Equidad en sistemas de recomendación de preguntas basados en representaciones de texto

Tesis de Grado de

Ingeniería en Informática

Director: Dr. Ing. Mariano G. Beiró

Co-director: Dr. Ing. J. Ignacio Alvarez-Hamelin

Alumno: Francisco López Destain *(Padrón #94.180)*

fdestain@fi.uba.ar

Facultad de Ingeniería, Universidad de Buenos Aires

Índice general

Capítulo 1

Introducción

En esta tesis estudiamos el problema de la equidad en sistemas de recomendación basados en representaciones de texto, y analizamos las posibilidades de mejorarla. Para ello, tomamos como caso de estudio la plataforma Stack Overflow, sobre la que construimos un sistema de recomendación de preguntas para los usuarios. Nuestro sistema de recomendación estará basado en un modelo de aprendizaje estadístico que utiliza como caracteristicas (*features*) de entrada, representaciones de texto de las preguntas. Con el sistema entrenado, analizaremos y propondremos mejoras a la equidad regional de los modelos, fomentando así las contribuciones de usuarios de países y regiones con distintos niveles de acceso a la tecnología.

En este capítulo introductorio presentamos las motivaciones del problema y las herramientas que utilizaremos durante el trabajo. Se describirá brevemente *StackOverflow*, luego se hará una introducción a los sistemas de recomendación y a cómo analizar su equidad, y por último se dará un panorama general de la estructura del resto del manuscrito.

1.1. Stack Overflow

Stack Exchange es una red de sitios de preguntas y respuestas sobre distintas áreas del conocimiento; en particular Stack Overflow se enfoca en temas de programación. La plataforma

ganó mucha popularidad entre desarrolladores, ya que permite resolver problemas a través del conocimiento agregado de la comunidad que trabaja en cada tecnología.

Las respuestas a las preguntas generadas por los usuarios sirven como referencia rápida en el trabajo diario de miles de desarrolladores, ya que las dudas y problemas se repiten y pueden ser aprovechadas por varios usuarios. Redactar las respuestas ayuda también a ganar experiencia en las tecnologías y fijar conocimientos.

Los usuarios, las preguntas y las respuestas forman parte de un sistema de reputación que ayuda a mantener la calidad de la plataforma. Por ejemplo, es posible identificar a los mejores contribuidores y a las preguntas mas recurrentes en un determinado tema. El sistema de reputación también permite que la plataforma sea automoderada. En el Capítulo 3 se describe la estructura del modelo de datos que utilizaremos para estudiar la plataforma.

Por último queremos notar que en los últimos años la proliferación de los *Large Language Models*, permitieron a los desarrolladores hacer preguntas a modelos públicos, como ChatGPT, obteniendo respuestas en segundos e incluyendo código específico para el proyecto en el que trabajan. Esto se manifestó en una caída en la participación de usuario en StackOverflow. Igualmente la plataforma sigue siendo uno de los principales sitios de preguntas y respuestas de tecnología, y en estos momentos se encuentra en proceso de integrarse con las funcionalidades de ChatGPT.

1.2. Equidad (*fairness*) en sistemas de recomendación

La abundancia de contenido en las nuevas plataformas hace necesarias herramientas para explorar y filtrar la información. Los sistemas de recomendación permiten en este sentido que los usuarios accedan a nuevos productos, películas, libros, música –*items* en general– de su interés, entre millones de opciones. Basándose en el comportamiento de consumo del usuario, o en usuarios similares, estos sistemas proponen nuevas recomendaciones que puedan resultar relevantes.

En el caso de Stack Overflow hay miles de preguntas nuevas a diario. Los usuarios interesados

en colaborar con la plataforma o en mejorar su reputación pueden buscar preguntas a responder a través de etiquetas, palabras clave y diferentes filtros. Para facilitarles la búsqueda a estos usuarios y guiarlos, es útil un sistema de recomendación de preguntas que tenga en cuenta sus posibilidades de responderlas. A su vez, sería importante garantizar el acceso a preguntas desafiantes que estén al alcance de sus conocimientos, para ayudarlos en su aprendizaje y desarrollo en nuevas tecnologías.

Muchos sistemas de recomendación están basados en sistemas predictivos, que se han popularizado a partir del crecimiento de la industria de los datos y del aprendizaje estadístico. El uso de estos sistemas automatizados para tomar decisiones o desarrollar negocios, trajo aparejado nuevos problemas: como estos modelos aprenden de información previa, podrían amplificar sesgos que ya estaban presentes en los datos, generando o potenciando situaciones de discriminación.

Ante este problema, en los últimos años tomó importancia el concepto de equidad –o *fairness* en inglés– en los sistemas automatizados. Parte de esta área estudia los sesgos en los diferentes modelos de aprendizaje automático y propone mecanismos para corregirlos. Estos sesgos pueden ser en algunos casos bastante evidentes: si un modelo predice una salida utilizando el género, edad o región de los usuarios, podría resultar sesgado y privilegiar a un grupo de usuarios sobre otro (por ejemplo, podría tener distintas tasas de error según el género). Pero también los sesgos pueden ser menos aparentes: aunque no se incluyan atributos sensibles, puede haber correlaciones entre ellos y otros atributos presentes, y de esta forma el modelo podría acceder a dichos atributos sensibles en forma indirecta.

En el caso particular de los sistemas de recomendación, el análisis de la equidad se vuelve crítico, ya que estos sistemas filtran información y determinan qué usuarios tienen acceso a cada contenido. En el caso de StackOverflow podría recomendar las preguntas a personas que ya tengan buenas respuestas para preguntas similares, poniendo obstáculos a usuarios que buscan aprender. Al considerar usuarios similares para las recomendaciones, una región puede acaparar preguntas en ciertas tecnologías, y crear también una barrera de entrada para usuarios de otras regiones que quieran incorporarse. Este fenómeno de refuerzo de patrones preexistentes puede

darse también en otras redes sociales, en el que pocos usuarios podrían acaparar la voz o inundar la plataforma con sus ideas, dejando de lado a voces minoritarias. Esto nos plantea una alerta respecto al funcionamiento de los algoritmos de recomendación que orquestan plataformas de uso masivo.

Para ejemplicar las consecuencias nocivas del uso no regulado de sistemas predictivos, mencionaremos el caso paradigmático de uso del sistema COMPAS para asistir al poder judicial en Estados Unidos. COMPAS se encargaba de predecir la probabilidad de reincidencia de criminales y, cómo se descubrió en [Larson and Kirchner, 2016] a partir de un estudio de ProPublica, tenía una tasa de falsos positivos mucho más alta para afroamericanos que para la población blanca, lo cual resultaba discriminatorio dado que una persona de origen afroamericano tenía más probabilidades de ser perjudicada erroneamente por el sistema con respecto a una persona blanca. Este comportamiento discriminatorio se debió en parte a haber puesto el foco sólo en el desempeño del sistema (en términos de porcentaje de casos correctamente clasificados) sin haber considerado la equidad del mismo durante su entrenamiento.

La equidad es un problema multicausal, la inequidad preexistente debida a desigualdades sociales puede verse amplificada por factores como la forma de interacción de los usuarios con los sistemas. En nuestro caso la inequidad debida a la barrera de acceso a la tecnología puede verse afectada por la frecuencia de interacción de los usuario con la plataforma. O sea que tener previamente pocas oportunidades para responder puede terminar en un ciclo negativo como se ve en [Li et al., 2022] donde muestra que los sistemas de recomendación tienen peor desempeño en usuarios inactivos. Esto puede ser un problema para los usuarios con alta barrera de acceso a la tecnología, ya que todavía no responden activamente preguntas en alguna etiqueta en la que quisieran comenzar a participar.

Para el estudio del concepto de equidad seguiremos la metodología presentada en [Barocas et al., 2023], que formaliza el concepto de atributo sensible, y distintas métricas y nociones estadísticas de equidad. El libro hace una crítica de la idea de equidad por desconocimiento (*fairness through unawareness*), que sugiere no incluir al atributo sensible para no generar sesgos. En cambio, se

plantean tres nociones generales de equidad que surgen de definiciones probabilísticas de las variables involucradas:

- Independencia: la predicción es independiente de los atributos sensibles (está vinculada con la métrica de equidad conocida como paridad demográfica).
- Separación: condicionada al valor real, la predicción realizada es independiente de los atributos sensibles (es conocida también como igualdad de oportunidades, porque implica iguales tasas de error para todos los grupos [Hardt et al., 2016]).
- Suficiencia: condicionado a la predicción, el valor real a predecir es independiente de los atributos sensibles (conocido también como calibración [Pleiss et al., 2017]).

1.3. Etapas del trabajo

En esta tesis construiremos un sistema de recomendación de preguntas, compuesto por un modelo predictivo basado en la representación del texto de las mismas. Para ello mediremos la cercanía entre las preguntas y los usuarios que las respondieron, a fin de recomendar posibles usuarios candidatos a responder una nueva pregunta. El objetivo será proponer candidatos que tengan la capacidad de responderlas.

El desempeño del modelo se evaluará contrastando los *rankings* de usuarios recomendados generados por el modelo contra el listado de aquellos que efectivamente respondieron la pregunta en la plataforma. El conjunto de datos se dividirá en un conjunto de entrenamiento y otro de pruebas. Realizaremos para ello una división temporal, de manera de no utilizar información futura en la predicción y garantizar así que el desempeño medido sea generalizable a futuras preguntas.

En el Capítulo 2 introducimos el estado del arte en lo relativo a representaciones de texto, el problema de *learning to rank* y el concepto de *fairness* (equidad), y las herramientas que utilizaremos en el trabajo.

En el Capítulo 3 revisamos la distribución de las diferentes dimensiones relevantes para el análisis

de los datos. También describimos el preprocesamiento realizado para estandarizar el nombre de los países a los que pertenecen los usuarios. Finalmente presentamos la partición de los datos en un conjunto de entrenamiento y un conjunto de pruebas para medir el desempeño y la equidad de los modelos.

En el Capítulo 4 explicamos cómo se entrenaron las diferentes representaciones y modelos para poner en funcionamiento el sistema. Además, analizaremos la equidad del sistema propuesto con respecto a las regiones de las que provienen los usuarios, y propondremos modificaciones para mejorar la equidad en el sistema, analizando cómo esto afecta su desempeño general. Por último proponemos algunas modificaciones al algoritmo para corrección de la equidad, que mejoran su comportamiento global.

Capítulo 2

Estado del arte

En la primera parte de este capítulo presentaremos las técnicas más comunes utilizadas hoy en día en sistemas de recomendación. Dado que una práctica común en las implementaciones es dividir el *pipeline* en dos etapas –una de selección de candidatos y otra posterior de *ranking* u ordenamiento–, comenzaremos el capítulo describiendo la etapa de selección. Luego, nos enfocaremos en los tipos de modelos para *ranking* y haremos una breve explicación de aquellos más relevantes. La última sección del capitulo estará dedicada al problema de equidad en sistemas de recomendación, y a la descripción de un algoritmo específico para corregirla.

La práctica de dividir a los sistemas de recomendación en dos etapas ha sido documentada por distintas empresas como Google [Covington et al., 2016], Alibaba [Feng et al., 2020, Wang et al., 2018], JD [Zhang et al., 2020] y Meta [Huang et al., 2020]. En la Figura 2.1 vemos un esquema de dichas etapas. La primer etapa busca filtrar de la base total de candidatos a una cantidad representativa para resolver la consulta que se está haciendo, buscando un alto *recall*. La segunda etapa trata de ordenar a los candidatos por algún *ranking* de importancia según la búsqueda. Dado que esta etapa devolverá a una cantidad k de candidatos más relevantes para la recomendación, se busca que tenga una alta precisión.

Los flujos mostrados en color verde en la figura 2.1 representan procesamientos realizados *offline*: no ocurren durante la resolución de una consulta específica en busca de recomendaciones, sino

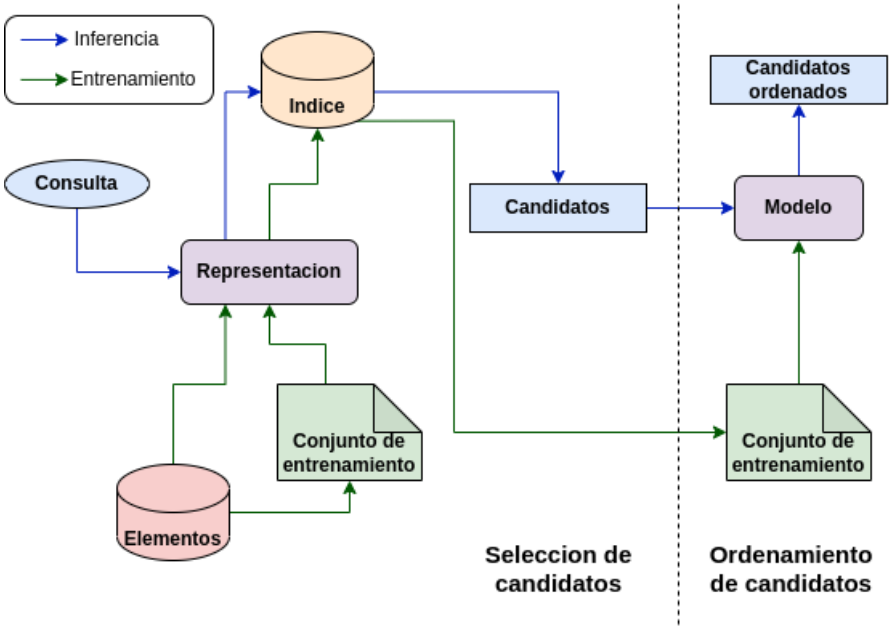


Figura 2.1: Etapas en un sistema de recomendación

que tienen lugar durante el entrenamiento del modelo. En esta etapa ocurre el entrenamiento de las representaciones para la selección de los candidatos, su carga en la base de datos en la que serán buscados, y el entrenamiento del *ranker*. Los flujos en color azul representan la resolución de una consulta: en este proceso se obtiene la representación de la consulta, con ella se selecciona un conjunto de candidatos que se envían al *ranker*, y de esta manera se obtiene una lista de resultados ordenada por relevancia.

2.1. Selección de candidatos

Para seleccionar a los candidatos buscaremos representaciones vectoriales de las preguntas. Una vez transformadas las preguntas en vectores, las indexaremos de manera que al buscar una nueva pregunta, la selección de candidatos se haga obteniendo el nuevo vector y extrayendo los k vectores cercanos, que representarán preguntas previamente indexadas. Para obtener estos vectores de forma rápida usaremos la biblioteca Faiss de Facebook [Douze et al., 2024], que esta preparada para hacer búsquedas de vecinos aproximadas de forma eficiente en memoria.

A continuación describiremos dos mecanismos de entre los más modernos para obtener representaciones vectoriales de texto que evaluamos en nuestro modelo: Doc2Vec y SentenceBERT.

2.1.1. Doc2Vec

Doc2Vec [Le and Mikolov, 2014] es un modelo que permite crear representaciones vectoriales de documentos de texto de forma no supervisada. Está basado en la idea de Word2Vec [Mikolov et al., 2013a, Mikolov et al., 2013b].

Word2Vec busca resolver el problema de modelado de lenguaje, donde dado un conjunto de palabras se intenta predecir cuál es la siguiente palabra más probable. Para hacer esto, el modelo de Word2Vec organiza los datos de entrenamiento con una ventana deslizante que se desplaza sobre los documentos del corpus; en cada iteración obtiene una palabra y su contexto. Tanto el contexto como la palabra se representan con un *one-hot-encoding*, un vector esparso en donde se le asigna un índice a cada palabra del vocabulario seleccionado. De esta forma, las palabras y el contexto representados tienen un valor 1 en los índices que corresponden a ellas, y un valor 0 en las restantes posiciones del vector.

Word2Vec se basa en la hipótesis distribucional [Firth, 1968, Harris, 1954], que dice que "elementos lingüísticos con distribuciones similares tienen significados similares", de esta manera palabras que compartan el mismo contexto deberían tener representaciones similares. Con el conjunto de datos armado, hay dos formas de aprender las representaciones:

Continuous bag-of-words model (CBOW) : trata de predecir la palabra central basándose en el contexto.

Skip-gram model busca predecir palabras del contexto a partir en la palabra central.

El modelo de Word2Vec está compuesto de una red neuronal de dos capas. En la figura 2.2 se puede ver la arquitectura correspondiente al modelo CBOW: como la palabras de entrada son un *one hot encode*, la multiplicación vectorial de la primera capa de la red neuronal va a devolver una columna de la matriz de pesos $\mathbf{W}$, por esto se la suele usar como una tabla de *lookup*. La

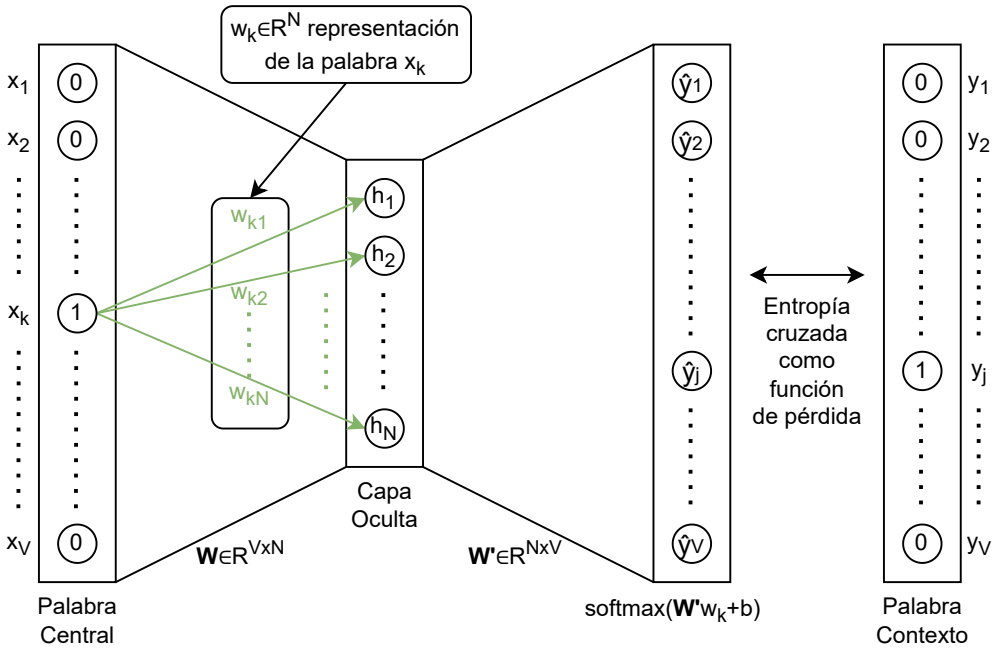


Figura 2.2: *Framework* para aprender representaciones de palabras en Word2Vec.

representación de la palabra se pasa por la segunda capa de la red neuronal, en donde se se obtiene la predicción para la palabra de contexto que se quería predecir, se compara el resultado con el *one hot encode* de la palabra esperada y usando entropía cruzada como función de pérdida se aplica *backpropagation* para actualizar los pesos. Como las palabras que aparecen en contextos parecidos van a tener los mismos *targets*, sus representaciones tienden a ser parecidas de forma de producir predicciones semejantes al pasar por la segunda capa, respetando así la hipótesis distribucional planteada anteriormente. Entonces si el modelo converge, la representación –como se buscaba– mantiene cerca las palabras que son semánticamente similares.

Word2Vec permite representan palabras individuales, pero para obtener la representación de un extracto de texto, o sea un conjunto ordenado de palabras de tamaño variable, se necesita un nuevo modelo. En la Figura 2.3 vemos como Doc2Vec extiende el modelo de Word2Vec –CBOW en este caso– La idea es agregar a la representación de entrada un vector para los documentos de entrenamiento que se entrenará en conjunto. Al terminar el entrenamiento se obtienen los vectores de las palabras para el vocabulario de los textos y la representación de los documentos

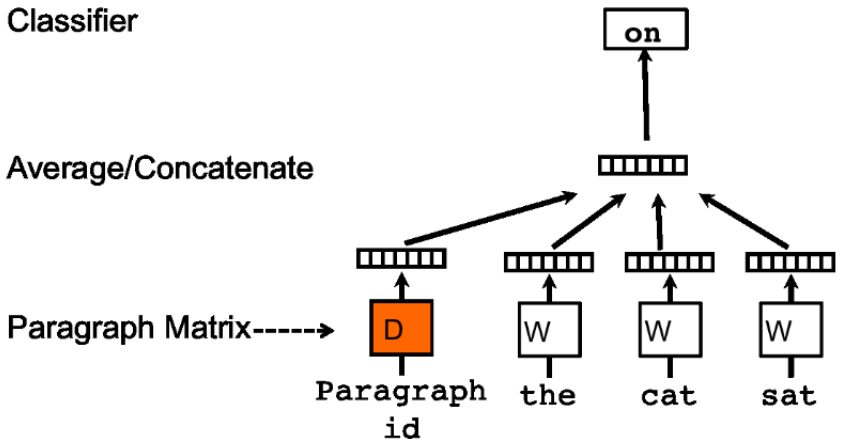


Figura 2.3: *Framework* para aprender representaciones de documentos en Doc2Vec para el modelo *Distributed Memory Model of Paragraph Vectors*. Figura extraída de [Le and Mikolov, 2014].

de entrenamientos.

El modelo anterior se denomina *Distributed Memory Model of Paragraph Vectors*, pero si se extiende la idea del *Skip-gram model* se llega al modelo *Distributed bag of words*, que se puede ver en la figura 2.4 en donde se usará solamente la representación de los documentos para predecir palabras al azar contenidas en los textos.

Cuando luego se busque la representación de un nuevo documento en el modelo ya entrenado –en tiempo de inferencia– se dejarán fijas las representaciones de las palabras y las últimas capas de la red neuronal, y se ajustarán pesos para el nuevo documento.

2.1.2. Sentence Bert

Otra forma de representar un extracto de texto es usando Sentence-BERT(SBERT) [Reimers and Gurevych, 2
SBERT se basa en la arquitectura de transformers [Vaswani et al., 2017], un modelo de secuencia a secuencia que permite incorporar información de contexto a cada palabra usando un mecanismo conocido como "atención".

La Figura 2.5 muestra como funciona el mecanismo de atención. En un primer paso, los vectores

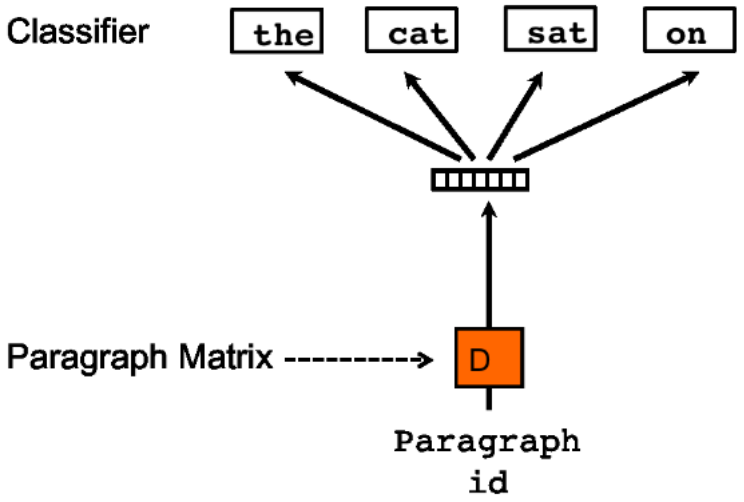


Figura 2.4: *Framework* para aprender representaciones de documentos en Doc2Vec para el modelo *Distributed bag of words*. Figura extraída de [Le and Mikolov, 2014].

de entrada *query, key and value* (Q, K, V) pasan por una red neuronal de una sola capa con activación lineal. Luego, los vectores resultantes de esta capa pasan por un *scaled dot-product attention*, en donde se obtiene una matriz de *scores* multiplicando las *queries* con las *keys*. Se escala esta matriz de *scores* para facilitar el flujo de los gradiente durante la etapa de *backpropagation*, y se aplica una *softmax* por fila para obtener los pesos de atención. Finalmente, se multiplica la matriz de valores por los pesos de atención para obtener los vectores de salida. Opcionalmente puede haber un enmascaramiento de la matriz de *scores*, previo al *softmax*, que consta de poner un símbolo de $-$ ínf en los valores a los que no hay que prestar atención, por ejemplo si no quisiéramos atender a valores futuros. Todo este procedimiento conforma una cabeza de atención; varias cabezas en paralelo –cada una con sus propios parámetros– son concatenadas y entran a una última red neuronal de una sola capa con activación lineal, para obtener la representación final de las entradas.

En la Figura 2.6 vemos que la arquitectura general de los *tranformers* está compuesta por un *encoder-decoder* con diferentes tipos de atención. Estos últimos se pueden categorizar en:

Encoder-decoder attention. En este caso la *query* viene del *decoder* y la *key* y *value* del

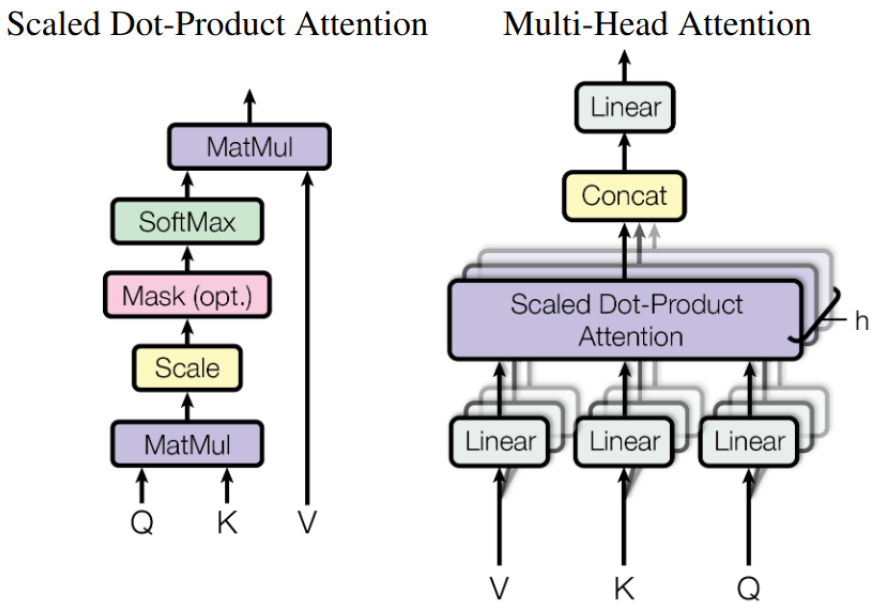


Figura 2.5: Mecanismos de atención en *transformers*, *Scaled Dot-Product Attention* a la izquierda y *Multi-Head Attention* a la derecha. Figura extraída de [Vaswani et al., 2017].

encoder. Esto permite al *decoder* tener atención sobre la secuencia de entrada.

Self-attention . Aquí *keys*, *values* y *queries* provienen de los mismos vectores de entrada, es esta manera se agrega la información de contexto a cada palabra.

Masked self-attention. Al igual que en self-attention *keys*, *values* y *queries* provienen de los mismos vectores, pero se enmascara la matriz de *scores* previo al cálculo de atención para evitar el flujo de información desde palabras futuras.

Siguiendo el diagrama de la Figura 2.6 las palabras se transforman en *embeddings* en la entrada, y se les agrega el *positional encoding* antes de ingresar al *encoder*. Los *positional encodings* permiten incorporar información del orden en que se encuentran las palabras, esto junto con los mecanismos de atención permite a los *transformers* entrenar en paralelo. De esta manera superan en velocidad de entrenamiento a las representaciones con redes neuronales recurrentes, que requieren ser desenrolladas para entrenar.

Los *embeddings* de las palabras de entrada sumados a los *positional encodings* ingresan al *encoder*.

Pasan por una capa de *multi-head attention* en donde los mismos vectores entran como Q, K y V. Luego, una conexión residual suma la salida de la *multi-head attention* con la entrada y se hace una *layer normalization*. A continuación se pasa por una *Position-wise Feed-Forward Network* –una red neuronal de dos capas con activacion ReLU– y finalmente esto se suma con otra conexión residual haciendo una nueva *layer normalization*. Toda esta descripción constituye un bloque del *encoder*. Los encoders se puede apilar N veces, conectando la salida de uno a la entrada del siguiente. A la salida del *encoder* habrá un vector por cada palabra de entrada, que ahora contendrá la información del contexto en que estaba.

El *decoder* en su entrada toma la última palabra de salida, agrega *embeddings posicionales*, pasa por una capa de *masked multi-head attention* para no contextualizar con palabras futuras, al resultado de esto último se le suma una conexión residual, y se aplica *layer normalization*. Estos vectores pasan por una capa de *encoder-decoder attention* entrando a Q, y usando K y V desde la salida del *encoder*. A este resultado se le suma otra conexión residual y se aplica *layer normalization*. Finalmente se pasa por una *Position-wise Feed-Forward Network* y se suma con otra conexión residual, con una última *layer normalization*, y se pasa por una capa lineal con una *softmax* que determina el token de salida.

Partiendo de la arquitectura de *transformers* surgieron diferentes modelos como GPT, XLM o BERT. BERT *Bidirectional Encoder Representations from Transformers* [Devlin et al., 2018] está compuesto por una pila de *encoders*. Como se ve en la Figura 2.7, el mismo se entrena en dos etapas: una primera etapa de pre-entrenamiento, intensivo en tiempo y computación, y una segunda etapa de *fine-tuning* para tareas especificas, que se puede entrenar más rápidamente aprovechando los parámetros aprendidos previamente.

El preentrenamiento se realiza con dos tareas a la vez: *Masked Language Model* y *Next-Sentence Prediction*. Para *Masked Language Model* se enmascaran ciertas palabras que van a constituir el objetivo de la predicción. Se enmascara un 15 % de las palabras, de las cuales para algunas se usa el token *[MASK]*, para otras se utiliza el token de una palabra al azar, y en otras simplemente se deja la misma palabra. Para *Next-Sentence Prediction* se busca predecir si un extracto de

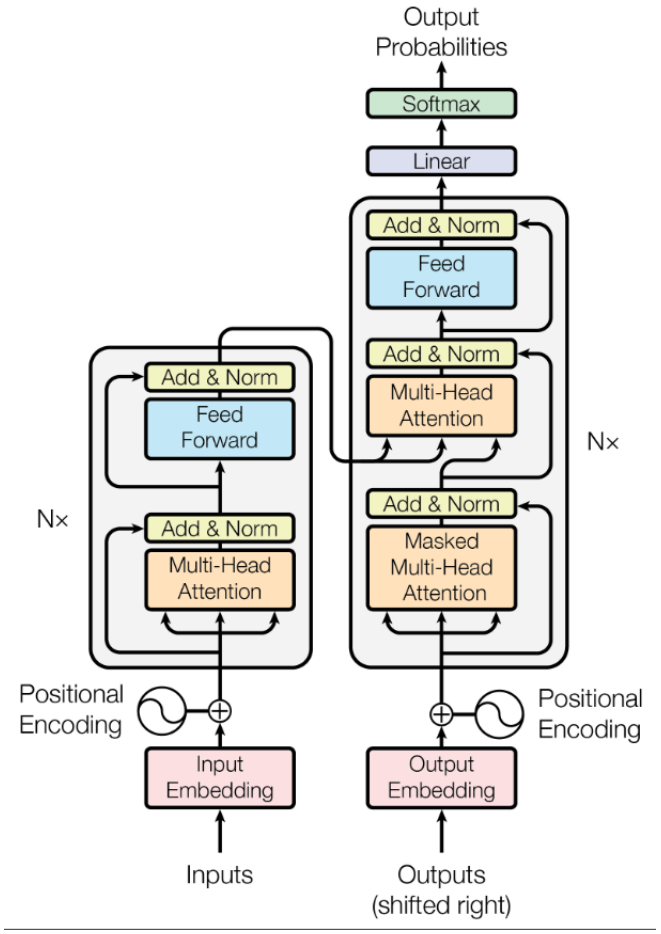


Figura 2.6: Arquitectura del modelo *transformers*. Figura extraída de [Vaswani et al., 2017].

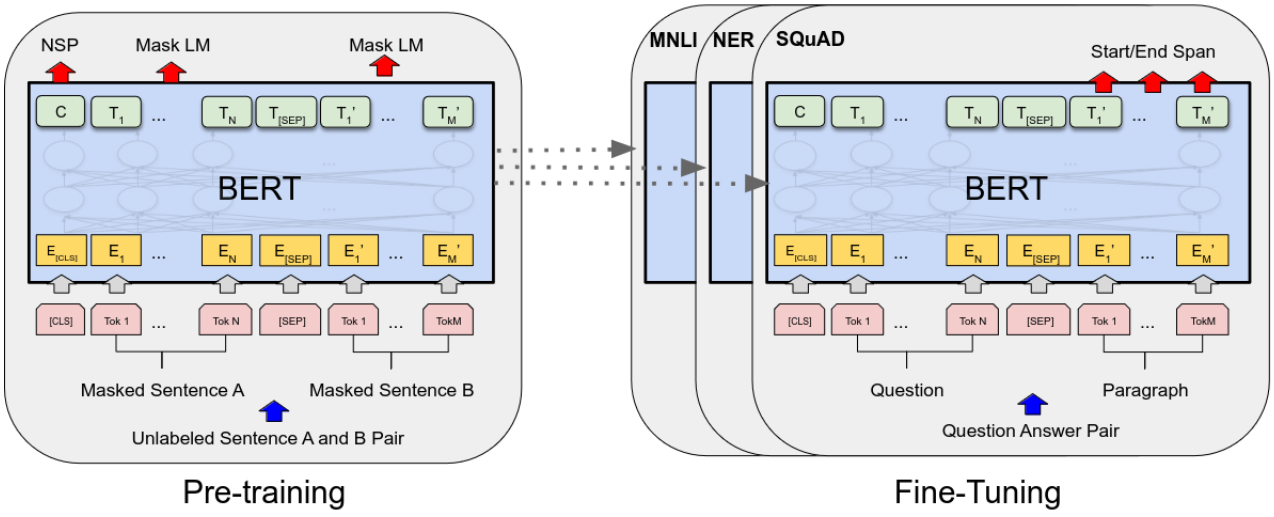


Figura 2.7: Procedimiento general para pre-train y fine-tuning de BERT. Ambas etapas usan la misma arquitectura cambiando las capas de salida. Los parámetros del modelo preentrenado se usan como valores iniciales para los modelos de las tareas especificas, durante el fine-tuning todos los parámetros son ajustados. [CLS] y [SEP] son tokens especiales, el primero se pasa en cada ejemplo y se usa para las tareas de clasificacion, el segundo sirve para separar las frases de entrada. Figura extraída de [Devlin et al., 2018].

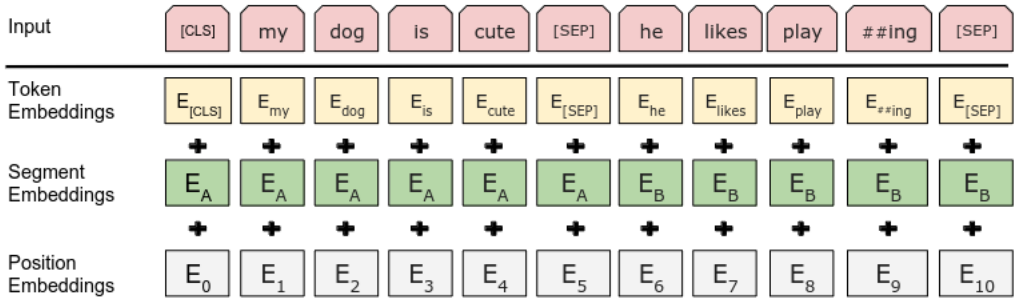


Figura 2.8: Representación de entrada BERT. Los *embeddings* de entrada son la suma de los *embeddings* de token, que representan a cada palabra de entrada, los *embeddings* de segmento, que representan a cada frase de entrada, y los *embeddings* de posición, que representan a cada posición en la frase de entrada. Figura extraída de [Devlin et al., 2018].

texto sigue a otro o no; para entrenarlo se toman 50 % de extractos contiguos y 50 % al azar.

Las entradas de BERT se pueden ver en la Figura 2.8, en donde los *embeddings* de entrada se componen de *embeddings* para los tokens, para los segmentos y para las posiciones. El token *[CLS]* siempre va en la primera posicion, y es el que vamos a usar para el resultado de la predicción hecha por *Next-Sentence Prediction*. El token *[SEP]* sirve para diferenciar en donde empiezan y terminan los segmentos de texto que estamos comparando. Además, se agregan los *embeddings* de segmento que permiten incorporar información del segmento al que pertenece cada palabra.

Una vez disponibles las salidas del último *encoder*, se conecta el estado final del token de entrada *[CLS]* con una red neuronal con una *softmax* para predecir si los segmentos son contiguos o no. De la misma manera el estado final de las palabras enmascaradas se conecta con una red neuronal con una *softmax* para predecir la palabra correcta.

Para hacer *fine-tuning* del modelo se agregan capas al final del último *encoder*, y se ajustan todos los parámetros *end-to-end*, aprovechando los parámetros pre-entrenados para ahorrar recursos, costos y tiempo.

BERT es funcional para hacer clasificaciones y regresiones; también se puede usar para comparar

la similaridad semántica entre dos frases. Si quisiéramos hacer una búsqueda semántica, deberíamos combinar de a pares el segmento del texto de consulta con cada uno de los segmentos en nuestro corpus, para poder compararlos y encontrar los más parecidos. Esto hace que el costo de buscar similaridades con BERT sea muy alto por la cantidad de pasadas de inferencia necesarias. *Sentence BERT* (SBERT) [Reimers and Gurevych, 2019] resuelve esto obteniendo una representación para cada segmento de texto que luego puede ser comparada usando distancia coseno.

En la Figura 2.9 se puede ver la arquitectura de SBERT para hacer el fine-tuning: los dos segmentos de entrada pasan por un modelo BERT que comparte sus parámetros, luego pasan por una capa de *pooling* para obtener una representación de cada segmento. El paper que introduce este modelo ofrece distintos esquemas de *pooling*: usar el estado final del token *[CLS]*, o usar *MEAN* o *MAX* sobre los estados finales de los *embeddings* de los tokens.

Para entrenar los pesos, los autores propusieron tres estructuras y funciones objetivo diferentes:

Classification Objective Function. Se concatenan las representaciones de los segmentos junto con su diferencia; luego se pasa por una red neuronal con *softmax* y se usa entropía cruzada como función de pérdida.

Regression Objective Function. Se toma la distancia coseno entre las representaciones de los dos segmentos, y se usa error cuadrático medio como función de pérdida.

Triplet Objective Function. Dada un segmento *anchor* (a), un segmento positivo (p) y uno negativo (n), la red busca que la distancia entre a y p sea menor que entre a y n.

Una vez que SBERT está entrenado se puede usar para obtener las representaciones de segmentos de textos e indexarlas; con esto se puede buscar similaridad semántica entre segmentos nuevos haciendo una búsqueda de vecinos cercanos en el índice.

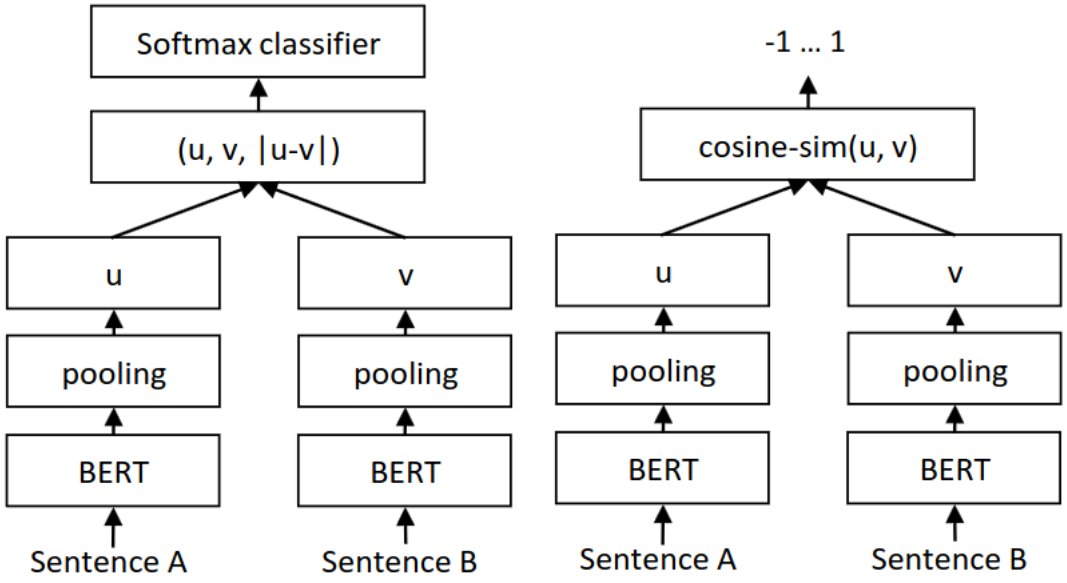


Figura 2.9: Arquitecturas propuestas en [Reimers and Gurevych, 2019] para *fine-tuning* de SBERT. A la izquierda una arquitectura de SBERT con función objetivo de clasificación para, por ejemplo, el *fine-tuning* para la tarea de Stanford Natural Language Inference (SNLI). A la derecha una arquitectura SBERT con función objetivo de regresión para, por ejemplo, calcular puntuaciones de similitud. En ambos casos las dos redes BERT para cada frase tienen los mismos pesos atados (estructura de red siamesa). Figura extraída de [Reimers and Gurevych, 2019].

2.2. Learning to rank

Siguiendo con la resolución de la consulta una vez que los candidatos fueron preseleccionados, corresponde ordenarlos para presentar primero aquellos más relevantes como resultado. Para resolver este ordenamiento de los candidatos vamos a utilizar modelos de *learning to rank*. *Learning to rank* es un conjunto de técnicas para construir modelos que se encargan de la creación o agregación de *rankings* en problemas de aprendizaje automático. Para quien desee profundizar en el uso de modelos de *learning to rank* en el contexto de recuperación de información y NLP, recomendamos el libro de [Li, 2022], en el que está basada la presente sección.

El problema de creación de *ranking* consiste en, dado un subconjunto de los candidatos totales y una consulta en particular, encontrar una función que asigne un puntaje a cada candidato de manera que los mismos se puedan ordenar por relevancia. La tarea de aprendizaje consiste en usar *features* de las consultas y de los candidatos, junto con ejemplos de consultas y candidatos esperados, y entrenar parámetros para obtener la función de puntaje. La consulta en nuestro caso se correspondería con las preguntas hechas en la plataforma StackOverflow, y los candidatos con los usuarios a recomendar para responderlas.

Para entrenar los modelos se usa un conjunto de datos dividido en subconjuntos de entrenamiento y prueba, en donde cada muestra de un conjunto esta compuesta de una consulta y de un conjunto de candidatos junto con la relevancia de cada uno. La relevancia se puede expresar de diferentes maneras, por ejemplo usando etiquetas ordenadas. En nuestro caso el *score* de las respuestas, que StackOverflow mantiene a partir de votaciones que los usuarios realizan, puede resultar útil como proxy del desempeño de un usuario al responder una pregunta, y por lo tanto lo utilizamos como métrica de la relevancia de ese usuario como candidato.

En este esquema, es necesario contar con una función que transforme una consulta y una lista de candidatos en *features*. Esa función puede usar características directas de la consulta y de los candidatos o bien, como en nuestro caso, ser una representación aprendida. Utilizaremos entonces las representaciones de texto vistas en el capitulo 2.1 para transformar las preguntas

en *features*, y los usuarios serán representados en base al texto de las respuestas que han dado.

2.2.1. Métricas

Para evaluar el desempeño de los modelos se necesitan métricas que permitan comparar los resultados; estas métricas comparan listas de *rankings* de la salida de los modelos contra la evidencia (*ground truth*) del conjunto de datos. A continuación describiremos tres de las métricas más utilizadas:

Discounted Cumulative Gain (DCG): Mide la bondad de un *ranking* en base a un conjunto de relevancia, y se puede plantear como:

$$DCG(k) = \sum_{j=1}^{k} G(j) \cdot D(j)$$

En donde $G(j)$ es la función de ganancia, que corresponde al premio que recibimos por el candidato que rankeamos en la posición j, y $D(j)$ es la función de descuento por posición, que hace que posiciones más alejadas del ranking sean menos relevantes. De esta forma, DCG representa la ganancia acumulada de acceder a la información desde el principio hasta la posición k con descuentos al avanzar en las posiciones.

Para poder conmensurar diferentes *rankings* es necesario normalizar DCG, dando lugar al *Normalized DCG*.

Normalized Discounted Cumulative Gain (NDCG): Se puede obtener como:

$$NDCG(k) = \frac{DCG(k)}{DCG_{max}(k)}$$

Donde $DCG_{max}(k)$ es el factor de normalización y se elige de manera que un *ranking* perfecto en la posición k tenga un $NDCG(k) = 1$. En un *ranking* perfecto los documentos son ordenados de manera descendente según su relevancia, de manera que puede haber más de un *ranking* perfecto si hay relevancias idénticas que se repiten.

Una función de ganancia se puede definir como una función exponencial de la relevancia, de forma que la utilidad de acceder a la información aumenta exponencialmente cuando aumenta la relevancia del candidato:

$$G(j) = 2^{y(j)} - 1\,,$$

en donde $y(j)$ es la relevancia del candidato en la posición j. La función de descuento se puede definir como una función logarítmica de la posición, y en ese caso la utilidad de acceder a la información disminuye logarítmicamente al aumentar la posición:

$$D(j) = \frac{1}{\log_2(1+j)}\,.$$

De esta forma, el DCG y NDCG para un conjunto ordenado de candidatos C en base a un ordenamiento $R : [1, k] \to C$ son:

$$DCG(R) = \sum_{j=1}^{k} \frac{2^{y(j)} - 1}{\log_2(1+j)}$$

$$NDCG(R) = \frac{DCG(R)}{DCG_{max}(R)}$$

Otra métrica usada en *information retrieval* es el *Mean Average Precision*:

Mean Average Precision (MAP): asume que el grado de relevancia es binario. Se parte por definir el *Average Precision* (AP) como de un conjunto ordenado de candidatos en base a un ordenamiento R como:

$$AP(R) = \frac{\sum_{j=1}^{k} P(j) \cdot y(j)}{\sum_{j=1}^{k} y(j)}$$

Donde AP es *Average Precision*, $y(j)$ es la relevancia que puede ser 1 o 0, representando relevante o irrelevante respectivamente. $P(j)$ representa la precisión hasta la posición j y se define como:

$$P(j) = \frac{\sum_{k=1}^{j} y(k)}{j}\,.$$

De esta forma, AP representa la precisión media sobre todas las posiciones de los candidatos para la consulta. Los valores de precisión media se promedian en un conjunto de consultas y sus respectivos candidatos ordenados, para obtener el MAP.

2.2.2. Enfoques de aprendizaje

La mayoría de los métodos de aprendizaje de *learning to rank* pueden ser categorizados en tres enfoques: *pointwise*, *pairwise* y *listwise*. Los enfoques *pointwise* y *pairwise* transforman el problema de *ranking* en un problema de clasificación o regresión que intenta predecir el score de cada uno de los candidatos en el ranking. El enfoque *listwise*, en cambio, toma una lista de candidatos y aprende un modelo para predecir a todo el ranking en su conjunto; este problema no es directamente asimilable a problemas previos. La principal diferencia entre modelos es la función de pérdida empleada por cada uno. En general, los enfoques *pairwise* y *listwise* tienen un desempeño superior a los métodos *pointwise*; por ejempolo, en el *Yahoo Learning to Rank Challenge* de primavera del 2010, LambdaMART –un método de enfoque *listwise*– logró el mejor desempeño [Chapelle and Chang, 2011].

En el enfoque *pointwise* se transforma el problema en una regresión o clasificación, ignorando la estructura del *ranking*. De esta forma, el problema se plantea como la predicción de una etiqueta asignada a cada candidato, y según si la etiqueta es una clase o un número esto resulta en un problema de clasificación o de regresión, respectivamente. En la parte superior de la Figura 2.10 se puede ver cómo el modelo *pointwise* toma cada documento y predice el *score*; luego la función de pérdida se calcula en relación al *score* real que debería tener ese documento. Durante la etapa de inferencia se predice el *score* de cada documento candidato y después se ordena la lista de candidatos en base a los *scores* predichos.

En el enfoque *pairwise* se transforma el problema en una clasificación o regresión de a pares; en este caso la estructura del grupo de candidatos también se ignora. El problema se plantea como la predicción de, dado un par de candidatos, cuál de los dos tiene una etiqueta de una clase superior o cual tiene un mejor *score*, resultando en un problema de clasificación o de regresión, respectivamente. De esta forma se entrena un modelo que aplicará los mismos pesos a las *features* de ambos candidatos, y luego tomará la diferencia en la salida intentando aproximar la diferencia real en el score de ambos candidatos. En la parte central de la Figura 2.10 se puede ver en nuestro caso cóimo el modelo *pairwse* toma pares de documentos y predice scores

para cada uno; luego toma la diferencia de *scores* y calcula la función de pérdida en relación la diferencia esperada que deberían tener los documentos. Durante la etapa de inferencia se pasa cada documento por el submodelo entrenado que predice los scores, y después se ordena la lista de candidatos en base a estos *scores* predichos.

En el enfoque *listwise* se toma la lista de candidatos tanto en el aprendizaje como en la predicción. De esta manera se toma en cuenta la estructura del grupo de candidatos en su conjunto. Esto tiene como ventaja que se pueden incorporar directamente métricas de evaluación de *Information Retrieval* a las funciones de pérdida. Entonces, la entrada al modelo será la lista de candidatos, y la salida será un *score* para cada candidato de la lista; esta es una nueva categoría de problema en *machine learning* y no puede ser mapeada directamente a técnicas preexistentes. En la Figura 2.10, en la parte inferior, se puede ver cómo el modelo *listwise* toma listas de documentos y predice listas de scores; luego la función de pérdida se calcula en relación a todos los *scores* esperados a la vez, conservando la estructura del grupo durante el aprendizaje. Durante inferencia se pasa la lista de candidatos entera al modelo y se obtienen directamente los resultados de los *scores* para cada candidato.

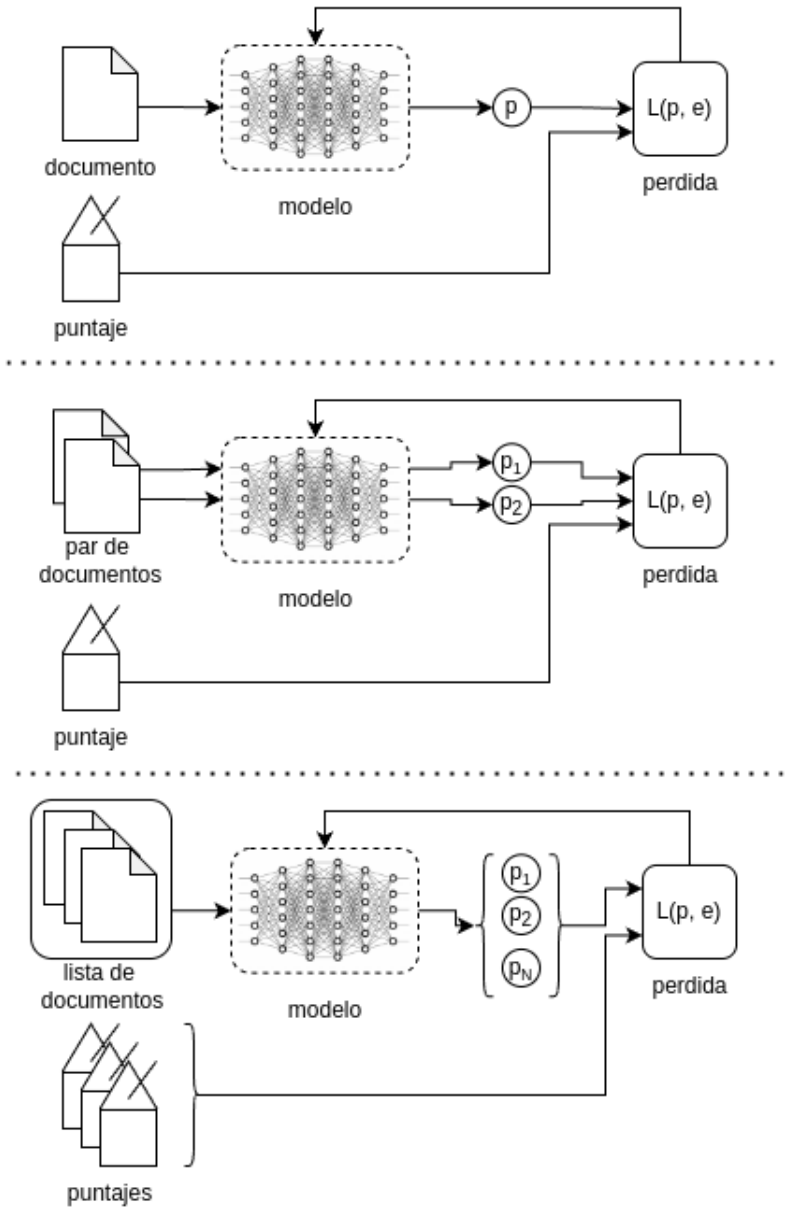


Figura 2.10: Diferentes estrategias en el problema de Learning to Rank: *(arriba) pointwise, (centro) pairwise, (abajo) listwise.*

2.2.3. RankNet, LambdaRank y LambdaMART

En este trabajo elegimos usar la implementación de XGBoost [Chen and Guestrin, 2016] del modelo LambdaMART, que como mencionamos anteriormente es unos los modelos con mejor

desempeño en LTR. LambdaMART es una adaptación de LambdaRank para *gradient boosting trees*, que a su vez se basa en un modelo previo: RankNet, que describiremos a continuación [Burges, 2010].

RankNet [Burges et al., 2005] es un modelo con enfoque pairwise, que incorpora la entropía cruzada como función de pérdida durante el aprendizaje. Dado un par de documentos (d_1, d_2) se intenta predecir la probabilidad de que uno de ellos esté mejor rankeado que otro, P_i, entrenando una función de *ranking* $f(x)$ que asigna *scores* a los candidatos. Luego se estima dicha probabilidad aplicando una función logística a la diferencia de los scores asignados por la función:

$$P_i = \frac{e^{f(d_1)-f(d_2)}}{1+e^{f(d_1)-f(d_2)}}$$

Luego, se utiliza la entropía cruzada para medir la distancia entre las distribución de probabilidad predicha y la esperada ($\overline{P_i}$), como:

$$L_i = -\overline{P_i}\log(P_i) - (1-\overline{P_i})\log(1-P_i)\,.$$

Usando la definición de P_i anterior podemos escribir la pérdida en función de $f(x)$:

$$L_i = -\overline{P_i}\cdot(f(d_1)-f(d_2)) + \log\left(1+e^{f(d_1)-f(d_2)}\right)\,.$$

RankNet utiliza redes neuronales como base para el modelo; en el *paper* original se usa una red de 3 capas con activaciones sigmoideas y un solo nodo de salida para modelar $f(x)$. En cada iteración se utiliza *backpropagation* para actualizar los parámetros según:

$$\theta_{nuevo} = \theta_{actual} - \eta\frac{\partial L_i}{\partial \theta_{actual}}\,,$$

en donde η es la tasa de aprendizaje. RankNet usa un conjunto de datos de validación para la selección de hiperparámetros, realizando una validación cruzada durante este proceso; de esta manera puede incorporar métricas de *Information Retrieval* en la evaluación.

Una ventaja de RankNet sobre métodos previos es que se puede hacer uso de técnicas de redes neuronales preexistentes. Como desventaja, la meta de aprendizaje puede no ser consistente con la tarea durante inferencia. Normalmente se evalúa a los *rankings* utilizando métricas como NDCG que se definen en una lista de candidatos; en cambio, el entrenamiento se hace buscando mejorar la precisión de la clasificación *pairwise*, que se define entre pares de candidatos. Esta disparidad de objetivos se puede mitigar en parte como se mencionó anteriormente durante la selección de hiperparámetros.

En la Figura 2.11 se puede ver el problema de objetivos en RankNet: las barras grises corresponden a candidatos irrelevantes y las azules a candidatos relevantes; mientras que el ordenamiento de las derecha tiene 13 errores de pares, el de la izquierda tiene 11. Si analizamos estos *rankings* con métricas de *Information Retrieval* las mismas le darán mas valor a los resultados que aparecen en las primeras posiciones, por ello es preferible utilizar el *ranking* que vemos a la derecha. Las flechas negras representan el gradiente de RankNet, que aumenta con el número de errores de pares, pero lo que sería preferible teniendo en cuenta la estructura global del *ranking* son los gradiente rojos. Esta modificación de los gradientes es lo que busca resolver LambdaRank [Burges et al., 2006, Donmez et al., 2009].

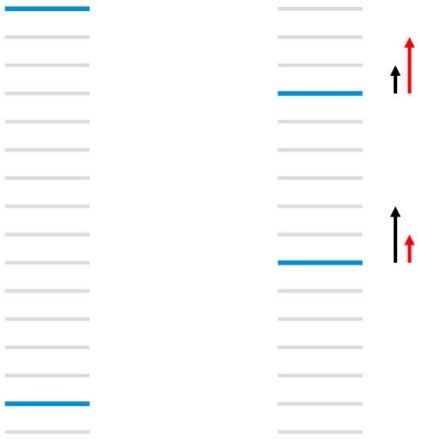


Figura 2.11: Esquematización de un conjunto de URL's ordenadas para una consulta determinada. Las barras de color azul oscuro y gris claro representan las urls relevantes y las que no lo son, respectivamente. Las flechas negras de la izquierda denotan los gradientes de RankNet, que aumentan con el número de errores por pares. Las flechas rojas de la derecha representan gradientes que consideran la información de la lista completa, estos gradientes son los que se busca obtener. Figura extraída de [Burges, 2010].

Un problema para incorporar métricas de *Information Retrieval* es que al depender del ordenamiento, las mismas no son continuas y diferenciables, por lo que no pueden ser minimizadas descendiendo por el gradiente. Para resolver este problema, en vez de cambiar la función de pérdida en sí, se modifican directamente los gradientes de la función de pérdida durante el entrenamiento. Esto resulta en una función de pérdida subrogada que termina calculándose en función de todos los documentos, y por lo tanto es una forma particular de enfoque *listwise*.

Entonces se define el gradiente de la función de pérdida subrogada, llamada función lambda - de allí el nombre LambdaRank -, de forma que incorpore información sobre cómo modifica el *ranking* general reordenar los dos documentos de entrada. Una opción directa es multiplicar el gradiente de la función de pérdida RankNet por el tamaño del cambio de alguna métrica de *Information Retrieval*, por ejemplo NDCG, que se obtiene de hacer el cambio de posiciones del par de candidatos evaluado.

Por último, LambdaMART reemplaza las redes neuronales usando el algoritmo MART Multiple Additive Regression Trees(MART), también conocido como Gradient Tree Boosting [Friedman, 2001]. MART se basa en la suma de múltiples árboles de regresión, que se crean sucesivamente buscando predecir los residuos del anterior, descendiendo por el gradiente de la función de pérdida. LambdaMART usa tanto la función lambda como los gradientes de MART. En experimentos comparativos LambdaMART mostró mayor eficiencia y precisión que LambdaRank [Wu et al., 2010].

2.3. Equidad (*fairness*)

Con el sistema completo entrenado nos interesará analizar si las decisiones que toma son justas. Para esto revisaremos los distintos aspectos que determinan si un sistema es justo o no, y cómo se puede corregirlo en caso de no serlo.

En la discusión general del tema que plantea [Barocas et al., 2023], nuestro éxito, felicidad y bienestar dependen de decisiones que toman otras personas y sistemas; esto pueden afectar profundamente el curso de nuestras vidas. Los sistemas de toma de decisión arbitrarios, inconsistentes o con fallas generan preocupación por el riesgo que tienen en limitar nuestra habilidad para conseguir las metas que nos proponemos y acceder a las oportunidades para las cuales estamos calificados.

Puede ser muy útil en tener un conjunto de reglas fijas, que se aplican consistentemente, pero un buen proceso de decisión debería tomar evidencia en cuenta. Identificar que tomar en cuenta puede suceder a través de la observación o, formalizando el proceso, se pueden armar pruebas estadísticas para analizar la evidencia histórica asociada con la decisión a tomar. En general los modelos estadísticos son mas precisos que los juicios de humanos, basados en intuición o experiencia.

Los modelos se pueden ajustar sobre las relaciones que intuimos como relevantes o también se puede dejar que los modelos observen todos los datos que hemos capturado e identifiquen que factores tienen relaciones estadísticas a la salida libremente, permitiendo así develar nuevos patrones, que los seres humanos podrían pasar por alto, en la evidencia histórica. Aunque

nosotros podemos realizar tareas complejas sin esfuerzo, como identificar objetos visualmente, pero no somos capaces de escribir un programa que enumere exhaustivamente los factores que nos permiten identificarlos. En vez de enseñar a una computadora a resolver la tarea a través de instrucciones explícitas, los modelos de aprendizaje estadístico superan este problema aprendiendo a partir de ejemplos.

Aprender es un proceso que requiere generalización desde los ejemplos. Esto implica que, para que la generalización de los ejemplos históricos a casos futuros sea buena necesitamos darle buenos ejemplos: suficientemente largos para ver patrones sutiles, suficientemente diversos para ver diferentes tipos de apariciones, suficientemente bien anotados para que tenga una verdad base confiable, etcetera. Entonces la toma de decisiones en base a la evidencia sólo es tan confiable como la evidencia en la que se basa; así el hecho de que el aprendizaje estadístico este basado en evidencia no asegura que va a llevar a decisiones precisas, confiables o equitativas.

Lo anterior es especialmente problemático cuando se usa para modelar el comportamiento humano, donde ejemplos históricos de los predictores modelados normalmente van a reflejar prejuicios en contra de ciertos grupos, esto puede hacer prevalecer estereotipos culturales y desigualdades demográficas preexistentes. Encontrar los patrones en los datos para obtener las predicciones mas probables normalmente implica repetir las mismas dinámicas. Otro problema es que, en contraste con la toma de decisiones a través de los seres humanos, los modelos estadísticos van a tratar de mejorar la precisión a todo costo, sin considerar si los atributos usados para la predicción son moralmente relevantes. Además es improbable que los seres humanos vayan en contra del sentido común cuando toman sus decisiones, algo que puede sucederle a un modelo entrenado con datos erróneos.

Por todas estas razones, los modelos de aprendizaje estadístico pueden resultar en decisiones que no son equitativas. En nuestro caso particular vamos a analizar si el sistema trata de forma equitativa a los usuarios que provienen de países con distintas barreras de acceso a la tecnología, usando como métrica *proxy* el índice de desarrollo humano.

Responder preguntas permite familiarizarse, explorar y corregir conocimientos previos sobre

nuevas tecnologías. Tener preferencias por usuarios países con índices de desarrollo humano mas altos al recomendar quienes responden nuevas preguntas, le saca oportunidades a los demás usuarios, reforzando la desigualdad en la barrera de acceso a la tecnología.

En esta sección vamos a describir métricas que permitan ver cómo afecta la equidad a nuestro sistema, para analizar si éste va a reproducir dinámicas de desigualdad previas. Luego vamos a revisar de que formas podríamos corregir la equidad de un sistema, y cual se adapta mejor a nuestro caso; para esto vamos a caracterizar los métodos encargados de la corrección de equidad en modelos. Por último describimos como funciona FA*IR, el método que elegimos para corregir la equidad en nuestro sistema.

2.3.1. Análisis y métricas

La discriminación por pertenecer a un grupo especifico de la población esta enfocada en categorías sociales que sirvieron como base para un trato adverso e injustificado en el pasado, esto da lugar a las llamadas categorías protegidas. Las mismas pueden incluir: sexo, etnia, religión, discapacidades o lugar de nacimiento. En muchos casos, los *features* que caracterizan a un individuo para ser representando en un modelo estadístico, codifican a su vez su pertenencia a un grupo protegido. Las variable aleatorias que capturan una o múltiples características que implican la pertenencia a un grupo protegido se llaman atributos sensibles.

Se podría pensar que una opción para evitar la inequidad seria borrar o ignorar atributos sensibles. El problema es que un conjunto de características pueden estar correlacionadas con el atributo sensible. En un espacio de características grande los atributos sensibles son generalmente redundantes dadas otras características. De esta manera, aunque removamos el atributo sensible el clasificador puede usarlo a través de la codificación redundante de las otras características. Por esto es un error ignorar los atributos sensibles si estamos buscando que un sistema sea justo.

Existen criterios estadísticos de no discriminación que intentan definir las relaciones entre variables aleatorias que describen a un clasificador. Son propiedades de la distribución conjunta de atributos sensibles (C), la variable a predecir (A) y la predicción del clasificador ($\hat{A}$). Se han

planteado muchos criterios que capturan diferentes aspectos e intuiciones de lo que es justo; todos en general intentan igualar alguna de las siguientes tres características a través de todos los grupos:

Tasa de aceptación : $P(\hat{A} = True)$.

Frecuencia de salida : $P(A = True|\hat{A} = \hat{a})$.

Tasa de error : $P(\hat{A} = True|A = False)$ y $P(\hat{A} = False|A = True)$.

Estos criterios se pueden generalizar usando declaraciones de independencia condicional como:

Independencia : $\hat{A} \perp C$.

Suficiencia : $A \perp C \mid \hat{A}$.

Separación : $\hat{A} \perp C \mid A$.

La independencia se puede explorar a través de definiciones equivalentes, por ejemplo la paridad demográfica, en donde la condición es equivalente a: $P(\hat{A} = True|C = c_0) = P(\hat{A} = True|C = c_1)$. Si se piensa el evento $\hat{A} = True$ como "aceptación", esta condición busca que la tasa de aceptación sea la misma en todos los grupos. La idea detrás de la paridad demográfica es la distribución proporcional de los recursos.

Una versión relajada de la suficiencia es la paridad predictiva, que requiere que tanto los falsos positivos como los verdaderos positivos para los aceptados sean los mismos a través de los distintos grupos: $P(A = True|\hat{A} = True \wedge C = c_0) = P(A = True|\hat{A} = True \wedge C = c_1)$, llamado *outcome test*. La paridad predictiva puede ser vista como la precisión condicionada a un valor del atributo sensible. La paridad predictiva se enfoca en el error de paridad que hay entre las personas que recibieron la misma decisión.

Por último, la igualdad de oportunidades se puede ver como una relajación de la separación de forma que: $P(\hat{A} = True|A = True \wedge C = c_0) = P(\hat{A} = True|A = True \wedge C = c_1)$. Esto se puede interpretar como el *recall* condicionado al valor del atributo sensible. Este criterio está asociado con acceder a las mismas oportunidades teniendo en cuenta las desigualdades

preexistentes en adquirir las habilidades necesarias para desenvolverse en la tarea a predecir.

2.3.2. Corrección de la equidad

Los métodos para la corrección de la equidad pueden ser categorizados según diferentes criterios, entre ellos: según el momento en que se hace la corrección, la estructura de grupo, el tipo de sesgo y la visión del mundo [Zehlike et al., 2022b].

Según en que momento se hace la corrección, se puede categorizar a los métodos como:

Pre-Procesamiento : Buscan ajustar el espacio de *features* para que los mismos no estén correlacionados con los atributos sensibles.

En-Entrenamiento : Utilizan restricciones durante el proceso de optimización.

Post-Procesamiento : Ajustan un clasificador ya entrenado, para que sus predicciones no estén correlacionadas con los valores de los atributos sensibles.

La estructura del grupo está compuesta por la cardinalidad de los atributos sensibles y el número de atributos sensibles. Con cardinalidad nos referimos a si los atributos sensibles son binarios, pudiendo tomas sólo dos valores, o no binarios, en cuyo caso se deberá determinar cuál/es de los múltiples valores serán los protegidos. El número refiere a la cantidad de atributos sensibles que el método puede manejar al mismo tiempo. Cuando se consideran muchos atributos al mismo tiempo, los mismos pueden ser tomados independientemente o en combinación conjunta. Se puede ver que los métodos que soportan un solo atributo sensible con cardinalidad no binaria también pueden ser utilizados para representar múltiples atributos sensibles, componiendo los atributos en la alta cardinalidad en el atributo sensible no binario.

El tipo de sesgo a mitigar puede ser preexistente, técnico o emergente. Los sesgos preexistentes son los que existen independientemente de los algoritmos y tienen sus orígenes en la sociedad. El sesgo técnico surge de restricciones o consideraciones técnicas, por ejemplo el tamaño de la pantalla o el sesgo de posición en los *rankings*. El sesgo emergente surge por el contexto de uso, normalmente cuando el sistema fue diseñado con otros usuarios en mente, o cuando conceptos

sociales derivan en el tiempo.

Un modelo se puede analizar como un paso dentro de un pipeline de decisiones, y este pipeline se puede plantear como mapeos entre tres espacios métricos: el espacio de construcción (EC), el espacio observado (EO) y el espacio de decisión (ED). Se define a la visión del mundo, o sistema de creencias, como los supuestos que realizamos sobre las propiedades de los mapeos entre estos espacios. El EC representa las propiedades no observables de un individuo, como la inteligencia o perseverancia, mientras que el EO representa las propiedades que podemos medir –por ejemplo un test de IQ como *proxy* de inteligencia o el promedio escolar como *proxy* de la perseverancia–. Este EO sirve como espacio de características para entrenar un modelo de decisión. Por último, el ED mapea el EO a un espacio en donde se toman decisiones, por ejemplo ser aceptado o rechazado para recibir una beca.

Los mapeos entre espacios son propensos a distorsiones, y los mapeos de EC a EO o ED son por definición no observables. Como las propiedades de estos mapeos no se pueden verificar, es necesario postular un sistema de creencias. Hay dos casos extremos de estos sistemas: WYSIWYG (*What you see is what you get*) y WAE (*We are all equal*). WYSIWYG asume que EC y EO son esencialmente el mismo espacio y cualquier distorsión entre ambos es a lo sumo de un mínimo valor ϵ. WAE, en cambio, asume que cualquier diferencia entre las distribuciones de utilidad de los diferentes grupos se dá por errores o sesgos en el proceso de observación. En el ejemplo de admisiones a becas la visión WAE supone que las diferencias en la distribuciones de los tests de IQ o el promedio escolar se deben a sesgos en el sistema escolar o a las pruebas de IQ.

Cuando se categoriza a los métodos de corrección de equidad con respecto a su visión del mundo, se analiza si el objetivo de equidad es la igualdad de resultados o la igualdad de tratamiento. Si la meta es alcanzar igualdad de resultados se asume que el EO no es confiable porque está sesgado o hay una distorsión entre EO y EC; este caso considera entonces una visión del mundo WAE. Si por otro lado, la meta es alcanzar igualdad de tratamiento, se asume que el mapeo entre EC y EO tiene poca distorsión, y entonces el método tiene una visión del mundo WYSIWYG. Por último, algunos métodos tienen una visión del mundo continua, y mediante parámetros del

algoritmo permiten variar entre una visión WAE y WYSIWYG.

En nuestro caso elegimos un método de corrección de equidad, buscando que su punto de operación sea durante el post-procesamiento, porque esto facilita aplicar las correcciones sobre el sistema ya entrenado. Además, buscamos un método que soporte mas de un atributo a la vez y que tenga extensiones a múltiples grupos protegidos, ya que la barrera de acceso a la tecnología puede estar afectada a su vez por otros atributos sensibles que no consideramos. También queremos que se enfoque en sesgos pre-existentes, para evitar repetir dinámicas que refuercen la barrera de acceso a la tecnología. Por último, dado que estamos tratando con sesgos preexistentes buscamos que el algoritmo tenga una visión del mundo WAE o continua, permitiendo ajustar un punto medio entre la visión WYSIWYG y WAE. En la siguiente subsección describimos el algoritmo que elegimos.

2.3.3. FA*IR

El algoritmo FA*IR [Zehlike et al., 2017] trabaja con una definición de equidad que se basa en el supuesto de que los *rankings* son justos cuando la decisión de elegir un candidato es tomada de una distribución Bernoulli que no esta impactada por los atributos sensibles del candidato [Zehlike et al., 2022c].

El algoritmo de FA*IR asegura que el número de candidatos protegidos no caiga debajo de un porcentaje requerido para cumplir equidad en ninguna posición del *ranking*. Aunque este método trabaja con un solo atributo sensible binario, en su trabajo posterior [Zehlike et al., 2022a] se propone una extensión al algoritmo permitiendo trabajar con múltiples grupos protegidos, soportando cardinalidad multinaria, y así también múltiples atributos sensibles como fue expuestos en la sección 2.3.2.

Este método intenta arreglar la independencia de la equidad, haciendo foco en la paridad demográfica. Un *ranking* se considera justo si la proporción de miembros del grupo protegido no cae debajo del mínimo esperado para que este evento provenga de un proceso Bernulli con p igual a la proporción de miembros protegidos en la población total.

p\k	1	2	3	4	5	6	7	8	9	10	11	12
0.1	0	0	0	0	0	0	0	0	0	0	0	0
0.2	0	0	0	0	0	0	0	0	0	0	1	1
0.3	0	0	0	0	0	0	1	1	1	1	1	2
0.4	0	0	0	0	1	1	1	1	2	2	2	3
0.5	0	0	0	1	1	1	2	2	3	3	3	4
0.6	0	0	1	1	2	2	3	3	4	4	5	5
0.7	0	1	1	2	2	3	3	4	5	5	6	6

Cuadro 2.1: Ejemplo de valores de $m_{\alpha,p}(k)$, el número mínimo de candidatos petenecientes al grupo protegido que deberán aparecer en las primeras k posiciones para que el ranking satisfaga el criterio de equidad de FA*IR con un $\alpha = 0{,}1$.

El hecho de que se use una proporción mínima para el grupo protegido sugiere que la visión del mundo que toma FA*IR es WAE. Pero ajustando el p-value, o la significancia estadística, de los tests de hipótesis podemos hacer variar la visión del mundo. Por ejemplo, tomando un valor bajo de significancia serán necesarios menos candidatos para cumplir la condición, adoptando así una visión WYSIWYG.

Para mejorar el desempeño el algoritmo precomputa una tabla que ya contiene el mínimo número de candidatos protegidos para cada posición en el *ranking* dada la proporción mínima esperada y la significancia deseada. Un ejemplo de ésto se muestra en la Tabla 2.1. Esto se hace calculando $F^{-1}(\alpha; k, p)$, la inversa de la función de distribución acumulada de una variable aleatoria binomial.

Con la tabla ya precomputada se ordenan los dos grupos por separado –protegidos y no protegidos– en orden descendente según su calificación predicha. Luego ambos grupos son combinados en un único *ranking*, tomando para cada posición el mejor de los dos grupos, pero cuando la cantidad de candidatos protegidos cae por debajo de los esperados por la tabla precomputada se agrega un candidato protegido en esa posición. En el Algoritmo 1 mostramos un pseudocódigo de

FA*IR.

Por último, la definición de equidad adoptada por FA*IR realiza implícitamente múltiples tests de hipótesis, k hasta la posición que se evalúa. Esto implica que se necesita conseguir un α_c que esté ajustado para calcular la tabla: si se usa $\alpha = \alpha_c$ se puede producir falsos negativos, rechazando *rankings* justos a una tasa mayor que α. El algoritmo tiene en cuenta este ajuste antes de precomputar las tablas con las cantidades de candidatos protegidos por posición.

Algorithm 1 El algoritmo FA*IR encuentra un ranking que maximiza la utilidad sujeta a la monotonicidad *in-group* y a las restricciones de equidad de grupo.

Input

k $\in [n]$, el tamaño de la lista a devolver.

q_i $\forall i \in [n]$, las calificaciones del candidato i.

g_i un indicador que es 1 si el candidato i está protegido.

p $\in]0, 1[$, la proporción mínima de elementos protegidos.

α_c $\in]0, 1[$, la significación ajustada para cada prueba de representación justa.

Output

τ que maximiza la utilidad y satisface la condición de equidad de grupo con los parámetros p, σ.

$P_0, P_1 \leftarrow$ cola de prioridad vacia con capacidad limitada de k elementos

for $i \leftarrow 1$ hasta n **do**

insertar i con valor q_i en la cola de prioridad P_{g_i}

for $i \leftarrow 1$ hasta k **do**

$m[i] \leftarrow F^{-1}(\alpha_c; i, p)$

$(t_p, t_n) \leftarrow (0, 0)$

while $t_p + t_n < k$ **do**

if $t_p < m[t_p + t_n + 1]$ **then**

$t_p \leftarrow t_p + 1$ ▷ Agregar un candidato protegido

$\tau[t_p + t_n] \leftarrow desencolar(P_1)$

else

if $q(ver(P_1)) \geq q(ver(P_0))$ **then** ▷ Agregar el mejor candidato disponible

$t_p \leftarrow t_p + 1$

$\tau[t_p + t_n] \leftarrow desencolar(P_1)$

else

$t_n \leftarrow t_n + 1$

$\tau[t_p + t_n] \leftarrow desencolar(P_0)$

return τ

Capítulo 3

Preprocesamiento y análisis exploratorio

En este capítulo describiremos cómo está compuesto el conjunto de datos, cómo lo preprocesamos para extraer los atributos de interés, y cómo se separó el conjunto en entrenamiento y pruebas. Todo el código realizado en esta sección así como en el resto del trabajo se encuentra disponible en el siguiente repositorio: https://github.com/flopezd/so_q_recom.

El conjunto de datos que utilizaremos es públicamente disponibilizado por Stack Exchange, que además ofrece una aplicación web para explorar sus bases de datos en https://data.stackexchange.com/; allí se pueden ejecutar consultas en lenguaje SQL de forma gratuita y sobre el conjunto completo de datos de la plataforma. Adicionalmente, en forma periódica Stack Exchange publica *dumps* de sus bases de datos de forma oficial en archive.org: https://archive.org/details/stackexchange. El conjunto de datos que extrajimos contiene aproximadamente 14 millones de usuarios, 23 millones de preguntas, 34 millones de respuestas, 87 millones de comentarios y 64 mil etiquetas a la fecha de la extracción (junio de 2023).

Como mencionamos anteriormente vamos a usar el Índice de Desarrollo Humano (IDH) como atributo sensible, siendo un *proxy* de la barrera de acceso a la tecnología. En la siguiente sección

se va a explicar como se extrajo los países a partir de la ubicación de los usuarios. Luego el atributo sensible binario se definirá en función de si los usuarios pertenecen a países de alto o bajo IDH. Dejamos la descripción de este proceso en el desarrollo del trabajo en la sección 4.3.1.

En las encuestas que presenta anualmente StackOverflow se observa que mas del 90 % de los usuarios son hombres, por eso el género fue uno de los candidatos evaluados para ser tomado como atributo sensible para contrastar los análisis de equidad. El conjunto de datos no contaba con género, entonces era necesario inferirlo. En [Yuenyong and Sinthupinyo, 2020] comparan diferentes modelos convencionales para predicción de género a partir de nombres y aliases, dando como resultado una presición de entre un 55 % y 70 %. La baja precisión iba a incorporar error al análisis de equidad, por lo que se decidió no incorporar el género como atributo sensible extra. Queda como una posible línea futura de trabajos si se puede obtener o inferir el género, contrastar el análisis o estudiar la interseccionalidad de ambos atributos.

Las preguntas en el conjunto de datos estaban con formato `html`. Buscando enfocarnos en el texto de las preguntas las preprocesamos usando BeautifulSoup para extraer el texto de los distintos *posts*. Mas del 99 % de los textos pudo ser extraído correctamente.

3.1. Extracción de países

Los usuarios de StackOverflow tienen en su perfil un campo de texto libre para indicar su ubicación. Este campo es opcional, y en nuestro conjunto de datos fue completado por más de 4 millones de usuarios, que representan un 21.31 % del total.

Las ubicaciones indicadas contienen nombres de ciudades, estados, países, abreviaciones o valores irrelevantes. Para hacer un análisis de este campo, extrajimos información a nivel de país. Este es el nivel geográfico que mantendremos a lo largo de toda esta tesis. Como referencia de los nombres de países y ciudades existentes en el mundo y sus escrituras en distintos idiomas utilizamos el dataset de geonames.

Para extraer el país construimos expresiones regulares, con las ciudades, estados y países a buscar

en los textos de las ubicaciones. Ante una coincidencia *(match)* asignamos el país asociado a esa *regex*. La búsqueda se hizo en el siguiente orden priorizando las coincidencias más largas:

- El nombre del país.
- El nombre administrativo de primer nivel (estados) en UTF-8 y ASCII.
- El nombre de la ciudad en UTF-8 y ASCII.
- Los nombres alternativos de la ciudades.
- El código de 2 letras del nombre del país (ISO-3166).

Con este procedimiento, hemos podido asignar un país al 95.26 % de las ubicaciones de los 4 millones de usuarios. En la figura 3.1 se puede ver la cantidad de usuarios asignados al top veinte de países identificados: en particular, India y Estados Unidos muestran una diferencia grande con el resto de los países, representando el 18 % y el 14 % del total de usuarios, respectivamente. El resto de la distribución de usuarios por país decrece de forma más suave.

Si bien el porcentaje de usuarios con país identificado resultó muy alto, creemos que se podrían hacer mejoras al algoritmo revisando las ubicaciones que no tuvieron coincidencias o dovolvieron un error. Se podrían también agregar chequeos por el código administrativo de primer nivel y por los nombres de código administrativo de segundo nivel (partidos). También se podría usar la Distancia de Levenshtein con algún umbral para encontrar las ubicaciones con errores de tipeo *(typos)*.

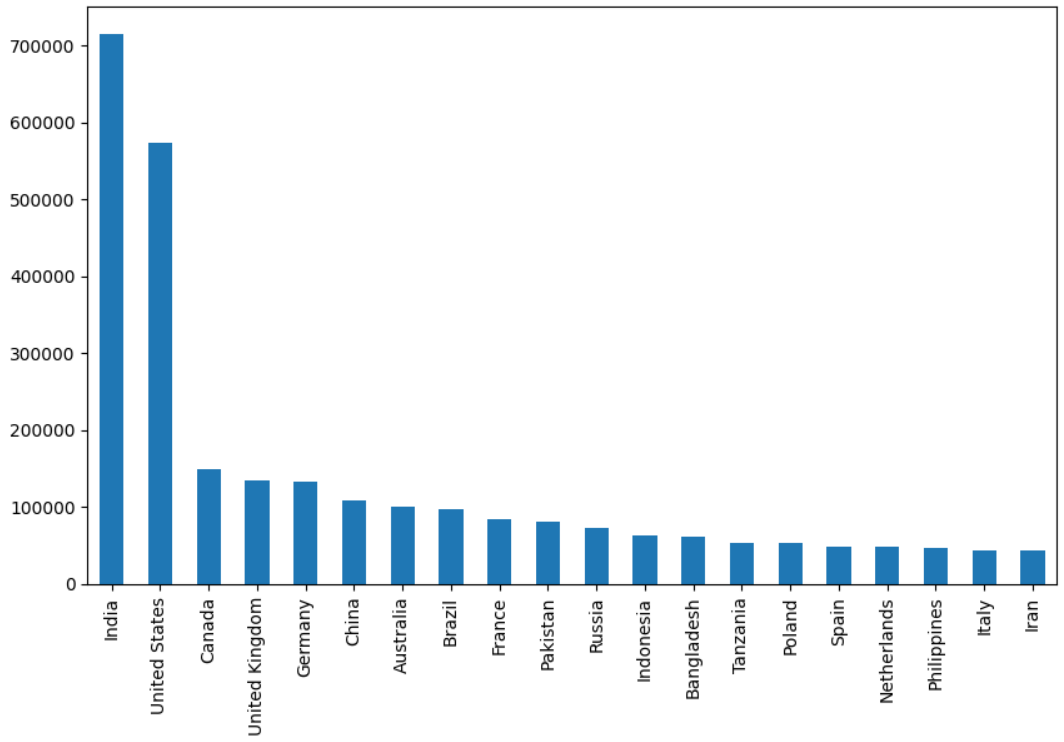
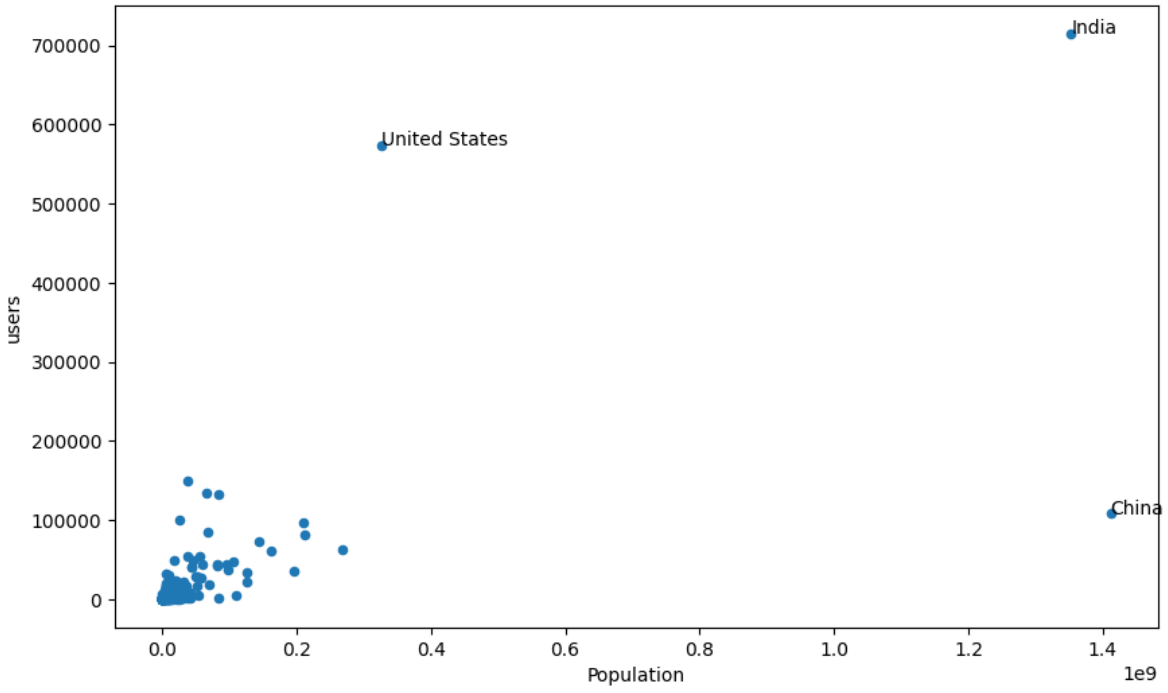
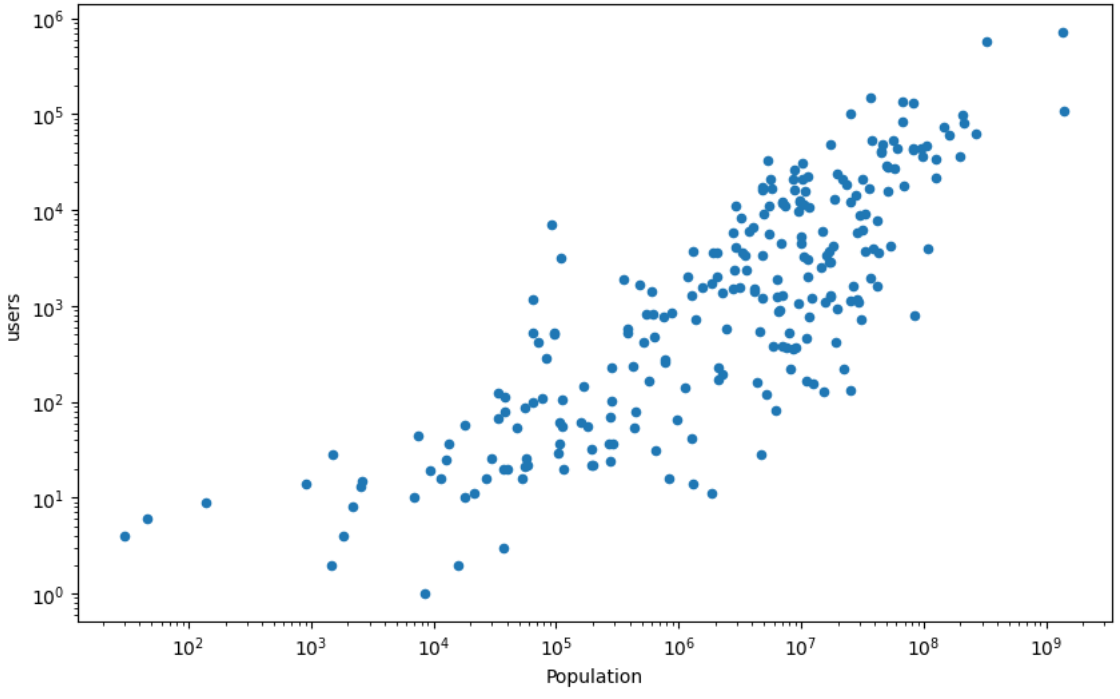


Figura 3.1: Cantidad de usuarios de StackOverflow por país

Para los análisis a lo largo del trabajo asumimos que completar ubicación era independiente del país del que proviene el usuario. En la figura 3.2 podemos ver la relación que hay entre cantidad de usuario y población de esos países, aunque hay un par de anomalías se puede ver que las variables son proporcionales. Hay dos países relevantes como anomalías: Estados Unidos que tenia sobrerepresentado la cantidad de usuario en relación a su población, posiblemente por el alto acceso a la tecnología, y China que tenia subrepresentada su cantidad de usuarios, posiblemente por sus restricciones al acceso del Internet público.



(a) Escala lineal.



(b) Escala logarítmica.

Figura 3.2: Población de cada país en relación a la cantidad de usuarios de StackOverflow por pais, en (a) escala lineal arriba y (b) logarítmica abajo.

3.2. Exploración de los datos

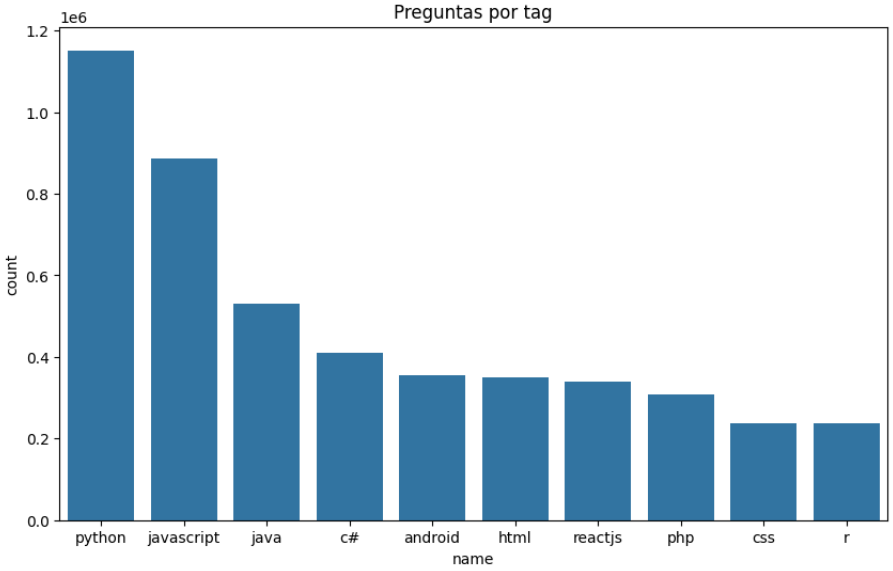


Figura 3.3: Cantidad de preguntas por *tag*(etiqueta), para los 10 *tags* con mayor cantidad de preguntas.

En la Figura 3.3 se puede ver las 10 etiquetas con mayor cantidad de preguntas con cada etiqueta, y en la Figura 3.4 de las respuestas a las preguntas. Como era esperable las primeras 10 etiquetas son las tecnologías mas usadas en la industria y la academia, además se ve que la proporción y preguntas y respuestas para cada tecnología es similar. En primer lugar se encuentra `python` con 1.17 millones de preguntas y 1.6 millones de respuestas. En promedio se ve que las preguntas de `python` tienen al menos una respuesta. Para facilitar el procesamiento de los datos y disminuir la variabilidad vamos a enfocarnos solo en las preguntas con etiquetas de `python`. Para extender las recomendaciones a otras etiquetas se puede hacer un ensamble de multiples sistemas, o agregar *features* que modelen a cada etiqueta a lo largo de los modelos del sistema.

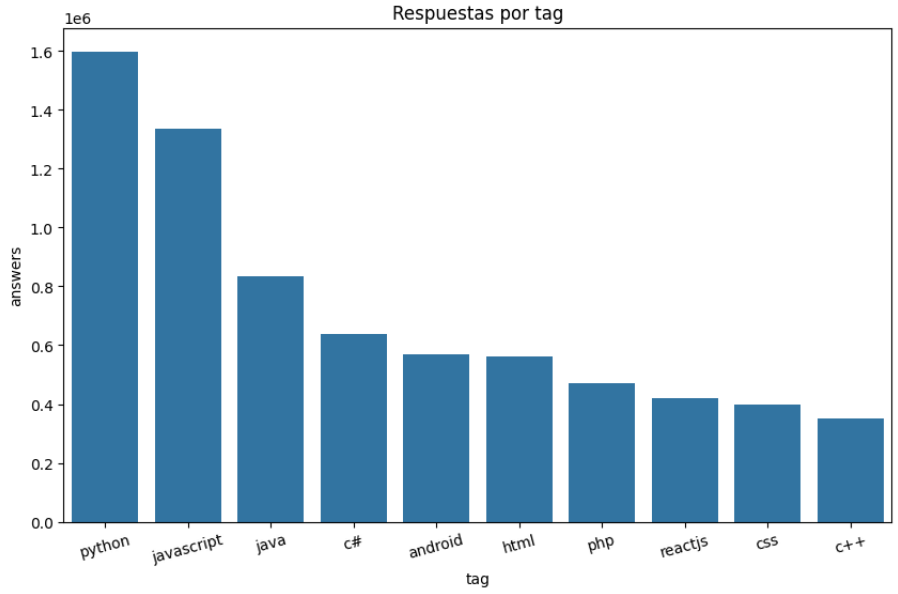


Figura 3.4: Cantidad de respuestas por *tag* (etiqueta), para los 10 *tags* con mayor cantidad de respuestas.

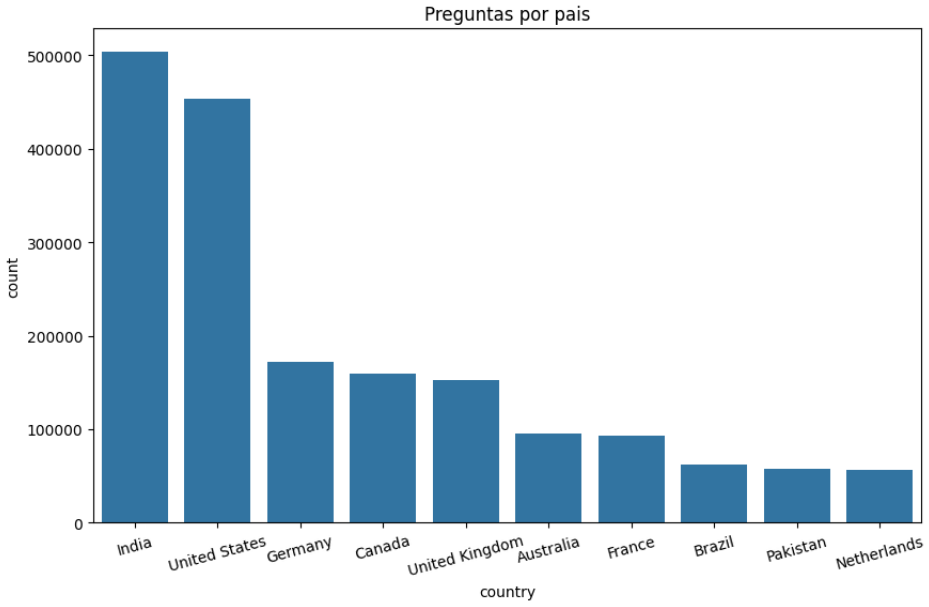


Figura 3.5: Cantidad de preguntas por país, para los 10 países con mayor cantidad de preguntas.

En la Figura 3.5 se puede ver los 10 paises con mayor número de preguntas, y en la Figura 3.6 de

las respuestas. Se ve que las preguntas y respuestas al ser menos en las primeras 10 posiciones están distribuidas mas balanceadamente que las etiquetas, donde las primeras etiquetas representan la mayoría de las preguntas y respuestas. Esto también se debe a que las etiquetas se solapan, y una pregunta puede tener múltiples etiquetas. Podemos ver que los países que aparecen en los primeros 10 son los países que identificaríamos con una baja barrera de acceso a la tecnología. Por último se ve que aunque India es el país que hace mas preguntas, Estados Unidos es el que mas las responde, y dado que podemos intuir que Estados Unidos tienen un mayor acceso a la tecnología, esto vuelve a uno de las postulados iniciales, donde suponemos que responder es la acción mas eficiente para tener acceso a una tecnología, contrariamente a lo que ocurriría con realizar preguntas.

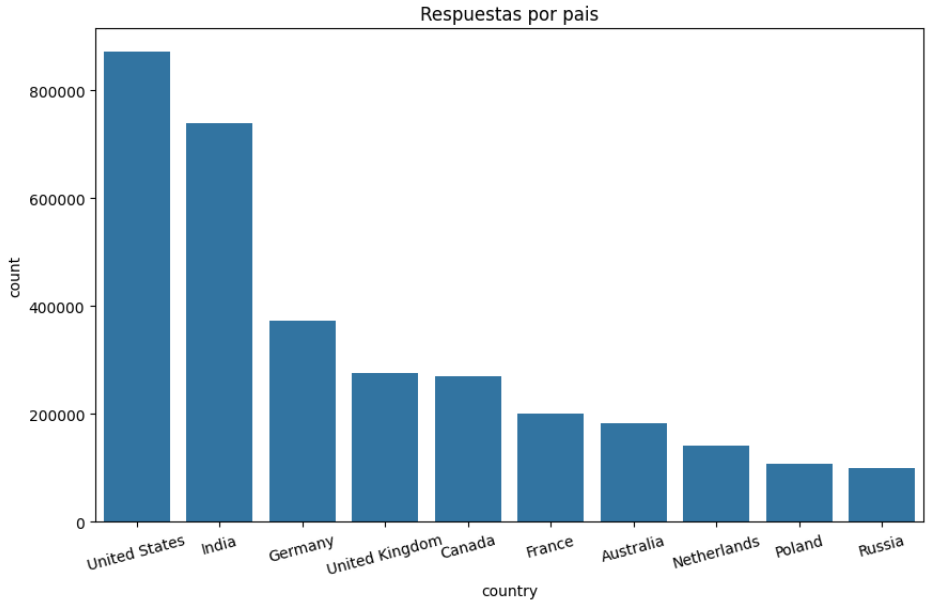


Figura 3.6: Cantidad de respuestas por país, para los 10 países con mayor cantidad de respuestas.

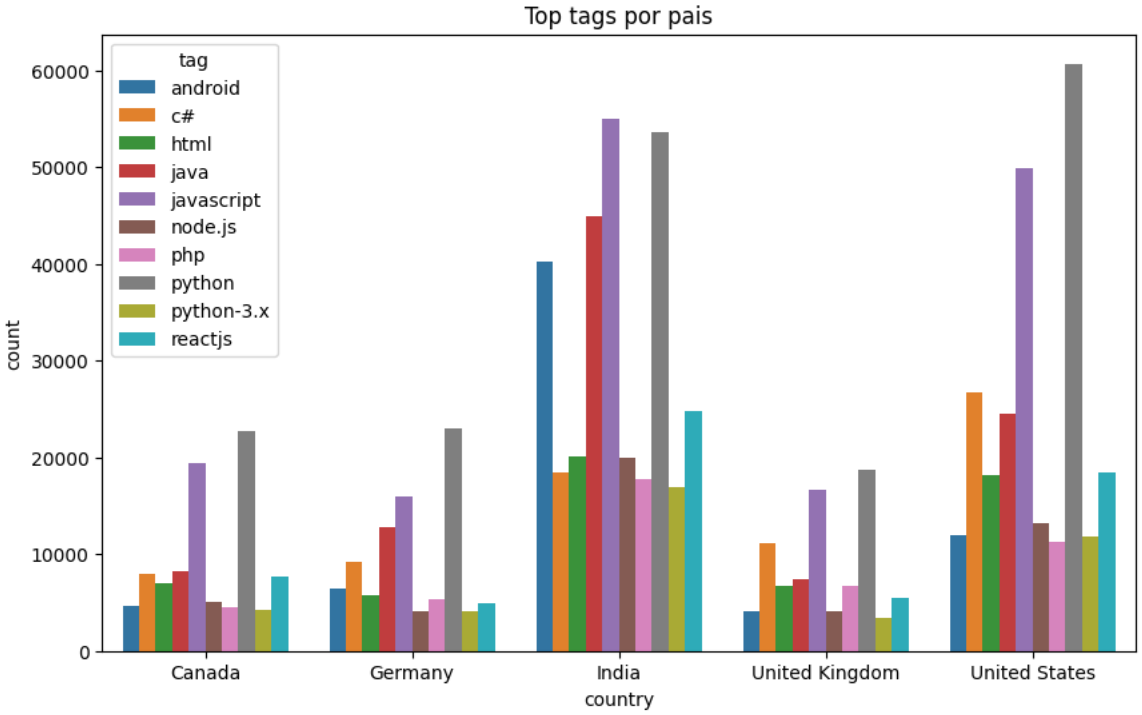


Figura 3.7: Cantidad de preguntas por *tags*(etiquetas) y país, para los 5 países con mayor cantidad de preguntas y los 10 *tags* mas usados, agrupados por pais.

En la Figura 3.7 vemos los 5 países con más preguntas con la distribución de las 10 etiquetas mas usadas, y en la Figura 3.8 mostramos la misma información, pero agrupada al inverso, con las distribución de países por cada tecnología. Notar que las etiquetas y los países elegidos son los que tienen mas preguntas independientemente, o sea los 5 países con mas preguntas y las 10 etiquetas con mas preguntas. Entonces, por ejemplo, para un país del gráfico podría haber alguna etiqueta con mas preguntas para ese país y que no este presente en este gráfico. Aquí vemos que en cada país las tecnologías dominantes son diferentes. También vemos que para cada tecnología los países dominantes siempre son India y Estados Unidos. Suponemos que esta sobrerepresentación de los países en las preguntas puede llevar a que los modelos tengan un mejor desempeño en ellos.

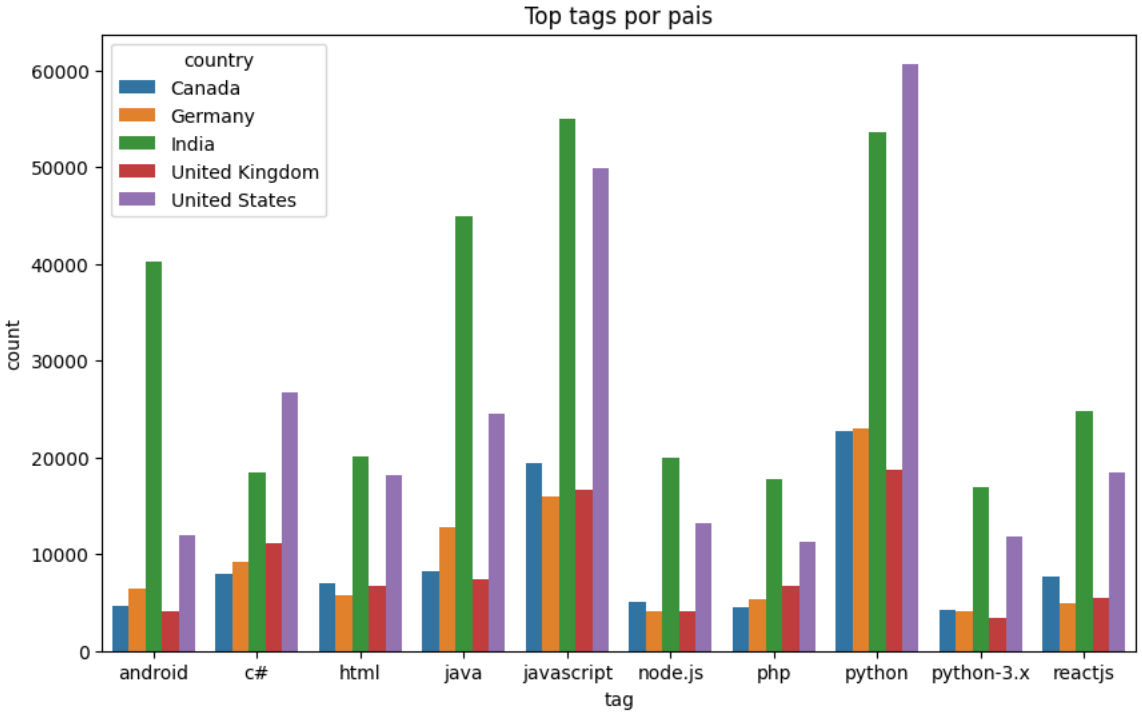


Figura 3.8: Cantidad de preguntas por *tags*(etiquetas) y país, para los 5 países con mayor cantidad de preguntas y los 10 *tags* más usados, agrupados por tag.

En la Figura 3.9 se ve la cantidad de respuestas aceptadas en promedio por país, se puede ver que son todos paises que tiene una barrera de acceso a la tecnologia bajo. Lo mismo se repite en la Figura 3.10 que muestra el puntaje promedio de las respuestas por pais, también se ve que los países que aparecen con mayor promedio son países con IDH elevado.

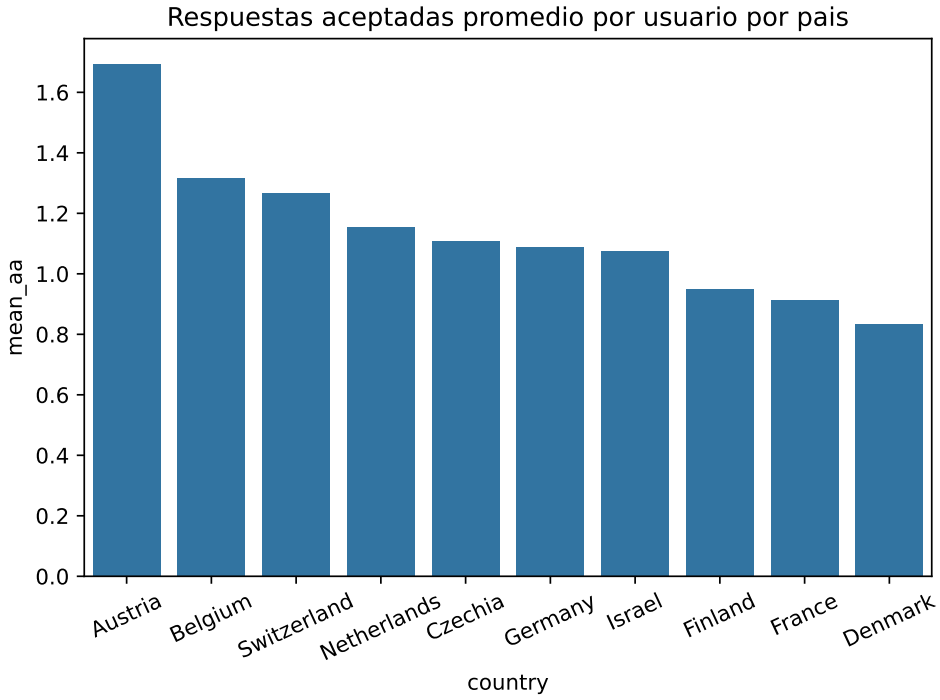


Figura 3.9: Cantidad de respuestas aceptadas en promedio por país, para los 10 países con mas respuestas aceptadas promedio.

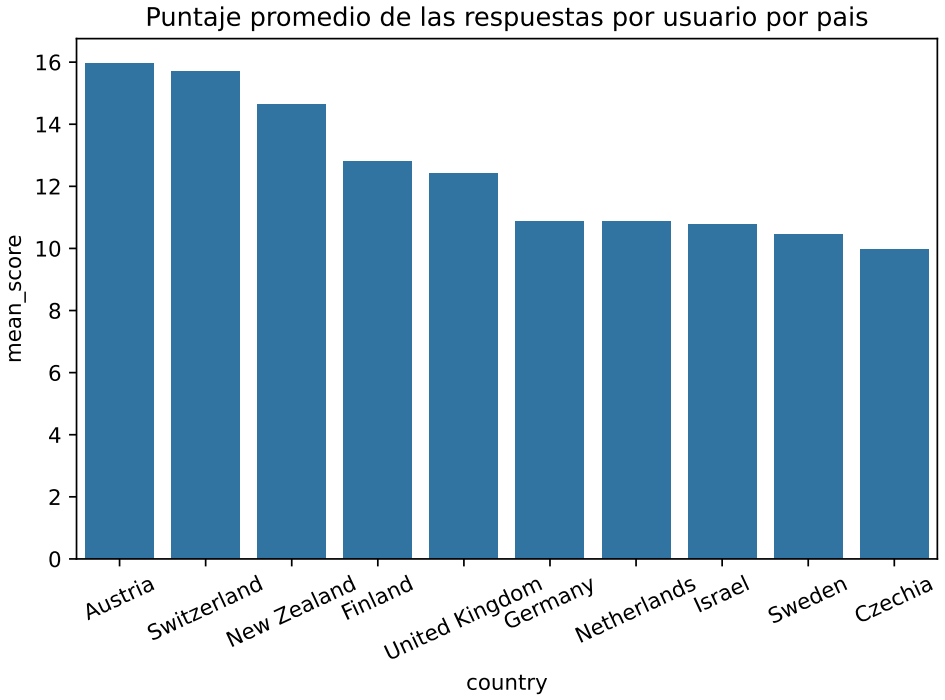


Figura 3.10: Puntaje promedio de las respuestas por país, para los 10 países con mejor promedio.

En contraste si vemos la Figura 3.11 que muestra la proporción de usuarios con respuestas que no tienen ninguna respuesta aceptada por cada país. En otras palabras, esto representa la proporción de personas que responden preguntas, pero cuyas respuestas nunca han sido marcadas como aceptadas hasta el momento. Podemos ver que los países con proporciones más altas son países que tiene una mayor barrera de acceso a la tecnología. Todos estos sesgos preexistentes hacen que sea una tarea desafiante crear un sistema que no los refleje o retroalimente: si los modelos simplemente tienden a los valores medios de las distribuciones observadas, se va a construir un sistema injusto.

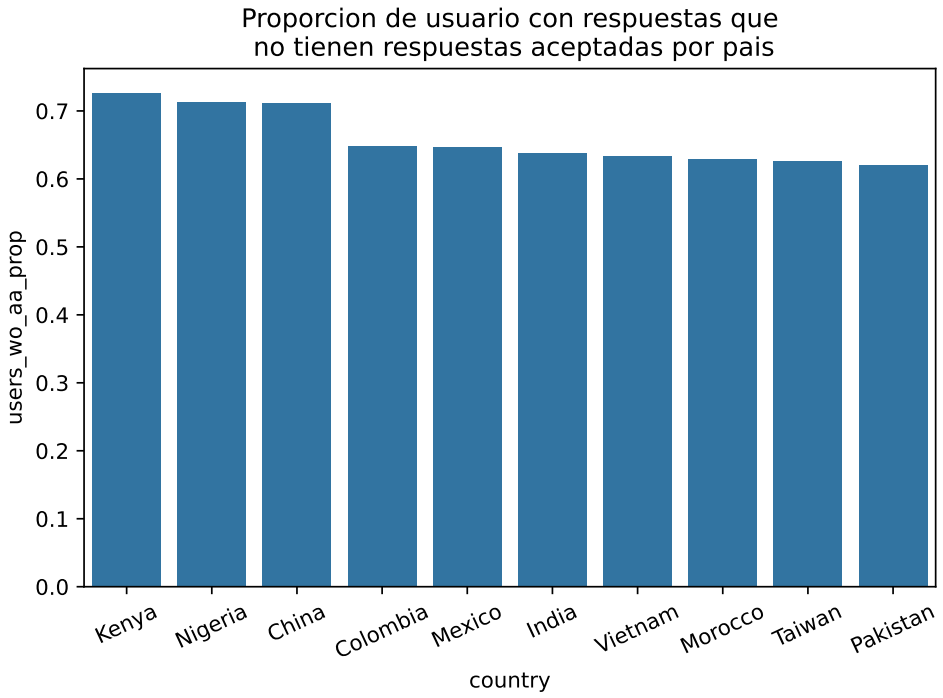


Figura 3.11: Proporción de usuarios con respuestas que no tienen ninguna respuesta aceptadas en cada país, para los 10 países con mayor proporción de respuestas escritas no aceptadas.

Los datos fueron separados en dos conjuntos: uno para los entrenamientos y otro para las pruebas. A lo largo del trabajo nos vamos a referir al conjunto de entrenamiento como el conjunto de los *posts* comprendidos desde 2019-01-01 hasta antes del 2022-01-01, con 896946 preguntas y 1360179 respuestas. Este conjunto de entrenamiento se va a usar para ajustar las representaciones de texto y el modelo de *learning to rank*. En algunos casos se uso un conjunto de validación formado por una muestra al azar del conjunto de entrenamiento, este conjunto se uso para controlar los entrenamiento, comparar resultados intermedio y asegurarse que no haya overfitting.

Las preguntas y respuestas posteriores al 2022-01-01 conforman el conjunto de pruebas, con 254688 preguntas y 254688 respuestas. El conjunto de pruebas se va a utilizar para las métricas de desempeño del sistema completo y para hacer los análisis de equidad. Cabe notar que en el conjunto de pruebas hay menos respuestas porque hay preguntas nuevas que todavía no habían

sido contestadas al momento de la captura, esto va a hacer que nuestras métricas de desempeño sean más exigentes con el sistema.

Capítulo 4

Desarrollo

Como mostramos en el Capítulo 2, para construir un sistema de recomendación con grandes volúmenes de datos es conveniente dividir el problema en dos etapas. La primer etapa es de alto *recall*, y en ella es necesario recuperar un gran número de candidatos. La segunda etapa, en cambio, se enfoca en la precisión, buscando filtrar los candidatos que no eran necesarios y ordenándolos para comenzar por los más relevantes.

En este capítulo comenzaremos mostrando las distintas representaciones que hemos explorado para la búsqueda vectorial de la primer etapa. Luego, mostraremos cómo resolvimos la segunda etapa utilizando modelos de *Learning To Rank* (LTR).

En una siguiente sección mostraremos los resultados de nuestro modelo: con el sistema completo ya montado hemos realizado mediciones de desempeño, y posteriormente nos enfocamos en calcular métricas de equidad. A este paso le siguió el postprocesado, a través del algoritmo FA*IR, de los resultados que entrega el sistema, y el análisis comparativo de la equidad antes y después de la corrección.

Hacia el final del capítulo mostraremos cómo modifica los ordenamientos el algoritmo FA*IR: en particular, en base a las observaciones sobre los resultados obtenidos, hemos propuesto cambios para que el ordenamiento global de FA*IR resulte más orgánico.

4.1. Generación de representaciones

En nuestro sistema necesitamos generar una representación para cada candidato a proponer, y que dicha representación se mantenga en una estructura índice en memoria que permita hacer búsquedas eficientes y escalables durante el uso. En este trabajo utilizamos la biblioteca Faiss de Facebook [Douze et al., 2024], que nos permite hacer búsquedas de vecinos aproximadas de forma rápida en memoria, una funcionalidad esencial para bases de datos vectoriales [Han et al., 2023].

Por otra parte, la representación generada deberá servir de sustento para extraer candidatos capaces de responder a una pregunta planteada. En otras palabras, deberá facilitar un alto *recall* (exhaustividad) en esta primer etapa del sistema. El *recall* lo medimos como la fracción de usuarios correctamente propuestos a través del índice, en relación al conjunto de usuarios que efectivamente respondieron la pregunta.

En tiempo de inferencia las representaciones se encuentran ya entrenadas y el índice construido. Al llegar al sistema una nueva pregunta, se obtendrá su representación y a través del índice se obtendrá una lista de usuarios candidatos a ser *rankeados* en la siguiente etapa del sistema.

Hemos evaluado dos estrategias para el armado de las representaciones:

Similaridad semántica de las preguntas Construimos las representaciones semánticas de las preguntas utilizando Doc2Vec [Le and Mikolov, 2014], entrenado sobre nuestros datos. Luego utilizamos un SBERT [Reimers and Gurevych, 2019] preentrenado y lo ajustamos con nuestro conjunto de datos. Insertamos en el índice de Faiss la representación semántica de cada pregunta, y al llegar una nueva pregunta obtenemos preguntas similares a partir del índice; de esta manera los usuarios candidatos a *rankear* serán los autores de las respuestas aceptadas de esas preguntas similares. El ajuste y los resultados de esta estrategia se describen en las subsecciones 4.1.1 y 4.1.2.

***Embeddings* de usuarios** Proponemos una metodología para construir *embeddings* de la habilidad de respuesta de los usuarios, y a la vez generar una transformación para las preguntas que las lleva al mismo espacio de habilidad de respuesta de los usuarios,

intentando ubicar las preguntas cerca de los usuarios que las respondieron. El índice en Faiss contiene en este caso los *embeddings* de los usuarios. De esta manera, al llegar una nueva pregunta se obtiene su representación semántica, se le aplica la transformación y se busca a partir de ella en el índice de usuarios a los candidatos a *rankear*. Esta estrategia y sus resultados se describen en la subsección 4.1.3.

4.1.1. Similaridad semántica con Doc2Vec

Para el ajusta Doc2Vec utilizamos el conjunto de entrenamiento, para generar los tokens de entrada se uso la función `simple_preprocess` de gensim[Wang et al., 2023]. El entrenamiento del modelo Doc2Vec se realizó con esta misma biblioteca.

Los siguientes análisis corresponden a un modelo Doc2Vec de 50 dimensiones entrenado por 50 *epochs* con 50 mil preguntas. El entrenamiento demoró al rededor de 7 minutos, y el tiempo de inferencia para la creación de todos los vectores a almacenar en el índice fue de 3 horas, en una computadora con 16Gb de RAM y un i5 de 16 núcleos.

Evaluamos la calidad de la representación a través del *recall* ofrecido por los primeros k candidatos sugeridos por el índice sobre un conjunto de validación. En la figura 4.1 mostramos el *recall* medio por pregunta obtenido para distintos valores de k. Observamos que el *recall* medio al extraer 10 mil candidatos es de 15 %, mientras que extrayendo 50 mil candidatos (que representan 5.57 % de los usuarios totales) se obtiene un *recall* del 25 %. Uno de los problemas que encontramos con este método y el de SBERT es que no todas las preguntas tenían respuestas aceptadas. Otro problema es que había usuarios que responden preguntas, pero nunca llegaron a tener una respuesta aceptada como pudimos ver en la sección 3.2.

Para entender mejor estos resultados analizaremos cómo están distribuidas relativamente las preguntas en el espacio de representación. En particular, mediremos la distancia entre pares de preguntas generados de distinta forma:

(1) Pares de preguntas (p_i, p_j) al azar.

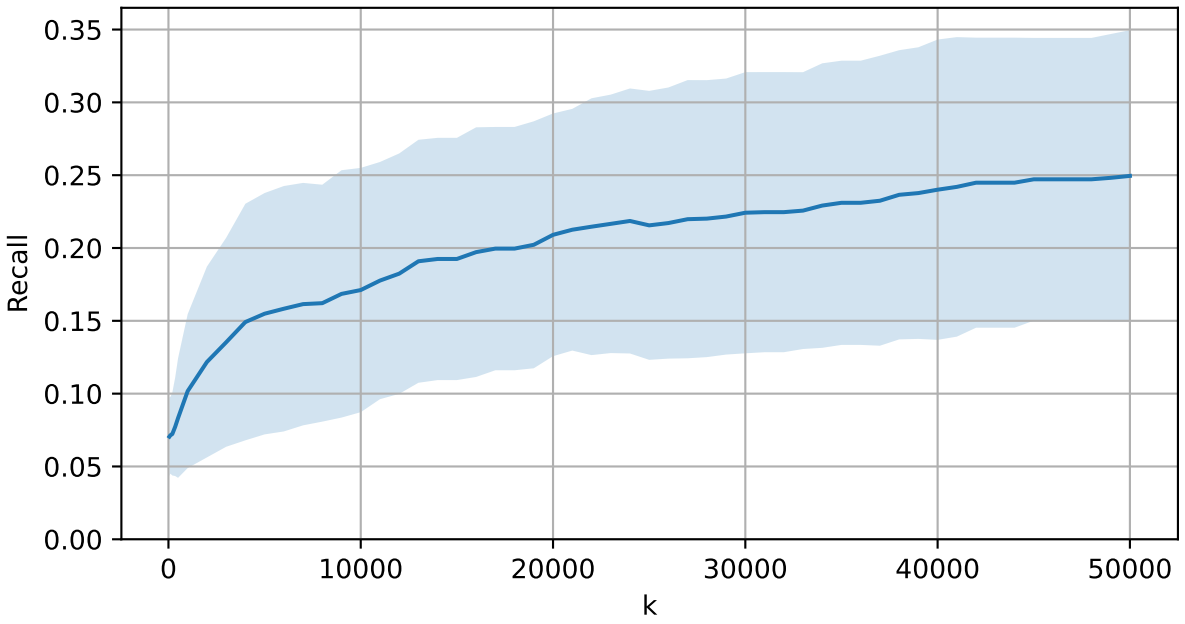


Figura 4.1: *Recall* medio y desvío estándar para las primeras k posiciones con la representación de las preguntas usando Doc2Vec. Para un conjunto de validación de preguntas al azar se busca las k preguntas mas similares y se obtiene los usuarios con la respuesta aceptada, si tienen, con estos usuarios se calcula cuantos fueron recuperados de los usuarios que respondieron las preguntas originales. Con las métricas de todas las preguntas se calcula la media y el desvío estándar para cada valor de k.

(2) Todos los pares de preguntas (p_i, p_j) tales que algún usuario de los que respondieron p_i tuvo la respuesta aceptada en p_j, para algunas preguntas p_i al azar. Estos pares representan el objetivo de nuestro modelo.

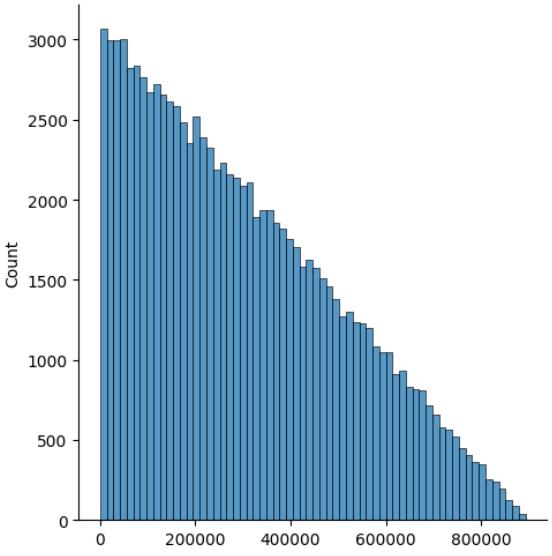
(3) Pares de preguntas $(p_i, c_j(p_i))$ obtenidas del índice, usando los primeros 50000 candidatos $c_j(p_i)$ de algunas preguntas p_i al azar.

En la figura 4.2 se muestran los histogramas de las distribuciones. Observamos que los pares generados totalmente al azar (1) muestran distancias mucho más grandes en relación a las obtenidas a través del índice o a las preguntas emparejadas. En los pares objetivo (2) observamos que la media es cercana 12, mientras que en los pares extraídos del índice (3) es cercana a 10. Esto nos señala que la representación está ubicando cerca a preguntas que son similares semánticamente, pero que no comparten los usuarios que buscamos y que se ven en los pares objetivo (2). O sea que preguntas que son parecidas semánticamente no necesariamente requieren las mismas habilidades para ser respondidas.

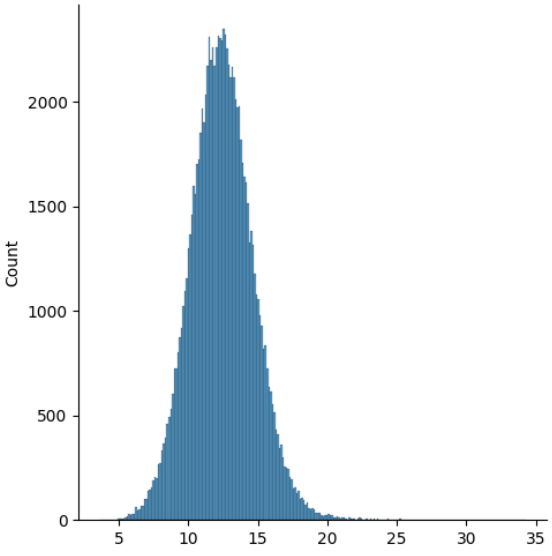
4.1.2. Similaridad semántica con SBERT

En el modelo previo observamos que usando sólo la similaridad semántica la representación ubica más cerca a pares de preguntas que no son nuestro objetivo. En este contexto, la incorporación de SBERT al proceso nos permite ajustar la representación semántica de los documentos, al incorporar en su mecanismo de *attention* (atención) a nuestros pares de documento objetivo como se vio en 2.1.2.

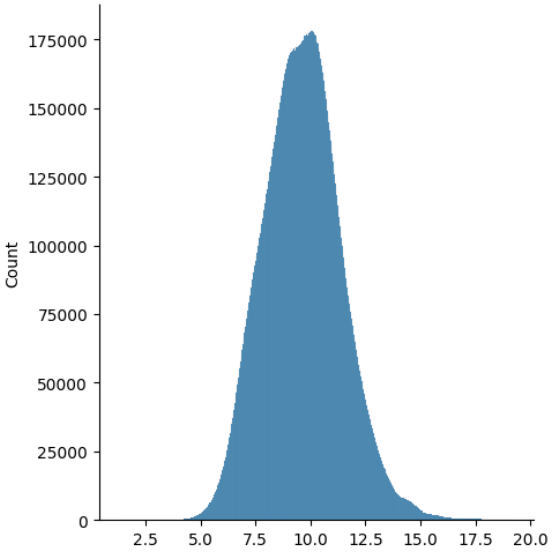
Utilizamos un modelo SBERT preentrenado (`nli-distilroberta-base-v2`[Liu et al., 2019] [Sanh et al., 2019]) y lo ajustamos con un conjunto de preguntas emparejadas similar al segundo grupo de la sección previa, en donde los usuarios que respondieron una pregunta son los que tenían la respuesta aceptada en la otra pregunta. De esta manera la representación de las preguntas incorpora la capacidad de los usuarios para representarlas. Entonces cuando se busquen candidatos en las preguntas cercanas los usuarios con respuestas aceptadas también van tener la capacidad de responder la pregunta original.



(a) Caso 1



(b) Caso 2



(c) Caso 3

Figura 4.2: (1) Distancias entre preguntas seleccionándolas al azar, usando Doc2Vec para la representación; (2) Distancias entre preguntas objetivo, usando Doc2Vec para la representación; (3)Distancias entre preguntas cercanas, seleccionando los candidatos por cercanía usando Doc2Vec para la representación.

Los siguientes análisis corresponden a un SBERT de 768 dimensiones entrenado durante una *epoch* con 200000 preguntas. El entrenamiento demoró 38 horas, mientras que crear todos los vectores para guardar en el índice tomó 37 horas, en una computadora con 16Gb de RAM y un i5 de 16 núcleos. La principal limitación de este método es que es poco escalable, y el ajuste resulta lento por la cantidad de parámetros que tiene el modelo. El *finetuning* de SBERT no alcanzo a capturar las dimensiones esperadas, como se ve en el resultados del *recall* que mostramos a continuación.

Nuevamente, para tener una idea de la calidad de la representación analizamos su *recall* para diferentes k candidatos extraídos del índice sobre un conjunto de validación. Se puede ver en la figura 4.3 que el *recall* medio trayendo 10000 documentos es de 16 %, y trayendo 50000 documentos, que representan 5.57 % de los usuarios totales, de 22.5 %. Es decir que este método trae mejores resultados que Doc2Vec para 10000 documentos, pero peores para 50000 documentos.

Dejamos como línea de trabajo futuro realizar entrenamientos por más tiempo o con un mejor hardware. Otra posibilidad a evaluar sería utilizar un arreglo alternativo incorporando una capa al final de SBERT y entrenar sólo esa capa con los documentos emparejados, el conjunto (2) de la sección 4.1.1, permitiendo entrenar mas rápido al ajustar menos parámetros.

4.1.3. Embeddings de usuarios

Como notamos en la sección anterior en la estrategia de similaridad semántica, el conjunto de usuarios candidato para una pregunta está limitado a aquellos que tuvieron una respuesta aceptada. Esto impone una cota superior en el *recall* que podemos obtener para una pregunta: aun si extrajéramos del índice el 100 % de las preguntas, el *recall* no necesariamente sería del 100 % porque hay usuarios que responden preguntas, pero nunca tuvieron una respuesta aceptada como vimos en la sección 3.2.

En esta nueva estrategia partimos de las representaciones de las preguntas generadas por Doc2Vec, dado que son más rápidas de generar, y construimos una red neuronal como la mostrada en la Figura 4.4 para crear *embeddings* de usuarios y una transformación de las preguntas. A partir

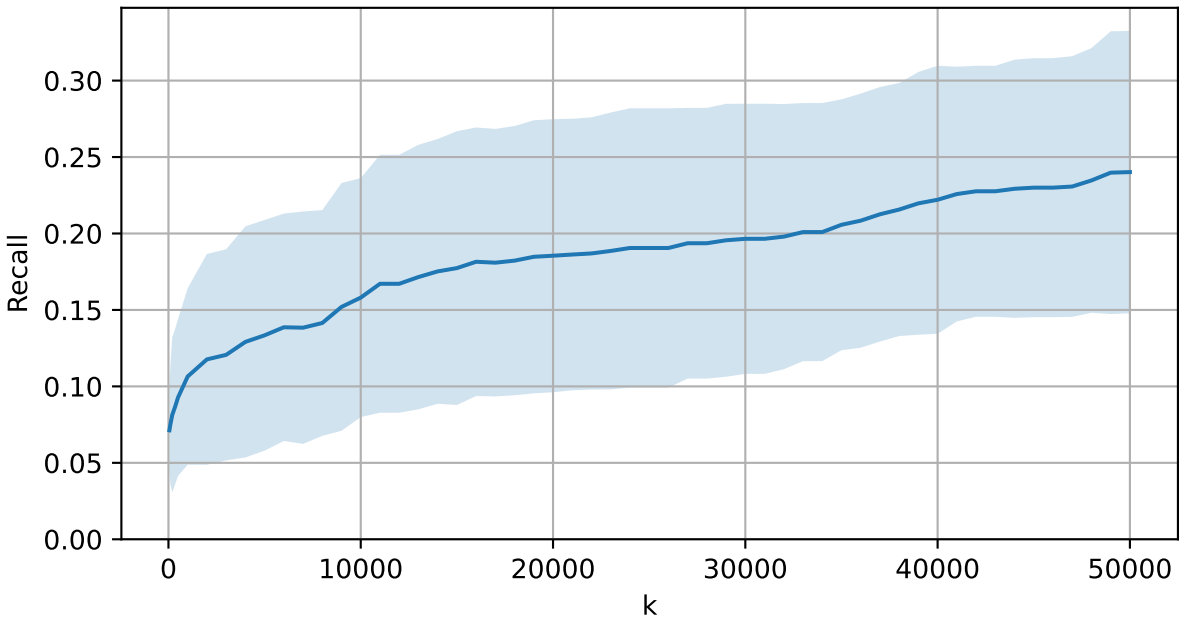


Figura 4.3: *Recall* medio y desvío estándar para las primeras k posiciones con la representación de las preguntas usando SBERT. Para un conjunto de validación de preguntas al azar se busca las k preguntas mas similares y se obtiene a los usuarios con la respuesta aceptada, si dicha respuesta aceptada existe. De aquí se calcula cuántos de los usuarios que respondieron las preguntas originales fueron recuperados. Finalmente, con las métricas de todas las preguntas se calcula la media y el desvío estándar para cada valor de k.

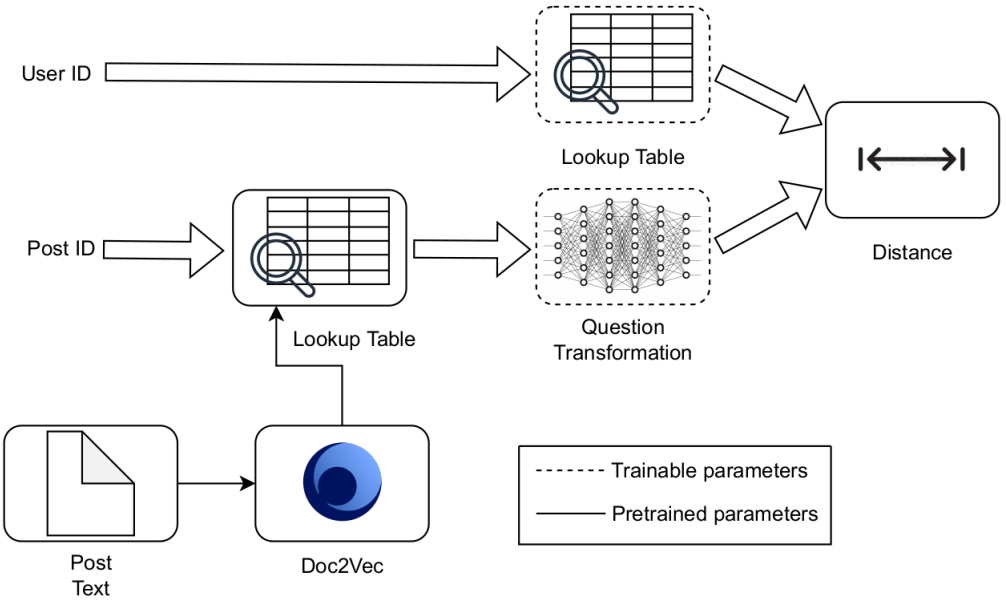


Figura 4.4: Arquitectura para entrenar embeddings de usuario y transformar las preguntas.

de la representación de las preguntas, con los parámetros preentrenados, se encadenan una red neuronal para transformar las preguntas llevándolas al mismo espacio en el que se va a crear una representación para la habilidad de respuesta de los usuarios. El modelo completo se ajusta con la distancia entre las preguntas y los usuarios que tienen la habilidad para responderlas, dejando cerca preguntas y usuarios que pueden responderlas.

En la Figura 4.5 se observa la estructura de la red neuronal usada para entrenar la transformación de las preguntas y el *embedding* de usuarios. La rama derecha genera el *embedding* de los usuarios, mientras que la rama izquierda transforma las preguntas al espacio de usuarios. Ambos *embeddings* están normalizados para que cuando se calcule la distancia euclídea sea una transformación monotónica de las distancia coseno. La etapa final del modelo calcula la distancia entre ambas y posteriormente su norma. Durante el entrenamiento la distancia objetivo entre una pregunta y un usuario será 0 si el usuario respondió la pregunta, o 2 en caso contrario.

El modelo se entrenó con 1 millón de preguntas y usuarios cercanos y 2 millones de preguntas y usuarios distantes durante 30 *epochs* con una dimensión de 10 para los usuarios, y demoró 2 horas en entrenar, en una computadora con 16Gb de RAM y un i5 de 16 núcleos. En la figura

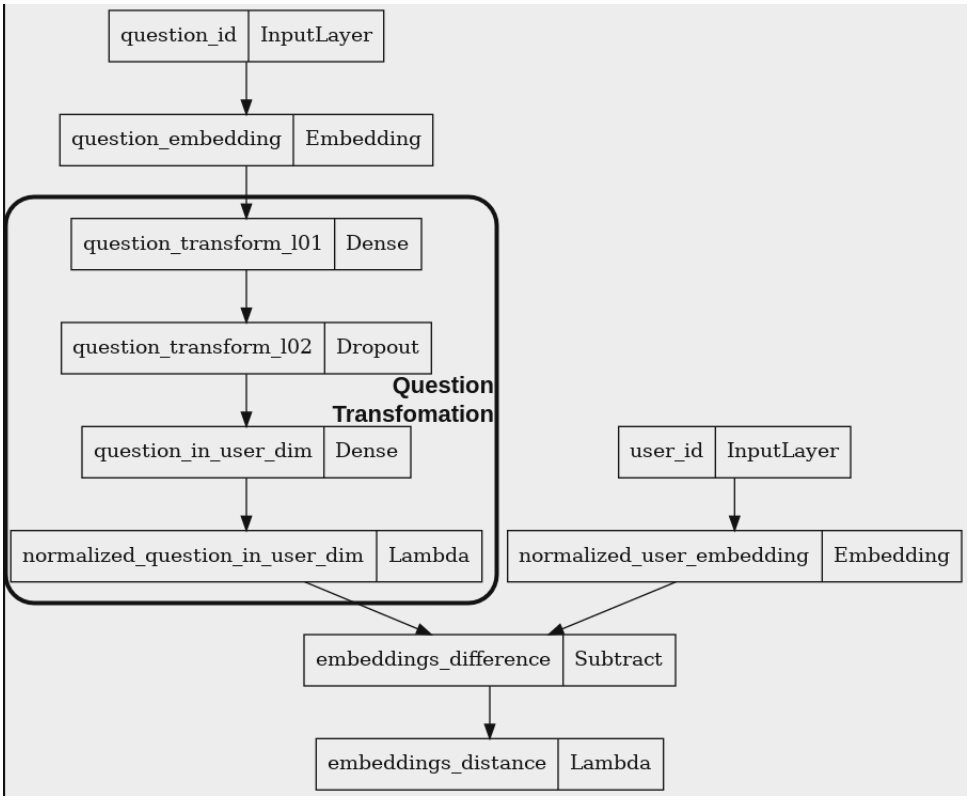


Figura 4.5: Capas de la arquitectura neuronal y operadores para entrenar los *embeddings* de usuario y la transformación de las preguntas.

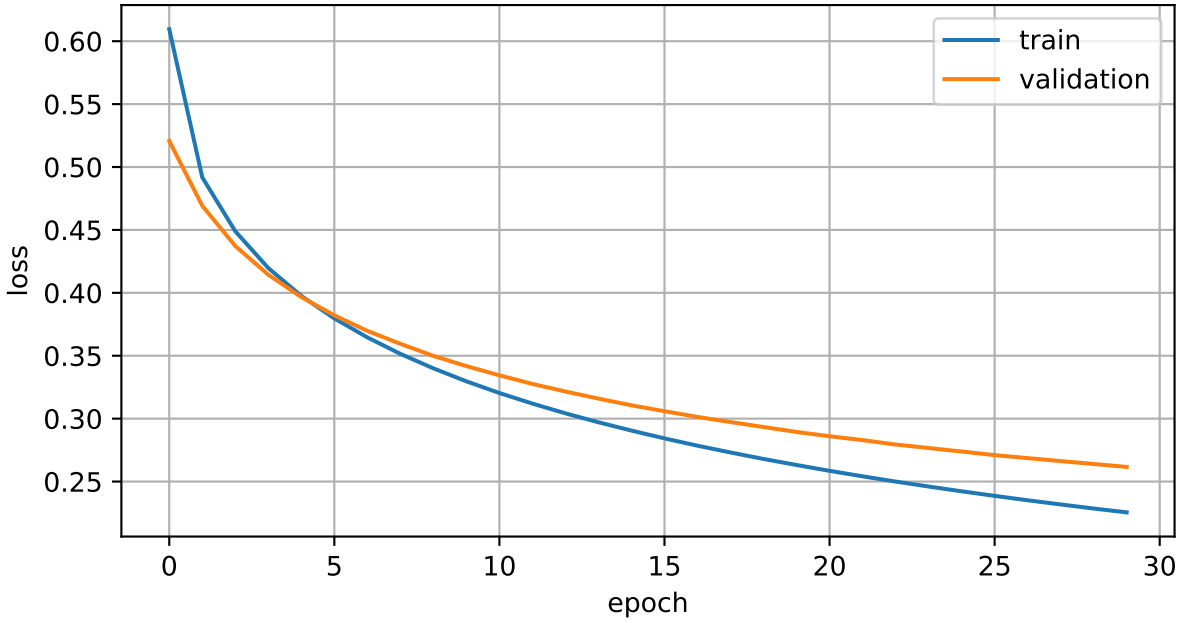


Figura 4.6: Valores de la función de pérdida (MSE) durante las *epochs* del entrenamiento del modelo de *embeddings* de usuarios para el conjunto de entrenamiento y el conjunto de validación.

4.6 se puede ver la pérdida a lo largo de las *epochs*, se puede ver que el entrenamiento converge y el conjunto de validación no muestra signos de *overfitting*. Las preguntas y usuarios distantes se construyeron tomando preguntas y usuarios al azar que no las hayan contestado.

Nuevamente para tener una idea de la calidad de la representación analizamos su *recall* los primeros k candidatos extraídos del índice sobre un conjunto de validación. Se puede ver en la figura 4.7 que el *recall* medio extrayendo 10000 usuarios es de 27 %, mientras que extrayendo 50000 usuarios (que representarían el 5.57 % de los usuarios totales) el *recall* pasa al 50 %. De esta manera, la metodología supera a las representaciones que sólo usaban la similaridad semántica. A su vez, esta metodología requiere menos espacio de almacenamiento en el índice, ya que solamente tiene que almacenar la representación de los usuarios, que son muchos menos que las preguntas; esto conlleva un mantenimiento más sencillo del modelo.

Otra ventaja de este esquema es que si el k que se trae del índice es igual al total de usuario el *recall* sera del 100 % sobre el conjunto de validación, porque va a traer a todos los usuarios,

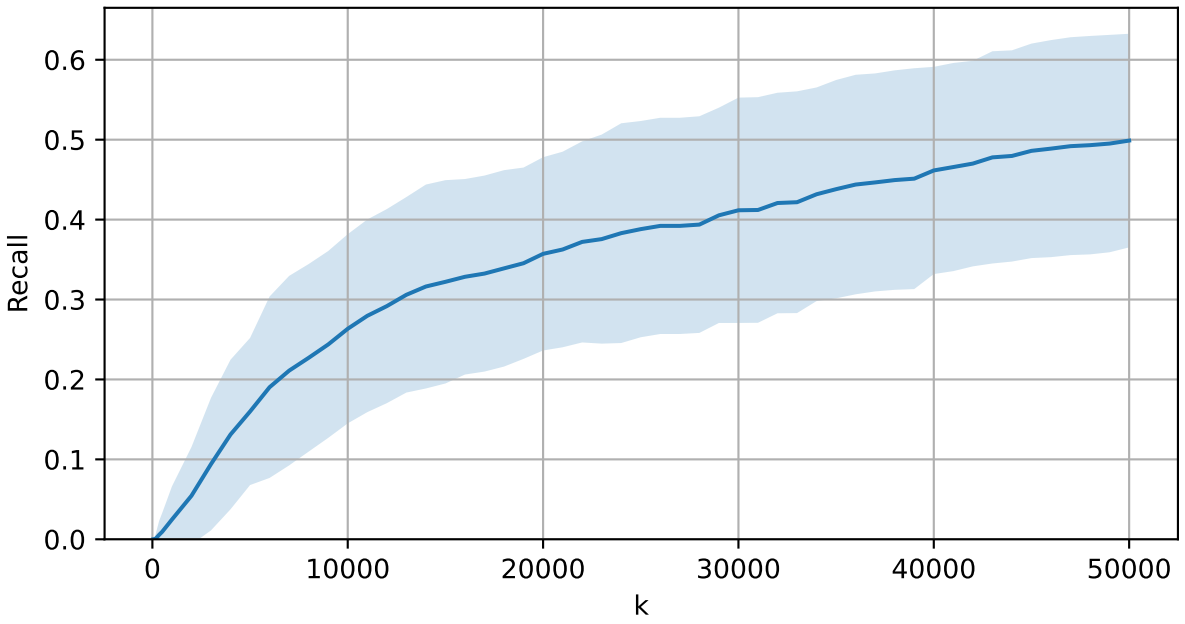


Figura 4.7: Recall medio y desvío estándar para las primeras k posiciones con la representación de las preguntas usando Doc2Vec junto con la transformación y el *embedding* de usuarios. Para un conjunto de validación de preguntas al azar se transforma al espacio de *emebeddings* de usuarios y se trae los k usuarios mas cercanos, con estos usuarios se calcula cuantos fueron recuperados de los usuarios que respondieron las preguntas originales. Con las métricas de todas las preguntas se calcula la media y el desvío estándar para cada valor de k.

incluyendo los de cualquier respuesta en particular. Por último, los usuarios nuevos o con pocas muestras pueden recibir como *embeddings* el promedio de los usuarios que compartan sus mismas etiquetas, tomando una visión del mundo WAE.

4.2. Implementación del algoritmo de Learning to Rank

Después de seleccionar los candidatos, necesitamos ordenarlos acorde a su probabilidad de responder de mejor manera la pregunta. Para esta tarea entrenamos un modelo de *learning to rank* usando los *scores* de las respuestas de usuarios previamente observados. En particular usamos el *XGBRanker* de la biblioteca *xgboost*, cuya implementación se basa en LambdaMART, visto en la Sección 2.2.3, que es un *pairwise ranker*.

Con la representación entrenada y los usuarios cargados en el índice, armamos un conjunto de datos de entrenamiento como se describe en la Sección 3.2. Los *features* usados para el modelo son los *embeddings* de usuarios entrenados para la representación, mientras que para las preguntas se calculó su representación con *doc2vec* y luego se obtuvieron sus *features* con la transformación usada anteriormente.

4.2.1. Entrenamiento

Se usó NDCG como función objetivo para el entrenamiento. En la Figura 4.8 vemos la pérdida en el conjunto de validación durante las iteraciones de entrenamiento. Se puede observar que el NDCG va mejorando con el transcurso de las *epochs*, y llegando a las 200 iteraciones la tasa de mejora comienza a aplanarse. Con el modelo ya entrenado se obtiene un NDCG de 0.7676 sobre el conjunto de pruebas.

4.2.2. Baseline

Para comparar el desempeño del sistema completo armamos una recomendación *baseline* con el conjunto de entrenamiento. Para este modelo *baseline*, la lista de candidatos iniciales era

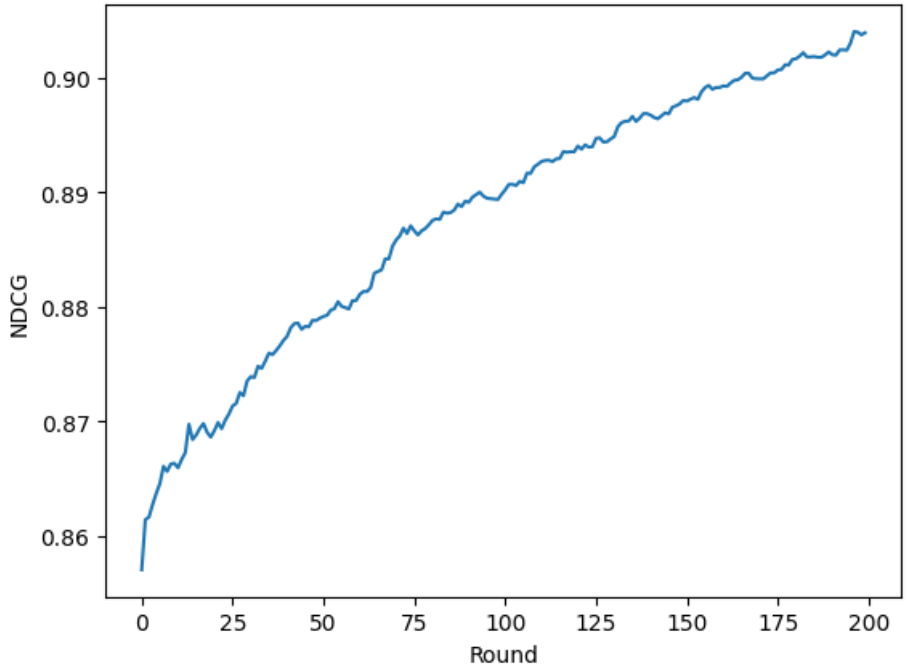


Figura 4.8: Valores de la función de pérdida (NDCG) durante los *rounds* del entrenamiento del modelo de *LTR* XGBoost Ranker sobre el conjunto de validación.

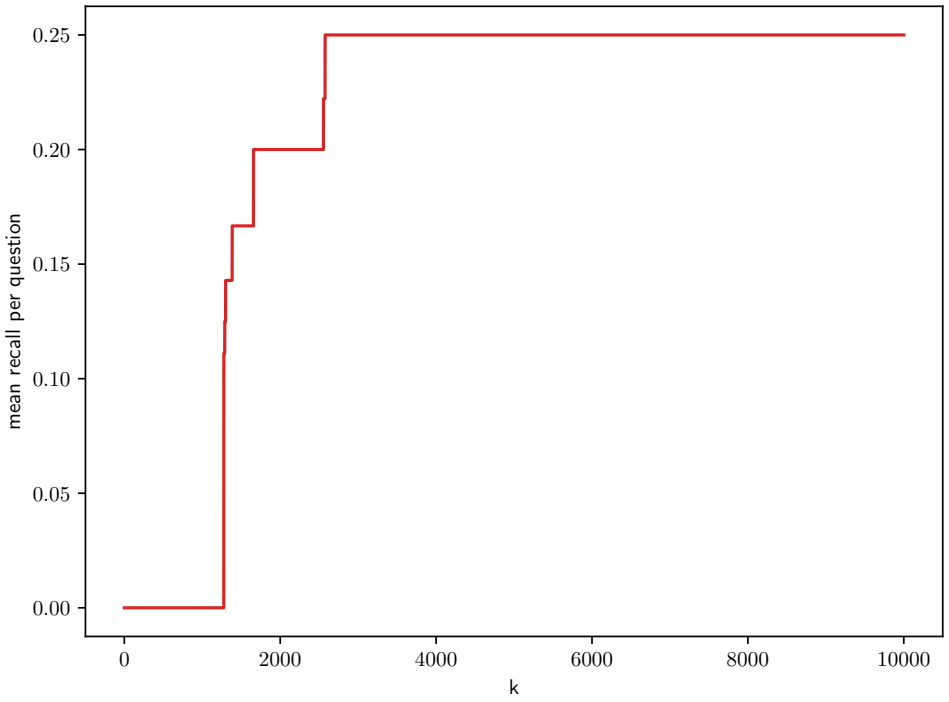


Figura 4.9: Recall medio sobre el conjunto de pruebas para la estrategia *baseline*. Esta estrategia toma los k candidatos con mayor *score* acumulado parar preguntas sobre `python` antes del 2022-01-01.

siempre la misma, y estaba conformada por los candidatos con mayor *score* acumulado parar preguntas sobre `python` antes del 2022-01-01.

En la Figura 4.9 se muestra el *recall* para los primeros k candidatos recomendados sobre el conjunto de pruebas. Encontramos que ninguno de los primeros 1275 candidatos recomendados respondieron las preguntas. Luego, el recall comienza a crecer hasta los 2575 candidatos, en donde llega al 25 %. Este es su máximo y ninguno de los siguientes candidatos mejora la media del *recall*.

4.2.3. Pruebas

Tomando el conjunto de pruebas medimos el *recall* del sistema completo, usando los mejores k candidatos de acuerdo al score extraído del *ranker*.

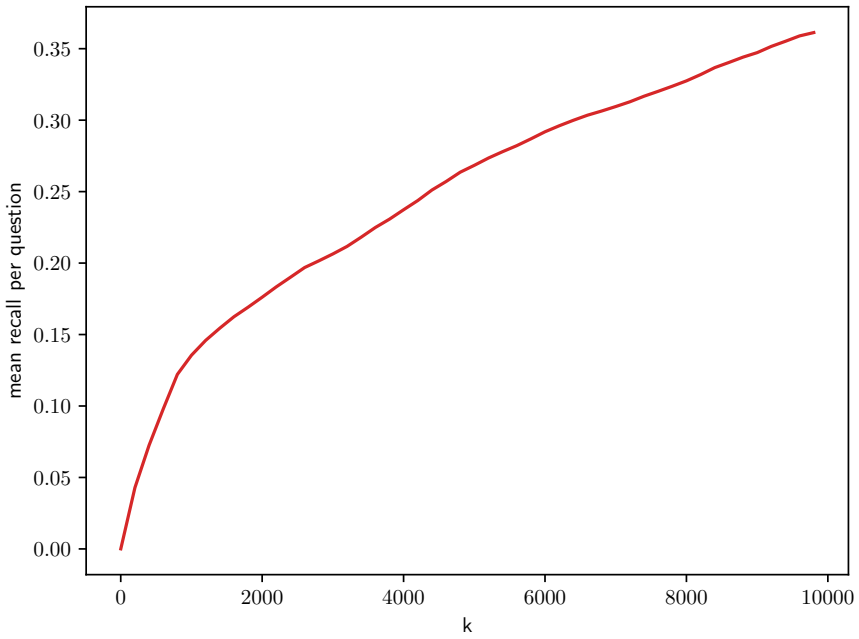


Figura 4.10: *Recall* medio para las primeras k posiciones para el sistema completo sobre el conjunto de pruebas. El sistema completo para cada pregunta la transforma al espacio del *embedding* de usuarios, toma los usuarios candidatos del índice, los pasa por el XGBoost Ranker entrenado y se queda con los primeros k candidatos con mejor puntaje.

Los resultados de esta evaluación se muestran en la Figura 4.10. Se puede ver que la *recall* mejora de forma sostenida al aumentar la cantidad de candidatos recomendados. Con mil candidatos recomendados se obtiene un 13 % de *recall*, y se llega a más de un 35 % para los primeros diez mil. Comparando con el baseline, para diez mil candidatos el *recall* de nuestro modelo es superior por una diferencia del 0,10.

Mirando las primeras posiciones (que son las más importantes para un sistema que va a notificar a usuarios, dado que no es factible inundar a una gran parte de la plataforma con recomendaciones por cada pregunta nueva), podemos ver que ya antes la posición 1000 el sistema supera ampliamente la recomendación *baseline*, que se mantenía en cero.

Otra observación relevante es que el baseline, recomendando una y otra vez a las mismas personas

que son las que mejor responden, generaría un sesgo que contradice *per se* a cualquier noción de equidad que queramos garantizar.

4.3. Resolución del problema de equidad en los *rankings*

Entre nuestros objetivos, nos interesa que el sistema de recomendación ofrezca las preguntas a usuarios potencialmente candidatos a responder entre una variedad de países con distinto nivel de desarrollo y/o penetración de la tecnología.

Dado que la cantidad de usuarios en cada país en StackOverflow es muy heterogénea, realizaremos una clasificación en dos grupos de países: de alto desarrollo humano y de medio/bajo desarrollo humano, los llamaremos IDH^+ e IDH^- respectivamente en el resto del documento.

Con el sistema completo entrenado como explicamos hasta aquí, y sin realizar ninguna corrección de equidad, hacemos una primera evaluación de su equidad con diferentes métricas que consideramos relevantes. En una segunda etapa elegimos un algoritmo de corrección de equidad para sistemas de recomendación y comparamos las métricas con las calculadas anteriormente.

De todos los algoritmos revisados para corregir la equidad elegimos FA*IR. Elegimos este algoritmo porque su punto de mitigación es el post-procesamiento; esto nos permite aplicar las correcciones sobre los resultados del sistema que ya teníamos entrenado. De esta manera, es mas fácil corregir sistemas que ya se encuentran en funcionamiento. Nosotros utilizamos la versión binaria del algoritmo, aunque en un *paper* posterior los autores extienden su algoritmo para soportar múltiples grupos protegidos. El método también permite tomar más de un atributo sensible a la vez, lo que nos posibilitaría en un futuro agregar otros atributos sensibles –por ejemplo, el género– al mismo tiempo que mantenemos el IDH. Finalmente, el algoritmo permite variar un parámetro para seleccionar la rigurosidad de los test de hipótesis, y por consiguiente la cantidad mínima de candidatos esperados en las posiciones para considerar justo al *ranking*, dándole así cierta flexibilidad. El algoritmo desarrollado en FA*IR está explicado en 2.3.

Analizamos cómo actúa el reordenamiento de FA*IR sobre los candidatos sugeridos y cómo esto

impacta en la paridad demográfica para las diferentes posiciones de la recomendación. Después comparamos la igualdad de oportunidades y la paridad predictiva, que pueden pensarse como una precisión y *recall* condicionadas por valor que toma el atributo sensible en los candidatos sugeridos.

Todos los resultados que mostramos en esta sección están calculados sobre el conjunto de prueba formado por los *posts* realizados a partir de 2022 como se describe en la sección 3.2.

4.3.1. División de los usuarios según Índice de Desarrollo Humano

Como *proxy* para medir la barrera de entrada a la tecnología de los países usamos el índice de desarrollo humano. De esta manera los usuarios que pertenecen a países con mayor IDH van a tener una barrera de entrada a las tecnologías menor. Entonces buscaríamos corregir la equidad del sistema para que las personas de países una barrera tecnológica mas alta pueda tener las mismas oportunidades.

Para determinar los países de alto y bajo IDH usamos un *dataset* del United Nations Development Programme para el año 2021. Después de corregir *typos* y traducir algunos nombres de países para que tengan el mismo nombre en el *dataset* y en la base de datos se armo la junta. De 3941484 usuarios con país se pudo asignar IDH a 3872918, mas de un 98

En la figura 4.11 se puede ver la distribución de IDH por usuarios. Se ve que la distribución no tiene una forma típica. El primer pico corresponde a india con un IDH de 0.633 con mas de 700 mil usuarios, el otro pico corresponde a estados únicos con un IDH de 0.921 y 600 mil usuarios.

Se tomo un umbral de IDH mayor a 0.92 para hacer la separación entre IDH^+ e IDH^-. Esto dejo en el grupo IDH^+ a los 20 países con mayor IDH, esto corresponde aproximadamente al 36 % de los usuarios totales.

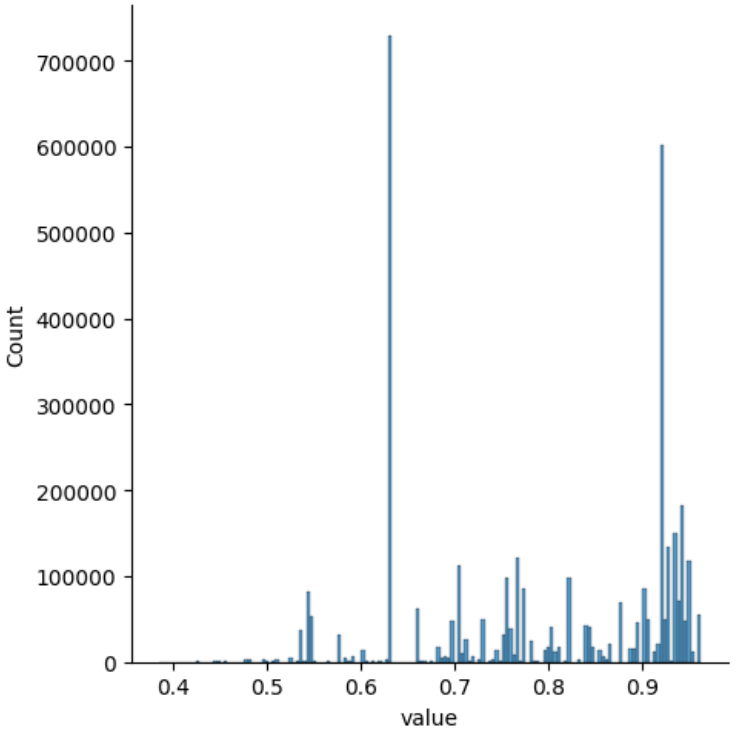


Figura 4.11: Distribución del índice de desarrollo humano por usuario.

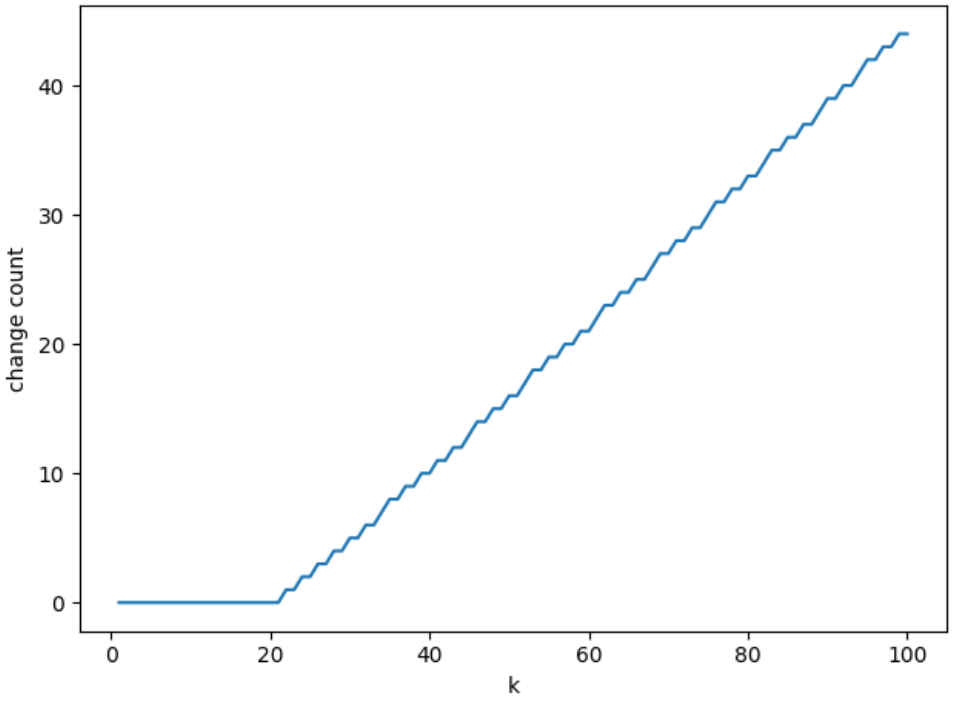


Figura 4.12: Cantidad de cambios hasta la posición k para el ranking generado para una pregunta particular.

4.3.2. Reordenamiento y paridad demográfica

Para entender como se comportaba FA*IR primero observamos la cantidad de cambios acumulados hasta la posición k. En la figura 4.12 se puede ver esta métrica para una pregunta en particular: observamos que el primer cambio se realiza en la posición 22, lo que significa que la cantidad de protegidos en las primeras 21 posiciones ya era justa en el ranking original que habíamos construído.

Continuando con el análisis de los cambios observamos las medias en la figura 4.13. Vemos que en las primeras 10 posiciones no hay cambios, después de la posición 30 se ve una tendencia linear hasta llegar a un poco mas de 40 cambios promedio en la posición número 100.

A continuación comparamos la cantidad acumulada de candidatos protegidos hasta la posición k, para el *ranking* antes original, después de haber corregido con FA*IR y el mínimo esperado por

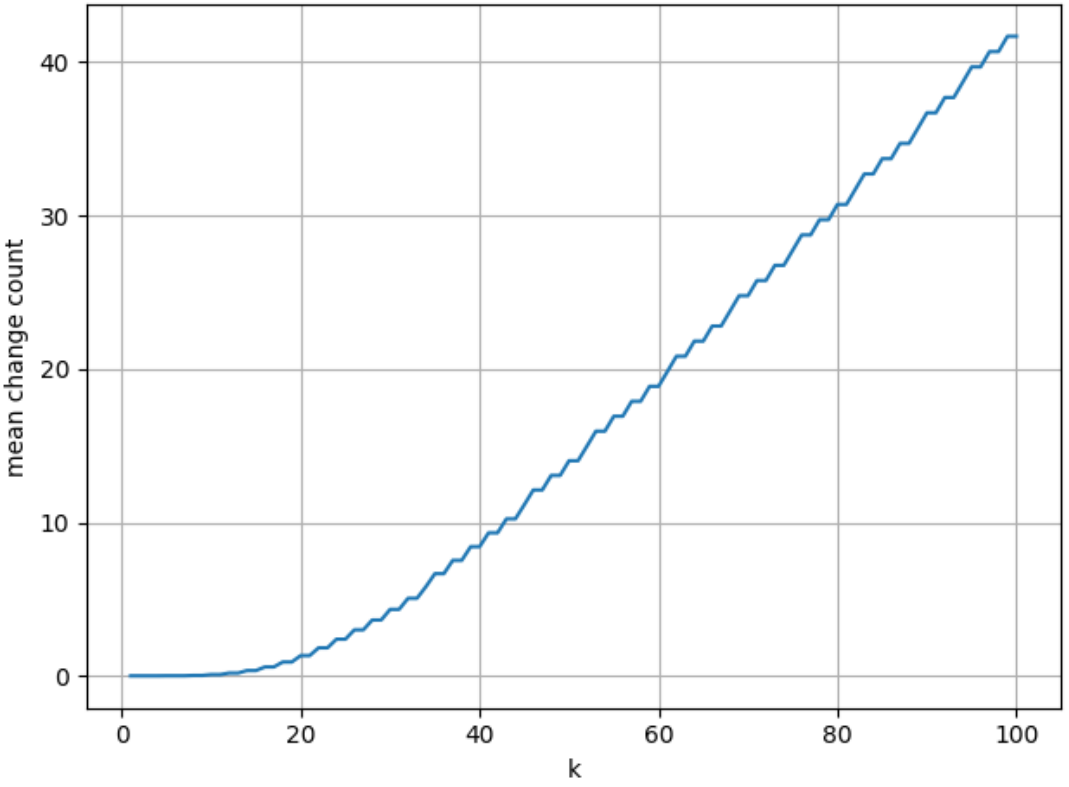


Figura 4.13: Cantidad media de cambios hasta la posición k promediando todas las preguntas.

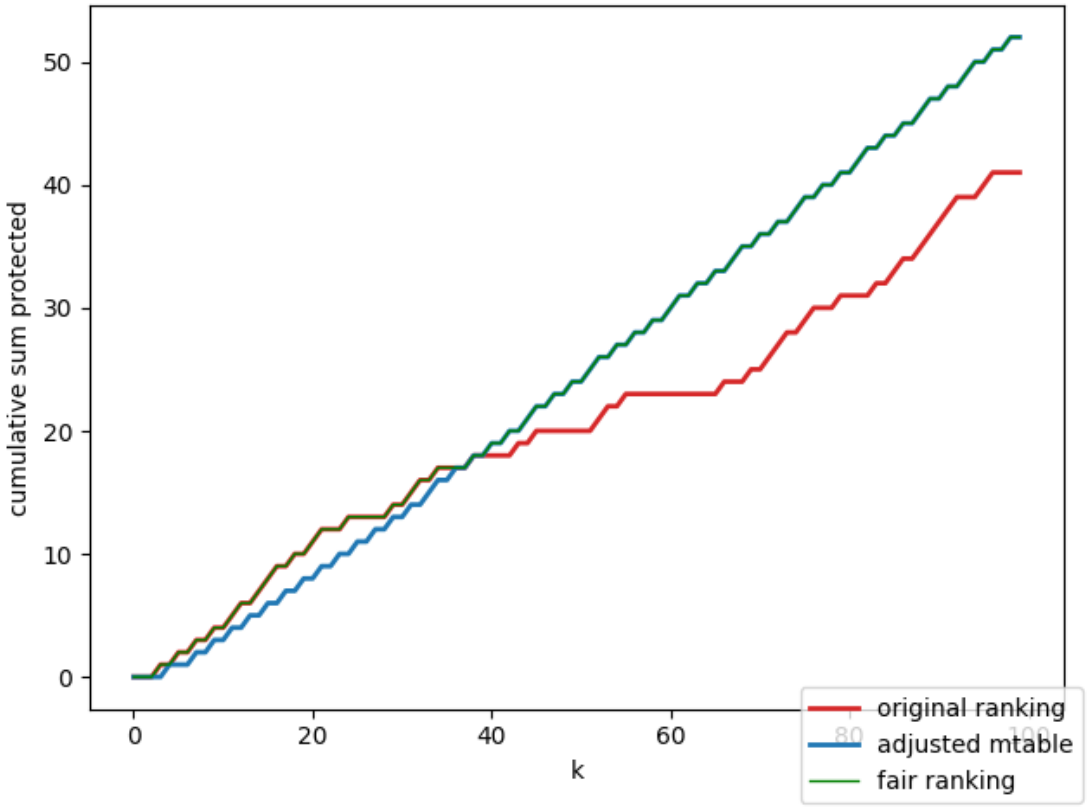


Figura 4.14: Suma acumulativa de candidatos protegidos hasta la posición k para un caso particular, en el *ranking* original, el justo y lo esperado por la *mtable*.

la *mtable*. En la figura 4.14 se gráfica un caso particular, se ve que al principio el ordenamiento con FA*IR sigue al original porque los valores están sobre el mínimo que espera la *mtable*, a partir de la posición 40 cuando el original queda por debajo de la *mtable* el ordenamiento con FA*IR sigue a la *mtable*.

En la figura 4.15 se ven las medias de la cantidad acumulada de candidatos protegidos hasta la posición k. Se ve que al rededor de la posición 20 la cantidad de candidatos protegidos en el *ranking* original empieza a queda por debajo de la *mtable*, en ese momento el *ranking* ordenando con FA*IR empieza a seguir los valores de la misma.

El nuevo ordenamiento generado por FA*IR afecta a la paridad demográfica. Para observar como fue el cambio en la figura 4.16 graficamos la proporción de candidatos en países IDH^{+}

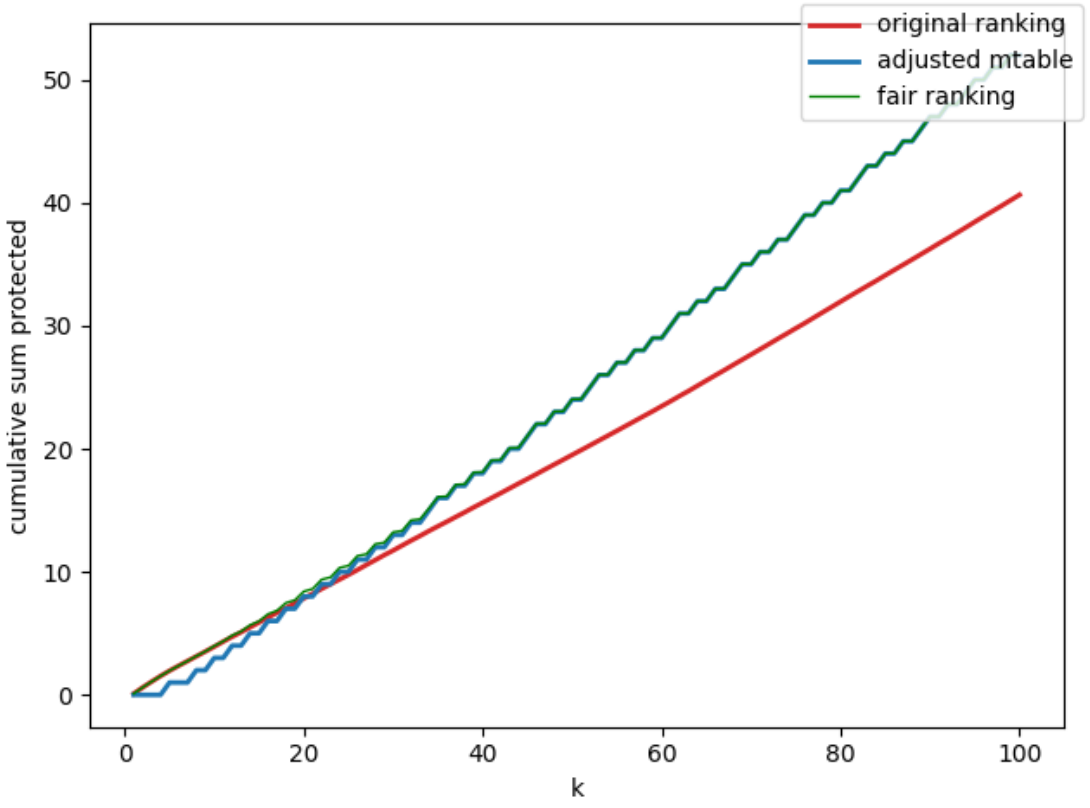


Figura 4.15: Media de la suma acumulativa de candidatos protegidos hasta la posición k, en el *ranking* original, el justo y lo esperado por la *mtable*.

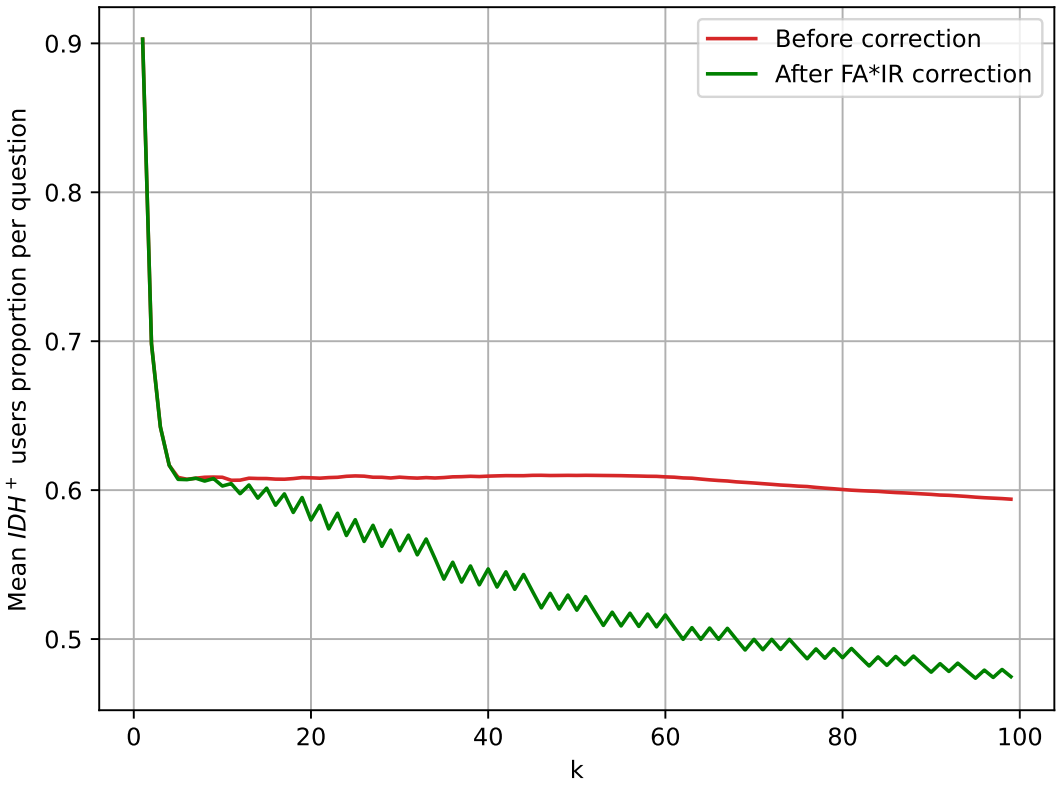


Figura 4.16: Proporción de candidatos de países desarrollados hasta la posición k.

acumulados hasta la posicion k comparando el ordenamiento original y el postprocesado con FA*IR. Se puede ver que en las primeras 5 posiciones la mayoría de los candidatos pertenecen a IDH^+. A partir de la quinta posición el ordenamiento original mantiene una proporcion de usuario IDH^+ alrededor del 60 %, superando el 36 % que seria justo. Dado que las primeras posiciones no tienen muchos cambios, la proporción de IDH^+ similar antes y después del postprocesado con FA*IR. Enfocándonos en los cambios que introdujo FA*IR en la Figura 4.17 podemos ver las mismas curvas para los k a partir de la quinta posición. Se puede ver que el después del post-procesamiento con FA*IR la proporción tiene una tendencia decreciente llegando a menos del 48 % en la posición 100. Con un k suficientemente grande debería llegar a su proporción justa del 36 %.

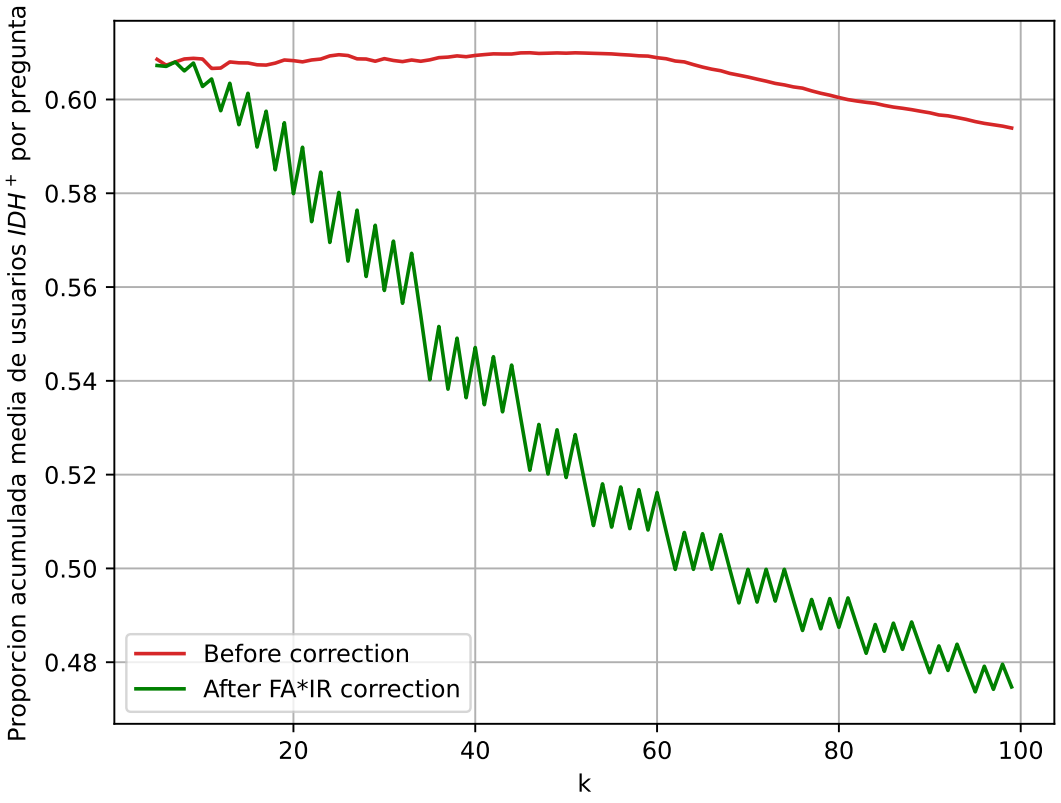
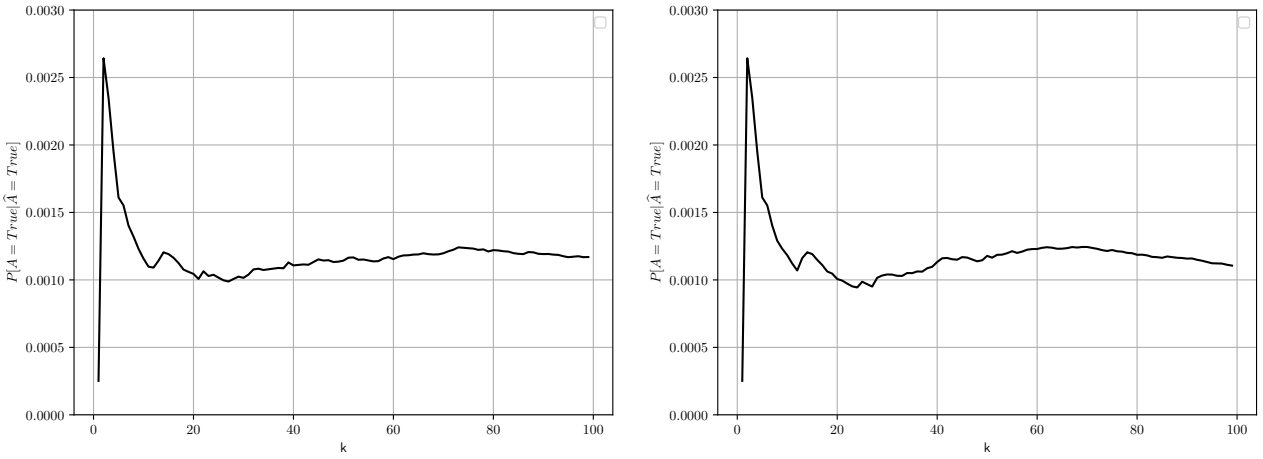


Figura 4.17: Proporción de candidatos de países desarrollados hasta la posición k, para k mayores a 4



(a) Antes de la corrección de equidad

(b) Después de la corrección de equidad

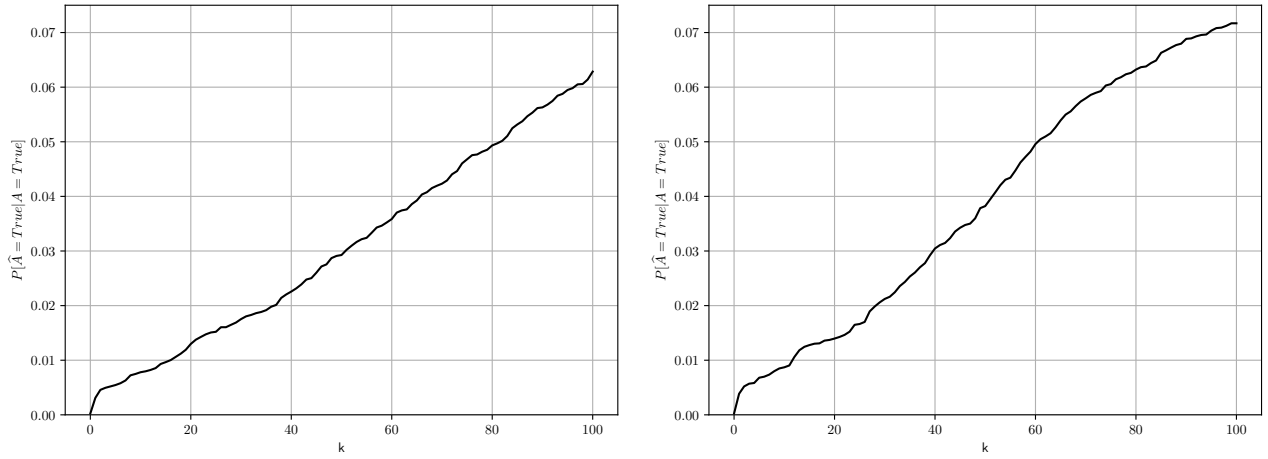
Figura 4.18: Cambio en la precisión con la corrección de equidad.

4.3.3. Performance

El primer análisis que realizaremos, antes de enfocarnos en la equidad, es cómo afecta el postprocesado con FA*IR al desempeño general del sistema. Para esto, en las figuras 4.18 y 4.19 comparamos la precisión y el *recall* para diferentes valores de k, antes (izquierda) y después (derecha) de aplicar la corrección de equidad, respectivamente.

En la figura 4.18 se puede ver que la precisión se mantiene similar, es decir, el reordenamiento no produjo grandes cambios en la precisión general del sistema, y los nuevos candidatos protegidos son de la misma calidad que los que estaban previamente en el listado.

Por otro lado, en la figura 4.19 se ve que hubo algunos cambios en el *recall*: para bajos valores de k, dado que hubo pocos reordenamientos, las curvas son parecidas. En cambio, para valores más grandes de k el reordenamiento produce una mejora visible en el *recall*. Esto podría deberse a que las puntuaciones de los usuarios que pertenecen a IDH^+ son en promedio mayores que las de aquellos que pertenecen a IDH^- (5.49 y 3.16 respectivamente), entonces el modelo de LTR los ordenó acorde con ello. Cuando el algoritmo de postprocesado los reordenada enfocado ahora en la paridad demográfica, podría estar adelantando usuarios de la clase protegida que



(a) Antes de la corrección de equidad

(b) Después de la corrección de equidad

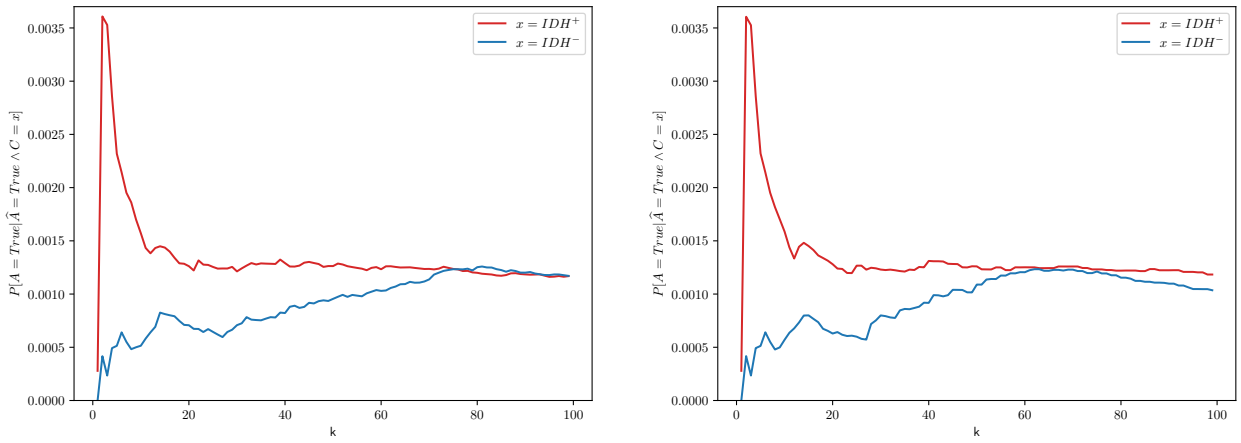
Figura 4.19: Cambio en la *recall* con la corrección de equidad.

habían quedado relegado a pesar de ser buenos candidatos, y por este motivo mejoraría el *recall*.

4.3.4. Igualdad de oportunidades y paridad predictiva

Después de analizar cómo cambiaban las métricas del sistema completo al usar FA*IR, buscamos entender ahora como afectó la mejora de equidad en la paridad demográfica (que es aquella implícita en el algoritmo FA*IR) a las métricas de igualdad de oportunidades y a la paridad predictiva.

Para analizar la paridad predictiva comparamos la precisión condicionada según los candidatos pertenezcan a IDH^+ o IDH^-, antes y después de aplicar FA*IR. En la figura 4.20 se pueden ver ambos gráficos. Comparando las figuras, encontramos que en las primera 20 posiciones de k las curvas de ambos grupos se mantienen iguales. Como mencionamos antes, esto podría deberse a que hay pocos reordenamientos en esas posiciones, a su vez es en estas posiciones en donde se nota mas marcadamente diferencia de precisiones entre los candidatos que pertenecen a IDH^+ y los que pertenecen a IDH^-, que privilegia a los primeros. Se puede ver que antes del reordenamiento (izquierda) a partir de la posición $k = 80$ los candidatos de ambos grupos tienen una precisión similar, mientras que después del reordenamiento esa precisión similar ya se



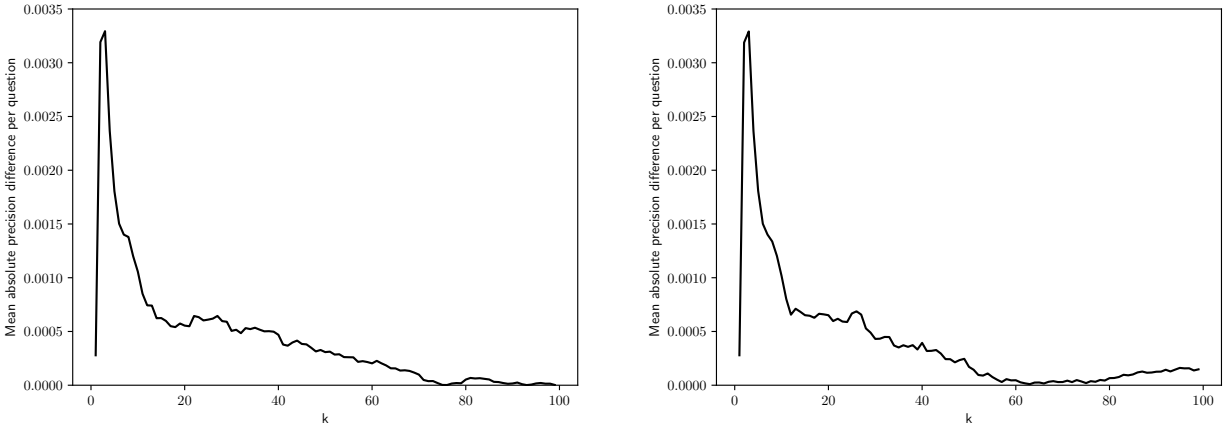
(a) Antes de la corrección de equidad

(b) Después de la corrección de equidad

Figura 4.20: Precisión según IDH antes y después de la corrección de equidad.

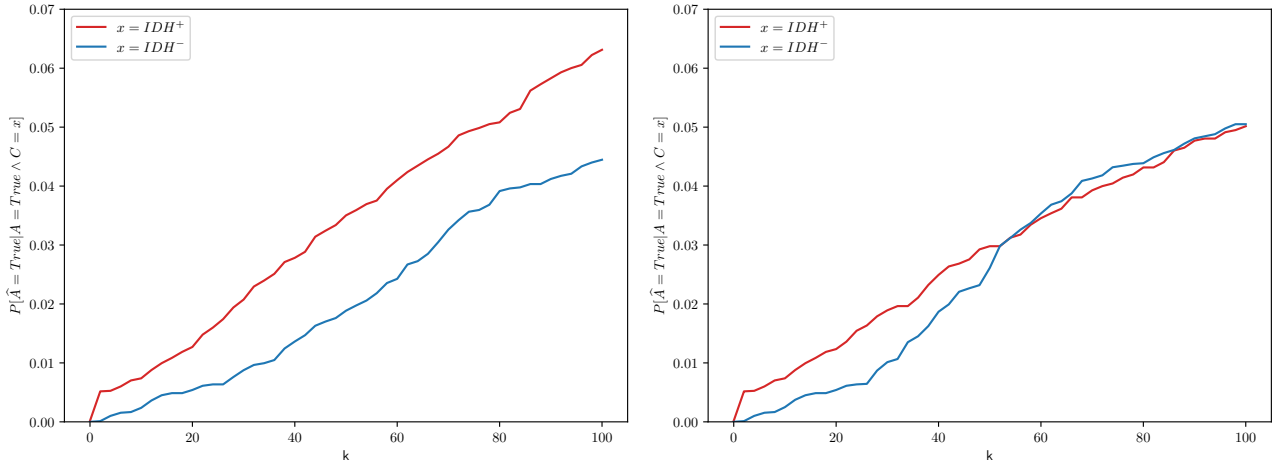
alcanza para $k = 60$. Sin embargo, haber adelantado los candidatos de mejor calidad hacia las posiciones bajas hace que en las posiciones posteriores la precisión del grupo IDH^- comience a bajar. En la figura 4.21 se puede ver la diferencia entre las precisiones de ambos grupos en valor absoluto. Aquí encontramos que la mayor diferencia se da en la quinta posición y es de $0,33\,\%$. En la posición $k = 40$ se observa una diferencia de cerca de $0,04\,\%$ tanto antes como después del reordenamiento, y a partir de esa posición las curvas se empiezan a diferenciar, achicándose la diferencia mas rápidamente después del postprocesado.

Por último, y con el fin de analizar la igualdad de oportunidades, en la figura 4.22 se pueden ver dos gráficos del *recall* condicionado a si los candidatos vienen o no de un país con alto IDH, antes y después de aplicar FA*IR. Nuevamente vemos que en las primeras 20 posiciones las curvas se mantienen iguales. Ya a partir de la posición $k = 40$ vemos que la brecha se reduce de $0,01$ a $0,006\,\%$. Finalmente, hacia $k = 60$ logra cerrarse de 0.015 a menos de 0.002. En la figura 4.23 se puede ver la diferencia en valor absoluto en el recall antes y después del reordenamiento. Observamos un cambio de tendencia en la posicion 40, endonde antes del postprocesado la brecha continuaba creciendo, mientras que después del reordenamiento la diferencia se hace más pequeña con valores crecientes de k.



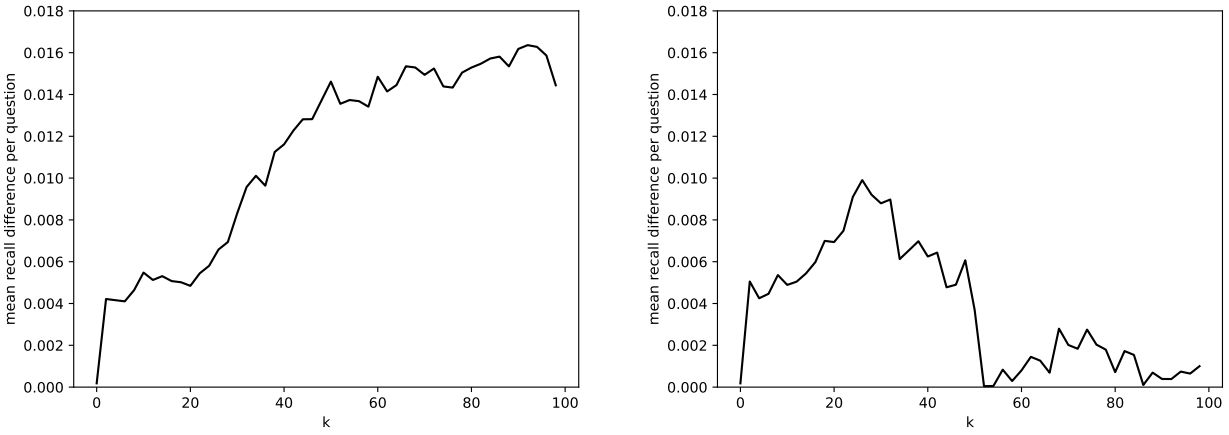
(a) Antes de la corrección de equidad (b) Después de la corrección de equidad

Figura 4.21: Diferencia absoluta de precisión entre países con alto y bajo IDH antes y después de la corrección de equidad.



(a) Antes de la corrección de equidad (b) Despues de la corrección de equidad

Figura 4.22: Recall segun IDH antes y despues de la correccion de equidad.



(a) Antes de la corrección de equidad

(b) Después de la corrección de equidad

Figura 4.23: Diferencia absoluta de recall entre países con alto y bajo IDH antes y después de la corrección de equidad.

4.4. Irregularidades introducidas por FA*IR

En esta sección exploramos el comportamiento de los cambios introducidos por el algoritmo FA*IR con el fin de entender que producía la forma de la curva de proporción de candidatos IDH^+ en la Figura 4.16. Con el problema identificado proponemos diferentes estrategias para mitigar la situación.

4.4.1. Análisis de las irregularidades

Para poder explicar los picos de la figura 4.16 analizamos cuál era la proporción de candidatos en cada posicion, sin acumular los candidatos hasta ese k. Esto se ve en la Figura 4.24 donde encontramos la proporción de usuarios que pertenecen a IDH^+ en el ordenamiento original y en el ordenamiento justo. Encontramos que el ordenamiento después de FA*IR difiere bastante de un ordenamiento orgánico. Se ve que a partir de $k = 50$ hay posiciones que son enteramente de candidatos protegidos y posiciones enteramente de candidatos no protegidos.

Para tratar de entender porque el algoritmo producía este irregularidad en las posiciones

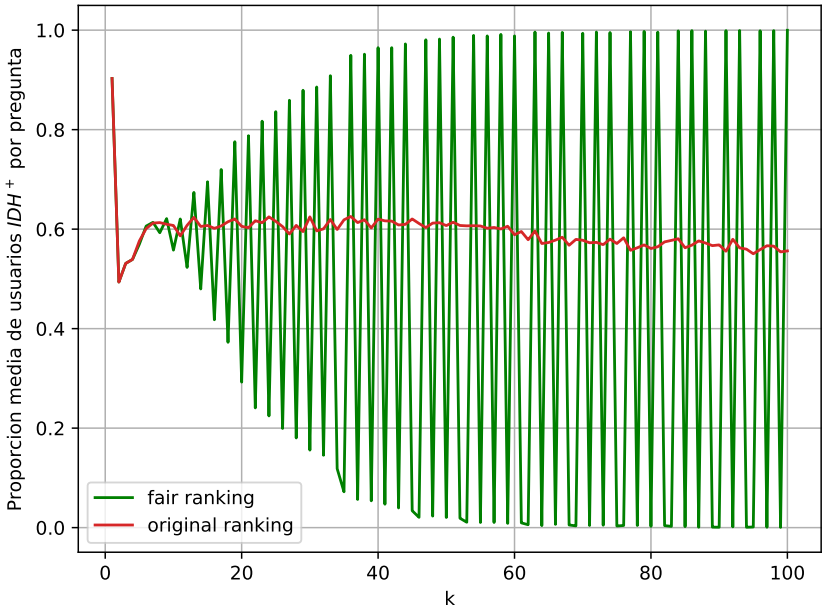


Figura 4.24: Proporción de candidatos de países desarrollados en la posición k.

analizamos la frecuencia relativa con la que el algoritmo FA*IR hace un cambio en cada posición. En la figura 4.25 se puede ver esta frecuencia relativa en nuestro conjunto de pruebas. En algunas posiciones observamos que nunca ocurren cambios: esto explica los picos en la proporción de candidatos protegidos en dichas posiciones. Como hay posiciones donde siempre hay cambios estas tienen la proporción total de esa posición con candidatos protegidos. Como contra parte en las posiciones donde la probabilidad de cambio es cero es donde se ven desplazados los candidatos no protegidos.

Revisando el algoritmo este fenómeno se da porque la decisión para poner o no un candidatos es determinista. En otras palabras, cuando según la *mtable* debe haber N candidatos protegidos y falta uno, el algoritmo siempre agrega el que falta; luego, si en la siguiente posición también necesita N protegidos, ya no va a agregar otro candidato protegido. Sin embargo, podría quedar un protegido en esa posición si así estaba en el ordenamiento original.

Mirando la Tabla 4.1, vemos por ejemplo que en la posición 32 espera que haya 14 protegidos. Como una posición antes ($k = 31$) debía haber al menos 13 protegidos, existe la posibilidad de

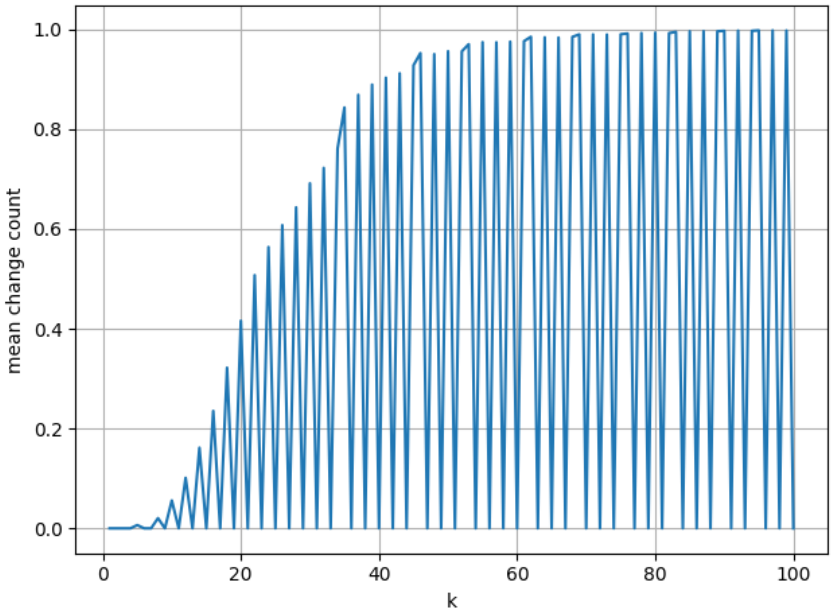


Figura 4.25: Frecuencia relativa con que FA*IR introduce cambios en la posición k.

que tenga que agregar uno en la posición 32. Pero luego, en la posición 33, la tabla también pide que haya 14 protegidos. Como en la posición 32 ya se aseguró de que hubiera 14, ya no agregará otro candidato protegido aquí. Esto explica la nula frecuencia relativa de cambios de la Figura 4.25 en la posición 33.

Otra situación se da entre las posición 33 y 35, en donde vemos un pico que ocupa dos posiciones. En la posición 34 la tabla indica que se necesitan 15 protegidos, y entonces puede agregar uno porque hasta la posición 33 se necesitaban sólo 14. Luego en la posición 35 se necesitan ya 16 protegidos, por lo que también hay chances de que se agregue un protegido. Al llegar a la posición 36, se mantiene el requisito de 16 protegidos, entonces la frecuencia relativa de cambios baja a cero.

Esto se traduce en lo que vemos en la Figura 4.1, donde la probabilidad es 0 después de esa posición viene un no protegido y se van acumulando acá, haciendo posiciones netamente no protegidas. Esto es un problema ya que genera posiciones preferenciales donde siempre aparecerán usuario del grupo privilegiado, como si fueran posiciones ”promocionadas”.

Posición k	#protegidos
31	13
32	14
33	14
34	15
35	16
36	16
37	17

Cuadro 4.1: Cantidad de protegidos por posición según la *mtable*.

4.4.2. Mitigación de las irregularidades

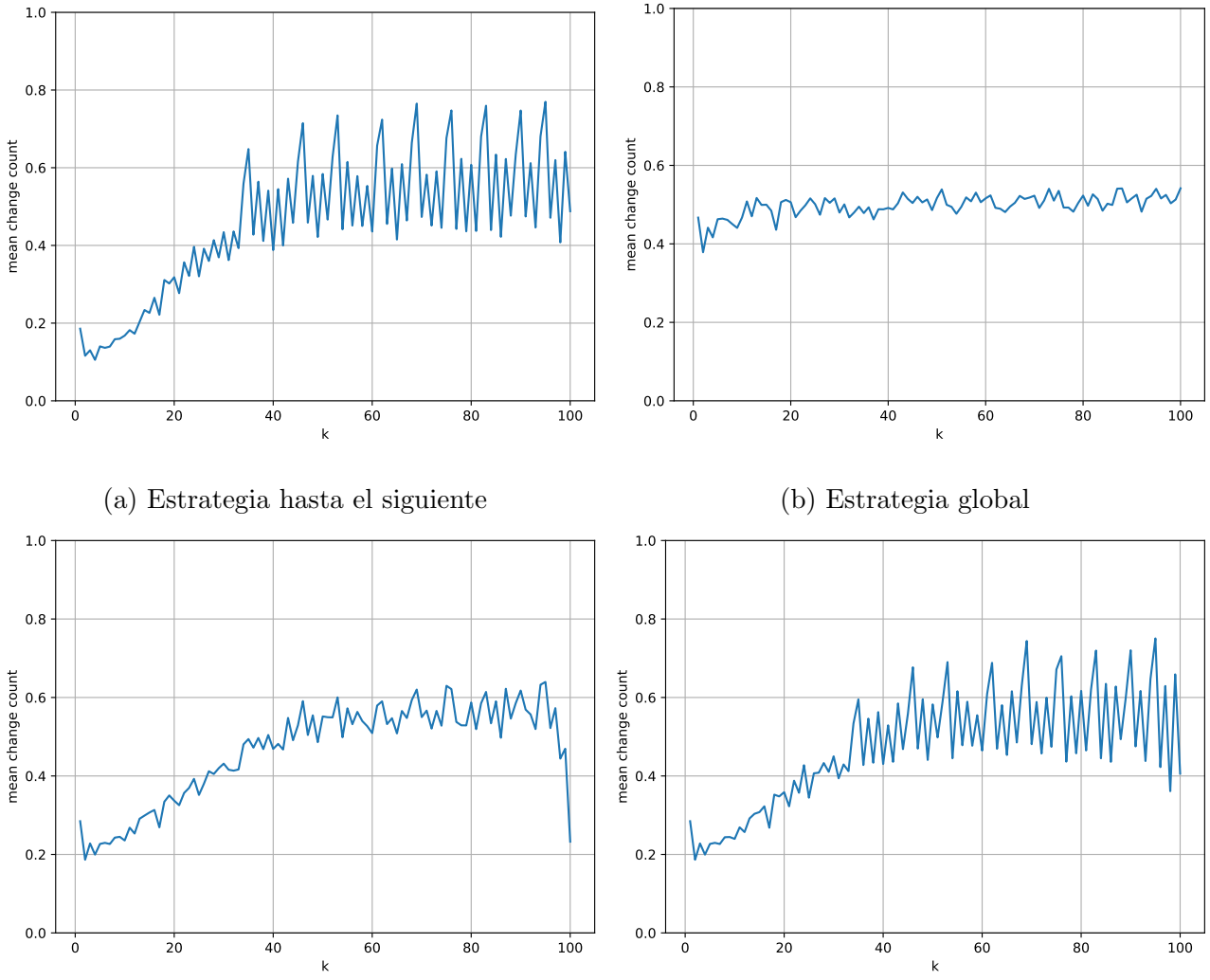
Para mejorar el comportamiento global de los ordenamientos probamos cuatro estrategias diferentes:

- **Próximo** asigna una probabilidad uniforme hasta la próxima posición en la cual haya que agregar un candidato.
- **Global** la probabilidad de meter un cambio es la cantidad de candidatos que faltan sobre la cantidad de posiciones en las cuales se pueden colocar.
- **Balanceado** es la media geométrica entre al estrategia del próximo y global.
- **Balanceado seguro** es igual al balanceado, pero agrega condiciones para que el ordenamiento sea justo.

La estrategia próximo garantiza que siempre se va a respetar la *mtable*, porque si llega a una posición donde ya no hay próximo para agregar el candidato la probabilidad va a ser uno. La estrategia global no asegura que vaya a agregarse candidatos como lo espera la *mtable*, pero distribuye los cambios mas uniformemente. La estrategia balanceada permite obtener los beneficios de ambas estrategias, pero todavía puede no ser justa. La estrategia balanceada

segura, arregla los casos que hacen que la balanceada no sea segura, agregando candidato si en alguna posición se esta por violar la *mtable*, también usa la probabilidad de próximo cuando la de global se vuelve cero.

En la Figura 4.26 se ven las probabilidades de cambio en cada posición para las diferentes estrategias. En la próxima (4.26a) las primeras posiciones tienen baja probabilidad y al final hay picos, porque una vez que llega hasta una posición donde no respeto la *mtable* solo puede distribuir uniformemente la probabilidad entre 2 o 3 posiciones. El global (4.26b) se mantiene estable con al rededor de 45 % de probabilidad de cambios en todas las posiciones. El balanceado (4.26c) se ve una probabilidad equilibrada entre las dos estrategia anteriores, con una caída en las ultimas posiciones, porque una vez que se alcanzan los candidatos que necesita ya no hace cambios, porque la probabilidad global se vuelve cero que hace cero la media geométrica. El balanceado seguro (4.26d) tiene una forma similar al balanceado agregando los picos de la próxima, pero atenuados, además aumenta la probabilidad de cambios en las posiciones bajas y las ultimas posiciones del balanceado.



(a) Estrategia hasta el siguiente

(b) Estrategia global

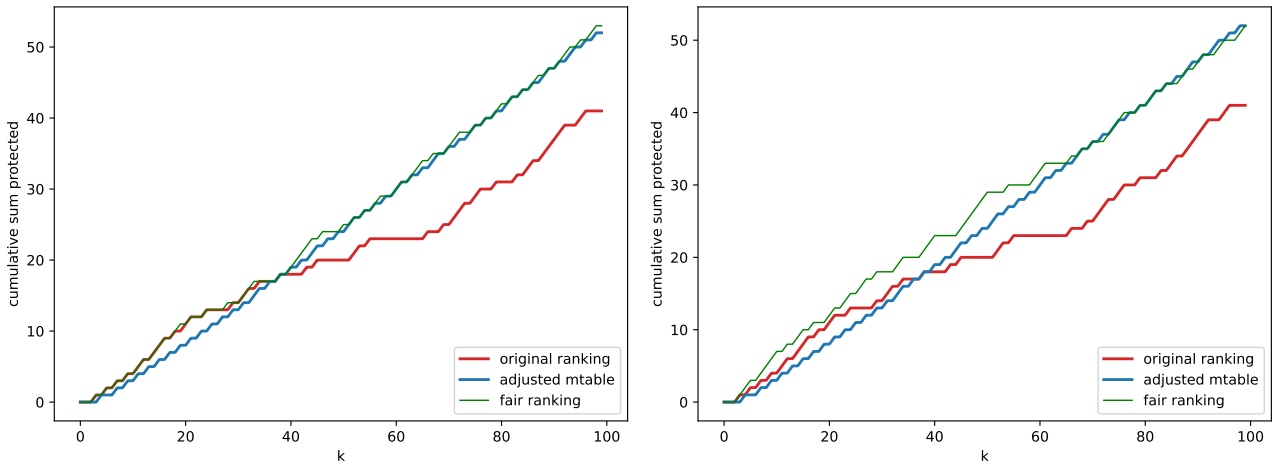
(c) Estrategia balanceada

(d) Estrategia balanceada segura

Figura 4.26: Frecuencia relativa de cambios introducidos en la posición k para las diferentes correcciones.

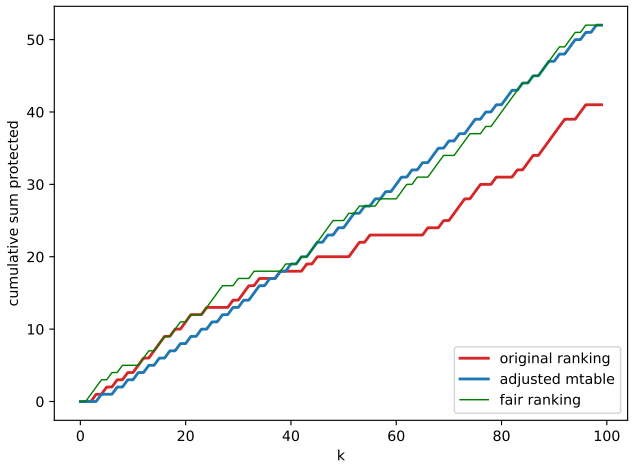
En la Figura 4.27 vemos como se acumulan los candidatos protegidos a lo largo de las posiciones para las distintas estrategias. Se ve que las estrategias próximo (4.27a) y balanceado seguro (4.27d) la cantidad de protegidos siempre esta por encima de la *mtable*. Mientras que en la estrategia global (4.27b) y la balanceada (4.27c) tienen posiciones en que estan por arriba y por debajo, por eso no siempre son justas. Las primeras estrategia hacen ordenamientos que siempre son justo, pero la global hace 545 y la balanceada 133 justas de las 3973 preguntas de prueba. Esto se puede ver claro en las medias que se ven en la Figura 4.28 mientras que las estrategia próximo (4.28a) y balanceado segura (4.28d) parecen interpolar los picos de la *mtable*, la global (4.28b) va por arriba de la *mtable* y al final queda por debajo, la balanceada (4.28c) se ve que a pesar de mejorar atraviesa algunos de los picos.

En la Figura 4.29 vemos las proporciones en cada posición. Se ve que la estrategia próximo (4.29a) tiene el problema en las posiciones altas donde como veiamos en las probabilidades cuando llega a una posicion en la que esta al limite de la *mtable* empieza a distribuir la proabbilidad entre menos posiciones y eso crea los picos que se ven. La estrategia global (4.29b) se ve mas pareja alrededor de 0.4. La estrategia balanceado (4.29c) tiene el problema que en la últimas posiciones como ya estan todos los candidatos la probabilidad se vuelve cero como vimos antes y eso dispara la proporcion de IDH^+. El balanceado seguro (4.29d) tiene los picos del próximo, pero mas atenuados, y arregla el problema de las ultimas posiciones del balanceado.

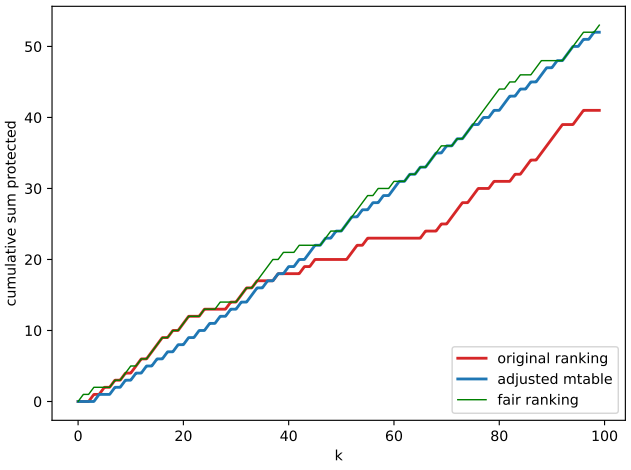


(a) Estrategia hasta el siguiente

(b) Estrategia global

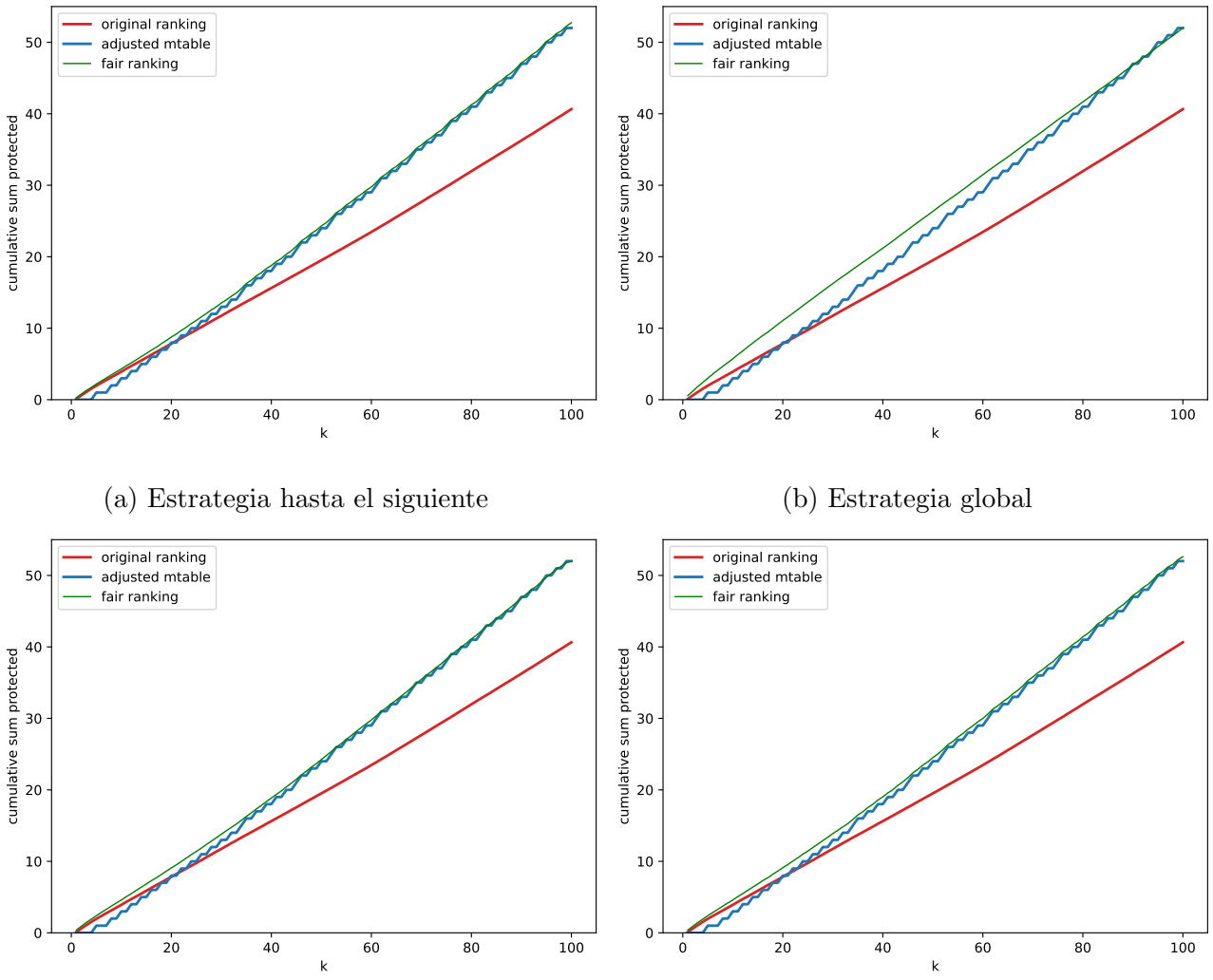


(c) Estrategia balanceada



(d) Estrategia balanceada segura

Figura 4.27: Suma acumulativa de candidatos protegidos hasta la posición k para un caso particular, en el *ranking* original, lo esperado por la *mtable* y las diferentes estrategias.



(a) Estrategia hasta el siguiente

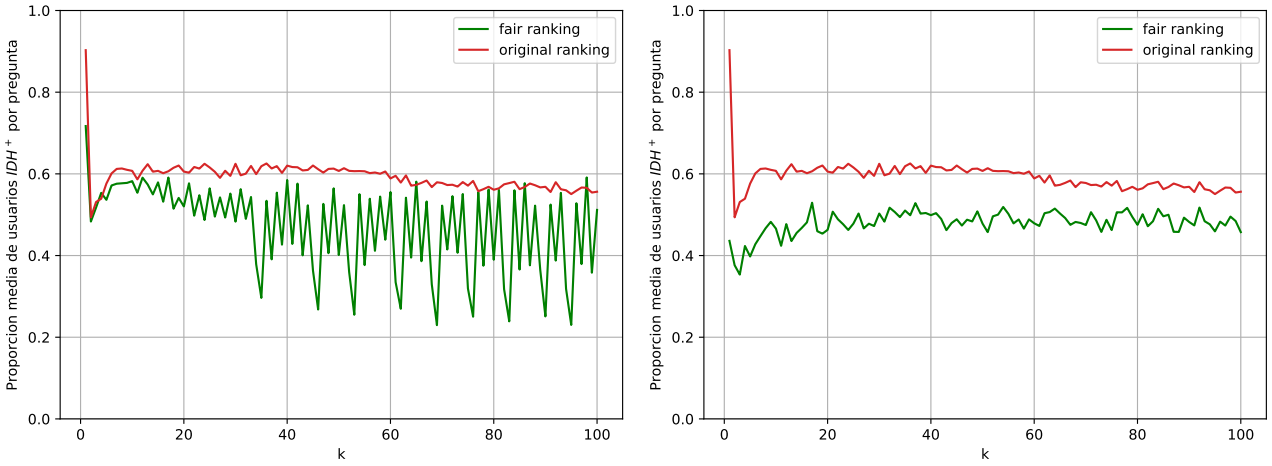
(b) Estrategia global

(c) Estrategia balanceada

(d) Estrategia balanceada segura

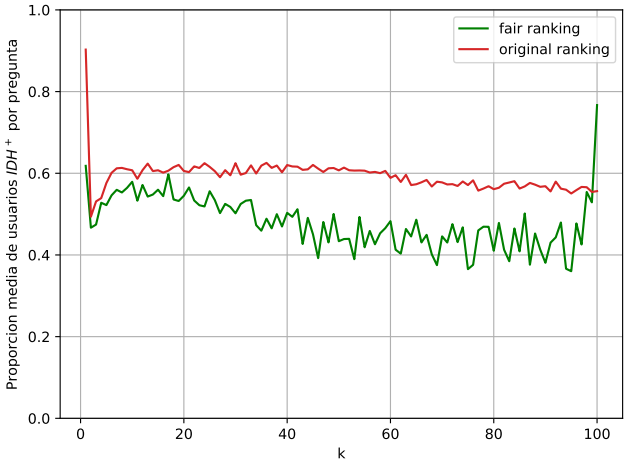
Figura 4.28: Media de la suma acumulativa de candidatos protegidos hasta la posición k, en el *ranking* original, lo esperado por la *mtable* y las diferentes estrategias.

Finalmente vemos las proporciones de IDH^+ acumuladas en la Figura 4.30, la estrategia próxima empieza con una proporción alta en posiciones bajas porque tiene baja probabilidad de cambios al principio, la global tiene una diferencia mas pronunciada al principio bajando a la mitad de 0.9 a 0.45, mientras que la estrategia balanceada y balanceada segura tienen un intermedio. En la Figura 4.31 se puede ver un foco en las diferencias con el ordenamiento original a partir de la posicion 5. La estrategia próximo baja de 0.56 a 0.48, la global empieza desde abajo, pero va subiendo porque mantiene la probabilidad de cambios, la balanceada comienza baja y aumenta hasta 20, y despues baja, la balancead segura es similar a la balanceada con picos en algunas posiciones que vienen de las condiciones para hacerla segura.

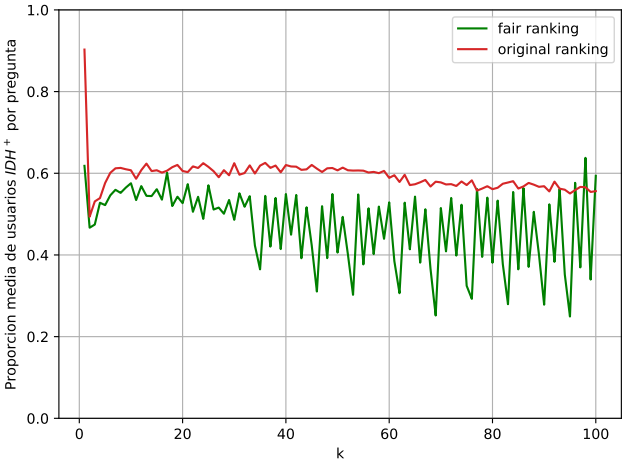


(a) Estrategia hasta el siguiente

(b) Estrategia global

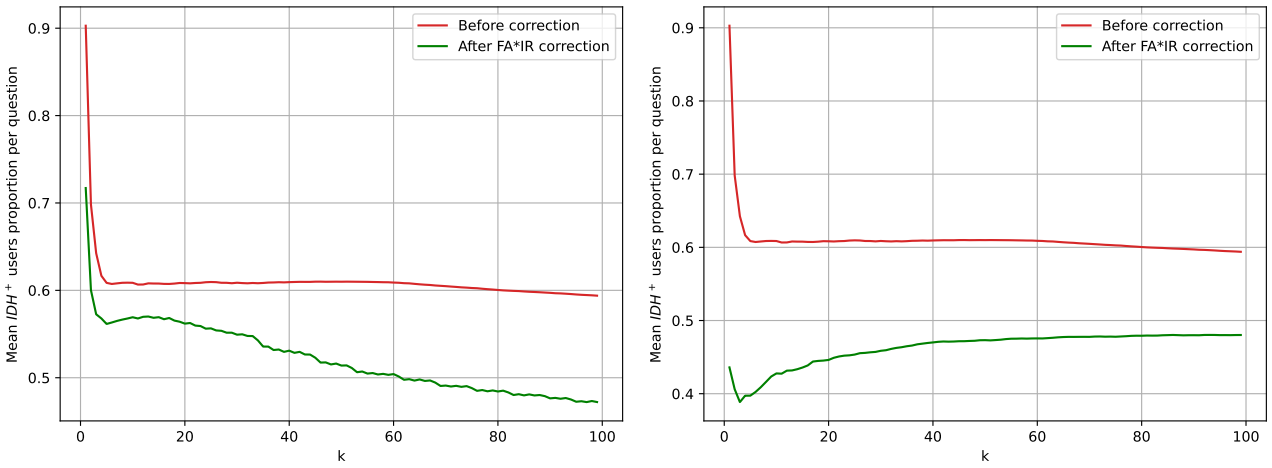


(c) Estrategia balanceada



(d) Estrategia balanceada segura

Figura 4.29: Proporción de candidatos de países desarrollados en la posición k para las diferentes correcciones.



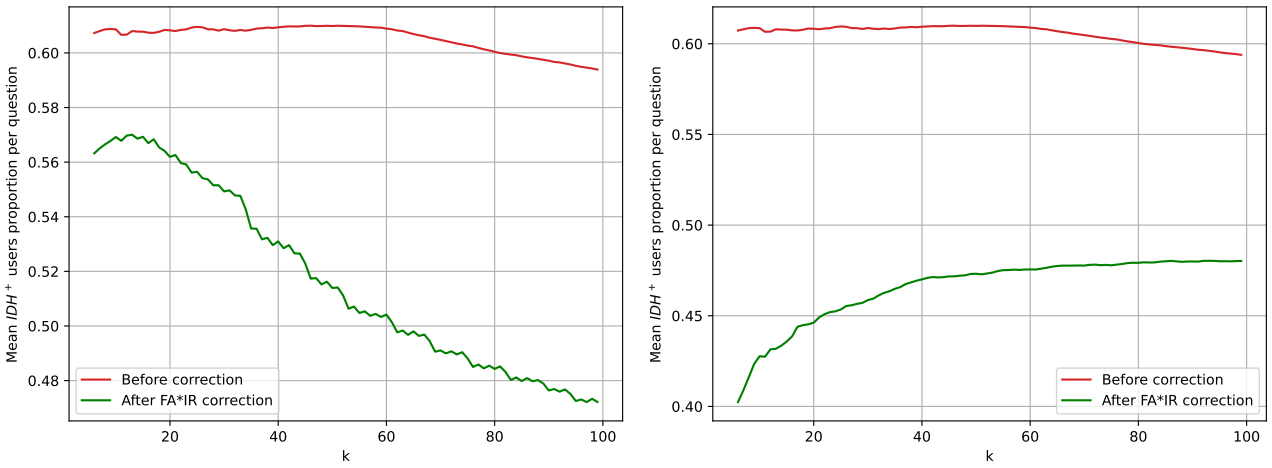
(a) Estrategia hasta el siguiente

(b) Estrategia global

(c) Estrategia balanceada

(d) Estrategia balanceada segura

Figura 4.30: Proporción de candidatos de países desarrollados hasta la posición k para las diferentes correcciones.



(a) Estrategia hasta el siguiente

(b) Estrategia global

(c) Estrategia balanceada

(d) Estrategia balanceada segura

Figura 4.31: Proporción de candidatos de países desarrollados hasta la posición k para las diferentes correcciones, para k mayores a 4.

Capítulo 5

Conclusión

En esta tesis se propuso un sistema de recomendaciones para la plataforma StackOverflow, explorando diferentes formas de representar a las preguntas y los usuarios que las iban a responder. Se analizó la equidad del sistema considerando diferentes aspectos, se aplico un algoritmo de corrección de la equidad, y se estudió el comportamiento del sistema después de las correcciones. Por último, mejoramos el comportamiento global del algoritmo de corrección de equidad.

En particular, observamos que:

- La similaridad semántica entre dos preguntas no garantiza que un usuario tengan la misma habilidad para responder ambas. Esto se manifestó en los modelos basados exclusivamente en Doc2Vec o SBERT.
- Para mejorar el *recall* de la primera etapa del sistema de recomendación, funcionó muy bien incorporar un *embedding* de baja dimensionalidad en relación a la dimensión de las preguntas, para representar la habilidad de responder de los usuarios. Esta representación, junto con una transformación que va del espacio de las preguntas al espacio de las habilidades necesarias para responderlas, logró mejorar el desempeño de esta primera etapa.

- El sistema sin ninguna intervención tiende a recomendar a personas del grupo privilegiado, más marcadamente en las primeras posiciones. Se pudo obtener un sistema más justo con el algoritmo de corrección de equidad FA*IR, sin embargo encontramos que las primeras posiciones siguieron siendo conflictivas.

- Aunque el algoritmo de corrección hacía foco en mejorar el criterio de independencia, también ayudó a mejorar la suficiencia y la separación. Además, en nuestro caso la corrección de equidad no afectó negativamente al desempeño.

- Con los cambios que hemos propuesto para el algoritmo de corrección, además de suavizar la equidad global logramos mejorar la equidad en las primeras posiciones, que eran las posiciones con mayor relevancia y diferencia.

Un problema que nos encontramos durante la tesis fue la disponibilidad de atributos sensibles en el conjunto de datos. Un atributo candidato para la corrección de equidad era el género; sin embargo no pudimos utilizarlo porque no contábamos con este dato, y los modelos que hacen predicción de genero a partir de nombres de usuario mostraban muy baja precisión.

Otro problema lo encontramos durante el entrenamiento de SBERT: a pesar de que debería ser posible hacer *finetuning* del modelo para que la representación se adapte a nuestro caso de uso, no logramos el entrenamiento suficiente para que converja a lo que necesitábamos.

Dejamos abiertas algunas posibles lineas de trabajo futuro en torno a la problemática de equidad en esta u otras plataformas:

- Refinar los algoritmos de asignación de paises, para capturar algunos casos que quedaron excluidos.

- Hacer los análisis de desempeño y equidad teniendo en cuenta a los usuario sin país como un tercer grupo de usuarios.

- Trabajar con otros *proxys* para la barrera de acceso a la tecnología, y comparar si se sostienen las mismas conclusiones que con el IDH que utilizamos.

- Analizar la interseccionalidad con otros atributos sensibles como género o edad.
- Ajustar SBERT por mas tiempo o con otro arreglo. Explorar los hiperparametros y mejorar los *embeddings* de usuarios.
- Contrastar con otros tipos de representaciones para los usuarios y las preguntas, por ejemplo incorporando métricas de las redes de preguntas y respuestas.

Bibliografía

[Barocas et al., 2023] Barocas, S., Hardt, M., and Narayanan, A. (2023). *Fairness and machine learning: Limitations and opportunities*. MIT Press.

[Burges et al., 2006] Burges, C., Ragno, R., and Le, Q. (2006). Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19.

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

[Burges, 2010] Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

[Chapelle and Chang, 2011] Chapelle, O. and Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pages 1–24. PMLR.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[Covington et al., 2016] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Donmez et al., 2009] Donmez, P., Svore, K. M., and Burges, C. J. (2009). On the local optimality of lambdarank. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 460–467.

[Douze et al., 2024] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2024). The faiss library.

[Feng et al., 2020] Feng, Y., Hu, B., Lv, F., Liu, Q., Zhang, Z., and Ou, W. (2020). Atbrg: Adaptive target-behavior relational graph network for effective recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2231–2240.

[Firth, 1968] Firth, J. (1968). A synopsis of linguistic theory, 1930-1955. In *Selected papers of JR Firth, 1952-59*.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[Han et al., 2023] Han, Y., Liu, C., and Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*.

[Hardt et al., 2016] Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Huang et al., 2020] Huang, J.-T., Sharma, A., Sun, S., Xia, L., Zhang, D., Pronin, P., Padmanabhan, J., Ottaviano, G., and Yang, L. (2020). Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.

[Larson and Kirchner, 2016] Larson, S. A. J. and Kirchner, L. (2016). There's software used across the country to predict future criminals. and it's biased against blacks. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

[Li, 2022] Li, H. (2022). *Learning to rank for information retrieval and natural language processing*. Springer Nature.

[Li et al., 2022] Li, Y., Chen, H., Xu, S., Ge, Y., Tan, J., Liu, S., and Zhang, Y. (2022). Fairness in recommendation: A survey. *arXiv preprint arXiv:2205.13619*.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[Pleiss et al., 2017] Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., and Weinberger, K. Q. (2017). On fairness and calibration. *Advances in neural information processing systems*, 30.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural*

information processing systems, 30.

[Wang et al., 2018] Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D. L. (2018). Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 839–848.

[Wang et al., 2023] Wang, L., Ling, Y., Yuan, Z., Shridhar, M., Bao, C., Qin, Y., Wang, B., Xu, H., and Wang, X. (2023). Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*.

[Wu et al., 2010] Wu, Q., Burges, C. J., Svore, K. M., and Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, 13:254–270.

[Yuenyong and Sinthupinyo, 2020] Yuenyong, S. and Sinthupinyo, S. (2020). Gender classification of thai facebook usernames. *International Journal of Machine Learning and Computing*, 10(5):618–623.

[Zehlike et al., 2017] Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., and Baeza-Yates, R. (2017). Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578.

[Zehlike et al., 2022a] Zehlike, M., Sühr, T., Baeza-Yates, R., Bonchi, F., Castillo, C., and Hajian, S. (2022a). Fair top-k ranking with multiple protected groups. *Information processing & management*, 59(1):102707.

[Zehlike et al., 2022b] Zehlike, M., Yang, K., and Stoyanovich, J. (2022b). Fairness in ranking, part i: Score-based ranking. *ACM Computing Surveys*, 55(6):1–36.

[Zehlike et al., 2022c] Zehlike, M., Yang, K., and Stoyanovich, J. (2022c). Fairness in ranking, part ii: Learning-to-rank and recommender systems. *ACM Computing Surveys*, 55(6):1–41.

[Zhang et al., 2020] Zhang, H., Wang, S., Zhang, K., Tang, Z., Jiang, Y., Xiao, Y., Yan, W., and Yang, W.-Y. (2020). Towards personalized and semantic retrieval: An end-to-end solution

for e-commerce search via embedding learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2407–2416.