Representaciones basadas en redes neuronales para tareas de recomendación

Pablo Pérez-Núñez
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
UO232368@uniovi.es

Oscar Luaces
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
oluaces@uniovi.es

Antonio Bahamonde
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
abahamonde@uniovi.es

Jorge Díez
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Gijón, España
jdiez@uniovi.es

***Resumen*—Los sistemas de recomendación tratan de conocer, de alguna manera, los gustos de los usuarios con el propósito de recomendar productos que sean de su agrado. La manera de representar a los usuarios tiene un rol importante en estos sistemas, ya que se espera que una buena representación facilite la correcta identificación del perfil de consumo de cada usuario, sintetizando, de alguna manera, sus gustos. No es menos importante la manera en la que se representan los productos, donde el contexto u orden en que se consumen podría ser relevante en su representación. En este artículo se analizan varias formas de representación basadas en redes neuronales y se aplican a dos tareas de recomendación diferentes en un conjunto de reproducciones musicales. Los resultados muestran que parece relevante considerar el orden de consumo para la codificación de los productos pero no para la codificación del perfil de los usuarios.**

***Index Terms*—perfiles de usuario, sistemas de recomendación, factorización, filtros colaborativos**

I. Introducción

La mayor parte de los recursos digitales que utilizamos hoy en día realizan recomendaciones a sus usuarios utilizando Sistemas de Recomendación ([1], [2]). Así sucede, por ejemplo, cuando entramos en la web de un periódico digital, donde el orden en el que están colocadas las noticias ya es, de alguna manera, una recomendación de lectura. Las listas ordenadas de noticias que aparecen en los márgenes (*"las noticias más leídas"* o *"las más comentadas"*) constituyen también una recomendación, basada en este caso, en la interacción de otros usuarios con la publicación digital. Otro ejemplo de recomendación podemos encontrarlo dentro de los artículos, en forma de enlaces, que nos conducen a otras noticias u otras recomendaciones del tipo *"noticias relacionadas"*.

Los sistemas de recomendación han encontrado en las tiendas online un campo de aplicación muy importante, con el objetivo de incrementar las ventas al tiempo que aumentar la satisfacción de los clientes. En este caso, lo que se recomienda son productos y las recomendaciones que se realizan son del estilo *"los clientes que vieron este producto también vieron…"*, *"comprados juntos habitualmente"* o incluso rankings de productos en función del número de ventas o de las críticas de los compradores. Podemos también encontrar recomendaciones en las plataformas de streaming de contenido multimedia, por ejemplo, Netflix o Spotify, en sitios web dedicados a organizar las reseñas de hoteles y restaurantes, como TripAdvisor, etc.

Algunos de los ejemplos antes mencionados son sistemas de recomendación elementales, no personalizados, que basan sus recomendaciones simplemente en comportamientos o gustos mayoritarios: si mucha gente ve una película y, además, las valoraciones que tiene ésta son muy buenas, entonces es bastante probable que se recomiende a los usuarios que todavía no la hayan visto.

Sin embargo, hay otros sistemas que basan sus recomendaciones teniendo en cuenta información adicional presente en el contexto. Uno de los más básicos ya se comentó en un párrafo anterior: cuando se detecta que un usuario está viendo información de un producto, el sistema le muestra productos que se parecen o que se compran junto con el que está viendo. Este sistema tiene en cuenta el producto que se está viendo para hacer una recomendación personalizada.

Hay otro tipo de recomendadores que almacenan un perfil para cada usuario en el que, de alguna manera, están contenidos sus gustos o preferencias [3]. Posteriormente, estos perfiles podrán combinarse para hacer recomendaciones personalizadas, con lo que se conseguirá una mayor satisfacción del usuario ante la recomendación. La forma en la que se calcule ese perfil podrá ser más o menos elaborada.

Una manera de tener en cuenta ese perfil de consumo consiste en representar cada usuario con un vector que codifique

El trabajo realizado en este artículo ha sido financiado, en parte, por el proyecto TIN2015-65069-C2-2-R del Ministerio de Economía y Competitividad y por los fondos FEDER.

la información relativa a su interacción[1] con los productos. Esto puede lograrse mediante, por ejemplo, la construcción de *embeddings* utilizando factorización de matrices [4]. De igual forma, los productos también pueden representarse por vectores que resuman la interacción que se ha tenido con ellos. En este artículo vamos a explorar los beneficios que pueden obtenerse al utilizar dos técnicas basadas en redes neuronales, *word2vec* [5] y *doc2vec* [6], para la codificación de usuarios y de productos. Estos algoritmos han sido diseñados originalmente para la codificación de palabras y documentos en procesos que requieren representar textos para tareas de aprendizaje. La codificación que obtienen está basada en el contexto de cada palabra en cada documento. El consumo de cierto tipo de productos también guarda relación con el contexto temporal, por ejemplo, la reproducción de una secuencia de canciones (*playlist*), por lo que podemos utilizar estas técnicas para las tareas de codificación. Con objeto de mostrar la relevancia de este orden secuencial, en este trabajo comparamos la codificación obtenida mediante estas técnicas frente a otra, que se obtiene al construir un embedding a partir de una codificación *one-hot*, donde no se tiene en cuenta el orden de consumo en manera alguna.

Una decisión que debe ser adoptada para codificar los perfiles de consumo de los usuarios es cuánto nos vamos a remontar en la secuencia temporal para construir y codificar dicho perfil. En este artículo hemos considerado y comparado la codificación que se obtiene considerando una trayectoria de consumo reciente y una trayectoria de consumo a largo plazo. Es posible que, en determinadas situaciones, sea interesante mantener estos dos perfiles, sobre todo cuando los productos para los que se quiere realizar recomendaciones tengan un comportamiento estacional. Ejemplos de este tipo pueden encontrarse en la alimentación, donde hay algunos platos que se consumen más durante el invierno y otros durante el verano.

Los sistemas de recomendación pueden clasificarse en dos grandes grupos: por una parte podemos construir *filtros colaborativos* [7], en los que se utiliza únicamente la información que se tiene sobre la interacción de los usuarios con los productos y este comportamiento puede darnos una idea de sus gustos o preferencias. Por otra parte, existen también sistemas de recomendación *basados en contenido* [8], donde se utiliza información conocida de los usuarios, como puede ser su ubicación, el sexo, etc. e información conocida de los productos, por ejemplo, género y fecha de estreno si se tratase de películas. En este tipo de sistemas se pueden tener perfiles de clientes y productos basándose en esos datos conocidos.

En este artículo hemos utilizado únicamente filtros colaborativos puesto que queremos estudiar la utilidad de codificar perfiles que resuman la interacción registrada entre usuarios y productos. Incluir información basada en contenido podría distorsionar este estudio, si bien es probable que los resultados en cuanto a precisión en la recomendación puedan ser mejores, pues dispondríamos de más información.

La organización del artículo se muestra a continuación: en la siguiente sección se hace una breve reseña de trabajos previos relacionados con la elaboración de perfiles. La Sección III se dedica a detallar cómo se pueden codificar los perfiles utilizando las técnicas incluidas en word2vec y doc2vec, que se utilizarán para codificar y comparar dos tipos de perfiles, uno a largo plazo (*perfil consolidado*) y otro a corto plazo (*perfil reciente*) para cada usuario. Seguidamente, en la Sección IV, se detallarán dos tareas de recomendación en las que se va a comprobar el rendimiento de estos perfiles. En la Sección V, dedicada a los resultados, se mostrará el rendimiento de los perfiles en las dos tareas planteadas, discutiendo los resultados obtenidos. Finalmente, se presentan las conclusiones y se apunta alguna línea futura de investigación en este contexto.

II. Trabajo relacionado

Son varios los trabajos que abordan la creación de perfiles utilizando la factorización de matrices como herramienta.

En [9] los autores presentan un algoritmo que proyecta cantantes y canciones en un nuevo espacio utilizando factorización de matrices. Las tareas de aprendizaje que se abordan en este trabajo están relacionadas con la predicción de artistas que han podido interpretar una canción, la predicción de canciones que han podido ser interpretadas por un determinado artista, la obtención de canciones similares (en algún sentido) a una dada o la obtención de artistas similares a uno dado. Estas preguntas son, realmente, tareas de recomendación y las proyecciones de cantantes y canciones pueden ser consideradas perfiles.

Algo similar se plantea en [10], donde en este caso se calculan los representaciones de usuarios y canciones con el objetivo de generar *playlists* del agrado de los usuarios.

La factorización de matrices también se ha utilizado para condensar los gustos de las personas respecto a productos alimenticios. En [4] los autores obtuvieron perfiles de consumidores útiles para conocer sus preferencias respecto al consumo de carne con diferentes periodos maduración.

Hasta donde nosotros sabemos, no hay trabajos en los que se plantee una codificación de los productos basándose en el contexto u orden en el que los usuarios interactúan con ellos. Tampoco tenemos constancia de trabajos en los que se discuta la necesidad de mantener dos perfiles para cada usuario, uno representando los gustos consolidados del usuario y otro representando los recientes.

III. Creación de perfiles a partir de la interacción de los usuarios con los productos

Supongamos un conjunto de usuarios, $\mathcal{U}$, y un conjunto de productos, $\mathcal{P}$. Además, conocemos las interacciones que ha tenido cada usuario con los productos y en qué orden se ha producido dicha interacción, de tal manera que para cada usuario disponemos de una lista ordenada de productos con los que ha interactuado a lo largo del tiempo:

$$\mathcal{D} = \{(\boldsymbol{u}, \boldsymbol{p}_1, \boldsymbol{p}_2, \ldots) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p}_i \in \mathcal{P}\} \quad (1)$$

[1] A lo largo de este artículo utilizamos la palabra *interacción* para referirnos a cualquier forma de consumo de un producto: la compra de un item, escuchar una canción, ver un producto en una tienda on-line, leer una noticia en una publicación digital, etc.

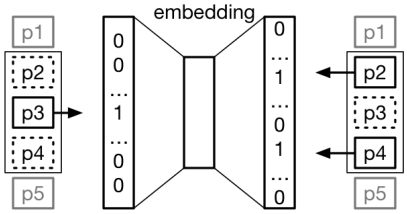

Figura 1. Arquitectura del word2vec utilizando skip-gram con una ventana de tamaño 1 para definir el contexto, que será la suma de los vectores one-hot de las palabras contenidas en el mismo

A partir de esta información aplicaremos los algoritmos word2vec y doc2vec para obtener los perfiles de productos y usuarios.

Los conjuntos $\mathcal{U}$ y $\mathcal{P}$ contienen únicamente los identificadores de los usuarios y productos, respectivamente. Una representación habitual para los elementos de estos conjuntos consiste en utilizar vectores *one-hot*, que son vectores cuya dimensión es igual a la cardinalidad del conjunto de elementos que representan. Un vector one-hot que represente al i-ésimo elemento del conjunto tiene un 1 en la componente i y todas las demás son 0. Los algoritmos word2vec y doc2vec, que describimos brevemente a continuación, se encargarán de recodificar estos vectores, calculando unos *encajes* (*embeddings*) en los que se tiene en cuenta la secuencia de aparición de los distintos elementos en los datos de entrada.

III-A. Codificación de los productos

Word2vec [5] es un algoritmo ideado para codificar las palabras de un corpus mediante vectores, de tal manera que éstos se disponen en el espacio de acuerdo a cómo se relacionan en los textos del corpus. Así, aquellas palabras con representaciones próximas suelen tener una relación fuerte.

Los vectores aprendidos para las palabras codifican ciertos patrones lingüísticos, que quedan patentes al realizar operaciones con los vectores resultantes, ya que, como se explica en [5], si al vector que representa la palabra *Madrid* se le resta el que representa la palabra *España* y se le suma el de *Francia*, resultará un vector muy cercano al que representa *París*.

Los autores plantean dos posibles estrategias para la recodificación de palabras, que resultan en dos tareas de aprendizaje:

- *Continuous Bag of Words (CBOW)*, donde a partir de las palabras del contexto se trata de predecir la central, y
- *Skip-gram*, donde a partir de la palabra central se trata de predecir las del contexto (Figura 1).

En ambos casos, el contexto se define mediante un tamaño de ventana y se representa como un vector que es la suma de los vectores one-hot de las palabras contenidas en él. La Figura 1 muestra un esquema de word2vec utilizando skip-gram, en el instante en que se presenta como entrada la palabra $\boldsymbol{p}_3$ y como salida deseada su contexto que, en este caso, son las palabras $\boldsymbol{p}_2$ y $\boldsymbol{p}_4$ o, más específicamente, el vector suma de los vectores que representan a dichas palabras. Entre la entrada y la salida hay una capa oculta, en la que obtendremos la representación buscada tras el proceso de entrenamiento.

Los autores afirman en su artículo que la estrategia skip-gram ofrece mejores resultados y, por tanto, esta estrategia es la que utilizaremos para la obtención de las representaciones de los productos en nuestro sistema de recomendación.

En un símil con el tratamiento de textos, nosotros consideramos la lista de interacciones con productos de la Ecuación (1) como si fuese una secuencia de palabras en un texto de forma que, para un determinado tamaño de ventana trataremos de aprender a predecir con qué productos ha habido interacción antes y después de la interacción con uno determinado.

III-B. Codificación de los usuarios

Tras aprender la codificación de los productos a partir de cómo los usuarios han interactuado con ellos, podemos obtener el perfil de cada usuario. Para ello proponemos utilizar el algoritmo doc2vec [6], que ha sido diseñado para codificar documentos o párrafos a partir de palabras codificadas mediante word2vec. En su artículo, los autores afirman que las representaciones obtenidas por este algoritmo mejoran el rendimiento en tareas de Recuperación de Información respecto al popular *Bag Of Words*. Para aprender la codificación de documentos, los autores también sugieren dos arquitecturas:

- *Distributed Memory version of Paragraph Vector (PV-DM)*, en la que se aprende a codificar cada documento en un proceso en el que se trata de predecir cuál es la siguiente palabra a una secuencia (ventana) de palabras del documento. La entrada al sistema es la concatenación del vector que representa al documento con los que representan a las palabras de la ventana seleccionada. El aprendizaje únicamente modifica la codificación de cada documento, minimizando el error de predicción, mientras que la codificación de las palabras (obtenida previamente mediante word2vec) se mantiene fija.
- *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)*, donde se toma un documento y se eligen al azar unas cuantas palabras del mismo. A partir únicamente del vector que representa el documento se aprende su codificación para predecir la presencia, sin importar el orden, de las palabras anteriormente elegidas.

El objetivo final en ambos casos es ir modificando durante el entrenamiento la representación de cada documento. En [6], los autores muestran el buen rendimiento de estas representaciones, proponiendo el cálculo de ambas, PV-DM y PV-DBOW, para su utilización de manera concatenada.

En nuestro trabajo consideraremos la lista de productos con los que un usuario ha interaccionado, Ecuación (1), como el equivalente a un documento, de manera que habrá un documento por cada usuario. Podremos entonces utilizar doc2vec para la obtención de un vector que codifique a cada usuario. En otras palabras, identificaremos el perfil de un usuario por la secuencia de productos con los que tuvo interacción.

III-C. Perfil consolidado y perfil reciente

Una vez que se han presentado las herramientas con las que trabajaremos, vamos a definir cómo se pueden obtener el perfil consolidado y el perfil reciente de los usuarios.

Esencialmente, el procedimiento es el mismo para ambos perfiles, pero la diferencia radica en los datos con que entrenaremos la red doc2vec. Así, para obtener un perfil consolidado utilizaremos toda la información disponible acerca de las interacciones del usuario con los productos, independientemente del momento en que se haya producido cada interacción. Nuestra intención es que el perfil consolidado registre o codifique todos los intereses del usuario a lo largo del tiempo.

Sin embargo, para efectuar cierto tipo de recomendaciones puede ser de poca utilidad la información relativa al consumo mucho tiempo atrás. Esto sucede, por ejemplo, en la recomendación de listas de reproducción de canciones, en las que, para añadir un nuevo tema del agrado del usuario parece obvio que es más importante tener en consideración las canciones que acaba de escuchar, que aquellas que escuchó hace semanas, o meses. Por esta razón hemos considerado la codificación de perfiles recientes, que se obtendrán entrenando doc2vec únicamente con las interacciones más recientes del usuario. Obviamente, el umbral de decisión que determina qué interacciones son recientes y cuáles no lo son dependerá del tipo de productos que se vayan a recomendar.

IV. Tareas de recomendación

Vamos a estudiar el rendimiento de los mecanismos de representación de productos y usuarios antes mencionados en dos problemas que se detallan a continuación. Los resultados (Sección V) se compararán con los obtenidos utilizando la codificación basada en vectores one-hot.

IV-A. Tarea 1: ¿interactuará con un determinado producto?

Para resolver esta tarea contamos con un conjunto de entrenamiento cuyos ejemplos son tripletas que contienen un usuario, un producto y si dicho usuario ha interactuado o no con dicho producto, esto es,

$$\mathcal{D}_1 = \{(\boldsymbol{u}, \boldsymbol{p}, z) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p} \in \mathcal{P}, z \in \{+1, -1\}\}. \quad (2)$$

El algoritmo que diseñamos para resolver esta tarea aprenderá utilizando la siguiente función logística:

$$\begin{gathered}\Pr(z|\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V}) = \sigma(z \cdot g(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V})), \\ \text{con } \sigma(x) = \frac{1}{1+e^{-x}},\end{gathered} \quad (3)$$

donde $\boldsymbol{W}$ y $\boldsymbol{V}$ son parámetros que deben ser encontrados utilizando la estimación *Máxima Probabilidad a Posteriori (MAP)*, y g es una función de compatibilidad entre los usuarios y los productos, definida como el producto escalar:

$$\mathrm{g}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V}) = \langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{p} \rangle. \quad (4)$$

Para enriquecer la expresividad del modelo aprendido se incorporarán términos independientes.

La función de pérdida a minimizar es, en este caso,

$$-\log \prod_{(\boldsymbol{u},\boldsymbol{p},z)} \Pr(z|\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{W}, \boldsymbol{V})) = \log\left(1 + e^{-z\langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{p}\rangle}\right), \quad (5)$$

también conocida como *softplus*.

IV-B. Tarea 2: ¿cuál será el siguiente producto?

La segunda de las tareas a resolver es tratar de anticiparnos a la interacción inmediatamente siguiente del usuario. En este caso el conjunto de entrenamiento estará formado por tripletas conteniendo el usuario, $\boldsymbol{u}$, el último producto con el que ha interactuado, $\boldsymbol{p}_i$ y el siguiente producto con el que interactúa a continuación, $\boldsymbol{p}_j$,

$$\mathcal{D}_2 = \{(\boldsymbol{u}, \boldsymbol{p}_i, \boldsymbol{p}_j) : \boldsymbol{u} \in \mathcal{U}, \boldsymbol{p}_i, \boldsymbol{p}_j \in \mathcal{P}\}. \quad (6)$$

En este caso el algoritmo podría diseñarse aprendiendo la siguiente función:

$$\Pr(y = j|\boldsymbol{u}, \boldsymbol{p}_i, \theta) = \frac{e^{v_j}}{\sum_k e^{v_k}} = \mathrm{softmax}(v)_j \quad (7)$$

siendo j el identificador del producto $\boldsymbol{p}_j$, k el de cualquier producto, θ el conjunto de parámetros (pesos de una red neuronal, por ejemplo) que se deben aprender y v el valor de salida de la red cuya entrada es la concatenación del usuario y el producto. También incorporamos un término independiente.

La función de pérdida habitual en problemas de este tipo es la *entropía cruzada* (*cross entropy*). Sin embargo, cuando el número de clases es elevado, algo común en los problemas de recomendación, el cálculo de $\mathrm{softmax}(v)$ es muy costoso, por lo que se recurre a estrategias de estimación del error, como la denominada *noise-contrastive estimation (NCE)* [11], que hemos utilizado en este trabajo.

V. Resultados experimentales

En esta sección describimos y analizamos los resultados experimentales que hemos obtenido utilizando distintas codificaciones para usuarios y productos. Más concretamente, hemos analizando el rendimiento de tres perfiles diferentes para los usuarios y dos para los productos.

V-A. Descripción del conjunto de datos

Para la experimentación hemos utilizado un conjunto de datos de la web *www.last.fm*, que han sido previamente publicados en el capítulo 3 del libro [12] y que se encuentran disponibles públicamente[2]. El conjunto contiene las reproducciones de música de 992 usuarios durante 5 años, aproximadamente. En total hay algo más de 19 millones de reproducciones sobre un conjunto de un millón y medio de canciones. En el conjunto también aparece la fecha y hora exactas de cada reproducción, con lo que no es complicado elaborar para cada usuario una lista ordenada como la que se muestra en la Ecuación (1).

V-B. Preparación de los experimentos

Hemos filtrado el conjunto, eliminando canciones repetidas o que se habían escuchado muy poco, quedándonos sólo con aquellas que se han escuchado al menos 10 veces, con lo que hemos reducido el conjunto a 312895 canciones.

Cada lista de reproducción asociada a un usuario fue separada en una parte para entrenar los perfiles (75 %) y otra para utilizar en las tareas propuestas (25 %), ver Figura 2.

[2]http://www.dtic.upf.edu/∼ocelma/MusicRecommendationDataset/lastfm-1K.html

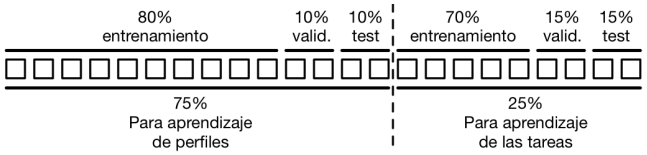

Figura 2. Método que se utiliza para la separación de la lista de reproducción de cada usuario en conjuntos de entrenamiento, validación y test

A su vez, cada una de estas dos partes fue separada en entrenamiento/validación/test, utilizando las partes de entrenamiento y validación para buscar los hiper-parámetros con mejor rendimiento. Los resultados que se muestran en las tablas son los obtenidos con estos modelos al evaluar su rendimiento sobre el conjunto de test.

Todos los algoritmos utilizados en este artículo fueron implementados utilizando la librería TensorFlow [13] sobre Python, y el optimizador SGD-Adam [14].

V-C. Obtención de perfiles

Para codificar las canciones utilizamos word2vec sobre el conjunto de datos correspondiente, usando un tamaño de ventana de 2 y con objeto de obtener vectores en un espacio de 64 dimensiones. Estos parámetros eran los que mejores resultados mostraban sobre el conjunto de validación.

Una vez codificadas las canciones, se codificaron los usuarios mediante la red doc2vec de dos maneras diferentes, una para obtener el perfil consolidado (P_{con}) y otra para el perfil reciente (P_{rec}). En ambos casos también utilizamos un espacio de 64 dimensiones, con una ventana de tamaño 2 (para la arquitectura PV-DM) y seleccionando todas las reproducciones del entrenamiento (para la arquitectura PV-DBOW). Al calcularse mediante los dos métodos propuestos en la Sección III-B y concatenarlos, finalmente los usuarios quedan proyectados en un espacio de 128 dimensiones. Para el cálculo del P_{con} se utilizaron todas las reproducciones reservadas para el aprendizaje de los perfiles, mientras que para el P_{rec} se utilizaron solo las reproducciones del último mes para cada usuario.

Finalmente, obtuvimos una codificación en la que los vectores de los usuarios y productos, en formato one-hot, son proyectados en un espacio de 64 dimensiones para las canciones y de 128 dimensiones para los usuarios. Utilizamos las mismas dimensiones que las de los perfiles obtenidos con word2vec y doc2vec con el fin de que los espacios tengan la misma capacidad expresiva y la comparación sea justa. La proyección se realiza a través de un encaje (embedding) que resulta del proceso de aprendizaje de cada tarea de recomendación, mediante el cálculo de dos matrices, X e Y, que proyectan cada usuario y cada canción en un espacio con las dimensiones antes especificadas (ver Figuras 3 y 4).

V-D. Tarea 1

Para la tarea de aprender un modelo capaz de indicarnos si un usuario va a escuchar una canción en el futuro, resolvemos el problema de optimización planteado en la Sección IV-A.

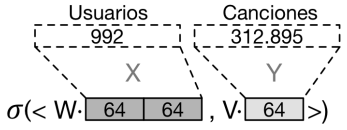

Figura 3. Red diseñada para la tarea 1, Ecuación (3). El vector que define el perfil del usuario (gris oscuro) y el de la canción (gris claro) pueden ser precalculados mediante doc2vec y word2vec, respectivamente o pueden aprenderse ad-hoc para resolver la tarea, a partir de la representación one-hot y optimizando los parámetros ($\boldsymbol{X}$ e $\boldsymbol{Y}$) necesarios.

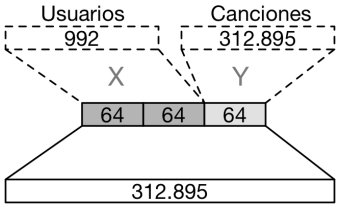

Figura 4. Red diseñada para la tarea 2, Ecuación (7). Las distintas codificaciones para usuarios y canciones son las mismas que para la tarea 1.

El conjunto $\mathcal{D}_1$ de la Ecuación (2) se construye utilizando la parte de entrenamiento de las reproducciones reservadas para el aprendizaje de las tareas (25 %), como se representa en la Figura 2. Las canciones contenidas en ese subconjunto de datos se etiquetan con $+1$, y, por cada una de ellas, se elegirá al azar una canción que el usuario no haya escuchado previamente, que se etiquetará con -1.

En la Tabla I se recogen los resultados, medidos en *Precision*, *Exhaustividad* (*Recall*) y *F1* (media armónica de las anteriores). En estos resultados se aprecia que la utilización del perfil de las canciones obtenido mediante word2vec favorece el aprendizaje. En cuanto a los perfiles de usuario, no parece que la incorporación de perfiles obtenidos mediante doc2vec mejoren el rendimiento, si bien parece más útil el perfil consolidado que el perfil reciente para este problema.

V-E. Tarea 2

Para resolver el problema de optimización de la Sección IV-B (predecir la siguiente canción a reproducir) utilizaremos una red con la estructura de la Figura 4, que entrenaremos con el conjunto $\mathcal{D}_2$ de la Ecuación (6), en el que los pares consecutivos de canciones reproducidas se construyen utilizando los datos reservados para el aprendizaje de tareas (25 %). El resto de datos se utiliza para obtener las codificaciones con word2vec y doc2vec, como en la tarea anterior.

Tabla I
TABLA DE RESULTADOS PARA LA TAREA 1

Representación		Precision	Recall	F1
Usuario	Canción			
one-hot	one-hot	77.5	73.8	75.6
one-hot	word2vec	**83.1**	**80.2**	**81.6**
P_{con}	word2vec	80.5	79.1	79.8
P_{rec}	word2vec	77.5	74.5	76.0

Tabla II
TABLA DE RESULTADOS PARA LA TAREA 2

Representación		Precisión						
Usuario	Canción	$P@5$	$P@10$	$P@20$	$P@50$	$P@100$	$P@1000$	Mediana
one-hot	one-hot	1.4	1.8	2.5	4.1	6.2	18.3	17476
one-hot	word2vec	7.7	**12.3**	**17.4**	**25.2**	**31.8**	**58.6**	**511**
P_{con}	word2vec	7.3	9.6	12.8	18.2	23.8	52.1	857
P_{rec}	word2vec	**8.3**	10.9	14.1	19.4	24.7	50.2	986

Para evaluar el rendimiento de las diferentes combinaciones de perfiles, se utilizó la medida *precision at x* ($P@x$), donde x varía en valores entre 5 y 1000. De esta manera podremos ver el porcentaje de veces que la canción que realmente va a escuchar el usuario se encuentra entre las x que más probabilidad les otorga el modelo aprendido. Los resultados de la Tabla II muestran que, nuevamente, la utilización de un perfil precalculado con word2vec para las canciones mejora el rendimiento considerablemente.

Para los perfiles de usuario, los mejores resultados se obtienen con la codificación resultante de la representación one-hot, calculada durante el aprendizaje de la tarea. Si enfrentamos el perfil reciente al consolidado, parece que el reciente ofrece mejor rendimiento, lo cual tiene sentido debido a la naturaleza de la tarea. Cabe destacar que los bajos valores de $P@5$ (por debajo del $10\,\%$ en todos los casos) se deben a que la tarea de recomendación es muy difícil: con tan sólo 5 recomendaciones se le pide al sistema acertar la siguiente canción entre 312895 alternativas posibles. La última columna de la tabla muestra la mediana de la posición que ocupa la canción que debería predecirse para acertar ($\boldsymbol{p}_j$ en la Ecuación 6) en el ranking que se obtiene atendiendo a la probabilidad predicha para cada canción. El valor 511 no es un mal resultado, teniendo en cuenta que hay más de trescientas mil canciones posibles.

Doc2vec, en su versión PV-DM, es entrenado para intentar predecir la siguiente canción, como en esta tarea 2. Los resultados que obtiene son ligeramente mejores, pero utiliza las dos últimas canciones escuchadas por el usuario.

VI. CONCLUSIONES Y TRABAJO FUTURO

Los Sistemas de Recomendación tienen muchas aplicaciones en la actualidad. Su éxito radica, principalmente, en la personalización que se está consiguiendo en las recomendaciones. Es por esto que la creación de perfiles de productos y usuarios cobra especial importancia.

En este artículo hemos evaluado el rendimiento de perfiles precalculados mediante el uso de las redes neuronales word2vec y doc2vec, y los hemos comparado con la codificación que se obtiene al calcular un embedding de los vectores de entrada a un espacio de forma que se optimiza un determinado objetivo. La comparación se ha efectuado en el contexto de dos tareas de aprendizaje: una de carácter más inmediato, predecir la siguiente canción a escuchar, y otra a más largo plazo, predecir si una canción se va a escuchar en el futuro.

Los resultados muestran que el rendimiento de los sistemas de recomendación mejora sustancialmente al introducir el perfil precalculado para las canciones mediante word2vec. Sin embargo, el perfil obtenido para los usuarios con doc2vec no mejora el rendimiento que se obtiene con la codificación obtenida a partir de vectores one-hot.

Como trabajo futuro, sería interesante obtener perfiles de usuario utilizando una red LSTM [15], ya que este tipo de redes es capaz de olvidar productos vistos hace tiempo.

AGRADECIMIENTOS

Agradecemos a NVIDIA Corporation la donación de la GPU Titan Xp utilizada en esta investigación.

REFERENCIAS

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, pp. 56–58, Mar. 1997.

[2] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*, pp. 1–35, Springer, 2011.

[3] J. Díez, D. Martínez-Rego, A. Alonso-Betanzos, O. Luaces, and A. Bahamonde, "Metrical representation of readers and articles in a digital newspaper," in *10th ACM Conference on Recommender Systems (RecSys 2016) Workshop on Profiling User Preferences for Dynamic Online and Real-Time Recommendations (RecProfile 2016)*, ACM, 2016.

[4] O. Luaces, J. Díez, T. Joachims, and A. Bahamonde, "Mapping preferences into euclidean space," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8588 – 8596, 2015.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[6] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014.

[7] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pp. 241–250, ACM, 2000.

[8] M. J. Pazzani and D. Billsus, "The adaptive web," ch. Content-based Recommendation Systems, pp. 325–341, Berlin: Springer-Verlag, 2007.

[9] J. Weston, S. Bengio, and P. Hamel, "Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval," *Journal of New Music Research*, vol. 40, no. 4, pp. 337–348, 2011.

[10] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 714–722, ACM, 2012.

[11] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *JMLR*, vol. 13, pp. 307–361, 2012.

[12] O. Celma, *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," mar 2016.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov 1997.