Caratula

Institución: Universidad de las ciencias informaticas ( UCI )

Título: Desarrollo e Implementación de un Sistema de Recomendación Basado en Deep Learning para el proyecto z17.

Indicación del tipo de trabajo del que se trata: El presente trabajo constituye un **desarrollo de software** de **investigación aplicada** en el área de la **inteligencia artificial**. Consiste en la creación de un **sistema de recomendación basado en Deep Learning** diseñado para el **proyecto z17.**

Autor: Alejandro Figueroa Rodríguez

Tutor(es):

Yadier Perdomo Cuevas

Aneyty Martin Garcia

Falero

Fecha de presentación: ---

Agradecimientos: ---

Dedicatoria: ---

Declaración de autoría:

Declaro por este medio que yo Alejandro Figueroa Rodriguez, con carné de identidad 01082968160soy el autor principal del trabajo titulado “Desarrollo e Implementación de un Sistema de Recomendación Basado en Deep Learning para el proyecto z17” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Resumen

En la era digital, la sobreabundancia de información ha generado la necesidad de sistemas inteligentes que ayuden a los usuarios a encontrar contenido relevante. Al aprovechar los avances en inteligencia artificial, estos sistemas permiten a los usuarios descubrir nuevos productos y servicios de manera más eficiente, mientras que las empresas pueden mejorar sus estrategias de marketing y aumentar sus ingresos. El deep learning ha revolucionado la forma en que interactuamos con las plataformas digitales, permitiendo una personalización sin precedentes y una gestión eficiente de grandes volúmenes de datos. Esta tecnología utiliza algoritmos avanzados para aprender de los datos de los usuarios y mejorar continuamente la experiencia de navegación. Como resultado, plataformas como YouTube, Netflix y Google Play Store pueden ofrecer recomendaciones altamente precisas que reflejan los intereses y preferencias individuales. Este estudio explora la creación y aplicación de un sistema de recomendación avanzado, utilizando técnicas de deep learning, para las plataformas digitales del proyecto Z17. La implementación de un sistema de recomendación basado en deep learning en estas plataformas no solo mejorará la interacción del usuario, sino que también impulsará la relevancia y el alcance de los productos de Z17 en el mercado global. El objetivo principal es mejorar la experiencia y satisfacción del usuario al ofrecer sugerencias altamente personalizadas y precisas, aumentando el tiempo de interacción con la plataforma y anticiparse a las necesidades de los usuarios manteniéndose a la vanguardia de una experiencia de usuario excepcional.

Palabras claves: Sistema de recomendación Deep Learning proyecto Z17, Inteligencia Artificial

Índice

Introducción

Situación problemática

1. Recopilar y buscar información sobre trabajos anteriores que traten sobre un tema similar al nuestro: Estudiar bien estos trabajos y formular una idea de qué manera se querrá hacer el recomendador.
2. Recopilar y procesar datos: Se debe recopilar información sobre los datos disponibles y sobre los usuarios que utilizarán el sistema de recomendación. Es importante procesar y analizar esta información para prepararla para el uso de los modelos.
3. Diseñar e implementar los modelo de aprendizaje profundo: Se debe diseñar y programar los modelos que utilizará el sistema de recomendación. Esto implica la implementación de algoritmos y técnicas de análisis de datos para encontrar similitudes entre los usuarios y proporcionar recomendaciones basadas en estas similitudes.
4. Validar y optimizar el modelo de recomendación: Es importante evaluar la precisión y efectividad del modelo de recomendación y optimizarlo para mejorar su rendimiento. Esto significa que hay que realizar pruebas y validaciones.
5. Finalmente, intentar integrar el sistema de recomendación en las plataformas.

**Capítulo 2**

* + 1. **Etapas.**

El sistema está compuesto por **3 etapas** fundamentales, Recuperación, Clasificación y Re-clasificación. Estas etapas están conectadas entre sí de manera secuencias respectivamente donde la salida de una es la entrada de la otra siendo la última etapa la encargada de devolver los datos a recomendar y la primera la encargada de obtener el corpus de elementos candidatos base inicial. A continuación cada una de las etapas:

* **Recuperación**: Esta etapa es responsable de seleccionar un conjunto inicial de cientos o miles de candidatos entre todos los posibles candidatos. El objetivo principal de esta etapa es eliminar de manera eficiente a todos los candidatos que no les interesan al usuario. Dado que se puede estar tratando con millones de candidatos, tiene que ser computacionalmente eficiente. Esta etapa puede estar compuesta por uno o varios modelos generadores de candidatos donde cada uno se base en los datos implícitos almacenados de los usuarios.
* **Clasificación**: Toma los resultados del modelo de recuperación y los ajusta para seleccionar el mejor puñado de recomendaciones posibles basándose en los datos explícitos que tenga disponible. Su tarea es reducir el conjunto de elementos que pueden interesar al usuario a una lista corta de posibles candidatos que ronde entre los cientos.
* **Re-clasificación:** En la etapa final el sistema puede volver a clasificar para considerar criterios o limitaciones adicionales como pueden ser la actualidad, la diversidad y la equidad. Estos son algunos de los diversos factores que pueden ayudar a mejorar considerablemente las recomendaciones de un sistema. La actualidad vela por que los elementos sean recientes y no generar candidatos antiguos. La diversidad valida que los candidatos sean diversos según las preferencias del usuario, todas las recomendaciones no pueden ser igual a lo que el usuario ve, ya que esto elimina el visibilidad del contenido algo fundamental en este tipo de sistemas. Por último la equidad evita sesgos en las recomendaciones como podrían ser sebos de clicks o falsas tendencias y permite que todos los usuarios sean tratados de manera justa.

Cada una de estas etapas está compuesta por 1 o varios modelos de redes neuronales profundas dependiendo de qué tan complejos sean los datos a utilizar.

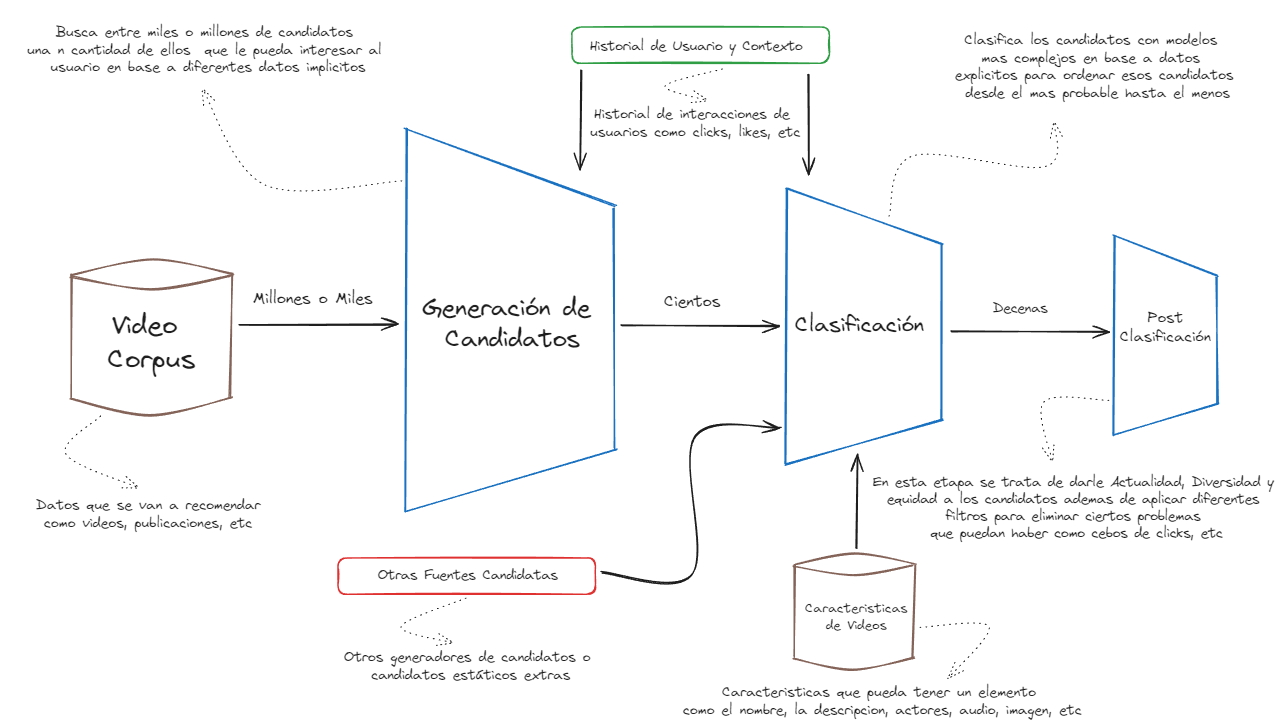


Figura 8: Arquitectura de capas del sistema.

<https://static.googleusercontent.com/media/research.google.com/es/ir/pubs/archive/45530.pdf>

<https://developers.google.com/machine-learning/recommendation?hl=es-419>

<https://www.tensorflow.org/recommenders/examples/quickstart?hl=es-419>

* + 1. **Modelos.**

Existen tres tipos de modelos utilizados en este sistema, Modelos de Dos Torres, Modelo de Recuperación y Modelo de Clasificación:

**Modelo de Dos Torres**: es una arquitectura de red neuronal diseñada específicamente para tareas de aprendizaje de representaciones y, en particular, para problemas de recomendación, búsqueda y clasificación. Su nombre proviene de la estructura visual que adopta: dos "torres" de redes neuronales que procesan independientemente dos tipos de datos, y luego combinan sus representaciones finales para obtener una puntuación o predicción.

En una arquitectura de dos torres, cada torre es una red neuronal que procesa características de entrada candidatas o de consulta para producir una representación integrada de esas características. Debido a que las representaciones de incrustación son simplemente vectores de la misma longitud, podemos calcular el producto escalar entre estos dos vectores para determinar qué tan cerca están. Esto significa que la orientación del espacio de incrustación está determinada por el producto escalar de cada par <query, candidato> en los ejemplos de entrenamiento.

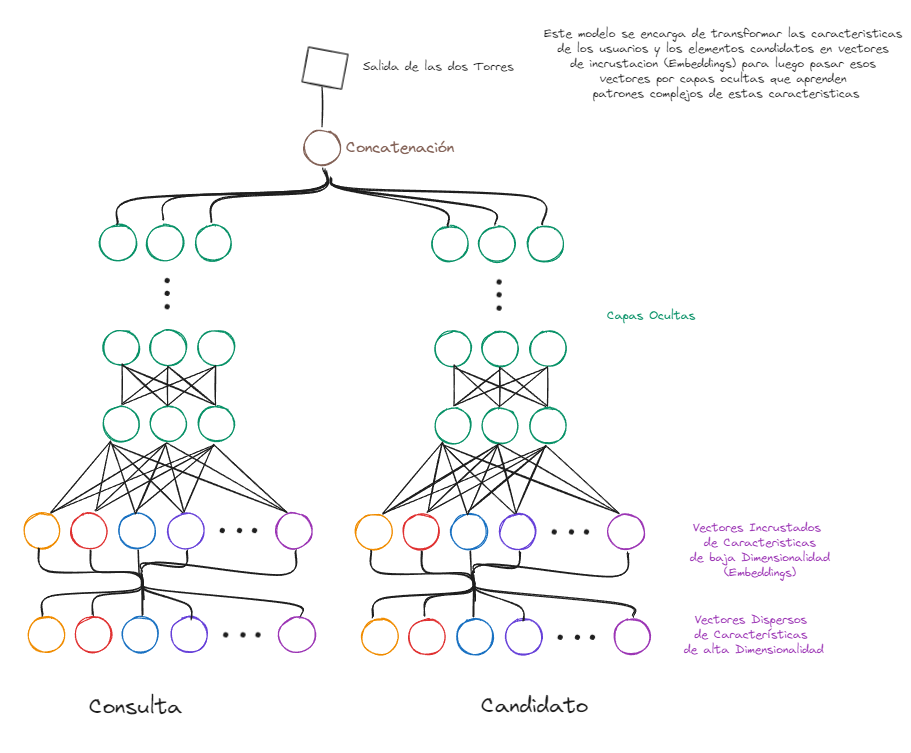


Figura 9: Arquitectura de una red neuronal de Dos Torres.

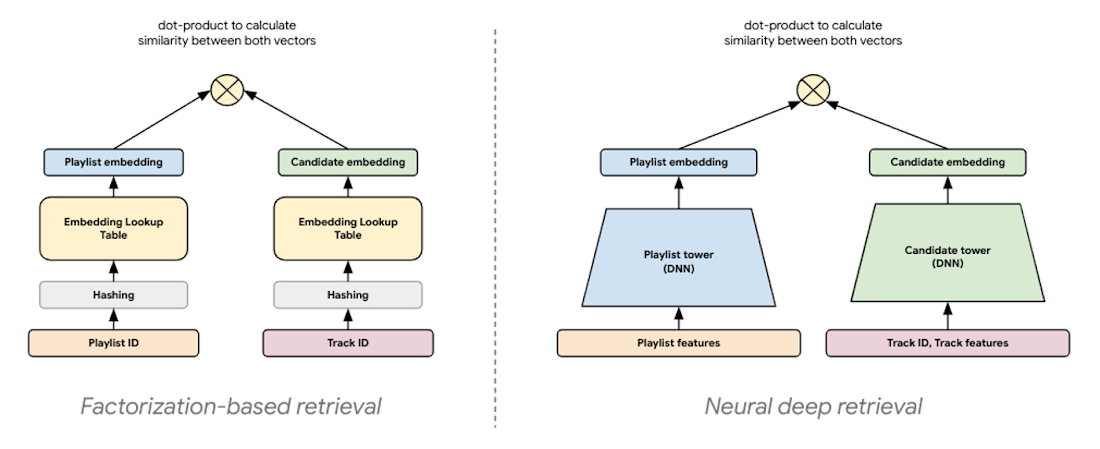


Figura 10: Las arquitecturas NDR (neural deep retrieval), como los codificadores de dos torres, son conceptualmente similares a los modelos de factorización. Ambas son técnicas de recuperación basadas en incrustaciones que calculan representaciones vectoriales de consultas y candidatos de dimensiones inferiores, donde la similitud entre estos dos vectores se determina calculando su producto escalar.

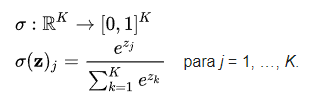
<https://cloud.google.com/blog/products/ai-machine-learning/scaling-deep-retrieval-tensorflow-two-towers-architecture>

**Modelo de Recuperación**: Es el encargado de leer el corpus de datos completo y recuperar unos miles de candidatos más probables a ser de interés para el usuario. Este tipo de modelos es entrenado con datos implícitos del usuario.

* **Representación**: Los modelos de recuperación a menudo se componen de dos sub modelos:
* Un modelo de consulta que calcula la representación de la consulta (normalmente un vector de incorporación de dimensionalidad fija) mediante funciones de consulta.
* Un modelo candidato que calcula la representación candidata (un vector de igual tamaño) utilizando las características candidatas.

Estos dos sub modelos son representados con el modelo de Dos Torres explicado anteriormente.

* **Función de activación de la capa de salida**: Se utiliza una capa de salida con función de activación Softmax <https://es.wikipedia.org/wiki/Funci%C3%B3n_SoftMax> para calcular la probabilidad de que a un usuario le interese cada candidato.



* **Métrica y función de coste**: Para medir la precisión se utiliza una métrica llamada **FactorizedTopK** específica diseñada para evaluar el desempeño de modelos de recomendación factorizados. Esta métrica se enfoca en medir la capacidad del modelo para predecir correctamente los elementos (por ejemplo, productos, películas) más relevantes para un usuario dado en una lista ordenada de recomendaciones.

Recibe la lista de candidatos donde para cada consulta (usuario) se obtiene la lista de candidatos predichos por el modelo y se compara esta lista con la lista real de candidatos seleccionados por el usuario. Luego se calcula la precisión para diferentes valores de k (generalmente 1, 5, 10, 50, 100). La precisión se define como el porcentaje de candidatos predichos que coinciden con los candidatos reales.

* **Servicio**: Este modelo una vez entrenado es explotado para construir un servicio eficiente mediante la construcción de un índice aproximado de vecinos más cercanos (ANN) <https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos> el cual se utiliza en producción para obtener los n candidatos en la etapa de recuperación.

El sistema de recomendación puede tener varios generadores de candidatos que usen diferentes fuentes, como de la siguiente manera:

* Elementos relacionados de un modelo de factorización de matrices.
* Funciones del usuario que tienen en cuenta la personalización.
* “Local” y "distante" items; es decir, que toman información geográfica a tener en cuenta.
* Artículos populares o en tendencia.
* Un gráfico social; es decir, los elementos recomendados por amigos.

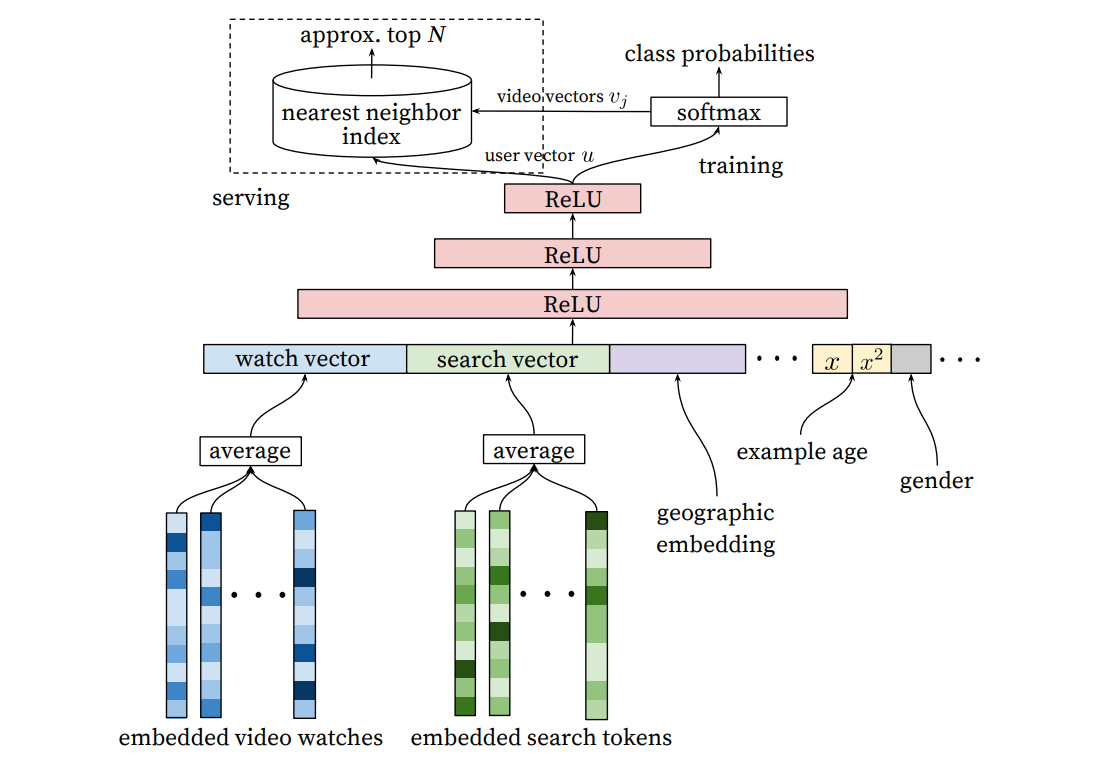


Figura 11: Arquitectura de un Modelo de Recuperación.

**Modelo de Clasificación:** Después de la generación del candidato, otro modelo califica y clasifica los candidatos para seleccionar el conjunto de elementos que se mostrarán.

El sistema combina estas diferentes fuentes en un grupo común de candidatos que se califican con un solo modelo y se clasifican según esa puntuación. Por ejemplo, el sistema puede entrenar un modelo para que prediga probabilidad de que un usuario mire un video en YouTube a partir de los datos explícitos del mismo como podrían ser los likes y teniendo en cuenta los siguientes factores:

Funciones de búsqueda (por ejemplo, historial de reproducciones del usuario, idioma, país y hora).

Elementos de video (por ejemplo, título, etiquetas, incorporación de video, duración).

* **Representación**: Este modelo utiliza la misma representación que el de Recuperación.
* **Función de activación de la capa de salida:** Este modelo puede variar en cuanto a la capa de salida ya que se podrían utilizar diferentes enfoques para clasificar a los candidatos dependiendo del set de datos que se pretenda utilizar.
* **Métrica y función de coste:** puede variar ya que se podrían utilizar diferentes enfoques para clasificar a los candidatos dependiendo del set de datos que se pretenda utilizar.
* **Servicio:** Se utiliza el propio modelo para hacer predicciones sobre los candidatos.



Figura 12: Arquitectura de un Modelo de Clasificación.

<https://www.tensorflow.org/recommenders?hl=es-419>

<https://developers.google.com/machine-learning/recommendation?hl=es-419>

<https://cloud.google.com/blog/products/ai-machine-learning/scaling-deep-retrieval-tensorflow-two-towers-architecture>

<https://www.youtube.com/playlist?list=PLQY2H8rRoyvy2MiyUBz5RWZr5MPFkV3qz>

<https://static.googleusercontent.com/media/research.google.com/es//pubs/archive/45530.pdf>

**Capitulo 2**

Requisitos Funcionales

1. Entrenar el sistema
2. Actualizar el sistema
3. Utilizar sistema
4. Mostrar sistema
5. Listar sistemas
6. Entrenar modelo
7. Actualizar modelo
8. Eliminar modelo
9. Listar modelos
10. Utilizar modelo
11. Crear etapa
12. Actualizar etapa
13. Eliminar etapa
14. Mostrar etapa
15. Listar etapas
16. Crear Configuraciones
17. Actualizar Configuraciones
18. Mostrar Configuraciones
19. Listar Configuraciones

Requisitos no funcionales

Usabilidad

El sistema para la gestión y aseguramiento material para la ONBC debe ser una aplicación web.

La aplicación debe presentar una interfaz agradable e intuitiva.

Confiabilidad

La información manejada por el sistema está protegida de acceso no autorizado de usuarios, definiéndose los permisos según sus roles.

Eficiencia

El sistema debe permitir que los usuarios interactúen con él de manera concurrente.

El tiempo de demora de una petición al servidor debe ser menor de cinco (5) segundos aproximadamente

Restricciones de Implementación y Diseño

El sistema debe ser desarrollado en su totalidad con tecnologías de código abierto

Software

Para el uso del sistema se requiere una PC cliente con cualquier sistema operativo, que se pueda instalar navegadores web para el uso de la aplicación.

La comunicación entre la PC cliente y el servidor de aplicaciones web se realiza a través del protocolo HTTPS.

Hardware

Teniendo en cuenta que este tipo de proyectos basados en redes neuronales profundas y análisis de grandes conjuntos de datos necesita de una muy buena potencia de cómputo para echar a andar de la cual no se dispone en el momento de implementación y prueba del mismo el autor ha establecido unos requisitos mínimos con los que cuenta para correr este sistema bajo ciertas condiciones

Se requiere un minimo de 80gb de disco duro, una tarjeta de red de 100MB, un procesador Core i3-4170 a 3.70GHz y 16GB de memoria RAM. Se requieren estos requisitos para un modelo de x parámetros, un corpus de entrenamiento de x filas y z columnas.