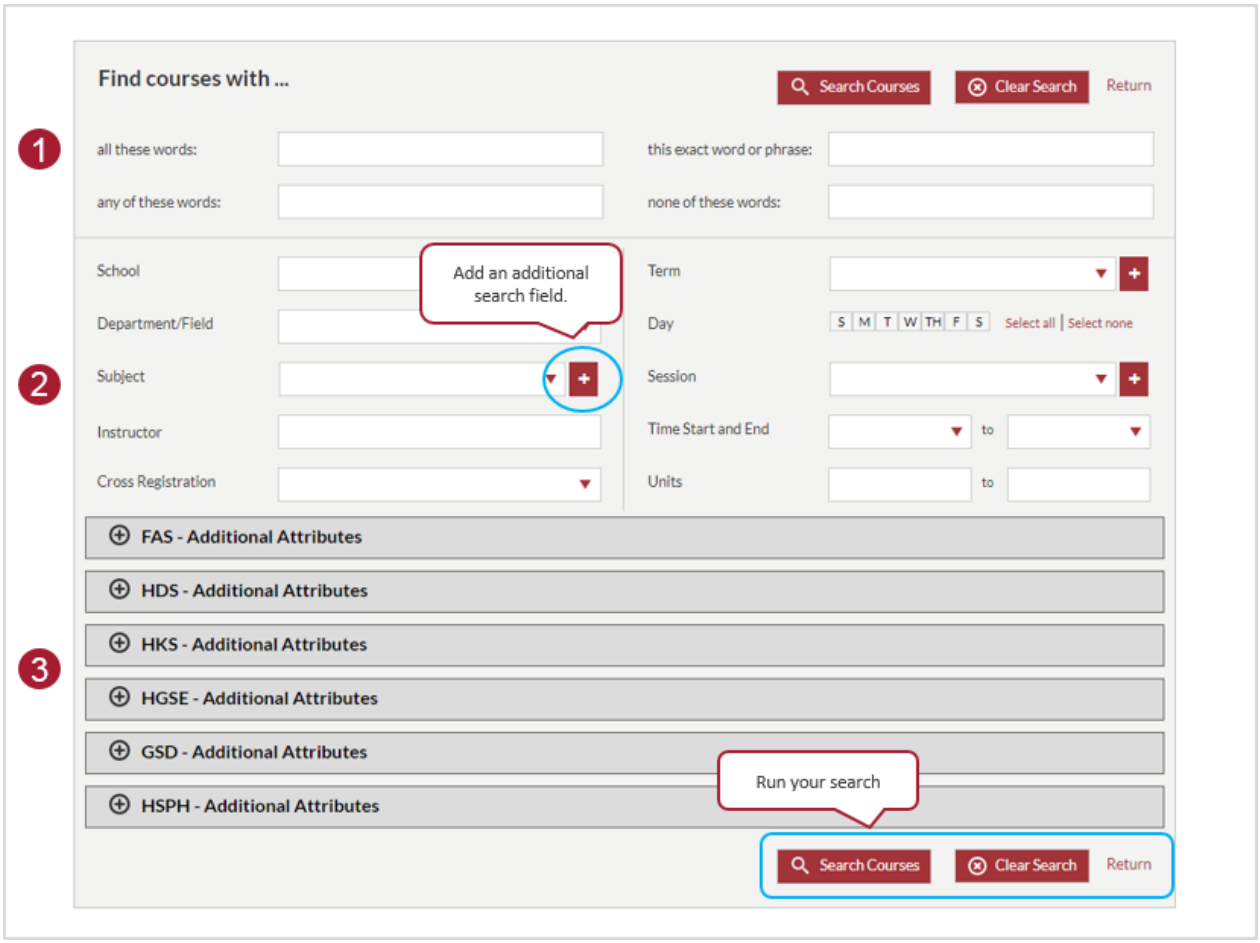

Figura 3.13: Búsqueda avanzada de cursos por nombre del sitio my.harvard de la Universidad de Harvard.

Como el caso de Harvard, existen otras universidades que utilizan un sistema de búsqueda parecido para brindar una herramienta a los estudiantes que permita indagar y conocer los cursos. Podrían tomarse como referencia para replicar en el caso de la facultad de FCFM, sin embargo, esto no es posible, debido a que a diferencia de la plataforma "my.harvard" que cuenta con una ficha exhaustivamente descriptiva para cada curso, Ucampus sólo cuenta con pocos datos descriptivos, como el nombre, requisitos, departamento, créditos y programa. Por lo demás, un sistema como el descrito no podría ser replicado, debido a que cada curso registrado en la plataforma de Harvard cuenta con una categorización de área o tópico según el departamento al cual pertenece, y es justo este atributo el que se desea obtener en el sistema de categorización propuesto en la presente memoria.

Capítulo 4

Análisis y diseño

4.1. Sistema de categorización

El sistema de categorización tiene como objetivo principal poder clasificar cursos a partir de las áreas de estudio a la cual pertenecen. Para cumplir con dicho propósito, esto puede ser estudiado como un problema de clasificación de aprendizaje automático, tanto supervisado como uno no supervisado. En la primera opción, se debe contar con un conjunto de datos de entrenamiento creado manualmente y en la segunda se debe hallar una manera de organizar los datos.

Para escoger bajo qué paradigma del aprendizaje automático se estudiará la categorización, se debe tener en cuenta el escenario y los recursos disponibles. Al considerar una metodología supervisada se deben crear etiquetas manualmente para construir un conjunto de datos de entrenamiento, esto es debido a que a partir de dicho conjunto será posible generar un modelo predictivo. Sin embargo, sólo la Escuela de Ingeniería cuenta con aproximadamente 3269 cursos distintos perteneciente a diversos departamentos, por lo que etiquetar cursos por áreas de estudio requiere tener conocimiento de los cursos y las áreas a las cuales pertenecen. Pese a ser un conjunto de entrenamiento en el cual sólo una parte de los cursos deben ser etiquetados, el trabajo requiere de especialistas de las distintas áreas por lo que dicho proceso sería laborioso. Adicionalmente, al extender el sistema de categorización al resto de facultades de la Universidad de Chile, la iteración y construcción del conjunto de datos de entrenamiento sería aún más compleja.

Para ejemplificar la complejidad de una preparación manual de un conjunto de entrenamiento se podría pensar en el efecto negativo de etiquetar reiterativamente de manera errada los cursos, perjudicando directamente las clasificaciones obtenidas por los modelos generados.

Finalmente, debido a los datos e información con la que se cuenta y a lo descrito previamente, el problema debe ser abordado como uno de aprendizaje no supervisado. Es decir, se debe hallar una manera para organizar o hacer "clustering" a partir de los distintos datos que se pueden obtener de cada uno de los ramos. Este es un problema de modelado de lenguaje típico en NLP. Por ello es que se abordará con las dos principales técnicas que se suelen utilizar: *Word embedding* y *Contextual Embedding*.

4.1.1. Identificando alternativas

Al tratar el problema como uno clasificación o modelado de texto, existen diversas alternativas con las cuales se puede tratar, sin embargo, debido a esta diversidad es que se debe estudiar y evaluar cada una de ellas. Entre las alternativas más populares se encuentran, la Asignación Latente de Dirichlet (LDA, en inglés), los métodos de "*Word embedding*", "Word2Vec" y "Doc2Vec" y una combinación de estos dos últimos, "Top2Vec".

4.1.1.1. Asignación Latente de Dirichlet

El modelado de tópicos en términos estadísticos, se utiliza para descubrir estructuras semánticas latentes a partir de una gran colección de documentos. Entre los métodos más populares basados en este principio, se pueden encontrar el Asignación Latente de Dirichlet y el Análisis probabilístico semántico latente, ambos explotan la inferencia estadística para descubrir patrones latentes en los datos. Sin embargo, pese a la popularidad de ambos, estos presentan debilidades, y la principal es que para discretizar los documentos, es necesario conocer la cantidad exacta de tópicos existentes, que es justamente uno de los objetivos a resolver. Por otro lado, con documentos breves, con pocas palabras, las observaciones que realizan estos métodos, son insignificantes para inferir un tópico o parámetros que permitan identificar parcialmente el documento dentro de una colección. Además, cursos cuyos programas no tengan una representación clara de los contenidos que se imparten, aumentarán el grado de dificultad al momento de categorizar, por otro lado, ambos métodos están basados en una representación de "Bag of words", por lo que se ignora el orden de las palabras y la misma semántica.

4.1.1.2. Word2Vec

Las representaciones distribuidas forman parte de diversas técnicas del procesamiento de lenguaje natural en *Machine Learning*. Estas se utilizan para generar representaciones vectoriales de palabras y documentos. Una de las técnicas más populares basada en este principio es "Word2vec" [3]. Básicamente este algoritmo introduce una representación distribuida de las palabras capturando la relación sintáctica y semántica entre ellas. Existen dos modelos para construir la representación distribuida en esta técnica, "Continuous Bag of Words" (CBOW) y "Skip Gram", y además, tienen un rendimiento diferente según el "corpus" al cual se desea operar con esta técnica. De acuerdo a Tomáš Mikolov, autor original del paper Word2Vec, Skip Gram trabaja mejor con conjuntos de datos pequeños, también encuentra representaciones de palabras raras o poco frecuentes con un mejor desempeño. Por otro lado, CBOW es más rápido y produce mejores representaciones para una mayor cantidad de palabras frecuentes, por tanto, considerando el tamaño del conjunto de datos con los que se trabaja, se

[3] Word2vec, es una de las técnica de Word Embbeding que ha sido estudiada y probada, su prueba está disponible en la sección de Anexo de la presente memoria.

ha optado por analizar las representaciones obtenidas por ambas técnicas, generando mejores representaciones al utilizar el modelo CBOW. Por otro lado, pese a poder contar con las representaciones de palabras de manera distribuida, aún se requiere de algún identificador asociado al grupo de palabras referidas a un documento, con el fin de poder realizar comparaciones entre un programa y otro.

4.1.1.3. Doc2Vec

Doc2Vec es una técnica basada en Word2Vec, el objetivo de este algoritmo es crear una representación numérica de un documento, independientemente de su tamaño. A diferencia de las palabras en Word2Vec, los documentos no están estructurados de manera lógica, sin embargo, Doc2Vec soluciona este problema de una forma sencilla, para esto, utiliza el modelo *Distributed Memory version of Paragraph Vector* (PV-DM). En términos simples, este modelo permite hallar la similitud entre documentos a partir del uso de un identificador por párrafo, oración o documento, así, es posible utilizar un algoritmo de clusterización como K-Means para hallar la similitud de las representaciones generadas por Doc2Vec y así encontrar los programas de cursos con similitud.

4.1.1.4. Top2Vec

Top2Vec [6] es un algoritmo que permite detectar los tópicos presentes en un texto, o un conjunto de documentos. Este algoritmo está constituido por pasos. En primera instancia, genera una representación vectorial de los documentos, a partir de un modelo generado por el algoritmo de Doc2Vec, o también, utilizando un modelo preentrenado como BERT multilingual o BETO. Al tener una representación vectorial para cada documento, reduce la dimensión de los vectores generados utilizando el algoritmo UMAP con el fin de mejorar la clusterización que se realiza posteriormente mediante el algoritmo HDBSCAN. Finalmente, asigna los tópicos a cada cluster identificando la palabra más representativa del mismo. A diferencia del uso de Doc2Vec junto a K-Means, Top2Vec optimiza las representaciones vectoriales, reduciendo la dimensionalidad, esto mejora la rapidez de clusterización junto con preservar la estructura global de la información. En términos simples, el tópico asignado a cada cluster no es más que el calculo de vectores como un centroide de cada área densa de documentos muy similares, en el que dicho centroide es considerado como el vector asociado a un documento más representativo del área.

4.1.2. Evaluación de los métodos

Dado a que es posible describir Top2Vec como una construcción en términos de los métodos de Word2Vec y Doc2Vec, basta con comparar y evaluar los modelos probabilísticos como LDA y PLSA junto a Top2Vec. Existen varios ventajes de Top2Vec por sobre los métodos tradicionales de topic modeling como LDA y PLSA, la primera y la más importante es la capacidad de hallar de manera automática la cantidad de tópicos, junto con que dichos tópicos son más representativos e informativos con respecto al corpus. Por otro lado, Top2Vec no requiere usar *stop-words* para preprocesar el corpus, permitiendo encontrar tópicos sin contar con un dominio en el idioma. Finalmente, el uso de una representación distribuida de palabras permite reducir los desafíos de métodos tradicionales basados en técnicas como *Bag Of Words*, que ignoran la semántica de las palabras.

Dimo Angelov, autor del algoritmo de Top2Vec realizó un experimento [6] para comparar y evaluar el rendimiento de su técnica junto a los modelos probabilísticos LDA y PLSA, para

esto utilizo como conjunto de datos las respuestas asociadas al sitio de Yahoo Answers, constituido por cerca de 1.3 millones de publicaciones. Estas pertenecen a 10 tópicos diferentes y existen 130.000 publicaciones por tópico. La cantidad de tópicos encontrados por Top2Vec fue de 2618, mientras que para entrenar los modelos LDA y PLSA, el costo computacional sólo permitió entrenar los modelos desde 10 a 100 tópicos con intervalos de 10.

Los resultados del experimento han sido expuestos en las Tabla 4.1 y Tabla 4.2, "*Probability-weighted amount of information*" (PWI), con una traducción literal al español como "Cantidad de información ponderada por probabilidad", en términos generales, es una medida numérica que expresa cuán relevantes son un conjunto de palabras para una colección de documentos. Por tanto, al analizar ambas tablas es evidente que las categorizaciones obtenidas por Top2Vec, poseen mejor PWI total, es decir, un 996.6, a diferencia del PWI total obtenido por LDA que es sólo de un 360.9. Además, analizando superficialmente las categorizaciones obtenidas por LDA, es posible identificar palabras no pertenecientes al lenguaje, o bien, que no poseen significado semántico, como lo es la segunda y tercer categorización de los tópicos del algoritmo LDA.

Tabla 4.1: Tópicos de 10 palabras generados por Top2Vec, entrenado en 20 grupos de información constituidos por 20 tópicos.

Topic Number	**Topic Words**	**PWI(T)**
1	pitching, pitchers, pitcher, hitter, batting, hit, hitters, baseball, batters, inning	74.2
2	bike, ride, riding, bikes, motorcycle, bikers, helmet, riders, countersteering, passenger	71.9
3	circuit, voltage, circuits, resistor, signal, khz, impedance, analog, diode, resistors	69.1
4	centris, ram, mhz, quadra, nubus, vram, iisi, lciii, cpu, fpu	62.4
5	patient, symptoms, patients, doctor, disease, treatment, jxp, therapy, skepticism, physician	59.1
6	koresh, fbi, compound, batf, davidians, atf, waco, raid, fire, bd	54.7
7	israel, arab, arabs, israeli, jews, palestinians, israelis, war, peace, occupied	54.3
8	orbit, space, launch, orbital, satellites, lunar, shuttle, spacecraft, moon, earth	53.7
9	clipper, nsa, encryption, encrypted, secure, keys, crypto, algorithm, escrow, scheme	51.6
10	controller, drives, drive, ide, scsi, floppy, bios, disk, jumpers, esdi	50.3
11	windows, drivers, ati, cica, driver, exe, card, autoexec, mode, ini	50.1
12	car, engine, cars, ford, brakes, honda, tires, valve, wheel, rear	49.7
13	hockey, playoffs, nhl, game, season, team, playoff, teams, scoring, play	48.0
14	gun, guns, firearms, laws, weapons, handgun, crime, amendment, handguns, firearm	46.6
15	window, application, xlib, manager, openwindows, motif, server, xview, client, clients	43.6
16	jesus, christ, god, bible, church, scripture, christians, scriptures, christian, heaven	38.9
17	postscript, format, printer, fonts, files, formats, font, truetype, bitmap, image	36.7
18	shipping, sale, offer, condition, asking, brand, sell, obo, price, selling	36.0
19	atheists, belief, religion, beliefs, god, christianity, truth, religions, believe, atheist	34.4
20	please, mail, post, email, posting, address, thanks, reply, interested, appreciate	11.3
		996.6

Tabla 4.2: Tópicos de 10 palabras generados por LDA, entrenado en 20 grupos de información constituidos por 20 tópicos.

Topic Number	**Topic Words**	**PWI(T)**
1	la, pit, gm, det, bos, tor, pts, chi, vs, min	49.9
2	hz, cx, ww, uw, qs, c_pl, lk, ck, ah	47.0
3	ax, max, pl, di, tm, ei, giz, wm, bhj, ey	42.6
4	db, period, goal, play, pp, shots, st, power, mov, bh	27.9
5	mk, mm, mp, mh, mu, mr, mj, mo, mq, mx	27.7
6	health, medical, new, study, research, disease, cancer, use, patients, drug	19.0
7	armenian, people, said, one, armenians, turkish, went, us, children, turkey	17.3
8	dos, windows, drive, card, system, disk, mb, scsi, pc, mac	16.7
9	file, program, window, files, image, jpeg, use, windows, display, color	15.7
10	government, president, law, would, mr, israel, state, rights, fbi, states	13.4
11	god, jesus, bible, church, christian, christ, christians, faith, lord, man	12.2
12	game, team, games, hockey, season, teams, league, nhl, new, players	12.0
13	space, nasa, earth, launch, shuttle, orbit, moon, satellite, solar, mission	11.8
14	edu, ftp, graphics, available, pub, image, mail, com, version, also	9.7
15	would, know, anyone, get, thanks, like, one, please, help, could	8.5
16	key, use, data, system, one, information, may, encryption, used, number	7.5
17	people, would, one, think, know, like, say, even, see, way	7.3
18	one, car, would, like, get, time, much, also, back, power	7.1
19	edu, com, please, list, mail, sale, send, email, price, offer	4.0
20	think, year, would, good, time, last, well, get, one, got	3.6
		360.9

Finalmente, tras realizar la comparación entre LDA y Top2Vec, este último resulta ser el método seleccionado para construir el sistema de categorización, utilizando como recursos dentro del mismo algoritmo los métodos de Word2Vec y Doc2Vec. Sin embargo, pese a escoger está opción, se debe hacer la comparación entre usar un modelo preentrenado como podría ser BETO, o entrenar un nuevo modelo a partir de Doc2Vec.

4.1.3. Abordando el problema con Top2Vec

Con los resultados del algoritmo Top2Vec expuestos previamente, es entonces que se ha optado por usar este algoritmo para resolver el problema planteado para un sistema de categorización. Cabe destacar, que se sugiere utilizar un modelo preentrenado como BETO, para un vocabulario extenso y variado, y entrenar un modelo basado en Doc2Vec para un vocabulario único, teniendo esto en cuenta, es posible que al considerar todos los programas de la facultad, se abarque un espectro amplio de ciencias y áreas de estudio, por lo que un modelo preentrenado podría abordar de manera más completa el corpus, como también podría pasar el caso en que los programas posean una descripción única del curso, por tanto el vocabulario resultaría ser más técnico. Por otro lado, se debe determinar la manera en que visualizaran los clusteres encontrados por el algoritmo, afortunadamente este cuenta con métodos para extraerlos desde un modelo.

4.1.4. Diseño del pipeline

Con la elección de Top2Vec como algoritmo a utilizar, se debe definir el pipeline con el cual se abordará el problema. En primer lugar se debe tener en cuenta la construcción de

un dataset, recolectando los datos, para esto Ucampus dispone de una API a la cual se le es posible consultar todos los cursos pertenecientes a la facultad, como también cada uno de sus datos descriptivos, entre ellos, su nombre, código, requisitos o créditos, y programa como documentos. Sin embargo, no son suficiente como para preparar un dataset robusto, por lo que la alternativa directa para obtener información de un curso es extraer la información contenida en su programa. Con el texto extraído, lo siguiente consta de realizar un preprocesamiento del texto debido a posibles problemas en la extracción. Al ser un documento "pdf", las librerías suelen extraer todo aquello que puede ser representado como un carácter, ocasionando que surjan palabras no pertenecientes a un idioma, entre ellas, palabras monosilábicas que no aportan un carácter descriptivo al texto. Por otro lado, existen palabras comunes entre los programas, y que podrían sesgar los resultados de la categorización, entre ellas, "controles", "evaluaciones", "nota", "prueba", "laboratorio", "unidades", "ayudantes", etc. Estas palabras deben ser purgadas manualmente, debido a que es posible, que dos programas posean de manera reiterada estas palabras y estén presente en ambos documentos a la vez, ocasionando que dos programas totalmente distintos puedan pertenecer a un mismo cluster.

Posteriormente, lo que sigue es entrenar y generar el modelo utilizando el algoritmo de Top2Vec. El algoritmo recibe como parámetro una lista con un conjunto de documentos, por tanto, dado que se cuenta con un dataset cuyas filas representan a cada uno de los cursos, las columnas de dichas filas deberán ser unificadas, concatenando en un sólo texto toda la información descriptiva. De esta manera, se construirá una lista a partir de la información de las filas unificadas y serán pasadas como parámetro al algoritmo. Por otro lado, el algoritmo también permite recibir un modelo preentrenado de *embedding* (Por ejemplo, BETO o modelos basados en BERT), esto es opcional debido a que por defecto el algoritmo entrenará un modelo de *document embedding* mediante Doc2Vec, sin embargo, con propósito de comparar los resultados y obtener el mejor, se deben probar ambas opciones.

Con el modelo entrenado, lo que sigue es almacenarlo para poder ser cargado y expuesto mediante un API REST. Para esto se debe obtener la información entrenada en el modelo, afortunadamente el algoritmo de Top2Vec cuenta con los métodos necesarios para obtenerla. Entre ellos, obtener la cantidad de tópicos hallados, los cursos que constituyen un tópico, palabras claves de los tópicos, e inclusive una búsqueda de los tópicos más familiares a un curso en específico. De esta manera, se debe crear un servicio API REST capaz de exponer mediante algún end-point[4] los métodos mencionados, para esto, se ha optado por crear una aplicación usando Flask como tecnología del framework, debido a que es posible compartir el mismo lenguaje de programación usado por la implementación de Top2Vec, Python. Por otro lado, al ser un framework minimalista, e incremental, permite crear algo simple sin la necesidad de darle mucho enfoque y dedicación, ya que no es el principal objetivo de la presente memoria.

Para ilustrar los resultados, se creará un front-end, constituido únicamente por una vista, en la cual se desplegará un formulario conformado por un campo de texto y un botón para enviar, junto con una sección gráfica que ilustre los resultados luego de enviar el formulario. El usuario objetivo es un estudiante que desea conocer los ramos que pertenezcan a un área de su interés, por lo que el campo de texto a rellenar debe ser capaz de procesar y validar palabras clave, nombre o código que haga referencia a un curso o un conjunto de cursos. Al

[4] Los end-points son las URLs de un API o un backend que responden a una petición HTTP.

enviar el formulario, este debe hacer una petición al servicio de API REST solicitando los cursos pertenecientes al tópico ingresado. Posteriormente, el API debe ser capaz de contestar mediante el "Respond" el o los cursos asociados al tópico estructurados en un JSON. Luego, al obtener respuesta de la petición en el front-end, esta debe ser ilustrada mediante un grafo (Ejemplo de grafo, Figura 4.1) en el cual se representara el curso solicitado o más familiar a su petición, junto a los cursos que estén relacionados, es decir, aquellos que pertenecen al misma área.

Un ejemplo del flujo descrito es el del diagrama de secuencia de la Figura 4.2, es importante notar que al ingresar un tópico que no obtiene resultados al hacer la petición al API REST, esta responderá con un error "HTTP 500", esto debido al diseño de Flask-Restful, por lo que se deberá hacer un manejo de eso errores en la implementación, para evitar confusiones entre una falla del servidor en la cual está alojado el API REST y la inexistencia de resultados. Se debe destacar que el diagrama dispuesto ejemplifica la secuencia a nivel general, por tanto, no han sido incluidas en él, validaciones en Front-end y el servicio API REST.

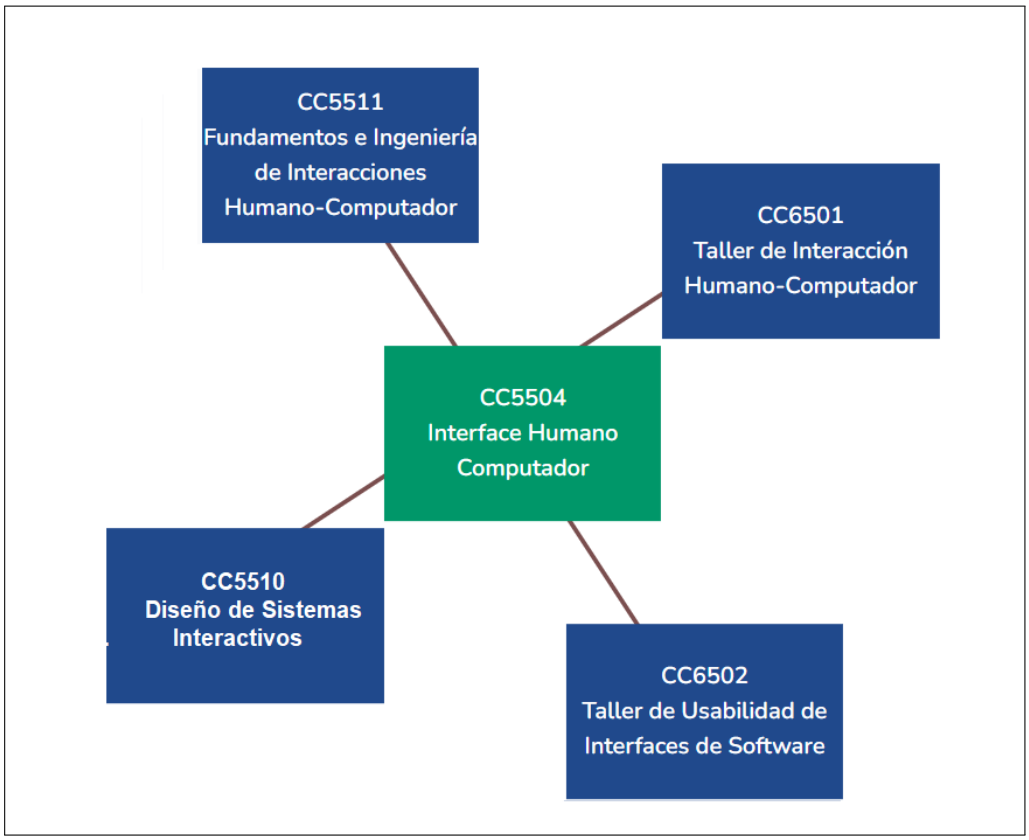

Figura 4.1: Prototipo de grafo representando el área de "Interacción Humano Computador" del departamento de Computación.

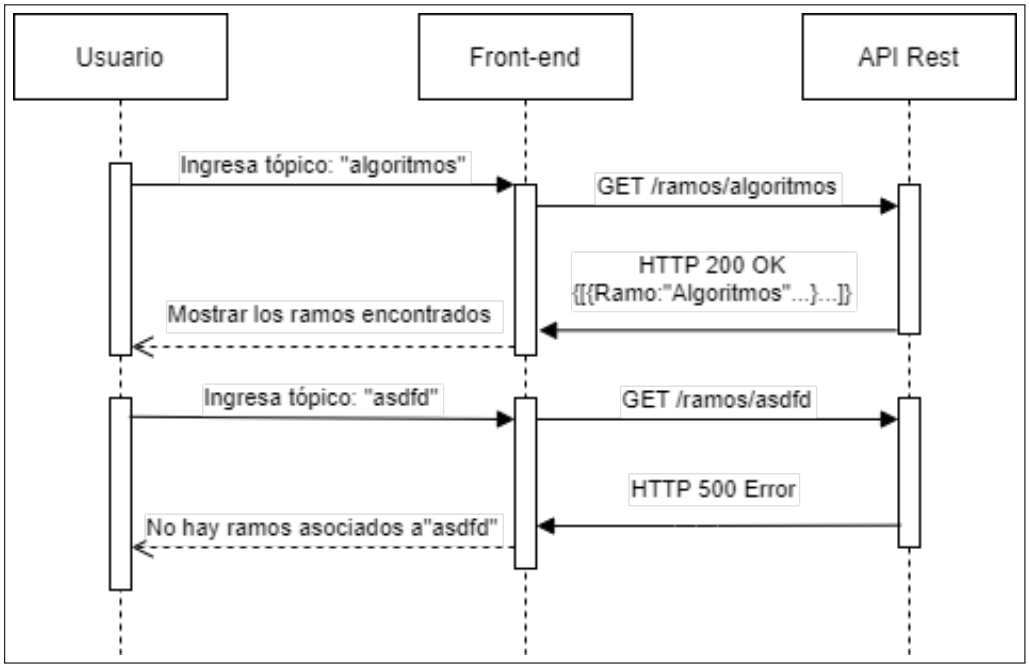

Figura 4.2: Ejemplo de diagrama de secuencia para sistema de categorización.

4.2. Sistema de recomendación

Los sistemas de recomendación, son algoritmos que intentan "predecir" las preferencias de un usuario. Se definen como entradas de recomendación dadas por los usuarios, que luego el sistema agrega y dirige a los destinatarios apropiados. Puede definirse además como un sistema que produce recomendaciones individualizadas como salida o tiene el efecto de guiar al usuario de forma personalizada a alternativas interesantes en un espacio más amplio de opciones posibles.

Existen estrategias de recomendación que son usadas con mayor frecuencia, también se les llaman tipos de motor, haciendo referencia a un motor de recomendación. Entre ellas podemos distinguir, "Popularity", que sugiere al usuario las opciones por popularidad, "Content-based", que, a partir de las opciones visitadas por el usuario, se intenta adivinar qué busca el usuario y así ofrecer alternativas similares, y la estrategia "Colaborative Filtering", que utiliza la información de todos los usuarios para identificar perfiles similares y aprender de los datos para recomendar opciones de manera individual.

El sistema de recomendación tiene como objetivo, establecer un perfil del estudiante a partir de los ramos cursados, asociados a las áreas de estudio a la cual pertenecen. Esto permite al estudiante, identificar aquellos ramos no cursados, que podrían ser de su interés. Para cumplir con este objetivo, se ha decidido que el sistema de recomendación debe ser estudiado mediante el uso de un filtro colaborativo basado en el usuario, debido a que se tiene a disposición información de los estudiantes, como historial de ramos y calificaciones por ramos, esto permite hallar similitudes entre los distintos perfiles de estudiantes, y así identificar aquellos que son más parecidos entre ellos. Con perfiles similares, es posible sugerir cursos asociados a un perfil que no hayan sido cursados por otro perfil similar, y que podrían representar un

grado interés al momento de realizar una inscripción académica.

Para entender cómo funcionará el filtro colaborativo basado en usuarios, consideremos un grupo de 4 estudiantes del Departamento de Computación, Juan, Pedro, Camila y Sofía, junto a 4 cursos del mismo departamento, Herramientas Computacionales para Ingeniería y Ciencias (CC1000), Desarrollo de Aplicaciones Web (CC5002), Taller de Programación Competitiva A (CC4005) y Taller de Hacking Competitivo (CC5325), tal como se aprecia en la Tabla 4.3. Veamos el grado de similitud que tiene Juan con sus compañeros, para esto se debe calcular primero el promedio de las notas de todos los estudiantes, excluyendo CC5325 que aún Juan no ha cursado, por tanto, esto lo calculamos como

Tabla 4.3: Tabla de ejemplo para sistema de recomendación

	CC1000	CC5002	CC4005	CC5325
Juan	6.5	4.2	6.0	
Pedro	6.6	5.6	4.9	4.1
Camila	6.6	6.2	7.0	6.3
Sofía	5.4	6.5	6.0	4.8

$$\bar{r_i} = \frac{\sum_c r_{ic}}{\sum c} \tag{4.1}$$

Luego tenemos que el promedio de cada uno de los estudiantes es,

$$\begin{aligned} \bar{r}_{Juan} &= 5.6 \\ \bar{r}_{Pedro} &= 5.7 \\ \bar{r}_{Camila} &= 6.6 \\ \bar{r}_{Sofía} &= 6.3 \end{aligned} \tag{4.2}$$

Posteriormente, se calcula la desviación de las notas con respecto al promedio, de la siguiente manera

$$r'_{ip} = r_{ip} - \bar{r}_i \tag{4.3}$$

Y se actualiza la tabla con la desviación obtenida para cada estudiante

Tabla 4.4: Tabla de ejemplo para sistema de recomendación

	CC1000	CC5002	CC4005
Juan (J)	1.1	-1.4	0.4
Pedro (P)	0.9	0.1	-0.8
Camila (C)	0.0	-0.4	0.4
Sofía (S)	-0.7	0.2	-0.3

Se define la formula de similitud de dos usuarios que han cursado los mismos ramos (tienen itemes en común), o también conocida como coeficiente de correlación de Pearson, además se puede interpretar como el coseno del ángulo entre las representaciones vectoriales de dos

estudiantes, pues $cos(\alpha)$ no es más que el coeficiente de correlación muestral de Pearson.

$$Sim(a,b) = \frac{\sum_p (r_{ap} - \bar{r}_a)(r_{ab} - \bar{r}_b)}{\sqrt{\sum (r_{ap} - \bar{r}_a)^2}\sqrt{\sum_p (r_{bp} - \bar{r}_b)^2}} \tag{4.4}$$

Con r_{up}: El puntaje del usuario u con respecto al curso p, y p: items.
Por tanto, al calcular la similitud de Juan con el resto de estudiantes obtenemos,

$$Sim(Juan, Pedro) = \frac{(1.1 * 0.9) + (-1.4 * 0.1) + (0.4 * -0.8)}{\sqrt{(1.1^2 + 1.4^2 + 0.4^2)}\sqrt{(0.9^2 + 0.1^2 + 0.8^2)}} = 0.24 \tag{4.5}$$

$$Sim(Juan, Camila) = \frac{(1.1 * 0.0) + (-1.4 * -0.4) + (0.4 * 0.4)}{\sqrt{(1.1^2 + 1.4^2 + 0.4^2)}\sqrt{(0.0^2 + 0.4^2 + 0.4^2)}} = 0.69 \tag{4.6}$$

$$Sim(Juan, Sofía) = \frac{(1.1 * -0.7) + (-1.4 * 0.2) + (0.4 * -0.3)}{\sqrt{(1.1^2 + 1.4^2 + 0.4^2)}\sqrt{(0.7^2 + 0.2^2 + 0.3^2)}} = -0.81 \tag{4.7}$$

Finalmente, para calcular la predicción del ramo no cursado por Juan se hace mediante

$$r_{up} = \bar{r}_u + \frac{\sum_{i \in estudiante} sim(u,i) * r_{ip}}{\sum_{i \in estudiante} |sim(u,i)|} \tag{4.8}$$

Por tanto, queda expresado como

$$r_{J,CC5325} = \tag{4.9}$$

$$\bar{r}_J + \frac{sim(J,P) * (r_{P,CC5325} - \bar{r}_P) + sim(J,C) * (r_{C,CC5325} - \bar{r}_C) + sim(J,S) * (r_{S,CC5325} - \bar{r}_S)}{|sim(P,CC5325)| + |sim(C,CC5325)| + |sim(S,CC5325)|} \tag{4.10}$$

Con J como Juan, C como Camila, P como pedro y S como Sofía. Por tanto la predicción de similitud del ramo CC5325 con Juan está dada por

$$r_{J,CC5325} = 5.6 + \frac{(0.24 * 1.6) + (0.69 * -0.3) + (-0.81 * 1.5)}{|0.24| + |0.69)| + |-0.81|} = 5.0 \tag{4.11}$$

De esta manera, se podría señalar que dado las notas de Juan, sus similitudes con sus compañeros y la notas de sus compañeros en el ramo CC5325, se predice que Juan obtendría una nota estimada 5.0 si es que cursara el ramo. Es claro, que Juan podría no obtener dicha nota, sin embargo, la nota puede ser considerada como un sistema de puntuación Elo, es decir, una calificación de la habilidad relativa del estudiante con respecto a sus compañeros, tal como en el ajedrez o en los videojuegos, este sistema de puntuación, permite asignar la capacidad del estudiante en un rango de notas.

4.2.1. Análisis de resultados

Considerando el ejemplo previamente expuesto, es claro identificar que el resultado obtenido es bastante general, y que por lo demás, podría resultar totalmente errado. Existen diversos factores que deben ser considerados para lograr resultados confiables en una predicción de este tipo, en primer lugar, los estudiantes son diferentes entre sí, y estos podrían obtener resultados distintos al esperado si es que cursaran el mismo ramo, un desempeño

sobresaliente o por lo contrario uno paupérrimo, entonces, ¿de qué serviría un sistema recomendador?. Para responder a la pregunta, se debe recordar el principal objetivo de este sistema, este es entregar un perfil al estudiante, es decir, otorgar una herramienta al usuario que le permita establecer referencias de aquellos ramos en los que podrían presentar más aptitudes. Bajo esta premisa, el resultado obtenido en el ejemplo podría servir como referencia, sin embargo, en la realidad existen n estudiantes, y la mayoría han cursado los mismos m ramos, esto quiere decir, que es muy probable que exista un estudiante que presente las mismas aptitudes y rendimiento que posea otro, por tanto, se podría establecer una relación mucho más clara, sin considerar problemas externos que afecten el desempeño del estudiante.

Complementando el análisis previo, al considerar como por ejemplo, un escenario en el cual pertenecen estudiantes del Departamento de computación, es claro que la mayoría de ramos cursados por los estudiantes coincidirán entre ellos, debido a que pertenecen a la misma carrera y poseen la misma malla, más aún, existirán cursos electivos que coincidirán entre subgrupos del grupo de estudiantes. En dichos subgrupos un estudiante "A" que no ha cursado "X" ramo electivo, podría estar interesado en cursar "X" debido a que comparte la mayoría de sus electivos con los compañeros del subgrupo, y dicho curso "X" ha sido cursado por ellos.

4.2.2. Diseño del pipeline

Con la elección del sistema de recomendación a partir de un filtro colaborativo basado en el usuario, hay que considerar que recursos son necesarios para obtener los resultados del ejemplo previamente expuesto. En primer lugar, se requiere el historial de ramos cursados, junto a las notas obtenidas en ellos, de cada estudiante de la facultad. Cabe destacar que, para resguardar la privacidad del estudiante, no se debe tener un identificador real, como el RUT o el nombre, por lo que se debe generar un identificador aleatorio del estudiante. Posteriormente, con el historial de notas, junto a un identificador se debe crear el dataset, para este paso se debe tener en consideración el ramo asociado por el estudiante, su nombre y código, de esta manera, un extracto del dataset estaría representado como el de la Tabla 4.5, en la cual se aprecian dos estudiantes con parte de su historial de ramos cursados y la nota obtenida en el promedio. Este dataset será construido mediante una dataframe de la librería para Python, Pandas, junto a Numpy como dependencia de la primera.

Tabla 4.5: Extracto del dataset de historial de cursos de estudiantes junto a su calificaciones.

id_persona	codigo	nombre_ramo	promedio
19585621	CC1000	Herramientas Computacionales para Ingeniería y Ciencias	6.2
19585621	CC3002	Metodologías de Diseño y Programación	6.0
19585621	CC3001	Algoritmos y Estructuras de Datos	5.8
19585621	CC5002	Desarrollo de Aplicaciones Web	6.9
2596348	CC1000	Herramientas Computacionales para Ingeniería y Ciencias	6.2
2596348	CC3001	Algoritmos y Estructuras de Datos	5.8
2596348	CC3002	Metodologías de Diseño y Programación	6.0
2596348	CC5002	Desarrollo de Aplicaciones Web	6.3
2596348	CC5325	Taller de Hacking Competitivo	6.1
3651987	CC1000	Herramientas Computacionales para Ingeniería y Ciencias	6.2
3651987	CC3001	Algoritmos y Estructuras de Datos	4.7
3651987	CC3002	Metodologías de Diseño y Programación	5.0
3651987	CC5002	Desarrollo de Aplicaciones Web	5.5

Luego, lo que sigue es crear un nuevo dataframe a partir del anterior, tal como la matriz de la Tabla 4.3, es decir, cada fila debe estar asociada a un estudiante, y cada columna a un curso, de esta manera, cada celda de una fila asociada a un usuario debe indicar la calificación que obtuvo el estudiante en el ramo asociado a la columna. Así, la matriz quedaría tal como se aprecia en la Tabla 4.6.

Tabla 4.6: Matriz de notas asociada a la Tabla 4.5

	CC1000	CC3001	CC3002	CC5002	CC5325
19585621	6.2	5.8	6.0	6.9	
2596348	6.2	5.8	6.0	6.3	6.1
3651987	6.2	4.7	5.0	5.5	

Con está último dataframe generado, los pasos a seguir son exactamente iguales a los expuestos en el ejemplo, por lo que se utilizaran funciones de la librería Pandas para obtener promedios, desviaciones de notas y poder unir dataframes. Por otro lado, para obtener la similitud del coseno, se usará la librería Scikit-learn, que posee funciones para obtener métricas, en particular aquellas que permiten computar la distancia matricial entre dos vectores.

Finalmente los resultados deberán ser expuestos mediante un API REST (Ejemplo Figura 4.3), conectado a una función que calcule la predicción de un ramo no cursado de un determinado estudiante, y así, tras realizar una petición a dicha API, obtener una nota referencial. Como la intención del sistema, es obtener un listado de cursos recomendados para el usuario,

el output de la función de predicción debe ser una lista. Posteriormente, la respuesta a la petición realizada al API, será un JSON con un listado de cursos recomendados junto a las calificaciones referenciales, dicho JSON deberá ser mostrado en una vista sencilla que permita probar al estudiante el funcionamiento del sistema. La vista debe contener un formulario cuyo input debe ser el identificador del estudiante, y luego de enviarlo, debe listar los cursos obtenidos por el API en una tabla.

Un ejemplo del flujo descrito es el del diagrama de secuencia de la Figura 4.3. Es importante destacar que por un acuerdo de confidencialidad y privacidad no se cuenta con un identificador real por estudiante, por lo que la validación de este sistema implementa una herramienta para poder identificarse en la interfaz y será estudiada posteriormente en la sección de Validación, por lo que el flujo descrito ignora la existencia de dicha herramienta. Por lo demás, el diagrama de secuencia muestra de manera general las acciones a realizar, sin considerar una validación del identificador.

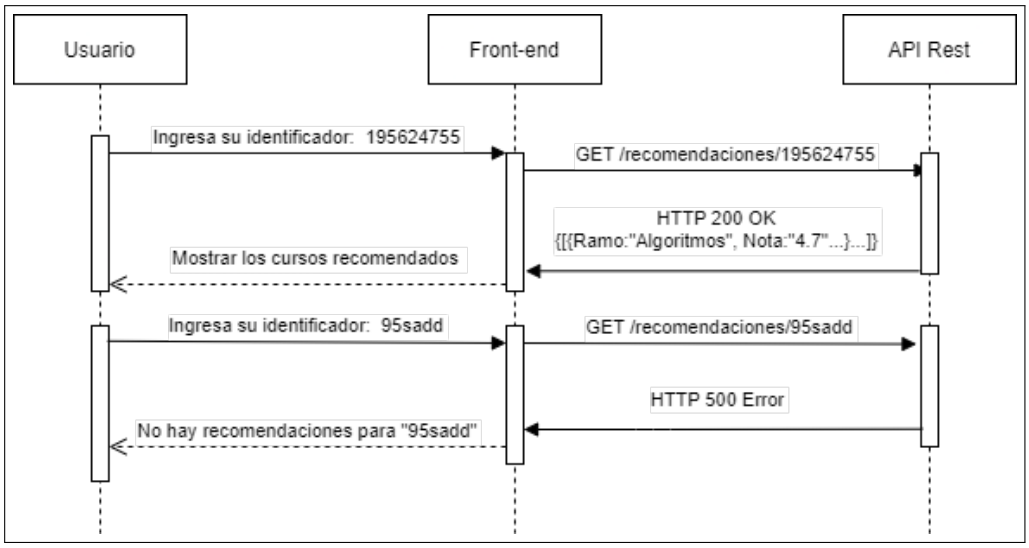

Figura 4.3: Ejemplo de diagrama de secuencia para sistema de recomendación.

4.3. Resumen

A lo largo de este capítulo se realizaron análisis para el sistema de categorización y el de recomendación. Para el sistema de categorización, primero se evaluó el escenario y los recursos dispuestos por Ucampus para cumplir los objetivos. Luego, a partir de los recursos se consideraron posibles métodos para categorizar, identificando cada una de las alternativas en el estado del arte para resolver problema de categorización. Posteriormente se compararon cada uno de los métodos con el propósito de hallar el que mejor resuelve el problema. Tras hallar la mejor alternativa, se diseñó el pipeline para resolver el problema, describiendo como procesar los datos y posteriormente usarlos.

Para el sistema de recomendación, de manera homologa al de categorización, primero se evaluaron los recursos dispuestos por Ucampus para cumplir los objetivos. Posteriormente, se explica mediante un ejemplo, el uso de un filtro colaborativo basado en usuarios que

permite resolver el problema. Tras obtener los resultados en el ejemplo, se realiza un análisis considerando los alcances del sistema y los objetivos propuestos. Luego, se diseña un pipeline para resolver el problema, describiendo como procesar los datos y posteriormente usarlos.

Capítulo 5

Implementación

5.1. Sistema de categorización

Para implementar el Sistema de categorización, primero se debe describir el pipeline en el cual se desarrolla, este comienza con una construcción del dataframe a partir de una recolección de datos, le sigue un preprocesamiento, posteriormente el entrenamiento del modelo y finalmente la disposición del modelo en el API REST. Se construirá usando Python como lenguaje de programación principal y la librería de Pandas, para la construcción de un dataframe.

Tabla 5.1: Primeras 6 filas del conjunto de datos: Listado de requisitos.

id_ramo	codigo	nombre	sct	programa_name	id_periodo
6154	CC1000	Herramientas Computacionales para Ingeniería y Ciencias	3	2020_1_CC1000.pdf	20201

Tabla 5.2: Continuación de tabla 5.1.

programa_url
https://Ucampus.uchile.cl/b/programas_cursos/bajar?id=48705

Para la construcción del dataframe, lo primero es recolectar los datos, para esto Ucampus dispone de un API a la cual se le es posible consultar todos los cursos pertenecientes a la facultad, como también cada uno de sus datos descriptivos, entre ellos, su nombre, código, requisitos o créditos, y programa como documento, sin embargo, no son suficiente como para preparar un *dataset* robusto, por lo que la alternativa directa para obtener información de un curso es extraer la información contenida en su programa. Dado a que la idea es obtener el texto de cada programa para almacenarlo en el dataframe, entonces el primer paso es añadir cada uno de los programas con sus datos descriptivos obtenidos en la API, tal como se ve en la tabla 5.1 y su continuación 5.2. Lo siguiente es recolectar cada uno de los documentos asociados a la url de programa_url de manera local.

Con los programas almacenados de forma local, y el dataframe construido, lo que sigue es obtener la información de cada uno de los documentos, para esto se utiliza la librería fitz de Python la cual permite extraer (Ejemplo de extracción en Figura 5.1) el texto plano de archivos *pdf*. Con el texto extraído se pueden apreciar caracteres sueltos y palabras monosilábicas, además se observan preposiciones, artículos y pronombres que no enriquecen la información recolectada, sin embargo, unos de los beneficios de usar el algoritmo de Top2Vec posteriormente, es que no hace falta utilizar técnicas de *stemming*, o *stop-words* para pre-procesar los datos, por lo que se realiza un preprocesamiento (Ejemplo de preprocesamiento en Figura 5.2) del texto pero sólo eliminando expresiones que no tienen sentido semántico, para esto se utiliza la librería *sPacy* que cuenta con herramientas de *tokenizado* y *Lematization*. Adicionalmente, existen palabras recurrentes en los programas de los cursos, entre ellas, "controles", "evaluaciones", "contenidos", "auxiliares", "ayudantes", etc. Estás palabras podrían inducir categorizaciones incorrectas al relacionar todos los cursos entre sí, por lo que se crea una lista de palabras en la que se restringen aproximadamente 150, aquellas que son propias de un programa de curso, identificadas previamente de manera manual.

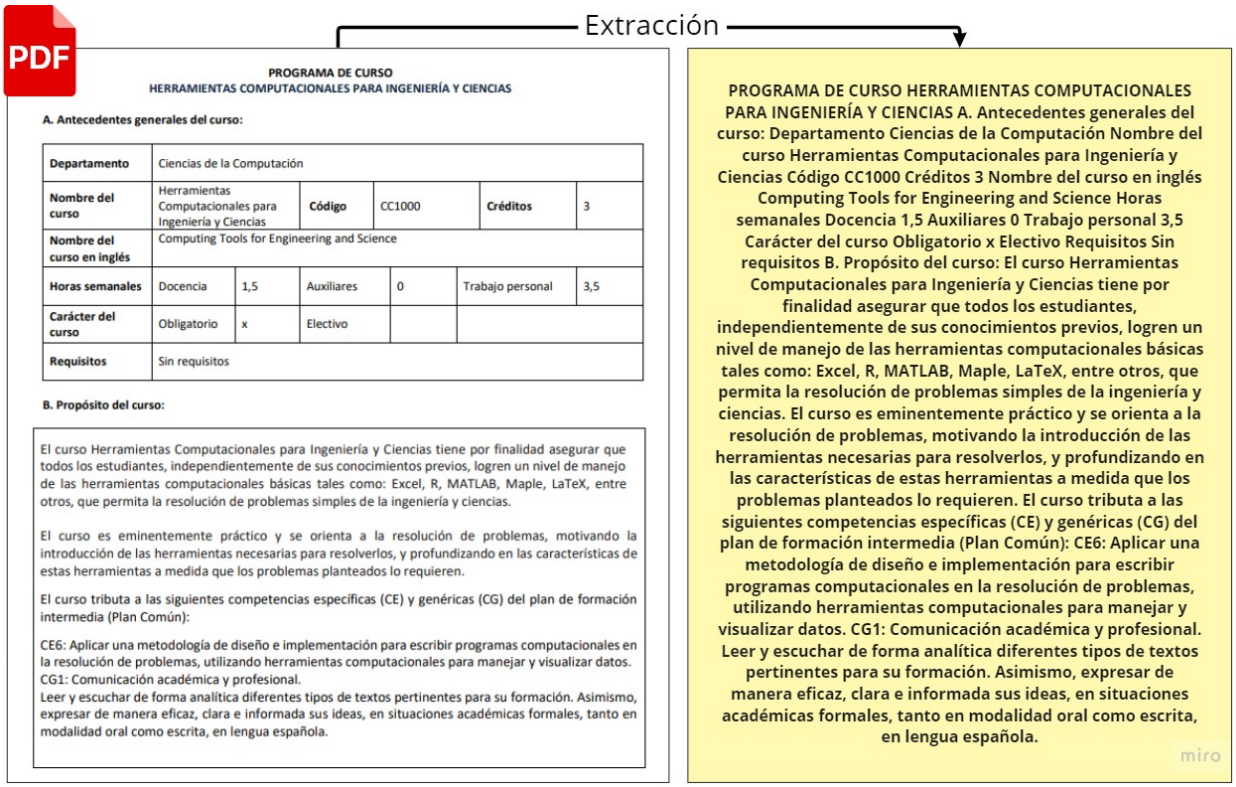

Figura 5.1: Ejemplo extracción de texto plano desde un documento pdf.

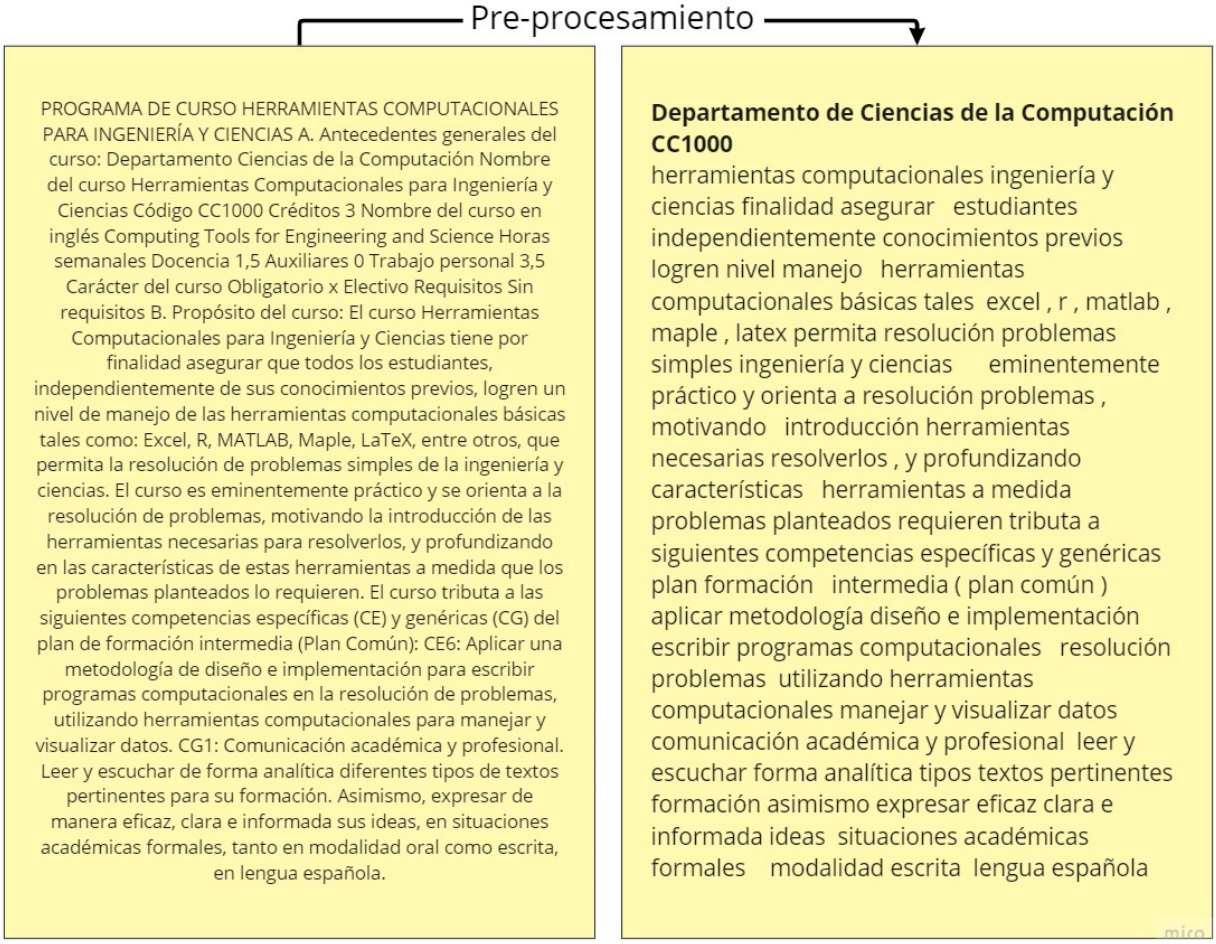

Figura 5.2: Ejemplo preprocesamiento de texto plano extraído de un documento pdf.

Con el texto plano de un programa extraído, y preprocesado, lo siguiente es agregarlo como un nuevo campo en una columna "desc" del curso asociado en el dataframe. Realizando este procedimiento para cada uno de los cursos se genera un dataset con toda la información descriptiva que será posible ser consultado posteriormente en un API.

5.1.1. Selección entre entrenar un modelo de embedding o usar uno preentrenado

El algoritmo de Top2Vec permite utilizar modelos de *embedding* preentrenados, o en su defecto, Doc2Vec se utilizará de forma predeterminada para generar modelos de embedding. Considerando la primera opción, y teniendo en cuenta de que los programas están en "español", surgen dos candidatos para modelos preentrenados, "Multilingual BERT" y "BETO", siendo la segunda alternativa, la opción escogida, esto debido a sus mejores resultados en tareas de clasificación como "MLDoc" por sobre "Multilingual BERT".

Por otro lado, tal como se menciona en el análisis, el rendimiento del modelo de embedding generado por algoritmo Doc2Vec podría ser mejor al considerar un escenario más técnico y especifico de datos. Al comparar los tópicos obtenidos por el algoritmo de Top2Vec usando los

Tabla 5.3: Nueva columna desc que muestra el texto extraído de un programa.

desc
Departamento de Ciencias de la Computación CC1000 Herramientas Computacionales para Ingeniería y Ciencias El curso Herramientas Computacionales para Ingeniería y Ciencias tiene por finalidad asegurar que todos los estudiantes, independientemente de sus conocimientos previos, logren un nivel de manejo de las herramientas computacionales básicas tales como: Excel, R, MATLAB, Maple, LaTeX, entre otros, que permita la resolución de problemas simples de la ingeniería y ciencias... Utiliza planillas de cálculo para el procesamiento, análisis y visualización de conjuntos de datos, considerando el uso de funciones de cálculo matemático, estadístico y de manejo de bases de datos... Utiliza MATLAB para calcular y analizar gráficos de funciones, correlacionando datos mediante gráficos de dispersión. Maneja vectores y matrices, carga archivos de datos, realiza cálculos numéricos y visualiza los resultados usando diversas técnicas, incluyendo gráficos y mapas de colores....

dos modelos de embedding, uno generado por Doc2Vec, y por otro lado el modelo "BETO", se observó a partir de un conjunto de datos con cursos exclusivos del Departamento de Computación, los tópicos obtenidos por Top2Vec usando un modelo generado por Doc2Vec tienen mayor representatividad y concordancia con aquellos que obtuvieron por el mismo algoritmo, pero usando BETO como modelo de embedding.

5.1.2. Generando modelo de Top2Vec

El algoritmo de Top2Vec entrena un modelo a partir de un conjunto de datos ingresados como parámetro, este es conocido como "Input corpus", y corresponde a una lista de "strings" . Cada uno de los "strings" en la lista, corresponde a los campos asociados a la columna "desc" agregada previamente. Adicionalmente, el algoritmo permite ingresar otros parámetros opcionales, entre ellos se han usado dos, el primero "speed", que permite variar la rapidez con la que se entrena el modelo. Mientras más rápido se entrena, menor será la calidad de los vectores generados en el modelo, debido a esto, se ha decidido por ingresar la menor rapidez "deep-learn", para obtener un modelo con mayor calidad. El segundo parámetro, "embedding_model", permite añadir un modelo preentrenado de *embedding*, y ha sido utilizado para probar la comparación descrita previamente.

Luego, el modelo entrenado cuenta con un método para guardarlo de manera local, y así usarlo posteriormente para ser leído con otros métodos de la misma librería de Top2Vec,

permitiendo hacer búsquedas por tópicos, obtener la cantidad de tópicos identificados, generar "Word Clouds" y hacer búsqueda de documentos por tópico o por palabras claves. De esta manera, es posible exponer estos métodos mediante un API REST.

5.2. Sistema de recomendación

La implementación del Sistema de recomendación debe considerar en primera instancia la recolección y preparación de los datos suministrados por Ucampus, para que posteriormente sean procesados como se ha descrito en el análisis.

5.2.1. Preparando el dataset

Ucampus a puesto a disposición un documento mediante en el cual se puede obtener el historial de notas de un grupo de estudiantes. Cada uno de ellos es identificado por un número aleatorio generado por Ucampus, exclusivamente para el presente trabajo. La tabla tiene como columnas, el código del curso, el identificador del estudiante y la nota del promedio obtenida y cuenta con 378672 registros de notas asociados a 20805 estudiantes distintos y que cursaron algunos de los 2431 cursos existentes, en la Tabla 5.4 se aprecia un extracto de la información suministrada. Por otro lado, se cuenta con el API REST de Ucampus, que permite obtener los datos de un curso (Tabla 5.1), esto permitirá unir la tabla con la información del curso a partir del código, cuando sea necesario al construir el API REST.

Tabla 5.4: Extracto de tabla suministrada por Ucampus del historial de cursos de estudiantes junto a su calificaciones.

RAMO	PERSONA	NOTA
IN7T2	179672143	6.9
CC73I	21033647	5.3
CC73I	20239664	6.5
IQ4305	587662599	4.6
CC73I	20053388	5.9
EH2201	47070522	6.1
CC63A	9721626	6.7

Posteriormente, a partir de la tabla suministrada, se necesita obtener los promedios de todos los cursos rendidos por cada estudiante, como también la desviación mencionada previamente. Para esto, se añaden dos columnas extra en el dataframe generado, la primera estará asociado al promedio (avg) de los cursos y la segunda a la desviación (adg) del curso asociado a la fila con respecto a su promedio. Luego, se obtiene el promedio de las calificaciones de un estudiante y se agrega a la nueva columna, de la misma, manera se hace con la desviación, tras obtener el promedio. Así, un extracto del nuevo dataframe debe ser como el de la Tabla 5.5, con las nuevas columnas de promedio (avg) y desviación (adg) para cada fila, y concluye la creación del dataframe con los recursos necesarios para construir el pipeline.

Tabla 5.5: Extracto del dataframe construido, en el cual se identifica el promedio de notas de los cursos de un estudiante y una desviación de la nota con respecto al promedio.

RAMO	PERSONA	NOTA	avg	adg
CC4101	21895882	5.2	5.5	-0.3
CC4002	21895882	5.1	5.5	-0.4
CC5404	21895882	6.7	5.5	1.2
CC4302	21895882	4.8	5.5	0.7

Finalmente para completar el dataframe, se deben agregar dos nuevas columnas, una con el nombre del curso y otra con el departamento al cual pertenece el ramo, pese a ser necesario para el API REST, agregar ambas columnas entrega una referencia y permite identificar mejor el ramo al estar generando nuevas tablas e ir validando resultados durante el desarrollo, de esta manera, el dataset queda como el de la Tabla 5.6

Tabla 5.6: Extracto del dataframe construido, en el cual se agregan dos nuevas columnas, una asociada al nombre del curso y otra al departamento asociado al ramo.

RAMO	PERSONA	NOMBRE_RAMO	DEPARTAMENTO	NOTA	avg	adg
CC4101	21895882	Lenguajes de Programación	Departamento de Ciencias de la Computación	5.2	5.5	-0.3
CC4002	21895882	Taller de Programación B	Departamento de Ciencias de la Computación	5.1	5.5	-0.4
CC5404	21895882	Taller de UML	Departamento de Ciencias de la Computación	6.7	5.5	1.2
CC4302	21895882	Sistemas Operativos	Departamento de Ciencias de la Computación	4.8	5.5	0.7

5.2.2. Estableciendo pipeline

Con el dataset preparado, lo que sigue es generar la matriz de notas, tal como la de la Tabla 4.6, para esto basta con pivotear la tabla usando como columnas los ramos, como filas los estudiantes con su identificador y valores en común, la diferencia de nota del estudiante en el respectivo curso con respecto a su promedio de notas en todos los ramos. Para hacer este procedimiento, Pandas cuenta con el método pivot_table, entregando como parámetros, valores, índices (filas) y columnas, obteniendo la matriz de la Figura 5.3.

RAMO	AA0050	AS2001	AS3000	AS3001	AS3003	AS3004	AS3010	AS3101	AS3103	AS3201	AS4101	AS4107
PERSONA												
1001458	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1001826	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1003159	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1004299	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1005309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figura 5.3: Extracto de matriz de notas obtenida tras pivotear el dataset usando como columnas los ramos, como filas los estudiantes con su identificador y valores en común las notas del estudiante en el respectivo curso.

Al pivotar la matriz existirán campos vacíos descritos como NaN en la Figura 5.3, estos campos se deben a que no todos los estudiantes han cursado todos los ramos existentes, por tanto, ambas matrices presentan una gran dispersión en los datos, lo que podría tener como consecuencia una incongruencia considerable en los resultados al obtener la similitud entre dos estudiantes que tengan cursados pocos o ningún ramo en común. Existen varias alternativas para reducir la dispersión de las matrices, una manera efectiva, pero poco escalable es usar una factorización matricial, sin embargo, al existir alrededor de 20000 estudiantes y 3200 ramos, y considerando el crecimiento anual de ambas cantidades, operar la matriz no sería una opción viable a futuro, por lo que quedan otras dos opciones, la primera es completar las celdas vacías con el promedio de las notas de toda la fila, es decir con el promedio de notas del estudiante, y una segunda opción sería completar las celdas con el promedio de notas de las columnas, es decir, el promedio de notas de todos los estudiantes obtenido en un curso. Ambas opciones parecen ser viables, sin embargo, existen diversos aspectos a considerar. En la primera opción el estudiante podría haber obtenido un promedio sobresaliente, pero la celda que se desea rellenar está asociada al curso con mayor dificultad de la carrera. Por otro lado, considerando la segunda opción, es posible que exista un curso cuyo promedio sea bajo, pero que el estudiante al cursarlo obtenga una nota superior al promedio asociado a dicho ramo con respecto al resto de estudiantes. Luego, es evidente que existen múltiples casos de borde para ambas opciones, pero hay un factor en común en todos ellos, este es la dificultad, y la métrica más representativa asociada es justamente el promedio de todos los estudiantes que han cursado el ramo.

Posteriormente, al considerar el promedio de todos los estudiantes que han cursado el ramo, para rellenar aquellas celdas vacías, es entonces que se obtiene una matriz como la de la Figura 5.4.

RAMO	AA0050	AS2001	AS3000	AS3001	AS3003	AS3004	AS3010	AS3101	AS3103	AS3201
PERSONA										
1001458	-2.960595e-16	1.617645e-16	9.868649e-17	-4.844610e-16	-1.803069e-16	2.297013e-17	1.572816e-16	5.207561e-17	3.289550e-17	-3.289550e-16
1001826	-2.960595e-16	1.617645e-16	9.868649e-17	-4.844610e-16	-1.803069e-16	2.297013e-17	1.572816e-16	5.207561e-17	3.289550e-17	-3.289550e-16
1003159	-2.960595e-16	1.617645e-16	9.868649e-17	-4.844610e-16	-1.803069e-16	2.297013e-17	1.572816e-16	5.207561e-17	3.289550e-17	-3.289550e-16
1004299	-2.960595e-16	1.617645e-16	9.868649e-17	-4.844610e-16	-1.803069e-16	2.297013e-17	1.572816e-16	5.207561e-17	3.289550e-17	-3.289550e-16
1005309	-2.960595e-16	1.617645e-16	9.868649e-17	-4.844610e-16	-1.803069e-16	2.297013e-17	1.572816e-16	5.207561e-17	3.289550e-17	-3.289550e-16

Figura 5.4: Extracto de matriz de notas obtenida tras rellenar celdas vacías con el promedio de todos los estudiantes que han cursado el ramo.

Luego, se debe obtener la similitud entre dos estudiantes, para esto basta con sacar la correlación de Pearson entre dos representaciones vectoriales de estudiantes. Scikit Learn cuenta con el método "cosine_similarity" que genera una matriz de estudiantes por estudiantes, y asigna el valor o puntaje de correlación entre ambos en la coordenada Estudiante_x, Estudiante_y.

PERSONA	1001458	1001826	1003159	1004299	1005309	1009605	1011181	1014065	1015449	1017028
PERSONA										
1001458	0.000000	0.968186	-0.964331	0.986591	0.454632	0.987832	0.997106	0.996688	-0.915244	0.996012
1001826	0.968186	0.000000	-0.939078	0.960755	0.442727	0.961964	0.970995	0.970588	-0.890650	0.969929
1003159	-0.964331	-0.939078	0.000000	-0.956930	-0.440964	-0.958134	-0.967130	-0.966724	0.887729	-0.966068
1004299	0.986591	0.960755	-0.956930	0.000000	0.451143	0.980250	0.989454	0.989039	-0.908220	0.988368
1005309	0.454632	0.442727	-0.440964	0.451143	0.000000	0.451711	0.455952	0.455761	-0.418518	0.455451

Figura 5.5: Extracto de matriz de similitud entre estudiantes, en la cual cada celda corresponde al puntaje de correlación entre ambos en la coordenada Estudiante_x, Estudiante_y. .

Tras obtener la similitud entre dos estudiantes, basta con listar los ramos cursados por cada uno y comprobar la similitud entre los cursos. Sin embargo, calcular la similitud entre todos los estudiantes podría significar una gran dificultad a medida que se añaden nuevos estudiantes al dataset. Por otro lado, el objetivo del sistema es encontrar perfiles similares de estudiantes con tal de recomendar cursos, es decir, si es que se quisiera buscar recomendaciones para un estudiante en particular, habría que compararlo con todo el resto de estudiantes, esto podría presentar inconvenientes puesto que la matriz generada, es de 20805 x 20805 celdas, y está creada a partir de un grupo menor al total de estudiantes inscritos de la facultad. Luego, comparar un estudiante con todo el resto, permitiría obtener resultados muchos más precisos, pero el calculo de la similitud podría tardar un tiempo excesivo, teniendo en cuenta de que son mucho más de 20805 estudiantes en el registro. Además, el objetivo es obtener la similitud del estudiante con otros, casi en tiempo real, al tratarse de una petición desde la plataforma o interfaz al API REST, por lo que obtener la respuesta podría tardar bastante

tiempo si es que no se optimiza el computo.

Al analizar el problema como uno de "big data", la optimización más común es usar vecindarios, es decir, encontrar subgrupos dentro de todos los estudiantes que posean un perfil parecido, de esta manera, podemos reducir significativamente la cantidad de datos para comparar. Luego, para hallar los vecindarios debemos ordenar la matriz obtenida a partir de los estudiantes cuyos puntajes sean similares, y posteriormente generar un dataframe con cada vecindario y los estudiantes que los componen, tal como se aprecia en la Figura 5.6.

PERSONA	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10
1001458	23979088	5594436	39019171	7997075	5992634	32074199	28746614	43013972	37172580	13370595
1001826	18577503	12788661	14670691	9395990	15683723	8083869	7458685	3868084	3415248	1452468
1003159	13892267	24369598	1507635	61445283	22283855	15755216	40700625	8320886	13463305	20697366
1004299	14885018	11713220	58191963	7877904	21891080	8092375	5425118	8268222	15352888	15886993
1005309	31533235	356035750	32023225	57938925	12948328	14766997	290807312	534293853	2853570	3752288

Figura 5.6: Extracto de dataframe de vecindarios, cuyas filas representan cada vecindario, y columnas el top n estudiante más similar al primer estudiante de la fila.

Lo que resta es obtener el listado de cursos sugeridos de acuerdo a la similitud que existe entre estudiantes, para eso, lo primero es generar una función cuyo propósito es identificar al estudiante **U** en el dataframe de los vecindarios. Luego, con el vecindario identificado, lo que sigue es identificar el listado de ramos cursados por el usuario **U**, y todos los ramos cursados por los estudiantes pertenecientes al vecindario, de esta manera, a partir de ambos listados, se puede construir un conjunto de ramos que no han sido cursados por el estudiante **U**, pero sí por estudiantes con perfiles similares, concatenando ambos listados y excluyendo los del estudiante **U**. Finalmente, se debe calcular la similitud del usuario **U** con los ramos del conjunto, para esto, tal como se muestra en el ejemplo, por cada ramo del conjunto se deben obtener las notas y desviaciones obtenidas por cada estudiante que cursó el ramo, y así junto a las notas del usuario **U** obtener la similitud del ramo, obteniendo una nota de referencia.

5.3. Integración de sistemas

Con ambos sistemas construidos, se debe crear una herramienta que integre y exponga los modelos generados, para esto se debe montar un servidor en el cual se desarrollará un back-end cuyo propósito es suministrar un API REST en el cual serán consultados los datos desde una interfaz Web o cliente.

5.3.1. Montando servidor y ambiente de desarrollo

Para montar el servidor *back-end* se ha suministrado una instancia de servidor remoto creado en *digitalocean* por parte de Ucampus, el cual cuenta con los requisitos de hardware necesarios para generar un modelo, y posteriormente disponibilizarlo mediante el API REST.

Además, debido a la cantidad reducida de interacciones y funcionalidades que podría tener el *back-end* es que un *framework* minimalista como lo es *Flask* resulta ser ideal para cumplir con los objetivos planteados.

Posteriormente, lo primero que se ha realizado es crear un entorno virtual en Python, con todas las dependencias necesarias para montar el back-end, siendo la principal *Flask*, *Flask-RESTful*, *Pandas*, *Numpy*, *Scikit-learn* y *Top2Vec*.

5.3.2. Preparando API REST (Lectura de modelos)

Para construir el API REST (Figura 5.7) se utiliza la extensión para *Flask*, *Flask-RESTful*. Esta extensión permite una abstracción ligera para el sistema ORM[5] de *Flask*, a su vez permite exponer un *end-point* a cada una de las funciones que permiten leer los modelos de los sistemas. De esta manera, al generar un end-point por función, estos pueden ser consultados mediante peticiones HTTP.

Para construir el API REST, por cada end-point que se quiera exponer se debe crear una clase, cuyo método será de tipo GET. Para el sistema de Categorización se han creado dos funciones, la primera para realizar una búsqueda por tópico y la segunda para realizar una búsqueda por palabras claves. Para cada función se debe crear una clase, cuyo método GET, debe recibir como parámetro, el tópico o la palabra clave, según sea el caso, y retornar un objeto JSON que es generado a partir de consultar los inputs en el modelo generado previamente por el sistema. Al consultar el end-point asociado la clase, se deben incluir como parámetros el input o la palabra clave, junto con un parámetro adicional que consiste en la cantidad de resultados. Al ser ingresados mediante la URL, la forma que debe tener el end-point asociado a la clase de búsqueda por tópico sería de la manera,

$$http://localhost:8080/sbt/<arg\_topic\_num>/<arg\_num\_res> \tag{5.1}$$

en la cual, el primer parámetro tras el dominio y el puerto, hace referencia a la clase asociada, en este caso la búsqueda por tópico (*Search by topic*) como "sbt", el segundo "arg_topic_num" corresponde al tópico a buscar y el tercero "arg_num_res" la cantidad de resultados a exponer. Para la otra clase, el end-point es idéntico salvo que posee como primer parámetro la búsqueda por palabra clave (*Search by keyword*) como "sbk" y como segundo parámetro, el tópico a buscar, "arg_keyword_num".

Por otro lado, para el sistema de recomendación el procedimiento para generar el end-point es el mismo, pero la función asociada corresponde a la búsqueda de cursos recomendados para un determinado estudiante, es decir, el end-point sería de la forma,

$$http://localhost:8080/sbs/<arg\_student> \tag{5.2}$$

siendo el primer parámetro, de especifique la búsqueda por estudiante (*Search by student*) como "sbs" y el segundo "arg_student" el estudiante. Es importante destacar, que los méto-

[5] El mapeo objeto-relacional (más conocido por su nombre en inglés, *Object-Relational mapping*, o sus siglas ORM) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

dos GET de las clases realizan las consultas y procedimientos pertinentes a los modelos ya generados para exponer los resultados.

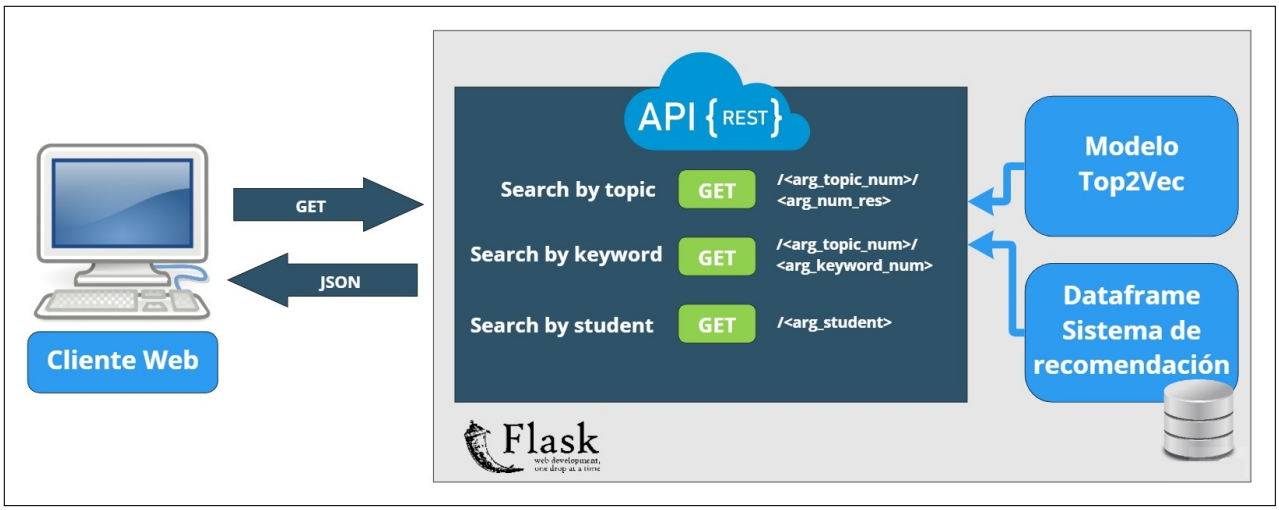

Figura 5.7: Arquitectura lógica del sistema API REST.

5.3.3. Montando interfaz básica para mostrar datos

La interfaz esta encargada de exponer de manera visual la información dispuesta por los modelos de los sistemas. Además, con propósitos de ilustrar sólo los resultados de los sistemas es que la distribución de la información es simple, de manera de ejemplificar casos de uso específicos para los sistemas. Esta utiliza HTML como lenguaje de marcado, CSS para la estructura y diseño del sitio, y JavaScript como lenguaje de programación, destacando el uso de la librería "jQuery" como principal recurso para la manipulación del "*Document Object Model*"[6] (DOM) y las funciones encargadas de realizar las peticiones al servicio API REST. Otras librerías a destacar son: "Bootstrap" para el diseño de una interfaz responsiva, uso de menús de superposición modales, botones, formularios, y estructuras del DOM prediseñadas, "DataTables.js", para la creación de tablas dinámicas para mostrar resultados y "Cytoscape.js" para la creación de grafos que ilustran la relación entre los cursos pertenecientes a un tópico en el sistema de categorización.

5.3.4. Flujo de credenciales (CORS)

Una de las principales dificultades al momento de establecer la comunicación con el API REST, es el intercambio de recursos de origen cruzado (CORS, siglas en inglés), esto se debe a que la mayoría de los navegadores requiere de páginas web con certificados SSL por la seguridad de la capa de transporte, junto con demostrar que la página web es segura. Por tanto, para poder realizar las pruebas con usuarios es necesario asegurar la navegación del mismo.

Para generar un end-point con certificados SSL, existen varias opciones, una es gestionar un dominio y comprar un certificado incluido, otra opción, orientada más al "testing" es ocupar un dominio temporal. Un dominio temporal como, por ejemplo, "Localtunnel", expone el servidor local al mundo, para poder compartir la URL con otras personas, asegurando su

[6] Document Object Model o DOM, es la estructura del documento HTML.

navegación por el sitio. “Localtunnel”, no necesita una configuración adicional de DNS o un proceso de *deploy*, basta con correr un servidor en “Node.js” para exponer el servidor local a un dominio público y certificado con SSL.

5.4. Resumen

En este capítulo se abordó la implementación de los sistemas de Categorización y Recomendación. Para el primer sistema, se describe como se establece el pipeline diseñado en el capítulo anterior, comenzando con la construcción del dataframe. Posteriormente se analiza la selección de un modelo generado por Doc2Vec por sobre uno preentrenado, para luego explicar cómo se ha generado el modelo de Top2Vec.

Luego, se explica la implementación del sistema de recomendación, comenzando por la preparación del dataset. Después se describe cada uno de los pasos del pipeline para construir los dataframes necesarios para consultar las recomendaciones.

Finalmente, se explica cómo se integran ambos sistemas, comenzando por la creación del servidor y el ambiente de desarrollo. Luego se describe la construcción del servicio API REST para leer el modelo generado para el sistema de Categorización, y los dataframes generados para el sistema de Recomendación. Posteriormente, se describe con qué recursos se ha construido la interfaz Web, para luego explicar cómo se han habilitados los certificados SSL y describir el flujo de credenciales para exponer la Web de manera pública.

Capítulo 6

Validación

Con el sistema de categorización y recomendación funcionando lo que resta es evaluar los resultados que arrojan cada uno. Para esto se debe tener en consideración que al abordar ambos problemas como uno de machine learning no supervisado, no se cuenta inicialmente con un dataset de entrenamiento que permite evaluar las categorizaciones y recomendaciones respectivamente. Crearlo significaría, cambiar el paradigma de *Machine learning* inicial por el cual se decidió proceder. Por lo demás, se debe contar con los conocimientos suficientes para poder crear categorizaciones congruentes, como también en el caso del sistema de recomendación se debe tener un dataset con las recomendaciones preestablecidas para un grupo de estudiantes, es decir, cada uno de ellos debe etiquetar aquellos ramos que podrían resultar ser una recomendación a partir de sus ramos cursados y notas, inclusive, debe tener noción de compañeros con perfiles similares para realizar el etiquetado, esto de acuerdo a como está basado el algoritmo a partir de un filtro colaborativo.

Por tanto, etiquetar para validar resulta ser una tarea compleja y exhaustiva, en el que el principal beneficio es utilizar métricas para evaluar la validez de los datos, su precisión, porcentaje de error, entre otras métricas. Sin embargo, en lugar de etiquetar y crear conjuntos de datos de entrenamiento, se evaluarán los resultados directamente con los potenciales usuarios, logrando obtener métricas mucho más representativas y flexibles debido a que son totalmente customizables.

Una validación directa con los estudiantes significa que estos deben evaluar los sistemas ingresando una entrada, esta deber ser procesada y generar una salida o resultado cuyo contenido debe ser evaluado por el usuario que está ocupando el sistema. La evaluación debe estar sujeta a criterios de validez, congruencias y utilidad de los datos obtenidos, por lo que se debe diseñar preguntas certeras para evaluar cada uno de ellos. Luego, uno de los objetivos deseados para los sistemas es que ambos estén dispuestos en la plataforma de Ucampus, por lo que deberá existir una interfaz que se comuniqué con los algoritmos que estarán corriendo en alguna instancia remota. Por otro lado, si se desean evaluar los sistemas, disponer una interfaz (como una página Web en la que puedan ingresar información) y obtener resultados, permitiría validar de manera asíncrona con los usuarios sin la necesidad de estar probando el algoritmo por cada estudiante que deseé validarlos. Además, permite emular la interacción que tendrá el usuario con las futuras vistas dispuestas en la plataforma de Ucampus de ambos sistemas.

6.1. Herramientas de validación

La validación de ambos sistemas ha sido realizada utilizando como herramienta una página web de tipo SPA (Single-page Application) como medio de prueba de concepto, por tanto, en ella se ha presentado una única vista con dos secciones, en la primera se presenta el sistema de categorización y en la segunda el de recomendación. Esta interfaz ha sido dispuesta con el fin de probar y medir estadísticamente la efectividad de los sistemas, es decir, validar que los objetivos propuestos han sido cumplidos, constatándolos con potenciales usuarios de la facultad, por lo que el diseño de la interfaz no conlleva consigo una evaluación de la experiencia de usuario. Por otro lado, el uso de la interfaz ha sido guiado mediante indicaciones textuales dispuestas en una ventana de superposición modal (Figura 6.1), que aparece al ingresar a la Web ó al abrirlo utilizando un botón de ayuda, además de las instrucciones, se ha creado un vídeo tutorial explicando brevemente cada sistema y como utilizarlos.

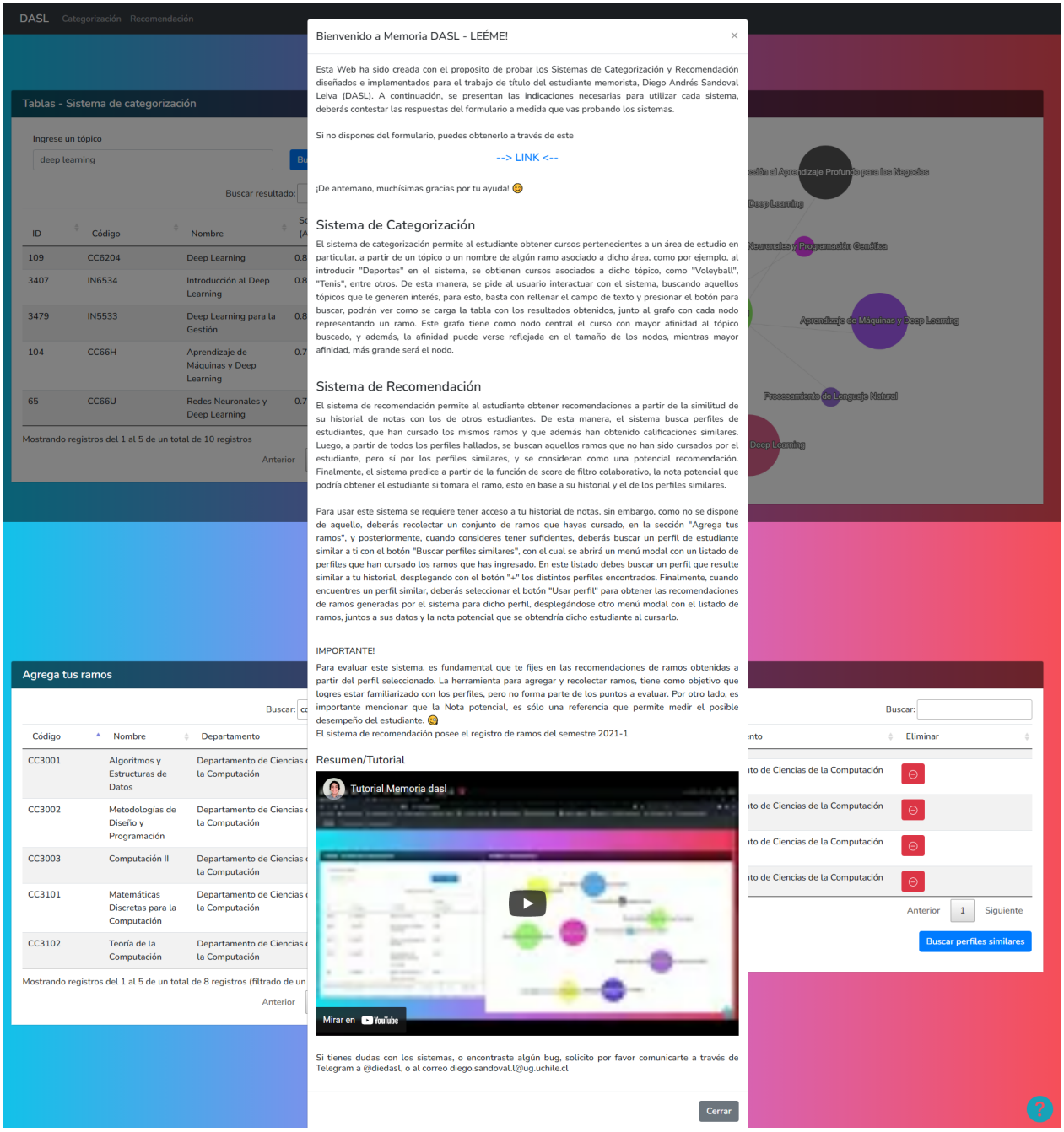

Figura 6.1: Ventana de superposición modal con la descripción de los sistemas, las instrucciones de uso y el video tutorial.

Para constatar y registrar las apreciaciones de los usuarios para ambos sistemas, se ha dispuesto un formulario a estudiantes de distintos departamentos de la facultad, en su mayoría estudiantes de últimos años de sus respectivas carreras, esto debido a que poseen mayor experiencia inscribiendo ramos y están más relacionados con las áreas de estudio y cursos de su carrera. Este formulario tiene como objetivo recolectar las apreciaciones de los estudiantes al usar ambos sistemas, preguntando principalmente sobre la confiabilidad de los resultados obtenidos, y de esta manera, constatar con el grupo de estudiantes que los objetivos propuestos inicialmente han sido cumplidos y así sacar conclusiones del trabajo realizado.

6.1.1. Sistema de categorización

La herramienta de validación para el sistema de categorización se ha dispuesto mediante interfaz de la Figura 6.2, en la parte izquierda se presenta un campo de texto en el cual se debe ingresar el tópico o campo de estudio al cual se desea obtener los cursos pertenecientes. De esta manera, el estudiante ingresa el tópico y luego debe presionar el botón de "Buscar" para que los resultados sean expuestos en la tabla inferior, cuyas columnas, ID, Código y Nombre permiten identificar los cursos de cada una de las filas. Por otro lado, la columna de "Score" o puntaje de afinidad entrega un puntaje entre 0 y 1, y muestra la afinidad que posee el curso con el tópico buscado. Es decir, mientras más cercano es el puntaje de un curso a 1 más representativo es del tópico, y en el caso contrario, mientras más cercano es a 0, tendrá menor representatividad del tópico sin embargo sigue perteneciendo a él. Por lo demás, la tabla cuenta con herramientas de búsqueda, ordenamiento y paginación para optimizar la observación de los resultados.

En la parte derecha de esta sección, el sistema cuenta con una representación mediante un grafo conexo y sin ciclos, o bien, un árbol de un sólo nivel, cuyo nodo padre es el curso con mayor puntaje de afinidad con respecto al tópico buscado, y los nodos adyacentes al padre constituyen el resto de cursos del tópico. El tamaño de cada nodo está dado por el puntaje de afinidad, por tanto, mientras mayor es el diámetro del nodo, más representativo es del tópico. La finalidad principal de esta representación es que el estudiante pueda navegar en el grafo para tener una perspectiva gráfica y más evidente de los resultados. En la Figura 6.2 se muestra un ejemplo de búsqueda para el tópico "deep learning", y los resultados son expuestos en la tabla y en el gráfico.

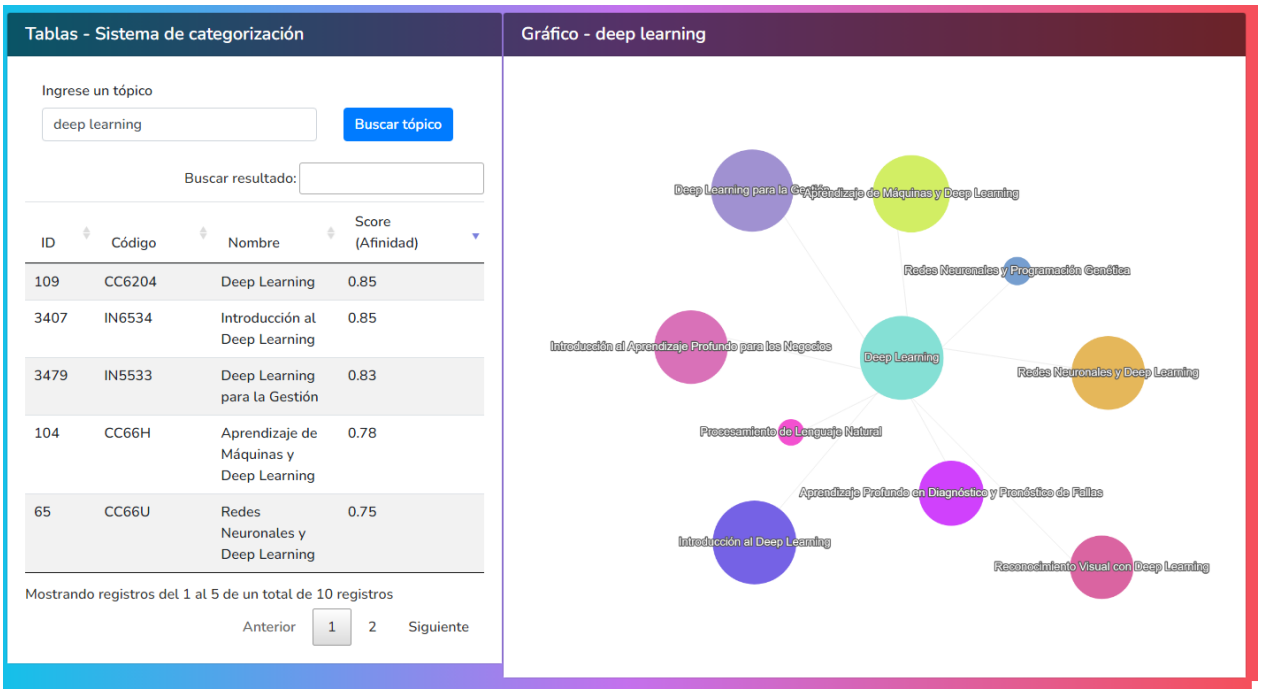

Figura 6.2: Herramienta de Sistema de Categorización, con la búsqueda del tópico "deep learning".

6.1.2. Sistema de recomendación

La herramienta de validación para el sistema de recomendación se ha dispuesto mediante interfaz de la Figura 6.3. Al no contar con acceso al historial de ramos de cada uno de los estudiantes por cuestión de privacidad, ni con un método de autenticación de Ucampus para utilizar las credenciales del estudiante y posteriormente obtener sus datos, se ha optado por crear una herramienta que permita a cada usuario de la validación construir su historial de cursos, con el fin, de que al recolectar una cantidad suficiente de ramos, el usuario logré hallar su propio historial, esto es posible, debido a que se cuenta con un registro de todos los estudiantes de la facultad junto a sus historiales, pero con un identificador cifrado por cada uno, para proteger su privacidad, de esta manera, el usuario al construir su historial podrá eventualmente hallar el suyo.

Para llevar lo anterior a cabo, en la parte izquierda de la sección el usuario puede buscar los cursos pertenecientes a su historial y agregarlos a la tabla del historial en construcción de la parte derecha mediante un botón verde con signo "'+". Si el usuario se equivoca al agregar un curso puede removerlo de la tabla que recolecta los cursos para construir el historial mediante otro botón rojo con signo "'-". Ambos botones pueden ser apreciados en la Figura 6.3. Al construir el historial se les ha sugerido a los usuarios que comiencen agregando sus ramos de especialidad y aquellos que crean ser más distintivos, como electivos, electivos humanistas y cursos de formación integral. Esto es debido a que si los usuarios comienzan la construcción agregando ramos comunes entre todos los estudiantes, como los cursos de plan común, tardaran más en hallar su propio registro.

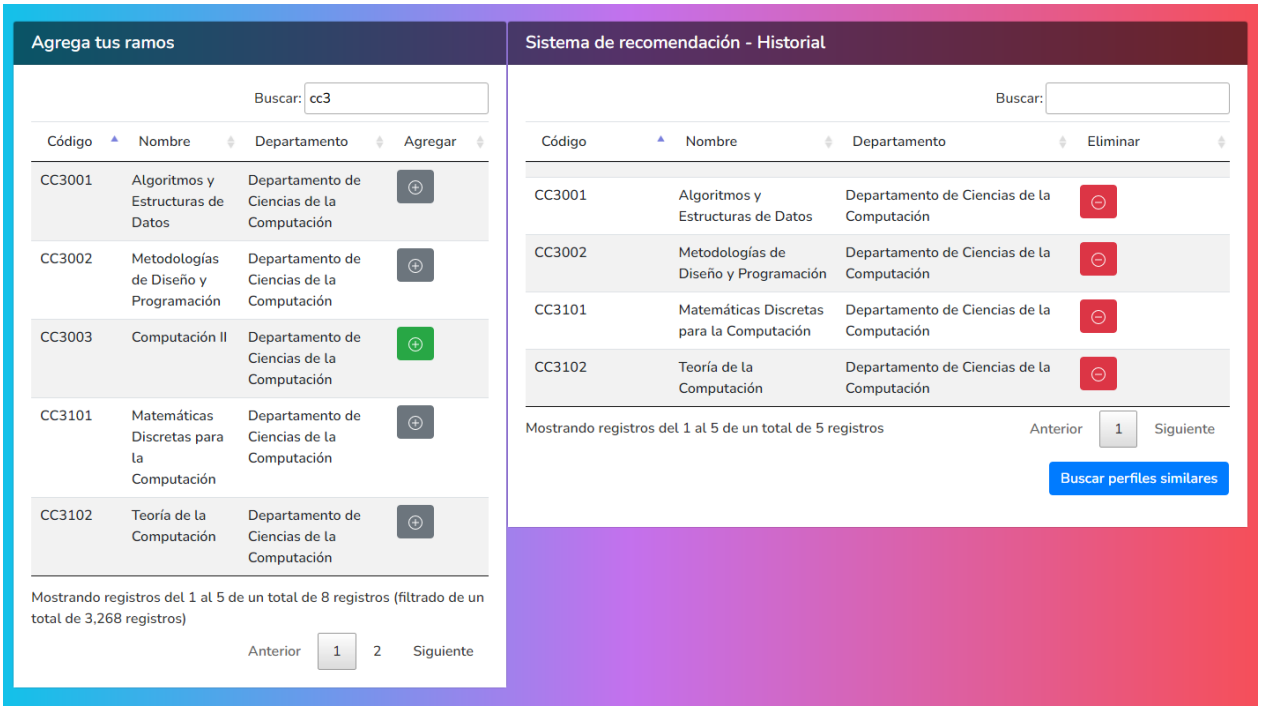

Figura 6.3: Herramienta de validación para sistema de recomendación, cuyo historial posee 4 cursos del Departamento de Computación.

Al ir agregando cursos al historial y al presionar el botón "Buscar perfiles similares", el usuario observará una nueva ventana superpuesta que despliega los resultados de todos los

perfiles de estudiantes similares. Es decir, estudiantes que también han cursado los ramos agregados por el usuario, tal como se aprecia en la Figura 6.4.

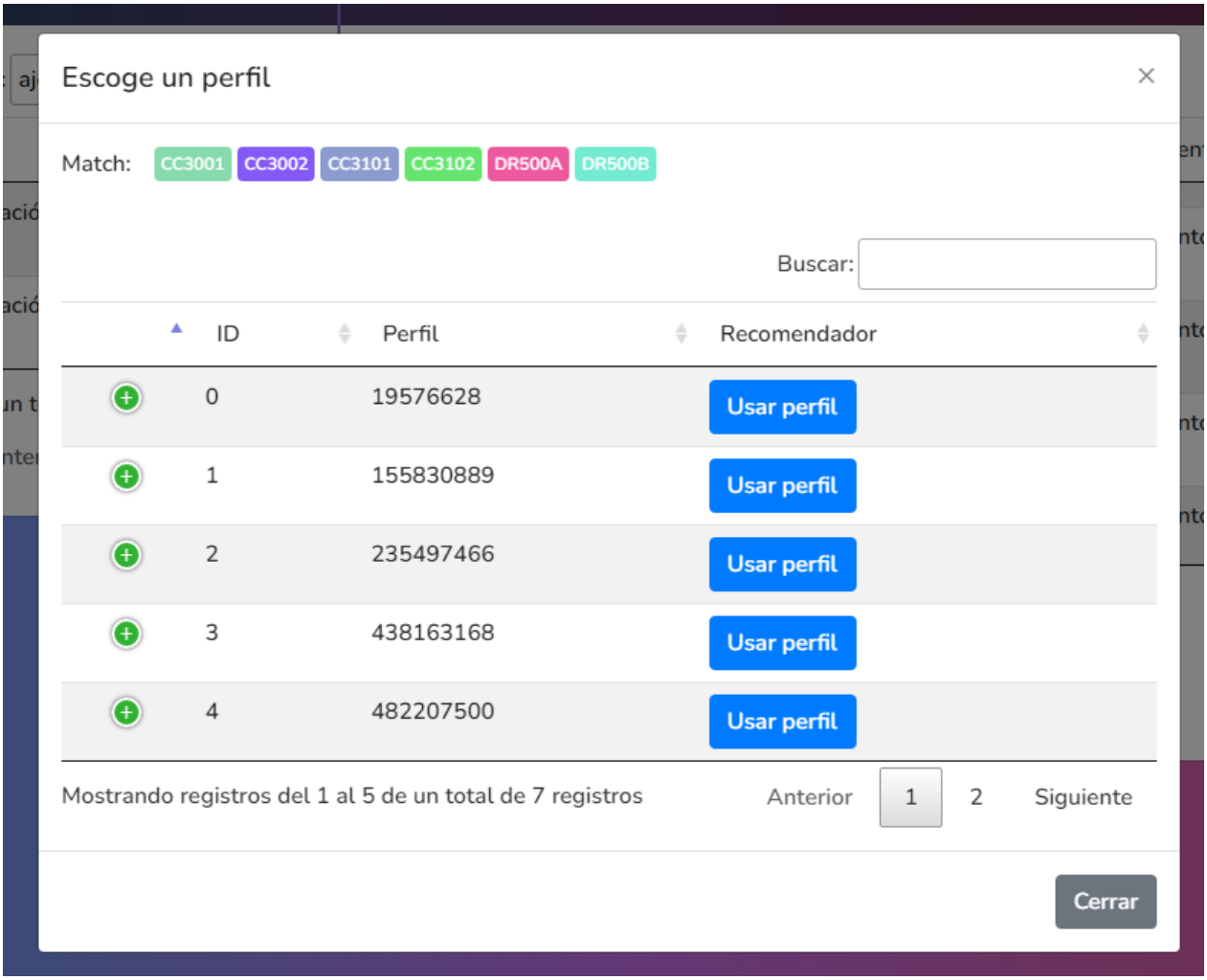

Figura 6.4: Ventana de superposición modal de los perfiles hallados tras construir un historial de cursos a partir de 6 cursos.

Luego, al cerrar la ventana y volver a la vista inicial de la Figura 6.3, a medida que el usuario vaya agregando más ramos, la cantidad de perfiles o registros mostrados en la ventana superpuesta se irán reduciendo hasta tal punto de hallar sólo un perfil, que será el suyo. En el ejemplo de Figura 6.4 se puede notar que al construir el historial agregando sólo 7 cursos (mostrados sobre la tabla, como "Match"), se ha encontrado el registro correspondiente al usuario, esto se verifica desplegando el historial del perfil presionando el circulo verde con "'+", tal como se aprecia en la Figura 6.5

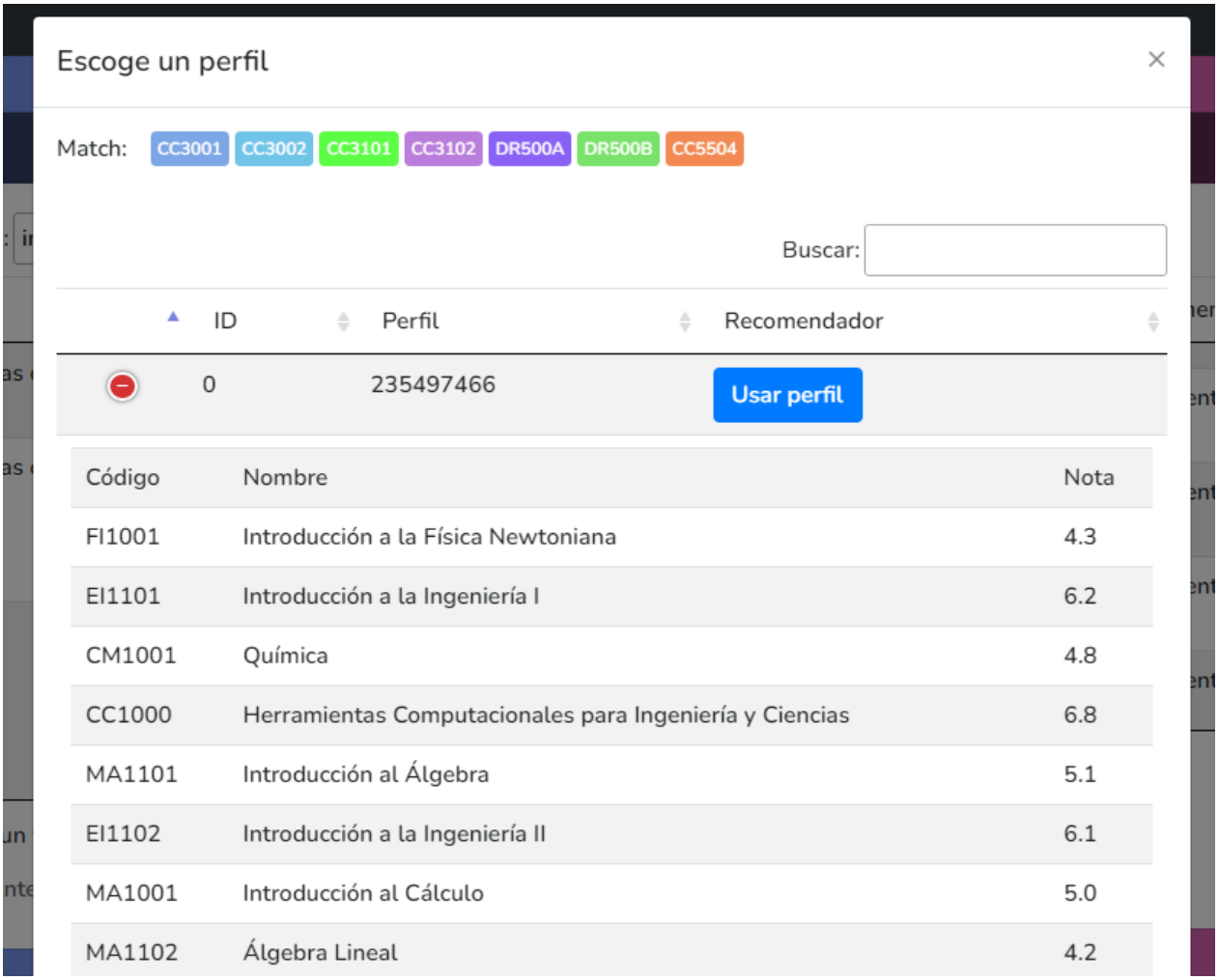

Figura 6.5: Ventana de superposición modal del único perfil hallado, al desplegar su historial de ramos.

Finalmente, tras obtener el usuario su historial de cursos, puede utilizar el sistema de recomendación presionando el botón “Usar perfil”. Luego se desplegará una nueva vista superpuesta con las recomendaciones dispuestas a través de una tabla con herramientas de paginación y búsqueda. En la tabla se dispone de los datos descriptivos de los cursos, como sus códigos, nombres, y departamentos al cual pertenecen, también se aprecia la columna asociada a la nota potencial, una nota de referencia que podría obtener el usuario si es que cursara el ramo, considerando su historial académico y el de perfiles similares.

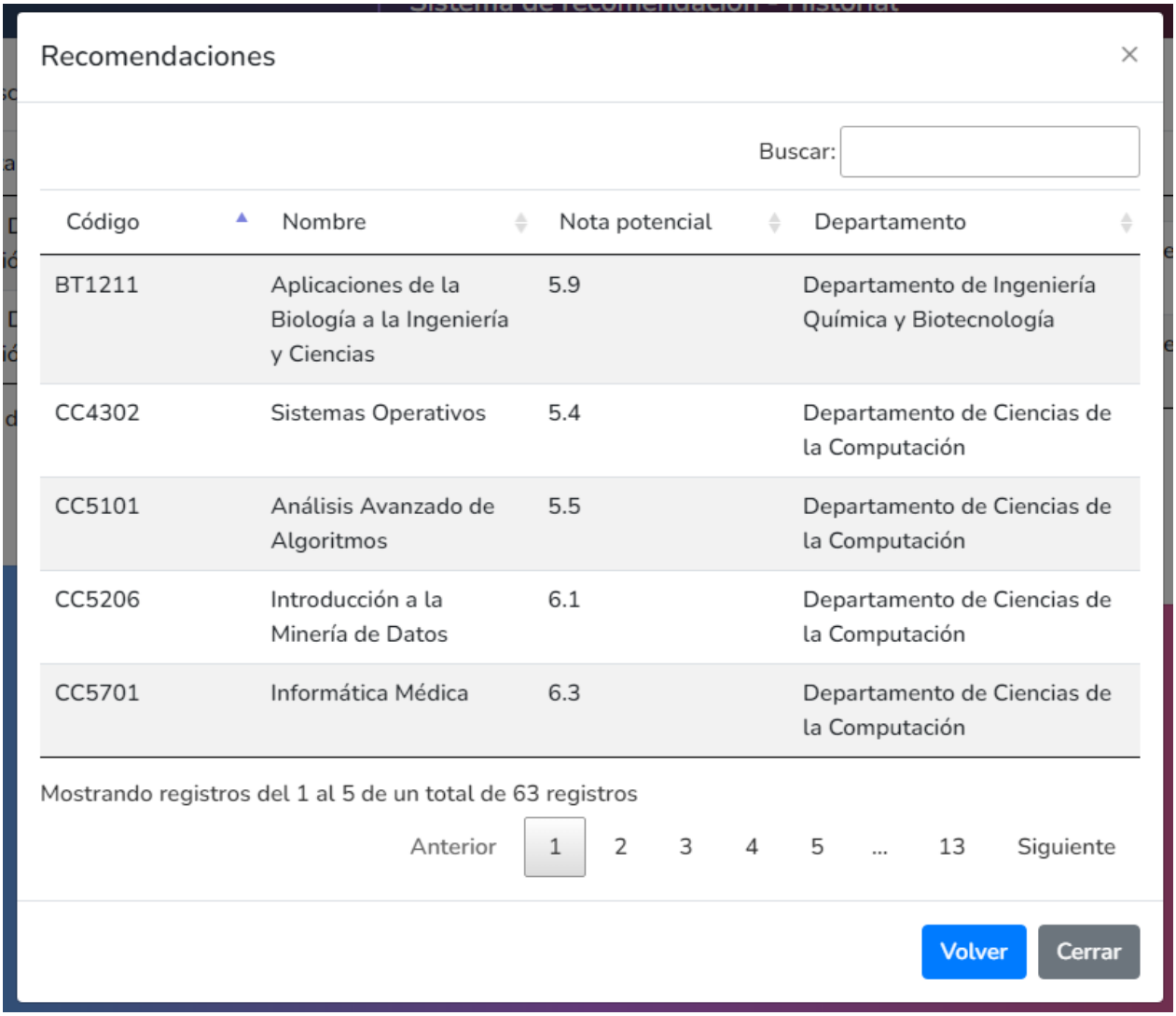

Figura 6.6: Ventana de superposición modal de recomendaciones.

6.2. Validación y resultados

La primera sección del formulario entregado a los estudiantes, muestra las indicaciones necesarias y el hipervínculo correspondiente a la página web con ambos sistemas. Ya que el objetivo de la interfaz no es evaluar la experiencia de usuario de la misma, se ha guiado al usuario mediante instrucciones básicas en el mismo formulario, en adición a las que han sido agregadas en la página web. El formulario consta de preguntas con respuestas y afirmaciones en las que el usuario debe responder mediante la escala de Likert para medir el nivel de acuerdo o desacuerdo de la afirmación, como también en algunas preguntas debe usar una escala evacuativa del 1 al 10.

La validación se ha realizado con 18 estudiantes, pertenecientes a departamentos como el Departamento de Ingeniería Industrial, el de Ciencias de la Comunicación, el de Ingeniería Civil (con estudiantes de la especialidad de Estructuras, Construcción y Geotécnica), el de Ingeniería Civil Eléctrica y el de Ingeniería Civil Mecánica, entre otros, por lo que existe una gran diversidad de áreas de estudios y recomendaciones que pueden ser constatadas por los

estudiantes en ambos sistemas.

6.2.1. Sistema de Categorización

El sistema de categorización ha sido usado para la búsqueda de 71 tópicos diferentes, entre los tópicos más buscados, se encuentran “Web”, “Machine Learning”, “Deportes”,“Datos”, “Computación”, entre otros (Se pueden apreciar en la Figura 6.7).

Figura 6.7: Tópicos buscados en el sistema de categorización, el tamaño de la palabra indica la relevancia en las búsquedas realizadas..

Como primera pregunta se ha consultado sobre la representatividad de los resultados, obteniendo los porcentajes expuestos en el gráfico de la Figura 6.8 un 50.0 % de respuestas que dicen estar muy de acuerdo con los cursos obtenidos por tópicos, le sigue un 50.0 % de gente que está de acuerdo.

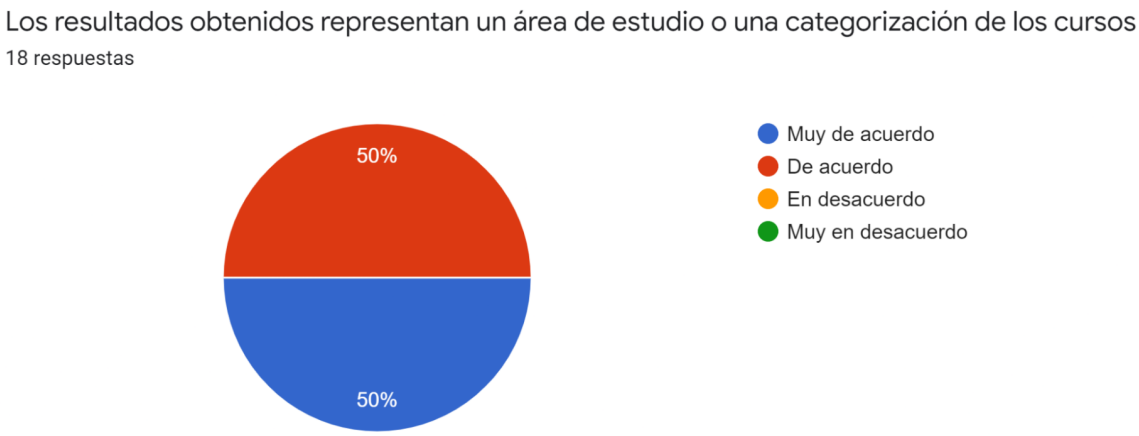

Figura 6.8: "Los resultados obtenidos representan un área de estudio o una categorización de los cursos".

En la misma línea con la pregunta anterior, se ha solicitado evaluar entre 1 a 10 la representatividad de los resultados de la categorización, obteniéndose un promedio de 8,35 (Figura 6.9). Además se ha consultado si el sistema permite conocer los cursos asociados a un tópico, y de esta manera poder descubrir nuevos cursos, obteniendo un 61.1 % de las respuestas como "Muy de acuerdo", un 33,3 % como "De acuerdo", y un 5,6 % como "En desacuerdo" tal como se aprecia en el gráfico de la Figura 6.10

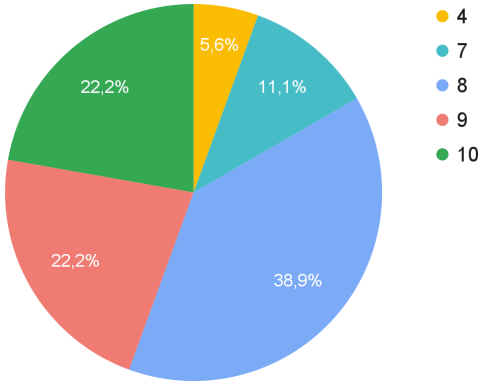

Figura 6.9: "Entre 1 a 10, siendo 1 la peor calificación, ¿Cómo evaluarías la categorización obtenida?"

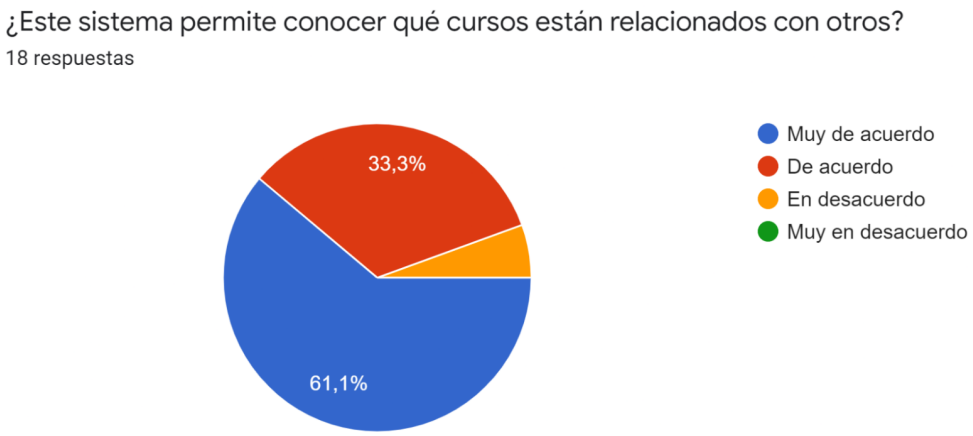

Figura 6.10: "¿Este sistema permite conocer qué cursos están relacionados con otros?"

Finalmente, se constata con el usuario que el sistema contribuye una herramienta al estudiante para la inscripción de ramos, obteniendo un 72.2 % de las respuestas como "Muy de acuerdo" y 27,8 % como "De acuerdo", tal como se aprecia en el gráfico de la Figura 6.11

Figura 6.11: "Este sistema, ¿Contribuye al estudiante una herramienta para poder inscribir los ramos?"

6.2.2. Sistema de Recomendación

En el sistema de recomendación los 15 de los 17 usuarios encuestados pudieron hallar sus historiales de notas, por lo que no tuvieron la necesidad de utilizar perfiles parecidos a los suyos para obtener las recomendaciones. La primera pregunta a este sistema evalúa las recomendaciones obtenidas en una escala de 1 a 10, obteniéndose una nota promedio de 8,06 (Figura 6.12).

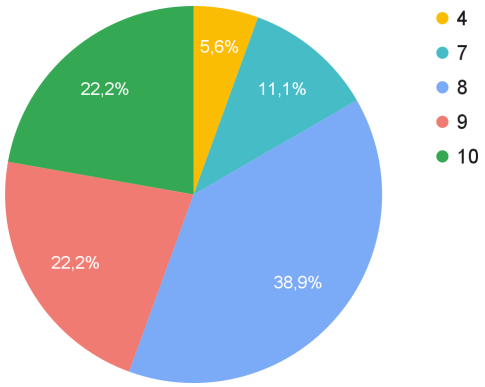

Figura 6.12: “Entre 1 a 10, siendo 1 la peor calificación, ¿Qué tan acertadas son las recomendaciones de tu departamento/especialidad?”

La siguiente pregunta mide si el estudiante está de acuerdo con que las recomendaciones obtenidas podrían resultar ser de su interés. Se obtuvo un 61,1 % de las respuestas como “Muy de acuerdo”, 33,3 % como “De acuerdo”, y un 5,9 % como “En desacuerdo” tal como se aprecia en el gráfico de la Figura 6.13

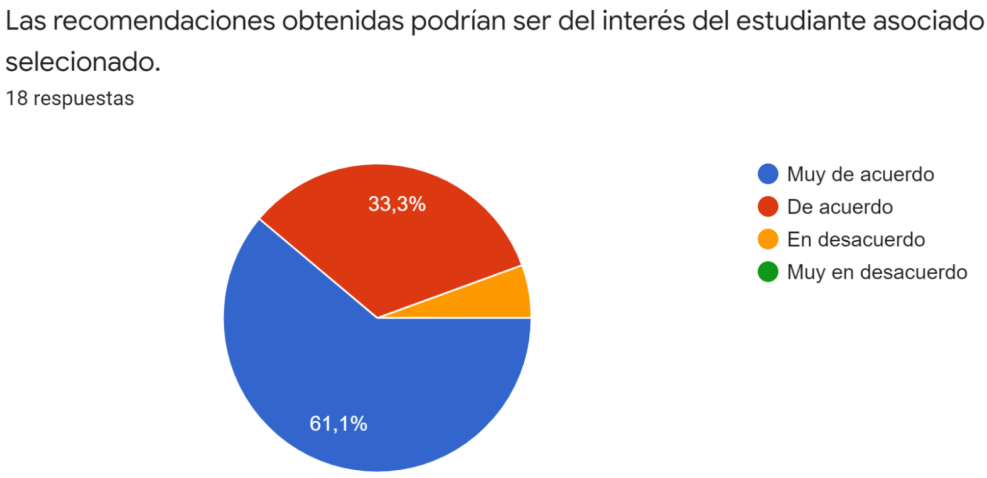

Figura 6.13: “Las recomendaciones obtenidas podrían ser del interés del estudiante asociado al perfil seleccionado.”

Finalmente la última pregunta a este sistema, permite constatar si el estudiante considera el sistema de recomendación como una herramienta para la inscripción de ramos. Se obtuvo un 61,1 % de las respuestas como “Muy de acuerdo”, 33,3 % como “De acuerdo”, y un 5,6 % como “En desacuerdo” tal como se aprecia en el gráfico de la Figura 6.13

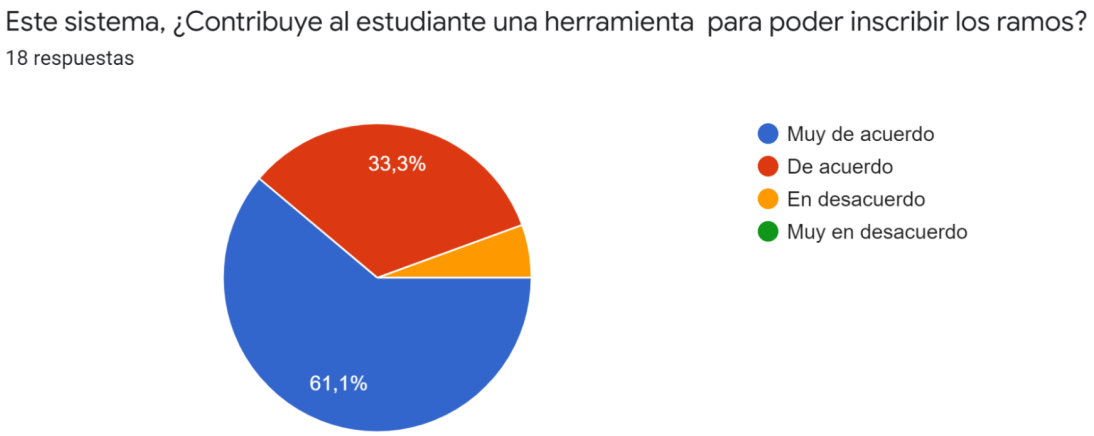

Figura 6.14: "Este sistema, ¿Contribuye al estudiante una herramienta para poder inscribir los ramos?"

6.3. Análisis de resultados

Con los resultados obtenidos en la validación de ambos sistemas se ha desarrollado un análisis que permite evidenciar la contribución de los sistemas para los estudiantes, como también distinguir los principales problemas asociados a cada uno, permitiendo identificar posibles mejoras para un eventual trabajo futuro.

6.3.1. Sistema de categorización

A partir de los resultados obtenidos en la validación de este sistema, se puede notar la variedad de tópicos buscados, debido a la diversidad de estudiantes encuestados, como también es evidente que la mayoría de ellos pertenece al departamento de computación. Por otro lado, con respecto a los tópicos obtenidos por el sistema, queda evidenciado por todos los encuestados que son representativos, evaluándolos con una nota promedio de 8,35 entre 1 a 10.

Luego, con el propósito de preguntar al usuario la utilidad del sistema para conocer la relación entre ramos, un 94,4 % de los estudiantes encuestados consideran que el sistema cumple en medida dicho propósito, sin embargo, el otro 5,6 % está en desacuerdo, esto podría deberse a que el modelo no ha sido lo suficientemente refinado. Constatando con los encuestados que han dicho estar en desacuerdo, estos dicen haber ingresado tópicos como "política", un tópico que es usado en múltiples cursos pertenecientes a diversos departamentos y que, por tanto, entrega cursos que no están relacionados entre sí, pero que sí estudian o mencionan política según sus programas. De esta manera, este tipo de términos que son de un aspecto más amplio y general resultan tener un mayor grado de dificultad para ser categorizados si es que no son acompañados por algún termino especifico, como podría ser "económica", buscando "política económica" en lugar de "política". Pese al problema mencionado, los estudiantes consideran que el sistema les contribuye una herramienta para poder inscribir los ramos, teniendo todos, un grado de acuerdo ante la afirmación.

6.3.2. Sistema de recomendación

De acuerdo a los resultados obtenidos para el sistema de recomendación, quedó evidenciado que, pese a no contar con algún sistema de autenticación para hacer la validación personalizada, la mayoría de los estudiantes pudo hallar su perfil con su historial de notas, de esta manera, las respuestas obtenidas para las preguntas tienen un alto grado de confiabilidad. De este modo, un 94,4 % de los estudiantes considera en cierta medida de acuerdo al nivel de acuerdo, que las recomendaciones obtenidas resultan ser de sus intereses, además las recomendaciones han sido evaluadas con una nota promedio de 8,06 entre una escala de 1 a 10, lo que evidencia la buena calidad de los resultados. Con respecto al 5,6 % de los encuestados que considera que las recomendaciones no han sido de sus intereses, han tenido apreciaciones acerca de los cursos recomendados que pertenecen al plan de estudio de su carrera, es decir, ramos obligatorios de la malla, esto se debe a que no existe un filtro en el sistema por ramos pertenecientes a un plan de estudio, sin embargo, quedará como trabajo futuro, por lo que eventualmente el sistema tendrá la opción de filtrar dichos cursos.

Por otra parte, se constata por la mayoría de los estudiantes, que el sistema de recomendación contribuye una herramienta para inscribir los ramos, sin embargo, un 5,6 % está en desacuerdo ante esta premisa, esto se ha constatado con los encuestados de dichas respuestas y es debido a la falta de filtros en la interfaz creada con propósito de validar el sistema. Filtros por cursos pertenecientes a un departamento, a un plan de estudio, por cantidad de créditos, por requisitos, entre otros, sin embargo, la retroalimentación obtenida por estos usuarios será eventualmente implementada en producción.

6.4. Resumen

En este capítulo se explica el motivo de la validación, y el por qué se ha realizado de a través de una interfaz con potenciales usuarios. Luego se describe la interfaz construida y explica brevemente el uso y las funcionalidades de la misma, tanto para el sistema de categorización como el de recomendación. Posteriormente, se muestran los resultados obtenidos por el formulario de validación y se realiza un análisis de los mismos, constatando los buenos y malos resultados.

Capítulo 7

Conclusiones

En esta memoria se ha trabajado en construir dos nuevas herramientas que permiten orientar a los estudiantes en la toma de decisiones al momento de realizar la selección de los ramos a cursar durante su carrera. Un sistema de categorización que permite al estudiante identificar el área de estudio al cual pertenecen los distintos cursos de la facultad en los que tiene interés y, en paralelo, un sistema de recomendación que permite establecer un perfil del estudiante a partir de sus ramos cursados, sus notas y el historial de otros estudiantes cuyos cursos y notas son similares.

La creación de ambos sistemas era importante debido a que la inscripción de cursos resulta ser un problema recurrente en los estudiantes ya que, ante la falta de experiencia y desconocimiento del contenido de los cursos, suelen optar por consultar a sus pares e inscribir los ramos a partir de opiniones ajenas. Si bien esta opción es una buena alternativa, podría desencadenar aún más problemas en la formación y especialización del estudiante al no tener en cuenta sus propios intereses.

Para el logro de los objetivos planteados, primero se evaluaron las alternativas en los algoritmos y tecnologías para construir el sistema de categorización, estudiando y probando cada uno de ellos, para luego realizar una comparación de los resultados. Tras conseguir, el algoritmo con mejores resultados, se procedió a construir el pipeline por el cual se construiría el modelo de categorización de cursos. Luego, con el modelo generado, lo siguiente fue evaluarlo y validarlo con algunos estudiantes y volver a iterar la construcción del mismo, variando y modificando las componentes del pipeline hasta generar un modelo aceptable con respecto al feedback y a la eficiencia para generar la categorización.

Una vez que se consiguió construir un modelo válido para el sistema de categorización, se procedió a seguir con el sistema de recomendación, construyéndose un pipeline a partir de un filtro colaborativo basado en el usuario, en el cual se utilizan los historiales de los cursos de los estudiantes para establecer el filtro. De la misma manera que con el pipeline del sistema de categorización, se generaron los recursos (en este caso el dataset) para correr el algoritmo, se validó con algunos estudiantes y se refino de manera iterativa hasta obtener resultados coherentes.

Para evaluar y validar de manera solida ambos sistemas, se desarrolló una página web que muestra ambos sistemas en una única vista y que funciona haciendo peticiones a un API REST que expone los métodos necesarios para leer e interactuar con el modelo del sistema de categorización y el dataset del sistema de recomendación. La página web ha sido utilizada como herramienta para validar con 18 estudiantes de carreras pertenecientes a distintos departamentos de la facultad, para constatar de distintas perspectivas y con mayor completitud los resultados obtenidos.

En conclusión, tras validar ambos sistemas y tomando en cuenta los objetivos del presente trabajo y los resultados obtenidos mediante las evaluaciones de los estudiantes que participaron en la validación, se puede decir que se lograron crear funcionalidades o herramientas que permiten orientar a los estudiantes en la toma de decisiones al momento de realizar la selección de los ramos a cursar durante su carrera a partir de factores como la oportunidad, el entorno y la personalidad del estudiante. Los sistemas creados contribuirán una herramienta al estudiante en su formación como profesional, permitiéndoles especializarse en las áreas de sus intereses.

Por otro lado, la construcción del sistema de categorización podría ser considerado un avance en el estado del arte con lo que respecta a su función, esto es debido a que el problema ha sido analizado desde una perspectiva no supervisada del aprendizaje de máquinas, lo que permite desconocer el tópico a cuál pertenece un curso, que resulta ser beneficioso para casos como el de Ucampus que no cuenta con atributos descriptivos que relacionen un curso con otro a nivel de contenido. La solución alternativa, supervisada, consiste en etiquetar cada uno de los cursos pertenecientes a la facultad, con un tópico, sin embargo, es un trabajo complejo, poco escalable y altamente demandante, considerando en primer lugar que se debe contar con gente capacitada y con conocimiento en los cursos y en los tópicos o áreas de estudio existentes para poder etiquetar. Por otro lado, existen cursos que, perteneciendo a especialidades y departamentos distintos, poseen relación entre sí, esta característica está implícita en el sistema desarrollado. Sin embargo, para tenerla en cuenta en esta solución alternativa, se requiere gente que este al tanto de la relación entre dichos cursos. Como la dificultad anterior, existen otras a tener en cuenta y que al abordar el problema como uno no supervisado, podrían resultar ser indiferentes. Adicionalmente, como se menciona en el Estado del Arte, universidades prestigiosas a nivel mundial, como Harvard, sólo cuentan con sistemas de búsquedas basado en consultas y uso de filtros, por lo que una herramienta como el sistema de categorización perfectamente podría implementarse en este tipo de universidades.

Con respecto al sistema de recomendación condice con el principio de filtro colaborativo utilizado en sistemas sofisticados de plataformas como “Netflix” o “Youtube”, por lo que utilizarlo en un escenario académico significa adecuarse a las tecnologías contemporáneas a las cuales la sociedad está hoy en día más relacionada, en especial los jóvenes estudiantes que suelen usar este tipo de plataformas. Por lo demás, obtener recomendaciones a partir de este principio permite obtener múltiples y variados cursos resultados, debido a la gran cantidad de estudiantes que presentan perfiles similares. Tal como sucede en las plataformas mencionadas que, al contar con muchos usuarios, permiten tener múltiples alternativas en la recomendación de series o vídeos.

7.1. Trabajo futuro

Si bien el trabajo realizado permitió construir y validar los resultados obtenidos de ambos sistemas, se espera que estos sean implementados en la plataforma de Ucampus, formando parte de una vista adicional que permita interactuar al estudiante tal como se hizo en la página web de la validación. Adicionalmente, al contar el sistema de recomendación con los datos descriptivos del estudiante, la búsqueda del perfil hecha en la validación podría volverse obsoleta, debido a que las recomendaciones serían obtenidas de manera inmediata con el identificador del usuario.

En cuanto al gráfico mostrado en el sistema de categorización, podría ser replicado en la vista de Ucampus, junto con agregar la lógica de los cursos que son requisitos de otros, como también agregar una ficha descriptiva del curso al presionar sobre un nodo. Con respecto al muestreo de los resultados mediante las tablas, se espera que sean agregados filtros que permitan al estudiante filtrar por departamento, por cursos obligatorios, por cursos pertenecientes a un plan de estudio y por cursos electivos.

Eventualmente al ser interfaces nuevas en la plataforma, se espera que sean diseñadas y validadas por los mismos estudiantes, evaluando en este caso la experiencia de usuario junto con resultados obtenidos. Considerando que al desplegar nuevas vistas en la plataforma estas pueden ser probadas por miles de usuarios, la retroalimentación permitiría refinar los datasets con los que se generan ambos sistemas. Adicionalmente, junto al sistema de recomendación, podría agregarse un registro de comentarios que permita a los estudiantes describir su experiencia tras cursar un ramo, reforzando la retroalimentación del estudiante con respecto al curso.

7.1.1. Mejoras al modelo del Sistema de Categorización

El pipeline diseñado para el sistema de categorización podría refinarse, en particular durante el preprocesamiento, esto es debido a que los programas de los cursos cuentan con bastantes palabras que no son “stop-words” y que suelen repetirse entre programas, por lo que afecta directamente en la categorización. Pese a contar con un diccionario que se encarga de reconocer dichas palabras, este podría extenderse aún más haciendo un análisis minucioso a los programas o a una plantilla de ellos.

Una mejora a largo plazo, que podría mejorar drásticamente los resultados de las categorizaciones, es construir un sistema “Scraper”, que busque el título del curso en alguna enciclopedia como Wikipedia, recolecte la información obtenida y la adjunte al dataset de los cursos junto a sus programas. De esta manera el dataset estaría siendo alimentado de dos fuentes: su programa, y la página de la enciclopedia. Eventualmente esto podría mejorar las categorizaciones obtenidas, sobre todo para aquellos cursos que presentan un programa con poco contenido.

7.1.2. Mejoras al modelo del Sistema de Recomendación

Si bien el filtro colaborativo basado en usuarios utilizado en el presente trabajo es uno de los métodos más utilizados existentes en el estado del arte, existe una gran variedad de mé-

todos que podrían ayudar a resolver el problema, uno de ellos es la mencionada factorización matricial que permite reducir la dispersión de las matrices del filtro colaborativo y pese a que ha sido descartada por la escalabilidad de los datos, se podría estudiar una mejor manera de operar las matrices. Otra alternativa a estudiar podría ser un sistema basado en redes neuronales profundas que resuelve la mayoría de limitaciones que posee la factorización matricial.

Por otro lado, con respecto a la validación de este sistema, se podría crear un conjunto de datos con historiales conocidos y de este modo crear métodos de evaluación offline, que permitan comparar las recomendaciones obtenidas por el sistema para un año especifico de los historiales de notas y posteriormente constatar si efectivamente las recomendaciones generadas fueron cursadas en los años posteriores por los estudiantes.

Bibliografía

[1] Borchert, M., "Career choice factors of high school students," Career choice factors, 2001.

[2] Blei, D. M., Ng, A. Y., y Jordan, M. I., "Latent dirichlet allocation," J. Mach. Learn. Res., vol. 3, p. 993–1022, 2003.

[3] Hofmann, T., "Learning the similarity of documents: An information-geometric approach to document retrieval and categorization," en Advances in Neural Information Processing Systems (Solla, S., Leen, T., y Müller, K., eds.), vol. 12, MIT Press, 1999, https://proceedings.neurips.cc/paper/1999/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf.

[4] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., y Dean, J., "Distributed representations of words and phrases and their compositionality," 2013, doi:10.48550/ARXIV.1310.4546.

[5] Mikolov, T., Chen, K., Corrado, G., y Dean, J., "Efficient estimation of word representations in vector space," Proceedings of Workshop at ICLR, vol. 2013, 2013.

[6] Angelov, D., "Top2vec: Distributed representations of topics," 2020, doi:10.48550/ARXIV.2008.09470.

[7] Rojas, J. P., "Spanish word embeddings," 2019, https://github.com/dccuchile/spanish-word-embeddings.

[8] Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K., "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, doi:10.48550/ARXIV.1810.04805.

[9] Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., y Pérez, J., "Spanish pre-trained bert model and evaluation data," en PML4DC at ICLR 2020, 2020.

[10] Jin, X. y Han, J., "K-means clustering," 2010, doi:10.1007/978-0-387-30164-8_425.

[11] Pokojski, W. y Pokojska, P., "Voronoi diagrams – inventor, method, applications," Polish Cartographical Review, vol. 50, pp. 141–150, 2018, doi:10.2478/pcr-2018-0009.

[12] Campello, R., Moulavi, D., y Sander, J., "Density-based clustering based on hierarchical density estimates," vol. 7819, pp. 160–172, 2013, doi:10.1007/978-3-642-37456-2_14.

[13] laurens van der Maaten y Geoffrey Hinton, "Visualizing data using t-sne," 2008.

[14] Hinton, G. y Roweis, S., "Stochastic neighbor embedding," 2008.

[15] McInnes, L., Healy, J., y Melville, J., "Umap: Uniform manifold approximation and projection for dimension reduction," 2018, doi:10.48550/ARXIV.1802.03426.

[16] Pascual, N. T., "Factors affecting high school students' career preference: a basis for career planning program," International Journal of Sciences: Basic and Applied Research, vol. 16, pp. 1–14, 2014.

[17] A., A. B., “Lematización basada en análisis no supervisado de corpus,”.

[18] Spathis, D., “Average words to represent documents with word2vec,” 2017, https://github.com/sdimi/average-word2vec.

ANEXO

Anexo A

Prueba de Word Embedding, usando Word2vec

A.1. Construcción de conjunto de datos de entrada

La mayor dificultad para comenzar a realizar el trabajo, es crear un buen conjunto de datos de entrada, cuyos elementos sean representativos, juntando todos los recursos posibles con tal de obtener resultados coherentes al momento de realizar las clasificaciones por cursos. Inicialmente se gestionaron recursos como: el listado de cursos de la Escuela de Ingeniería, el listado de requisitos de cada uno de los ramos y los programas de cada curso. Ambos listados se representaban como documentos con extensión csv, y debido a que contaban con más de 3000 filas se optó por acotar los cursos a sólo aquellos que pertenecen al Departamento de Ciencias de la Computación, quedando un total de 134 filas correspondientes a los cursos. Para esto se utilizó el programa *Excel* que cuenta con las herramientas necesarias para exportar los documentos acotados, con un determinado delimitador y *encoding* en particular, para este caso se escogió "latin-1" debido al uso de signos ortográficos, obteniendo dos conjuntos de datos como los descritos en las Tablas 1 y 2.

Tabla A.1: Primeras 6 filas del conjunto de datos: Listado de requisitos.

Código	Nombre	Requisitos
CC1002	Introducción a la Programación	CR3
CC3001	Algoritmos y Estructuras de Datos	CC1002
CC3002	Metodologías de Diseño y Programación	CC3001/CC30A
CC3003	Computación II	CC1001,MA1101
CC3101	Matemáticas Discretas para la Computación	CC1002,MA1101
CC3102	Teoría de la Computación	CC3101,CC3001

Con ambos listados acotados, lo siguiente fue crear un *script* en *Python* para automatizar la extracción del contenido de cada uno de los programas de los cursos a texto plano. Para esto se hizo uso de librerías como *PyPDF2* y *PyMuPDF*, ambas librerías permiten pasar documentos con extensión pdf a texto plano, sin embargo, se optó por probar ambas, debido

Tabla A.2: Primeras 2 filas del conjunto de datos: Listado de cursos.

Código	Nombre	Departamento
CC1000	Herramientas Computacionales para Ingeniería y Ciencias	Departamento de Ciencias de la Computación
CC1001	Computación I	Departamento de Ciencias de la Computación

a que existieron documentos que generaban errores al usar una o la otra, en su defecto, se debió pasar el contenido manualmente de los documentos que no funcionaban con ninguna de las dos opciones. Posteriormente se construyó un *dataframe* de *Pandas*, cuyas columnas corresponden al nombre del curso y otra columna con todos los elementos descriptivos de un curso de manera concatenada, generando de esta manera una descripción general para cada ramo, tal como se puede apreciar en la Tabla 3.

Tabla A.3: Primera fila del *dataframe* de cursos, cuyas columnas "Descripción", poseen elementos descriptivos del curso, en este ejemplo se han reducido para ilustrar el contenido.

Nombre	Descripción (Ejemplo)
Herramientas Computacionales para Ingeniería y Ciencias	Departamento de Ciencias de la Computación CC1000 Herramientas Computacionales para Ingeniería y Ciencias El curso Herramientas Computacionales para Ingeniería y Ciencias tiene por finalidad asegurar que todos los estudiantes, independientemente de sus conocimientos previos, logren un nivel de manejo de las herramientas computacionales básicas tales como: Excel, R, MATLAB, Maple, LaTeX, entre otros, que permita la resolución de problemas simples de la ingeniería y ciencias... Utiliza planillas de cálculo para el procesamiento, análisis y visualización de conjuntos de datos, considerando el uso de funciones de cálculo matemático, estadístico y de manejo de bases de datos... Utiliza MATLAB para calcular y analizar gráficos de funciones, correlacionando datos mediante gráficos de dispersión. Maneja vectores y matrices, carga archivos de datos, realiza cálculos numéricos y visualiza los resultados usando diversas técnicas, incluyendo gráficos y mapas de colores....

A.1.1. Preparando Word2vec

Existen ramos cuyos nombres no tienen palabras en común, como lo son, por ejemplo, "Diseño de Sistemas Interactivos" (CC5510) e "Interface Humano Computador" (CC5504), ambos pertenecientes al área de Interacción Humano-Computador (HCI en inglés), que además presentan programas cuyos contenidos son bastante parecidos. Sin embargo, si es que no se tiene conocimiento del área al cual pertenecen o no se revisan los programas de estos cursos, no sería posible establecer una relación que permita vincularlos de algún modo. Por otro lado, existen ramos que son extensiones de otros ya existentes como lo es el caso de "Taller de Programación A (CC4001)" y "Taller de Programación Competitiva A" (CC4005), para este ejemplo es claro que debe existir una correlación entre ambos. Si consideramos la técnica de *Word2vec* descrita en el presente informe, tanto para el primer como el segundo ejemplo debiese existir una cercanía entre los vectores asociados a los nombres de cada ramo. Para corroborar esta hipótesis se realizará un experimento a partir del conjunto de datos de entrada y el uso del algoritmo de *Word2vec* incluido en la biblioteca *Gensimn* en *Python*.

Para esta prueba se estará utilizando un modelo pre-entrenado de *Word Embedding* llamado *Spanish Billion Words Corpus* (SBWC) de Cristian Cardellino, cuyo corpus consta de aproximadamente 1.4 billones de palabras en español. Entre las dependencias a utilizar estarán Numpy, Pandas, Nltk, Gensim y Spacy.

A.1.1.1. preprocesamiento

El primer procedimiento que se debe realizar es el de tokenización, este consiste esencialmente en dividir el corpus completo en unidades más pequeñas, como palabras o términos individuales. Cada una de estas unidades se llamarán tokens. El corpus está constituido a partir de la concatenación de cada una de las descripciones asociadas a un curso que son obtenidas del *dataframe* creado previamente, este será tokenizado mediante el modelo en español "es_core_news_lg" de la librería de *Spacy*, que cuenta con un *pipeline* entrenado, con componentes de tokenización, *stemming*, lematización, entre otros.

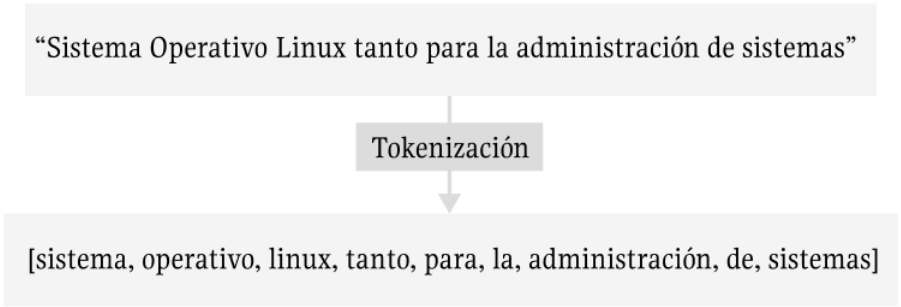

Figura A.1: Ejemplo de tokenización para una frase del corpus de descripciones concatenados.

Del ejemplo de la Figura 2, se puede notar que existen palabras como el artículo *la* que no aporta en términos generales para diferenciar y clasificar entre dos ramos, este tipo de palabras son las llamadas *stop-Words*, que corresponden a todas aquellas palabras que no tienen un significado por sí solas, sino que actúan como conectores acompañando o modificando a otras, este tipo de palabras están conformados por artículos, pronombres, proposiciones e

incluso algunos verbos.

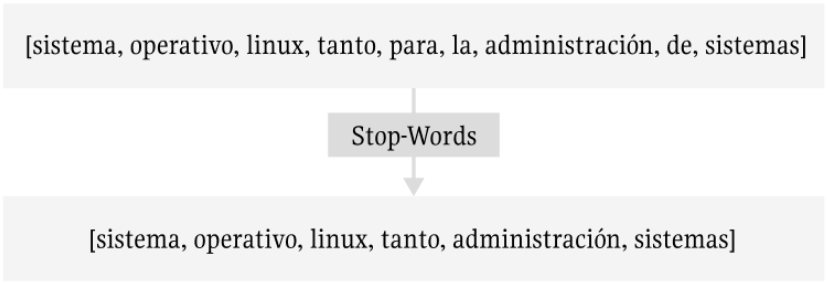

Figura A.2: Ejemplo de *stop-Words* para una frase del corpus de descripciones concatenados.

Luego, del ejemplo de *Stop-Words* de la Figura 3, se puede notar que existe la palabra *sistema* y su versión plural *sistemas*, sin embargo, la presencia entre ambas formas morfológicas puede no significar una gran diferencia en los resultados finales, por lo que se debe realizar una *lematización* [17] de los tokens. Este proceso de lematización consiste en transformar todas aquellas palabras de un texto que están en una forma flexionada o derivativa a una forma normal o lema. El lema es el representante de todas las formas flexionadas de una misma palabra. Por tanto, para el ejemplo descrito, el lema de "sistemas" es "sistema".

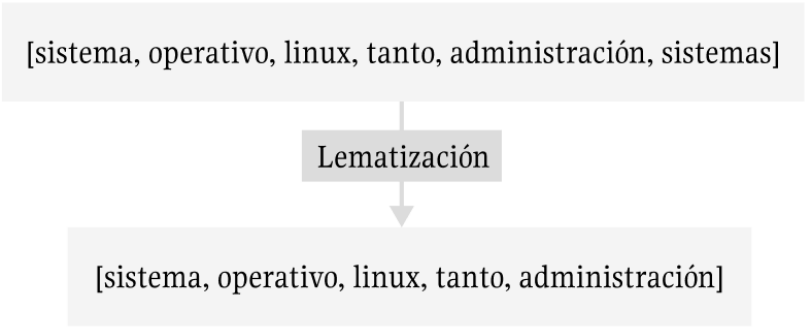

Figura A.3: Ejemplo de Lematización para una frase del corpus de descripciones concatenados.

A.1.1.2. Procesamiento de palabras

Con el preprocesamiento completo, a continuación, necesitamos cargar un modelo *Word2vec* previamente entrenado, para esto se utiliza el modelo descrito previamente, *Spanish Billion Words Corpus* (SBWC). Este modelo es cargado mediante el método de carga de la biblioteca *Gensim* que permite cargar modelos creados con algoritmos como *GloVe*, *Fastext* y por supuesto, *Word2vec*. Con el modelo cargado, se puede conocer los vectores asociados a las palabras que pertenecen al *dataframe* construido previamente, por ejemplo, según el modelo cargado, los vectores asociados a las palabras, "curso", "herramientas" y "computacionales", son los descritos en la Figura 5.

A continuación, se hace una reducción de los vectores de palabras obtenidos usando el algoritmo de t-SNE de la biblioteca *sklearn*, para ver si surge algún patrón en común entre

	0	1	2	3	4	5	6	7	8	9	...
curso	-0.073458	-0.272631	-0.271238	-0.107281	-0.068484	-0.676420	-0.029671	0.299533	-0.074630	-0.022121	...
herramientas	-0.360432	0.077423	-0.068517	0.129947	-0.501643	-0.602788	0.530834	0.233599	0.587406	0.296430	...
computacionales	-0.743668	-0.160593	0.058954	0.069514	-0.560875	-0.019577	-0.100859	-0.112902	0.129541	0.172288	...

3 rows × 300 columns

Figura A.4: Vectores de 300 columnas, asociados a las palabras; “curso”, “herramientas” y “computacionales”.

ellas, en este paso, se debió iterar más de una vez, variando cada uno de los parámetros que admite el algoritmo, para así refinar los resultados obtenidos.

Posteriormente, lo que sigue es representar gráficamente los resultados utilizando la librería de *matplotlib*. Como se puede apreciar en la Figura 6, palabras relativas a la computación están agrupadas en una sección del grafo, sin embargo, se puede notar un evidente fallo en el proceso de lematización, pues la palabra “computacionales” debiese tener como lema “computacional”, esto puede haber ocurrido debido a que el modelo en español de *Spacy* no cuenta con este lematizado en particular, lo mismo sucede con la palabra “ciencia”, que se aprecia en la Figura 7.

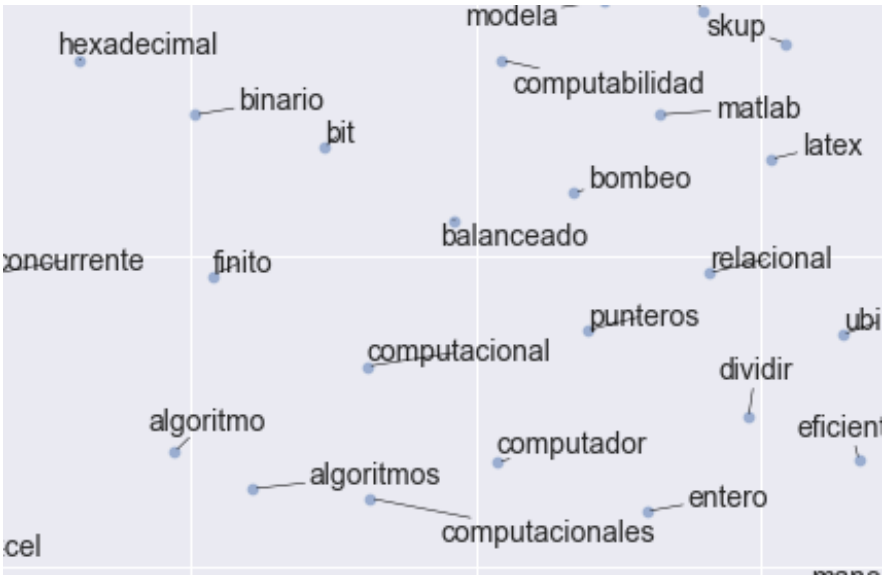

Figura A.5: Extracto de representación gráfica de t-SNE.

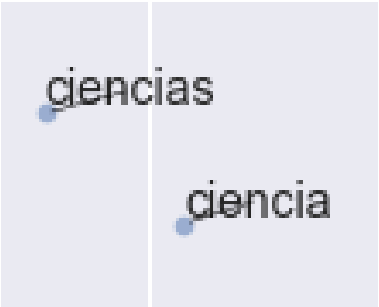

Figura A.6: Problema de lematizado para palabra “ciencia”.

A.1.1.3. Word Embeddings promediado

Tenemos una idea de cómo funciona la técnica de *Word Embedding* aplicada a este conjunto de palabras. Lo que sigue, es extenderlo a los títulos de los cursos, para esto, se debe rehacer el paso de preprocesamiento para mantener los nombres de ramos intactos. Para realizar esta extensión, *Dimitris Spathis* [18] ha creado una serie de funciones que permiten promediar los vectores por palabras. Luego al representar gráficamente el resultado de los vectores asociados a los nombres de los cursos, se obtiene el gráfico de la Figura 8.

A.1.1.4. Resultados preliminares

De la Figura 8, se pueden distinguir algunos patrones, como lo es la cercanía de los cursos asociados a “Programación competitiva”, una relación entre “Computación I” y “Computación II”, y otra entre “Estructura de datos comprimidos” y “Algoritmo y estructura de datos”. Por otro lado, hay cursos que pese a estar cercanos, no cuentan con alguna relación directa para asociarlos a un área de estudio, este problema puede deberse a múltiples factores, como el mencionado problema de lematización, una mala configuración de los parámetros del algoritmo *t-SNE*, una poca representatividad del idioma del modelo en español de *Spacy*, problemas con el uso del algoritmo de *Word2vec*, o en su defecto con la técnica de *Word Embedding*.

En síntesis, en esta prueba se logró un gran acercamiento para un sistema de categorización, nos permite tener una idea de cómo funciona la técnica de *Word Embedding* y que factores hay que tener en consideración. Por otro lado, se debe tener en cuenta que a partir de este proceso se debe crear el modelo que permite relacionar los ramos cursados por un estudiante, a los *clusters* de cursos formados por la técnica a utilizar, por lo que, para obtener relaciones fidedignas, el proceso debe ser refinado, probando y variando cada uno de los factores, técnicas y algoritmos existentes.

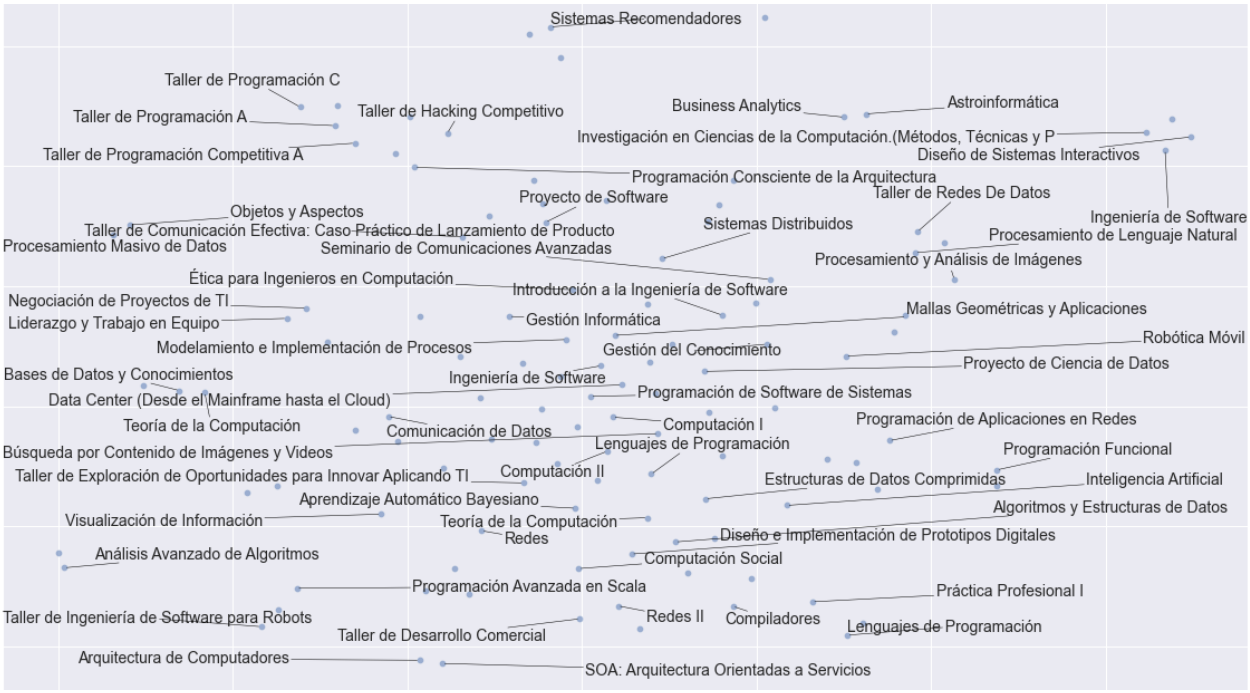

Figura A.7: Representación gráfica de *t-SNE*, para títulos de cursos.