



**Universidad
Europea**

LAUREATE INTERNATIONAL UNIVERSITIES

Estudio sobre Sistemas de Recomendación y Predicción basados en el procesamiento del lenguaje natural

Hugo Ferrando Seage

Escuela de Arquitectura, Ingeniería y Diseño

julio 2017

Director: Dr. Esteban García-Cuesta

Resumen

Debido a la gran cantidad de contenidos ofrecidos por las plataformas de Video On Demand (VOD) es necesario ofrecer a los usuarios un sistema de recomendación que tenga en cuenta sus gustos para ofrecer una buena usabilidad.

Existen varias técnicas para tal fin como el filtrado colaborativo o el filtrado por contenido, pero tradicionalmente ha sido necesario tener descriptores de las películas, como sus géneros, o una buena cantidad de puntuaciones numéricas de muchos usuarios diferentes.

Gracias a los avances en las técnicas del procesamiento del lenguaje natural quiero explorar la posibilidad de construir un recomendador que se base solamente en críticas escritas por diferentes usuarios. Además pensamos que es posible construir un modelo colaborativo que se base en descriptores sacados de esas críticas en vez de puntuaciones.

Hugo Ferrando Seage
octubre 2016

Abstract

Due to the amount of content offered by Video-On-Demand platforms, it's necessary to offer users a powerful recommendation system capable of taking into account their tastes to offer good usability.

There are several techniques used to achieve this, such as content or collaborative filtering, though traditionally, it's been necessary to use film features, such as their genres or a good amount of numerical scores by several different users.

Thanks to the advancements in the natural language processing techniques, it's now possible to build such a system using only film reviews written by users. In addition, we think it's possible to not only build a content filtering based model, but a collaborative filtering one, using the different features extracted in the natural language processing.

Hugo Ferrando Seage
October 2016

Agradecimientos

Quiero dar gracias a mis padres por todas las oportunidades que me han brindado. Sin ellos no podría haber llegado hasta aquí.

También quiero mencionar a mis compañeros del practicum y a todos los miembros del Big Data Lab de la Universidad Europea de Madrid.

Índice general

1. Introducción	13
1.1. ¿Qué es un sistema de recomendación?	13
1.2. Filtrado Colaborativo	13
1.3. Filtrado por contenido	14
1.4. Evaluación de modelos y otras consideraciones	15
1.5. Antecedentes	15
1.6. Objetivos	16
1.7. Estructura	17
2. Metodología	19
3. Recolección de datos	21
4. Limpieza de textos	25
4.1. POS Tagger	25
4.2. Reemplazar sustantivos por hiperónimos	25
4.3. Eliminación de nombres propios	26
4.4. Filtrar Stopwords	26
4.5. Stemmer	26
5. LSA	29
5.1. TF-IDF	29
5.2. Descomposición en valores singulares	30
5.3. Similitud del coseno	33
6. Doc2Vec	35
6.1. Word2Vec	35
6.2. Doc2Vec	39
7. E-Modelo	41

8. Optimización	45
8.1. LSA	45
8.2. Doc2Vec	52
9. Resultados	59
10. Interfaz	63
11. Conclusión	67
11.1. Mejoras	68
Index	77

Glosario

ALS	Alternating Least Squares. 41, 42
API	Application Programming Interface. 16, 19, 63
CBOW	Continous Bag of Words. 35, 36, 40
ES6	ECMAScript 6. 63
Ground Truth	Datos con los que comparar un modelo para estimar su efectividad. 15, 45
JSON	Javascript Object Notation. 11, 22, 30
LDA	Latent Dirichlet Allocation. 68
LSA	Latent Semantic Analysis. 14, 16, 19, 22, 26, 29, 31–33, 45, 52, 63, 65, 73
MongoDB	Base de datos no relacional basada en archivos JSON. 19
MVP	Minimum Viable Product. 19
one-hot	Vector donde todos los elementos son 0 exceptuando una posición con un 1. 37, 75
REST	Representational state transfer. 16, 19, 63
Softmax	Función logística que transforma los valores de un vector de numero reales a valores en el rango [0, 1]. La suma de todas los valores debe ser igual a 1 (distribución de probabilidad). 37
SPA	Single Page Application. 63

SSR	Server-Side Rendering. 63
SVD	Singular Value Decomposition. 14, 30, 33
testing A/B	Método para evaluar un cambio en un sistema, como puede ser un cambio en un algoritmo de recomendación, a una parte de los usuarios. Después de un plazo se miden los cambio en el comportamiento de los usuarios. 15
TF-IDF	Term Frequency-Inverse Document Frequency. 14, 29, 30, 45, 73
VOD	Video On Demand. 3
Word Embedding	Técnicas de procesamiento de lenguaje natural donde cada palabra del vocabulario es transformada en un vector de valores reales. 35

Capítulo 1

Introducción

1.1. ¿Qué es un sistema de recomendación?

Un sistema de recomendación es cualquier software capaz de recomendar un producto o servicio a un usuario en particular en base a algún criterio, generalmente del propio producto, usuario o alguna combinación de ambos.

Los sistemas de recomendación se pueden aplicar a una multitud de ámbitos: **películas**, series de TV, libros, restaurantes, hoteles, búsquedas online, seguros, sistemas expertos[20], servicios financieros[6], etc.

El software debe ser capaz de hacer un ranking de los diferentes ítems a recomendar y ofrecerlos al usuario. Si las recomendaciones son buenas ayudan en la navegación y el descubrimiento del producto y aumentan la satisfacción del cliente.

A continuación se describen las dos grandes categorías en las que se dividen.

1.2. Filtrado Colaborativo

El filtrado colaborativo trata de emparejar usuarios con gustos similares para después hacer recomendaciones. Los tres pilares son (1) deben participar muchas personas, (2) los usuarios deben representar sus gustos de alguna manera y (3) los algoritmos deben poder encontrar personas con gustos parecidos.[5]

Dentro de estos sistemas existen tres tipos:

- Basados en memoria
- Basados en modelo
- Híbridos

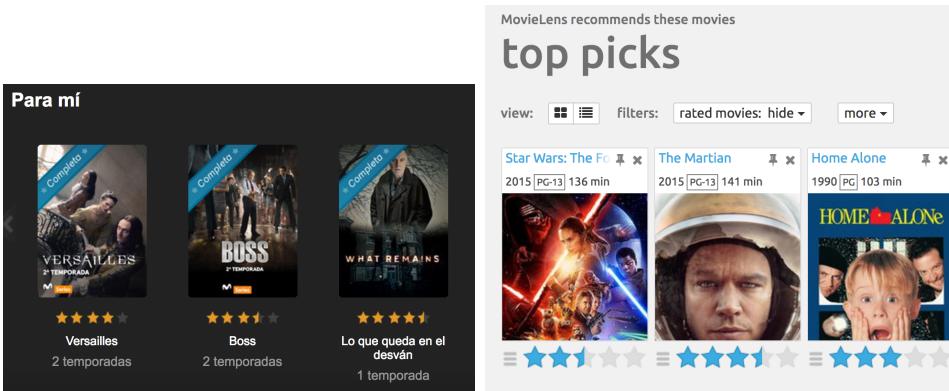
Los basados en memoria usan datos de evaluación de los usuarios (como una puntuación numérica) y calcula la similitud entre usuarios usando algoritmos como *Nearest Neighbour*.

Los basados en modelo usan algoritmos de aprendizaje automático y pueden usar tipos de datos más diversos que los basados en memoria. Los algoritmos incluyen redes bayesianas, LSA, SVD, clustering, cadenas de markov etc. Una de las principales ventajas de estos modelos es su fácil escalabilidad[10] y su capacidad para obtener patrones intrínsecos a los comportamientos homogéneos[13], en este caso, asociados a los usuarios.

Los híbridos usan una combinación de ambos modelos para dar mejores resultados.

Servicios como Netflix, Movistar+ (figura 1.1a) y Movielfense (figura 1.1b) usan filtrado colaborativo para hacer recomendaciones personalizadas a nivel de usuario usando diversas técnicas.

Figura 1.1: Sistemas de recomendación basados en filtrado colaborativo



(a) Recomendaciones personalizadas en Movistar+

(b) Recomendaciones personalizadas en Movielfense

1.3. Filtrado por contenido

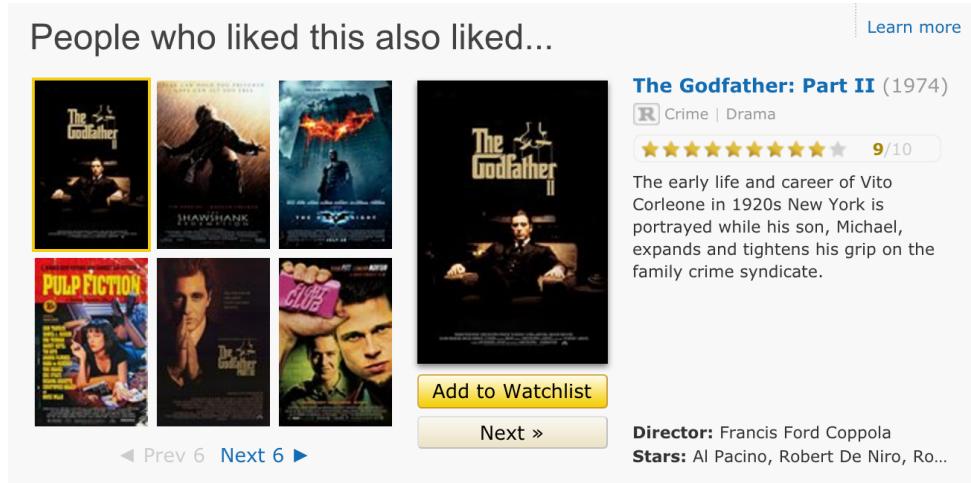
Este tipo de filtrado relaciona los gustos de un usuario con las propiedades de un ítem sin tener en cuenta a otros usuarios.

Las características de los ítems pueden ser de todo tipo. En las películas podría ser los géneros que le corresponden, en restaurantes el tipo de comida que sirven, etc

Para realizar las recomendaciones los sistemas se suelen basar en acciones anteriores de un usuario para poder recomendar algo (al comprar unos neumáticos recomendar un recambio de aceite, por ejemplo), o se puede crear un modelo en base a los descriptores de los ítems usando redes neuronales, TF-IDF, redes bayesianas, clusters, árboles de decisión etc.

IMDb (figura 1.2) usa este tipo de sistema para hacer sus recomendaciones al visitar la página de una película.

Figura 1.2: Recomendaciones por película en IMDb



1.4. Evaluación de modelos y otras consideraciones

La evaluación de estos modelos puede presentar algún problema, ya que no se suele contar un '*Ground Truth*' contra los que evaluar la precisión del modelo. Por esto se suele evaluar de manera manual y preguntando por la satisfacción de los usuarios o usando testing A/B[15].

Además, a la hora de recomendar, los usuarios también suelen valorar otros aspectos que el sistema de recomendación no tome en cuenta.

Por ejemplo, es interesante que las recomendaciones sorprendan[9] al usuario y que sean lo suficientemente diferentes entre sí. También es interesante hacer recomendaciones diferentes teniendo en cuenta el lugar donde el usuario vive o su edad. Esto puede suponer problemas ya que el usuario se puede dar cuenta de que el sistema no respeta su privacidad.

1.5. Antecedentes

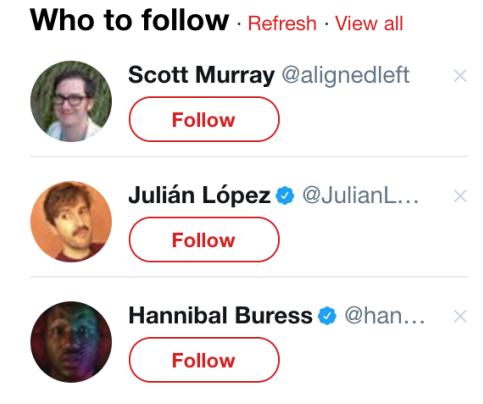
Existen varias técnicas usadas a día de hoy para recomendar películas. Varios servicios cuentan con recomendaciones por película como IMDb (figura 1.2) y Movistar+. Otros servicios ofrecen recomendaciones con filtrado colaborativo a cada usuario como Netflix y Movistar+ (figura 1.1a). Además existen servicios que se dedican exclusivamente a hacer estas recomendaciones como Movielense (figura 1.1b), Jinni y Taste.io.

Muchos de estos servicios se basan en las puntuaciones o visualizaciones de los contenidos vistos por el usuario para realizar un filtrado colaborativo. Otros servicios usan tags, como los ofrecidos por IMDb y los géneros, para ver que películas tienen más cosas en común y así poder recomendarlas.

En otros ámbitos se usan sistemas similares. Last.fm hace recomendaciones en base a la música que escuchan sus usuarios. Twitter es capaz de recomendar varias cuentas para

seguir a sus usuarios usando una combinación de muchas señales[16], como se puede ver en la figura 1.3.

Figura 1.3: Recomendaciones en Twitter



1.6. Objetivos

El objetivo principal es la creación de una aplicación web donde un usuario pueda escoger una película y obtenga 10 películas recomendadas usando diferentes algoritmos basados en el procesamiento de lenguaje natural.

En concreto se implementarán modelos basados en LSA y Doc2Vec, además de realizar una optimización de sus parámetros para su evaluación. También se describirá un tercer modelo híbrido llamado E-Modelo[22], que, pese a haber completado su implementación, no se ha incluido en el recomendador.

También se implementa una interfaz común donde poder hacer peticiones a los diferentes modelos usando una API REST, además de la interfaz web desde donde probar los algoritmos. Ésta última debe ser rápida y lo suficientemente intuitiva para que un usuario normal lo pueda usar. La API REST es pública y puede servir para la creación de otros servicios ajenos a este proyecto.

Para la creación de modelos es necesario tener un buen conjunto de datos con los que realizar el entrenamiento. Para ello se realiza otro programa que descarga los datos necesarios de IMDb mediante un crawler.

El código del proyecto está publicado en tres repositorios de GitHub (todo el código bajo licencia MIT):

- *Frontend* <https://github.com/hugo19941994/movie-pepper-back>
- *Backend* <https://github.com/hugo19941994/movie-pepper-front>
- *Documentación* <https://github.com/hugo19941994/movie-pepper-doc>

El proyecto está en producción y se puede visitar en <https://moviepepper.hugofs.com>.

1.7. Estructura

En primer lugar se describe la metodología seguida para el desarrollo del proyecto, mientras que los capítulos siguientes se centran en la parte mas técnica del proyecto y describen los diferentes procesos y métodos usados. Esto incluye como se consiguen los datos para el recomendador, como se filtra el texto y los algoritmos usados para la creación de los modelos.

En segundo lugar se presentan unas gráficas con los resultados al variar los parámetros de los dos algoritmos implementados y unos resultados finales con los modelos finalizados.

El capítulo *Interfaz* habla sobre la web desde donde se pueden probar los modelos y como se ha creado.

Finalmente se presentan unas conclusiones sobre el proyecto y se habla sobre partes que se podrían mejorar.

Capítulo 2

Metodología

La metodología de este proyecto ha sido ágil, basado en la creación de MVPs. Gracias a esta metodología se pueden evaluar diferentes caminos e hipótesis posibles con una cantidad de recursos limitados. También reduce los riesgos frente a posibles fallos o experimentos.

Cada iteración del proyecto constaba con unos objetivos claros y un alcance limitado. Después de su realización se lanzaban una serie de pruebas para verificar su correcto funcionamiento y se analizaban los resultados obtenidos. Si los resultados no fueron satisfactorios se descartaba esta iteración y se realizaba alguna modificación para poder ir mejorando.

En primer lugar realizamos la fase de recolección de datos. Se probaron tres sitios diferentes: Wikipedia, IMDb[23] y FilmAffinity. En Wikipedia es demasiado difícil sacar la información de las diferentes películas manera precisa. Tanto FilmAffinity como IMDb[23] tienen contenidos más estructurados. FilmAffinity tiene tags de género más precisos, pero IMDb[23] cuenta con muchos más usuarios y por lo tanto más variedad de tags y de críticas.

Se empezó usando una combinación de las 3 fuentes, pero tras hacer pruebas se vio que con los datos de IMDb[23] era suficiente y simplificaba el código.

También se usó MongoDB para guardar toda la información, pero finalmente se descartó por el mismo motivo de arriba. Complicaba el código y en este proyecto no aportaba muchos beneficios, sobre todo al eliminar la información de FilmAffinity y Wikipedia.

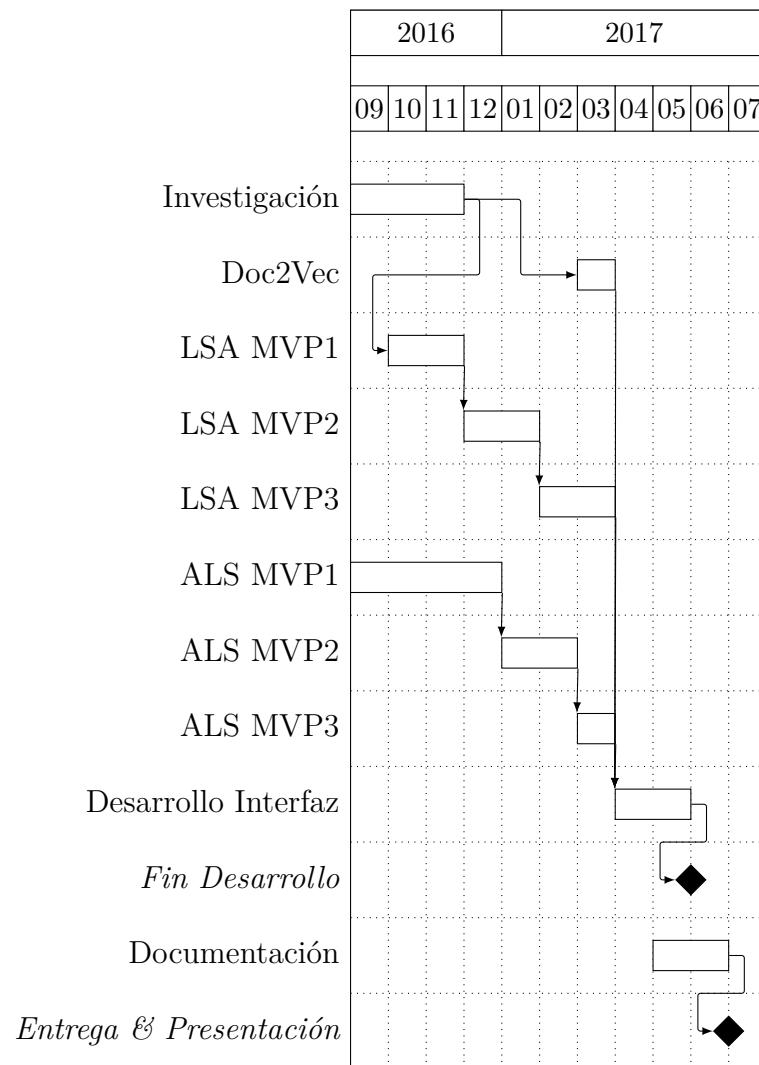
Como orden de implementación se realizó primero el modelo LSA y en paralelo se realizó el proyecto de E-Modelo[22]. Debido a las resultados del E-Modelo se decidió reemplazarlo con Doc2Vec.

Cuando todos los modelos fueron optimizados y evaluados se realizó la interfaz web, el API REST y este documento.

Hay que mencionar que la inspiración para este proyecto fue la realización de tareas de mejora e investigación en un motor de recomendación de películas al hacer el practicum,

aunque el código en sí no guarda ninguna relación.

Figura 2.1: Diagrama de Gantt con tiempos del proyecto

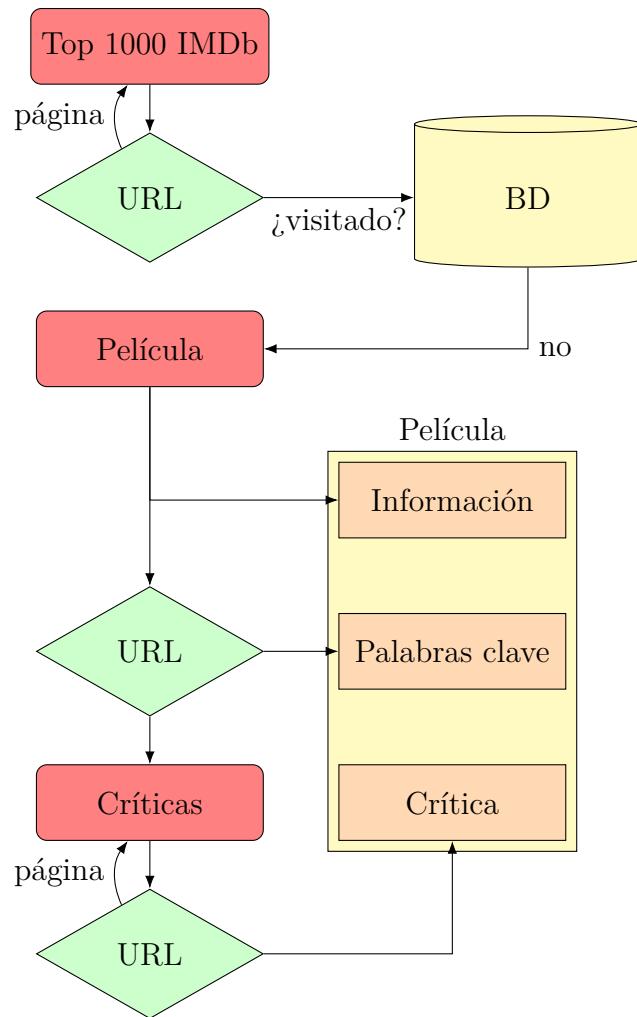


Capítulo 3

Recolección de datos

El primer paso es la obtención de datos con los que hacer las recomendaciones y entrenar los modelos. Para esta tarea se ha usado Scrapy[27], un paquete de Python con el que crear crawlers de Internet.

Figura 3.1: Diagrama de flujo de la descarga de datos



El crawler inicia a indexar en la pagina de top 1000 películas por valoración de los usuarios (link). El crawler descarga los siguientes datos de cada película.

- Título
- Directores
- Año
- Género
- Sinopsis
- Críticas
- Palabras claves
- Carátula
- URL

El crawler entra en todos los enlaces de la pagina inicial. Si el siguiente enlace es la siguiente lista de películas repite el proceso. Si el enlace pertenece a una película se descarga los datos y busca la pagina de las keywords, sinopsis y criticas. Si el enlace no es la ficha de una película ni parte de la lista se descarta. Este proceso termina cuando no queda ningún enlace por indexar. Este proceso se realiza de manera asíncrona (explicar) usando Python 3.6.

Como criticas se descargan como máximo las 25 con mejores valoraciones. Esto, junto a la sinopsis forman los datos con los que entrenar tanto el modelo Doc2Vec como el LSA.

Todo el texto descargado esta en inglés. Si algunos de los datos estuviese en otro idioma sería necesario traducirlo o descartarlo. En una primera versión se descargaban datos tanto en inglés como en español y se traducía usando la acrshortapi de traducción de Microsoft. Finalmente se eliminó, ya que por lo general la mayoría de contenido del top 1000 de IMDb[23] es de habla inglesa, al igual que la mayoría de críticas escritas allí. Eliminando los datos en castellano se simplificaba el código y se elimina una dependencia de un servicio externo.

Toda la información se guarda en ficheros JSON. Un ejemplo de documento es:

```
{  
    "url": "http://www.imdb.com/title/tt0816692/?ref_=adv_li_tt",  
    "title": "Interstellar",  
    "rating": "8.6",  
    "genres": [  
        "Adventure",  
        "Drama",  
        "Sci-Fi"  
    ],  
    "year": "2014",  
    "director": "Christopher Nolan",
```

```

"poster":  

    → "https://images-na.ssl-images-amazon.com/images/M/MV5B.jpg",  

"keywords": [  

    "space travel",  

    "father daughter relationship",  

    "wormhole",  

    "black hole",  

    "saving the world",  

    "astronaut",  

    "relativity",  

    "..."  

],  

"plot": "A group of elderly people are giving interviews about  

    → having lived in a climate of crop blight and constant dust  

    → reminiscent of The Great Depression of the 1930's. The first one  

    → seen is an elderly woman stating her father was a farmer, but  

    → did not start out that way. The scene changes. We are introduced  

    → to a farmer and widower named Joe Cooper (Matthew McConaughey).  

    → He is a college-educated former NASA test pilot and engineer  

    → ...",  

"reviews": [  

    "I am wondering if the people who write 'intertstellar' 10-star  

    → reviews are actually sane, or if they base their opinion on a  

    → movie library completely deprived of everything that is  

    → sci-fi. Basically if the sic-fi component - most of which does  

    → not really happen until the end -is removed from the plot the  

    → movie by and large is a classic Hollywood soap opera. The  

    → worn-out rag of cliché family relations ...",  

    "(((WOW))) ... !!! It's almost impossible to put in words, but we  

    → have to try and give everyone what he truly deserve. In my  

    → modest opinion, Interstellar is the best Sci-Fi in human  

    → history. Believe it or not. It's the simple and the  

    → complicated, It's the usual and the different...",  

    "..."  

]
}

```


Capítulo 4

Limpieza de textos

El preprocessado de los textos es una parte esencial del procesamiento del lenguaje natural. Sin este paso la precisión del modelo es mucho menor ya que el modelo tendrá mucho ruido. Es importante eliminar todas las ambigüedades posibles para poder relacionar los conceptos que existen dentro de los textos y así ayudar a los algoritmos a cumplir su función.

Algunas de las técnicas más comunes incluyen eliminar tiempos verbales, signos de puntuación, distinción entre palabras singulares y plurales, masculinas y femeninas, mayúsculas y minúsculas, palabras muy comunes, etc.

4.1. POS Tagger

En primer lugar sepáramos cada documento en frases. Se usa un POS Tagger entrenado con el Penn Treebank tagset[25] (el tagger por defecto de NLTK[7]):

Zeus is a Greek God.

Quedaría tagueado así

Figura 4.1: Resultados del POS tagging

NNP	VBZ	DT	NN	NNP
Zeus	is	a	Greek	God .

Donde NNP es nombre propio, VBZ un verbo, DT determinante y NN nombre común

4.2. Reemplazar sustantivos por hiperónimos

Cualquier sustantivo se intenta reemplazar por un hiperónimo (palabra de la misma familia pero de categoría más general, como animal es a perro o audio a altavoz). Para

obtener el hiperónimo de forma precisa usamos la función lesk[3] de NLTK[7], que detecta el contexto de una palabra (es capaz de diferenciar gato de herramienta a gato de animal dependiendo de las palabras que rodean a la palabra en cuestión).

El resultado de lesk[3] es una lista ordenada por relevancia de hiperónimos del blob. Se reemplaza por el primer resultado (el más relevante).

Así el texto se convertirá en:

```
Zeus is a country deity.
```

4.3. Eliminación de nombres propios

Los nombres propios se eliminan porque se detectó que si dos personajes de películas diferentes se llamaban igual la similitud resultante de los cálculos con LSA muy alta, aunque las películas no tengan nada que ver. Para evitar esto se eliminan.

Así el texto se convertirá en:

```
is a country deity.
```

4.4. Filtrar Stopwords

En este paso se elimina cualquier token que coincida con los stopwords de la lista inglesa de NLTK[7] y cualquier número. Esta lista incluye palabras muy comunes como ‘i’, ‘me’, ‘my’, ‘am’, ‘those’, ‘few’, etc.

También se eliminan los signos de puntuación y se pasan todas las palabras a minúsculas.

Nuestro texto anterior ahora se convertirá en:

```
country deity
```

4.5. Stemmer

El ultimo paso del filtrado del texto es pasar el SnowBall[28] stemmer de NLTK[7]. En este caso Snowball usa el algoritmo de Porter[2].

El stemming reduce cualquier palabra a su raíz, eliminando las partes de palabras que las hacen plurales, masculino o femenino y diferentes tiempos verbales, entre otros. Esto permite a nuestro recomendador interpretar palabras similares en los diferentes documentos como iguales. Algunos ejemplos:

consign ⇒ consign
consigned ⇒ consign
consigning ⇒ consign
consignment ⇒ consign

Siguiendo con el mismo ejemplo:

counti deiti

Esto relacionará palabras como counties, country, countries, etc.

Capítulo 5

LSA

Latent Semantic Analysis (LSA) es el algoritmo en el que se basa el primer modelo de recomendación implementado en el proyecto. Se trata de una serie de técnicas usadas para determinar la distribución semántica y la relación entre documentos extrayendo conceptos. La hipótesis principal de LSA es que las palabras que tienen alguna relación semántica ocurrirán en documentos que también tienen una relación.

Estas técnicas se empezaron a desarrollar en los años 60[1], pero se popularizaron en los años 90[4]. A día de hoy sigue siendo una de las técnicas más usadas dentro del procesamiento del lenguaje natural.

5.1. TF-IDF

El Term Frequency-Inverse Document Frequency (TF-IDF) calcula lo relevante que es una palabra dentro de un conjunto de documentos. Construye una matriz de relevancia por documento, donde cada casilla representa la relevancia de un término en él. Cuanto más aparece un término en un documento, más relevante es, pero cuanto más aparece en global, menos peso tendrá. El 83 % de los recomendadores que usan textos se basan en este algoritmo[21].

Algunos de los parámetros que se pueden ajustar son el número mínimo y máximo de veces que un término puede aparecer en el conjunto para entrar dentro del cálculo. También se puede ajustar el número de palabras máximas a analizar. Si ese número se sobrepasa se eliminan las palabras con menos relevancia. En el capítulo 8 se ajustarán los parámetros para ver cuáles son los idóneos para esta tarea.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (5.1)$$

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (5.2)$$

$$idf(t, d) = \log\left(\frac{|D|}{1 + \{d \in D : t \in d\}}\right) \quad (5.3)$$

Para cada conjunto de documentos filtrados se hace el TF-IDF y quedan las palabras mas relevantes de cada película. Un pequeño ejemplo sería:

Figura 5.1: Ejemplo matriz TF-IDF

	says	just	room	dead	asks	ship	mother
The Matrix	0,39	0,16	0,19	0,01	0,25	0,79	0,27
Alien	0,12	0,12	0,06	0,46	0,21	0,07	0,83
$tfdif =$ Serenity	0,46	0,55	0,15	0,55	0,22	0,27	0,11
Casablanca	0,00	0,60	0,51	0,00	0,00	0,60	0,00
Amelie	0,41	0,00	0,35	0,83	0,00	0,00	0,00

Esto crea una matriz donde cada fila corresponde a un documento del corpus, mientras que las columnas representan las palabras del vocabulario. Cada casilla corresponde a la relevancia de esa palabra en ese documento. En nuestro caso guardamos cada fila en el JSON de la películas correspondiente. Si fuese necesario, en este paso se pueden extraer los tokens más relevantes de cada documento para su posterior uso. Cuando se ejecute SVD se perderán esos datos, pero para el recomendador no va a ser necesario guardarlos.

El código para realizar la matriz TF-IDF se basa en la implementación de scikit-learn[26]

5.2. Descomposición en valores singulares

Con Singular Value Decomposition (SVD) reducimos la dimensionalidad de la matriz calculada con TF-IDF. Esto acelera de una manera muy significativa los cálculos y además soluciona el problema de la polisemia. Existen varias formas de calcular el SVD, pero scikit-learn implementa el algoritmo descrito en «Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions»

SVD busca descomponer una matriz natural en 3 matrices diferentes diferentes. A representa la matriz de documentos donde cada fila representa los diferentes términos y las columnas representan los diferentes documentos.[11]

Figura 5.2: Ejemplo matriz LSA

$$A = \begin{matrix} & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \text{Action} & \left\{ \begin{array}{l} \text{action} \\ \text{gun} \\ \text{shoot} \\ \text{run} \\ \text{love} \\ \text{peace} \\ \text{kiss} \end{array} \right\} & \left(\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right) & = U \cdot \Sigma \cdot V^T \end{matrix}$$

Figura 5.3: Matriz LSA palabra a categoría

$$U = \begin{matrix} & \text{Sci-Fi topic} & \text{Romance topic} & \text{Ruido} \\ & \text{action} & 0,13 & 0,02 & -0,01 \\ & \text{gun} & 0,41 & 0,07 & -0,03 \\ & \text{shoot} & 0,55 & 0,09 & -0,04 \\ & \text{run} & 0,68 & 0,11 & -0,05 \\ & \text{love} & 0,15 & -0,59 & 0,65 \\ & \text{peace} & 0,07 & -0,73 & 0,67 \\ & \text{kiss} & 0,07 & -0,29 & 0,32 \end{matrix}$$

Donde U se considera la matriz palabra-a-categoría

Figura 5.4: Ejemplo matriz Σ de LSA

$$\Sigma = \begin{pmatrix} & \text{Sci-Fi topic} & \text{Romance topic} & \text{Ruido} \\ & 12,4 & 0 & 0 \\ & 0 & 9,5 & 0 \\ & 0 & 0 & 1,3 \end{pmatrix}$$

Donde Σ se considera la matriz de importancia de cada categoría

Figura 5.5: Matriz LSA documento a categoría

$$V^T = \begin{matrix} & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \text{Sci-Fi topic} & 0,56 & 0,59 & 0,56 & 0,09 & 0,09 \\ \text{Romance topic} & 0,12 & -0,02 & 0,12 & -0,69 & -0,69 \\ \text{Ruido} & 0,40 & -0,80 & 0,40 & 0,09 & 0,09 \end{matrix}$$

V^T se considera la matriz de documento-a-categoría

Es importante recordar que para cualquier matriz natural A es posible calcular su SVD donde: U, Σ & V son únicos U & V son ortonormales:

$$U^T \cdot U = I \quad (5.4)$$

$$V^T \cdot V = I \quad (5.5)$$

Σ es diagonal y sus valores singulares están en orden descendiente ($\sigma_1 > \sigma_2 > \sigma_3 \dots$)

Estas matrices se guardan en el mismo documento dentro de la base de datos y se usan a la hora de recomendar haciendo la distancia del coseno entre las matrices de dos películas. La implementación del LSA también se basa en la de scikit-learn[26]

También es normal reducir la dimensionalidad eliminando los conceptos que tengan menor relevancia en nuestros cálculos. En este ejemplo, el concepto llamado ‘Ruido’ tiene una relevancia mucho menor que los otros dos conceptos (figura 5.4), y por tanto se puede eliminar ya que no aportará mucha información (figura 5.6).

Figura 5.6: Matriz LSA documento a categoría reducida

$$V^T = \begin{matrix} & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \text{Sci-Fi topic} & 0,56 & 0,59 & 0,56 & 0,09 & 0,09 \\ \text{Romance topic} & 0,12 & -0,02 & 0,12 & -0,69 & -0,69 \end{matrix}$$

V^T después de la eliminación del concepto ‘Ruido’

5.3. Similitud del coseno

Para obtener los valores finales de LSA se debe calcular la similitud del coseno entre los vectores de documentos. Esto nos da un valor entre -1 y 1 pero como nos movemos en un espacio positivo los valores finales serán entre 0 y 1. En las figuras 5.9b y 5.9a se puede ver una representación gráfica en 2D.

$$\cos(\theta) = \frac{\sum_{i=1}^n A_i \cdot B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}} \quad (5.6)$$

Siguiendo con el ejemplo anterior:

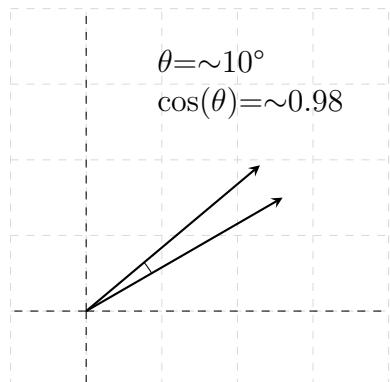
Figura 5.7: Alta similitud entre Matrix y Alien

$$\cos \left(\begin{pmatrix} 0,56 \\ 0,12 \end{pmatrix}, \begin{pmatrix} 0,59 \\ -0,02 \end{pmatrix} \right) = 0,97 \quad (5.7)$$

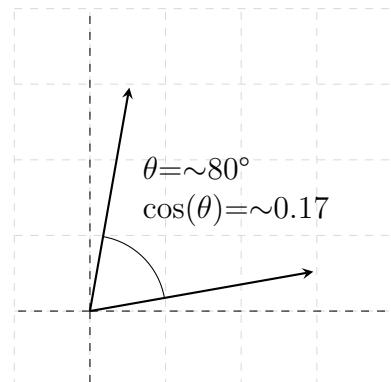
Figura 5.8: Baja similitud entre Matrix y Amelie

$$\cos \left(\begin{pmatrix} 0,56 \\ 0,12 \end{pmatrix}, \begin{pmatrix} 0,09 \\ -0,69 \end{pmatrix} \right) = -0,08 \quad (5.8)$$

Figura 5.9: Visualización de la similitud del coseno



(a) Alta similitud del coseno



(b) Baja similitud del coseno

Capítulo 6

Doc2Vec

Word2Vec[17] es un algoritmo de tipo Word Embedding creado por Google en 2013. Doc2Vec es un algoritmo que se basa en Word2Vec pero trabajando sobre múltiples documentos para así tener vectores de párrafo en vez de vectores de palabras. Fue desarrollado en 2014 por los mismos investigadores de Google[18], aunque no lo llamaron Doc2Vec (así se llama la implementación en Gensim[12]).

Este algoritmo se va a usar para tratar de descubrir que películas tienen más relación entre sí y así poder crear otro modelo de recomendación similar al creado en el capítulo 5.

Para explicar el funcionamiento de Doc2Vec es esencial hablar antes de Word2Vec.

6.1. Word2Vec

Word2Vec crea una red neuronal con una única capa oculta. Existen dos modelos, Continous Bag of Words (CBOW) y Skip-Gram, el segundo dando, por lo general, mejores resultados.

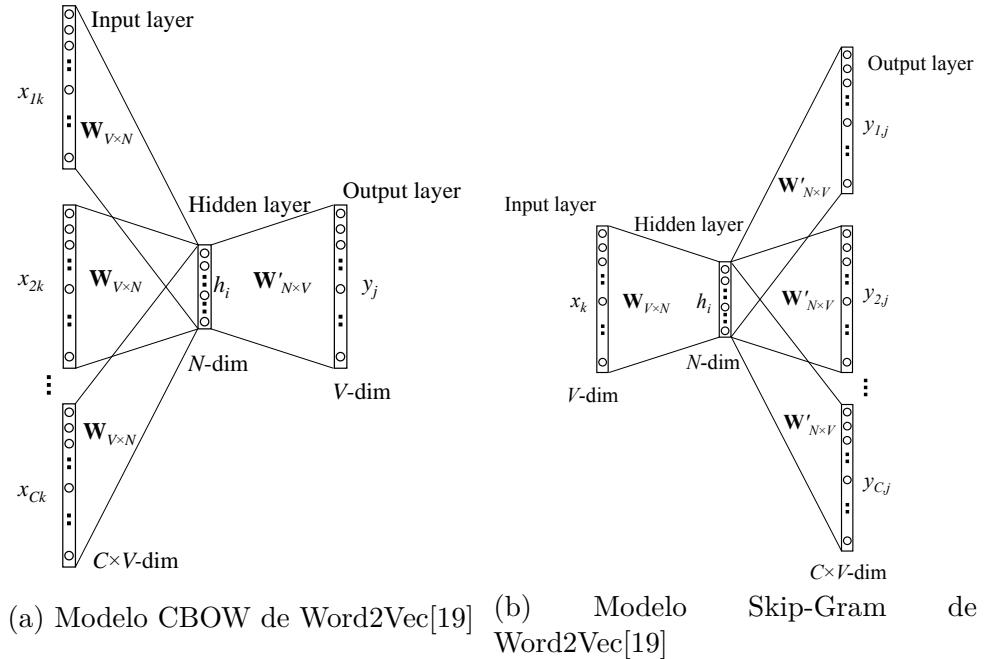
El objetivo del modelo CBOW es predecir la posibilidad de que una palabra aparezca en un contexto dado. Por el contrario, el objetivo de un modelo Skip-Gram es el de predecir que palabras aparecen en el contexto de una palabra. A continuación se explica el funcionamiento, a alto nivel, del modelo Skip-Gram.

En primer lugar definamos el contexto de las palabras en un corpus. Se elige una ventana de palabras (normalmente un valor entre 2 y 5). Para cada palabra de nuestro corpus creamos parejas de palabras dentro de su ventana por el lado izquierdo y derecho, como se muestra en la tabla 6.1.

Con las palabras y sus contextos definidos se puede empezar a entrenar nuestro modelo. La figura 6.1b muestra una representación gráfica de la red neuronal que se va a entrenar.

El entrenamiento necesitará 3 capas (input, oculta y output). El número de neuronas en el input y el output será igual al número de palabras en el vocabulario, mientras que el número de neuronas en la capa oculta es un parámetro del modelo que se puede

Figura 6.1: Modelos Word2Vec



Cuadro 6.1: Ventana deslizante en Word2Vec

Source Text	Training Samples
The quick brown fox jumps over the lazy dog	(‘the’, ‘quick’) (‘the’, ‘brown’)
The quick brown fox jumps over the lazy dog	(‘quick’, ‘the’) (‘quick’, ‘brown’) (‘quick’, ‘fox’)
The quick brown fox jumps over the lazy dog	(‘brown’, ‘the’) (‘brown’, ‘quick’) (‘brown’, ‘fox’) (‘brown’, ‘jumps’)
The quick brown fox jumps over the lazy dog	(‘fox’, ‘quick’) (‘fox’, ‘brown’) (‘fox’, ‘jumps’) (‘fox’, ‘over’)

Cuadro 6.2: Vectores one-hot

Palabra	Posición por orden alfabético	Vector
fox	2/3	[0, 1, 0]
dog	1/3	[1, 0, 0]
zebra	3/3	[0, 0, 1]

ajustar. Los investigadores que desarrollaron este modelo recomiendan elegir entre 50 y 500 neuronas[17].

En el input pasarán vectores one-hot que representan cada palabra (normalmente en orden alfabético), como se puede ver en la figura 6.2.

Las matrices W y W' se inicializan con unos valores aleatorios, como en cualquier red neuronal. Esta red tiene una función de activación lineal que lo único que hace es pasar a la capa oculta la fila que representa la palabra del input de la matriz W (figura 6.2).

Figura 6.2: Función de activación lineal en Word2Vec

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & 12 & 19 \\ 11 & 18 & 25 \end{pmatrix} = \begin{pmatrix} 10 & 12 & 19 \end{pmatrix}$$

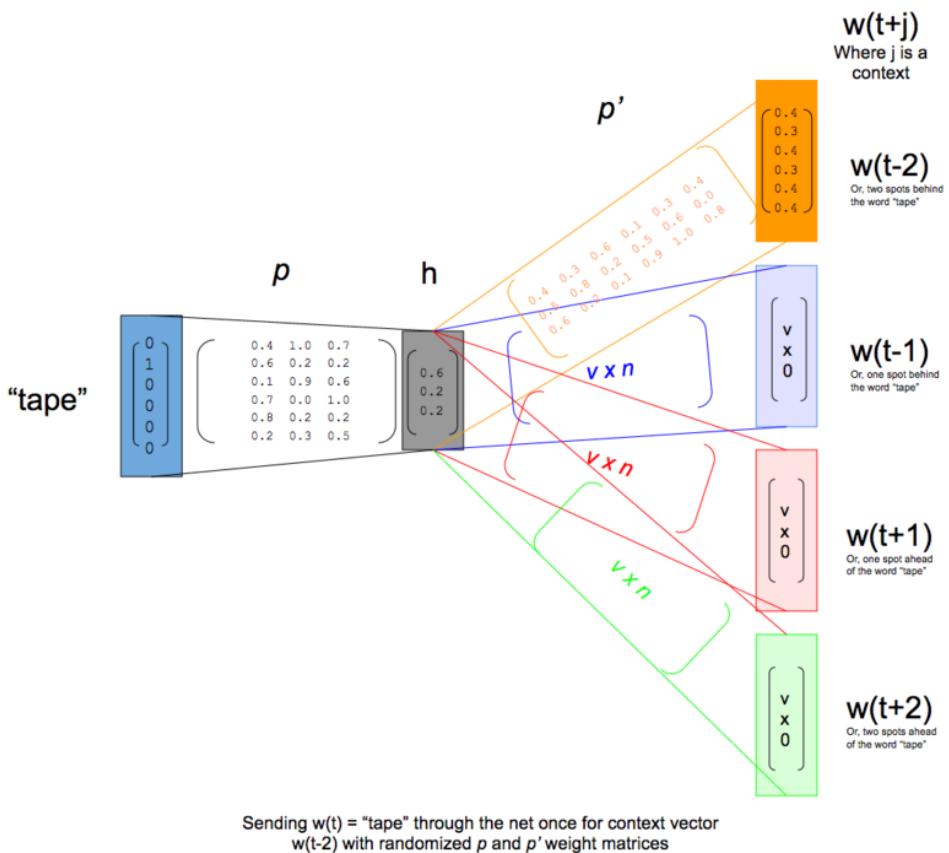
$$X_3 \cdot W_{VxN} = h_3$$

En el output habrá tantos vectores como hayamos configurado en nuestra ventana (el tamaño del contexto).

En cada iteración de entrenamiento se siguen los siguientes pasos (la figura 6.3 puede ayudar a visualizar el algoritmo):

- Multiplicación de h y W'
- Convertir resultado a probabilidades usando Softmax
- Calcular el error entre el resultado y el objetivo (usando los datos reales)
- Actualizar pesos en W y W' usando backpropagation

Figura 6.3: Representación visual del modelo Skip-Gram[24]



Para realizar el cálculo probabilístico se minimiza una función de pérdida usando un descenso estocástico[19].

En la práctica, usar softmax con todo el vocabulario no es realista. Para solventar éste problema se usan unas optimizaciones como usar un softmax jerárquico en vez del normal, desechar palabras poco usadas y usar ‘negative subsampling’, entre otras.

La probabilidad de que una palabra se quede en el vocabulario es:

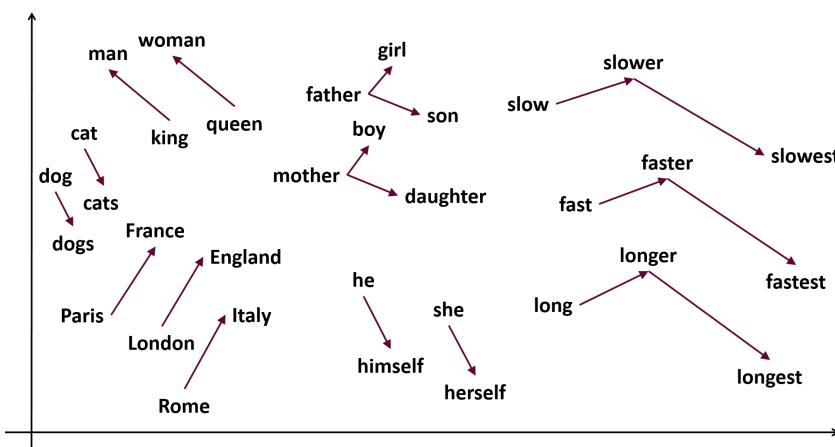
$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad (6.1)$$

Donde t es un threshold definido por el usuario, w_i una palabra en el corpus y $f(w_i)$

Tras repetir los pasos el número de iteraciones de veces programado la creación del modelo habrá concluido y nuestras probabilidades se encontrarán en la matriz W (en la figura 6.1b entre nuestro vector de entrada y la capa oculta).

Los resultados del modelo son unos vectores para cada palabra de nuestro vocabulario, que se puede comparar con otras para ver su similitud.

Figura 6.4: Vectores de resultado de Word2Vec en 2D[29]



En la tabla 6.3 se pueden ver algunos resultados reales del modelo entrenado con críticas de IMDb. La figura 6.4 representa los vectores de un modelo general, visualizado en un plano 2D.

6.2. Doc2Vec

Conceptualmente, Doc2Vec es un algoritmo muy similar a Word2Vec, pero distribuyendo el vocabulario entre varios documentos (o párrafos).

Existen dos modelos diferentes que, de alguna manera, extienden las dos implementaciones de Word2Vec: Distributed Memory Model & Distributed Bag of Words. En muchas tareas DBOW funciona mejor, aunque una combinación de los dos modelos da resultados más consistentes[18].

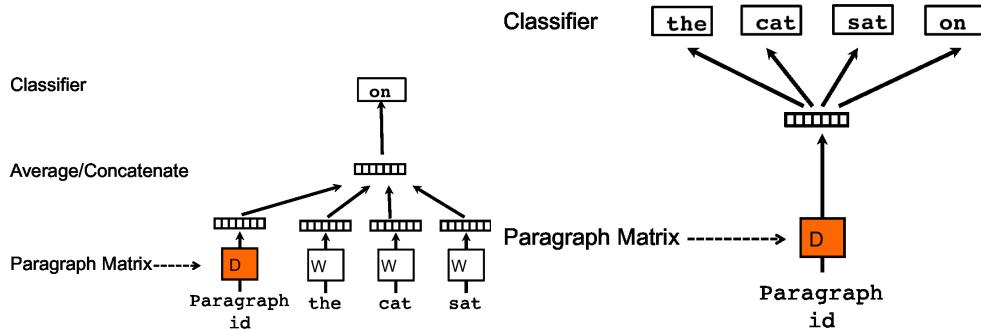
Cuadro 6.3: Resultados Word2Vec

Palabra clave	Palabras similares				
mafia	dealer	banker	mob	mexican	drug
axe	expedition	undercover	biker	employee	outlaw
hair	clothes	coat	eyes	teeth	makeup
airplane	announcer	sanctuary	engine	accident	ambushed
plant	retrieve	investigate	thwart	unearth	escape
air	automatic	oil	swinging	oxygen	ocean

El Distributed Memory Model se basa en el modelo CBOW, pero añadiendo en el input el id del documento correspondiente y agregando el vector resultante al vector W.

El Distributed Bag of Words Model (figura 6.5b) se basa en el modelo Skip-Gram de Word2Vec. En el input, en vez de aparecer una palabra aparece el id de un documento. La red luego se entrena con un subconjunto de contextos del documento.

Figura 6.5: Modelos Doc2Vec



En ambos casos el resultado del modelo vuelven a ser vectores correspondientes a los documentos con los que se ha entrenado y se pueden hacer las mismas comparaciones que con los vectores de Word2Vec.

Capítulo 7

E-Modelo

E-Modelo[22] es un algoritmo de recomendación híbrido que intenta aunar el procesamiento de lenguaje natural y los conceptos extraídos de textos con un filtrado colaborativo. Para el filtrado colaborativo se usa Alternating Least Squares (ALS) (la implementación distribuida de Apache Spark).

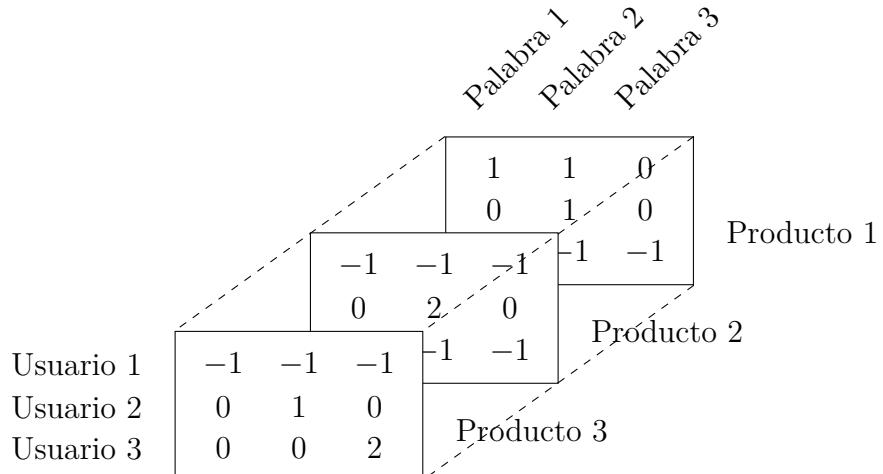
En primer lugar es necesario obtener un buen corpus de datos de usuarios y textos. Existen varios datasets como los de Amazon y IMDb. Para los experimentos realizados se usó un dataset con datos de Amazon, pero el algoritmo se podría aplicar a otros ámbitos.

El dataset contiene textos de críticas de usuarios de un producto en particular. Cada producto y cada usuario tiene un identificador único.

Hacemos un procesamiento de los diferentes textos extrayendo palabras clave. El proceso es similar al explicado en el capítulo 4.

La matriz del filtrado colaborativo es una matriz tridimensional donde los ejes corresponden a usuarios, productos y palabras.

Figura 7.1: Matriz 3D del E-Modelo



Para calcular el resultado de ALS es necesario reducir a dos dimensiones la matriz del

E-Modelo. Para ello pasamos las palabras a columnas, para representar productos-palabra.

Figura 7.2: Matriz E-Modelo de 2 dimensiones

	Producto 1			Producto 2			Producto 3		
	Palabra 1	Palabra 2	Palabra 3	Palabra 1	Palabra 2	Palabra 3	Palabra 1	Palabra 2	Palabra 3
Usuario 1	1	1	0	-1	-1	-1	-1	-1	-1
Usuario 2	0	1	0	0	2	0	0	1	0
Usuario 3	-1	-1	-1	-1	-1	-1	0	0	2

Los -1 representan huecos, lo que significa que el usuario no ha escrito una crítica para ese producto en concreto. Los 0 son para palabras del vocabulario que un usuario no ha usado en una de sus críticas.

En matrices más grandes se hace un filtrado posterior para eliminar palabras cuyas columnas sean demasiado dispersas para reducir la cantidad de ceros.

Luego se pasa a un ALS normal. Como el cálculo de matrices muy grandes es muy elevado se usó un cluster de varios ordenadores para acelerar el análisis.

Antes de crear el modelo final se optimizan probando una serie de parámetros del modelo ALS usando un 10 % de los datos para validación, un 20 % para test y un 70 % para el entrenamiento. Los vectores son escogidos al azar. Una vez probados los parámetros se crea el modelo final con la combinación que mejor resultado dio.

Los parámetros son:

- Rank
- Lambda
- Número de iteraciones
- Seed

El resultado es una matriz donde los -1 que había en la matriz original se han convertido en valores estimados. Esos valores representan con qué frecuencia un usuario usará una palabra en una crítica de un producto que no ha comprado. Si usamos palabras que describen gustos, como pueden ser los adjetivos calificativos, se puede distinguir cuando una persona va a estar satisfecho con su compra y cuando no.

Resaltar que en este caso la recomendación vendría dada una vez se realiza la predicción de palabras para un usuario y un producto a través de un consiguiente análisis de dichas palabras (por ejemplo, de sentimiento). Este modelo ha sido probado en cuanto a su eficiencia en predicción[22] llegando tener una mejor tasa en precisión cercana al 40 %,

pero todavía no es comparable con los ratios de precisión obtenidos en predicción de valoración dado su estado incipiente.

Capítulo 8

Optimización

Antes de obtener unos resultados definitivos es necesario buscar los mejores parámetros de los diferentes algoritmos. No es fácil evaluar un modelo como éste, ya que no existen unos datos que usar como '*Ground Truth*'.

Se decidió optimizar el modelo usando películas películas que, a mi parecer, son similares entre sí y analizar qué parámetros ajustaban más el resultado. Se ha usado una película de ciencia ficción, una de acción y una de animación infantil.

En las gráficas todos los puntos se han movido lateralmente unos milímetros para que no se solapen en el caso de que su posición sea la misma.

Las líneas en las gráficas se han generado usando una regresión lineal simple para poder ver la tendencia de los modelos más fácilmente.

8.1. LSA

En el modelo LSA se pueden ajustar cuatro parámetros.

- Minimum Document Frequency: El mínimo número de veces que una palabra debe aparecer en el corpus de documentos para ser usado en el cálculo de TF-IDF.
- Maximum Document Frequency: Igual que el parámetro anterior, pero para el máximo número de veces que debe aparecer una palabra.
- Número de features: Máximo número de features (palabras) en la matriz de TF-IDF. Las palabras menos relevantes se descartan si superan este número.
- Número de componentes: Número de dimensiones al que reducir los componentes usando LSA

Los parámetros elegidos han sido maximum document frequency 200, minimum document frequency 5, número de componentes 1000 y número de features sin límite.

Figura 8.1: Resultados al modificar ‘Minimum Document Frequency’ (Apocalypse Now)

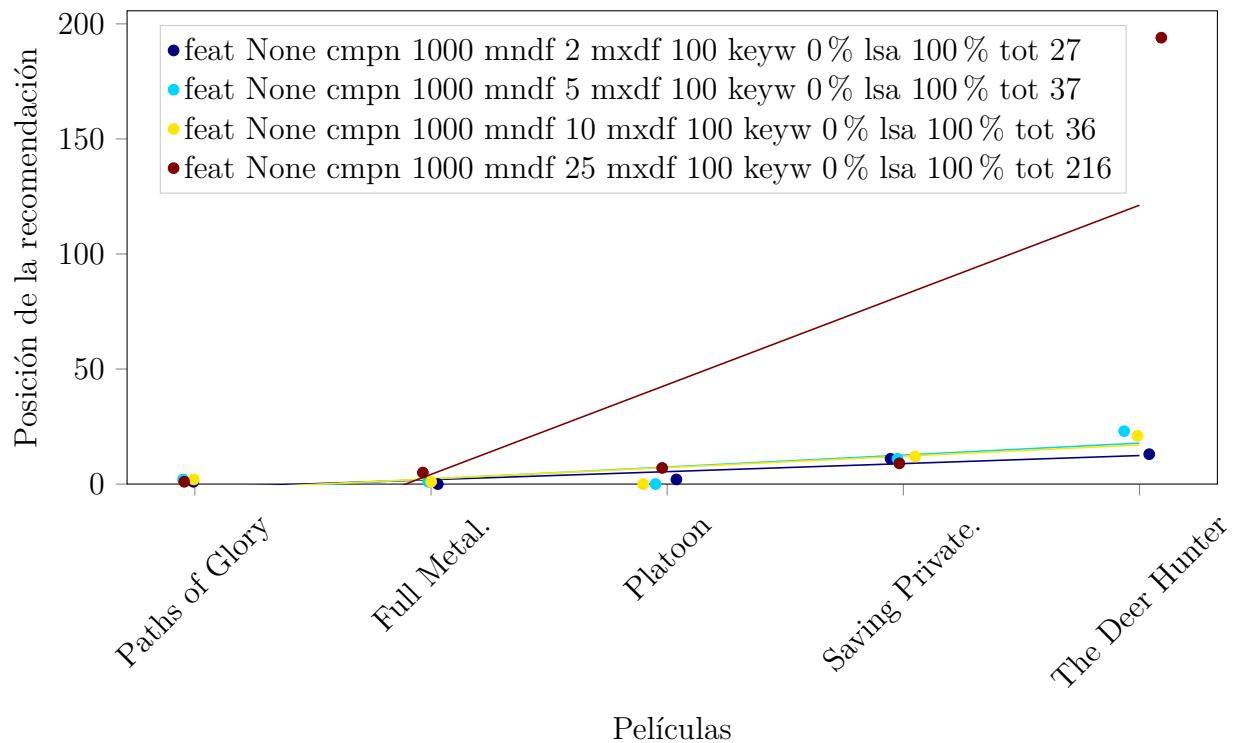


Figura 8.2: Resultados al modificar ‘Minimum Document Frequency’ (2001: A Space Odyssey)

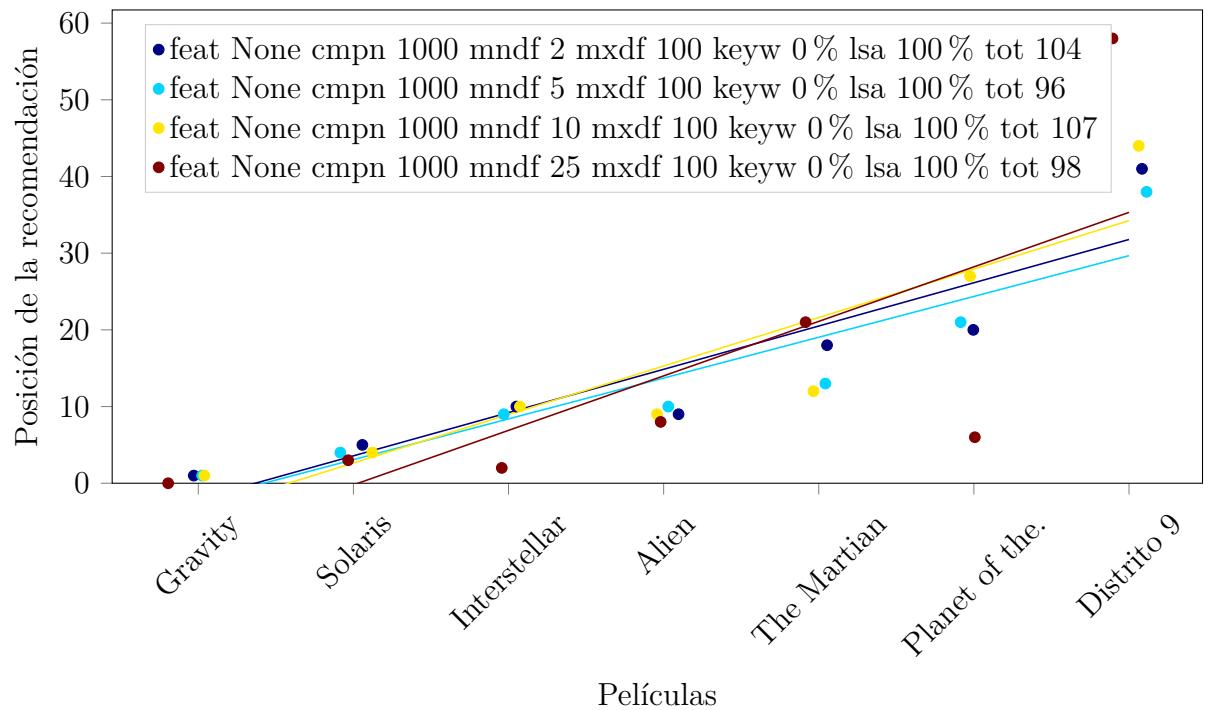


Figura 8.3: Resultados al modificar ‘Minimum Document Frequency’ (Ratatouille)

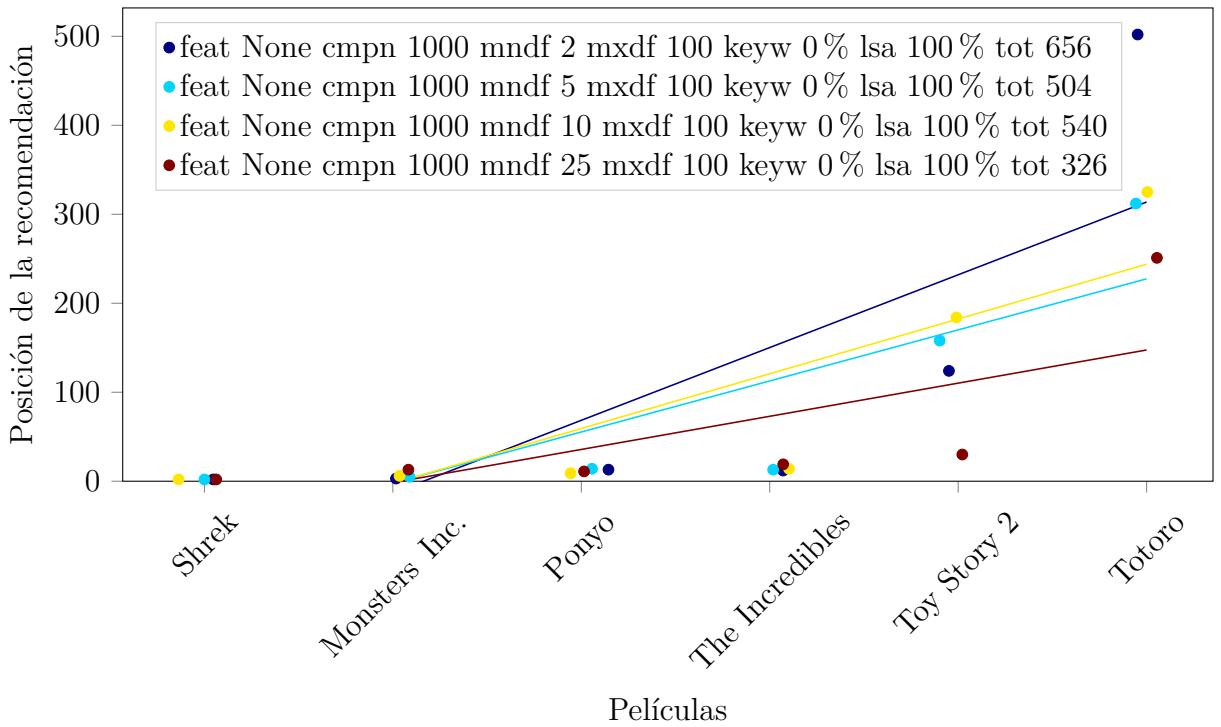


Figura 8.4: Resultados al modificar ‘Maximum Document Frequency’ (Apocalypse Now)

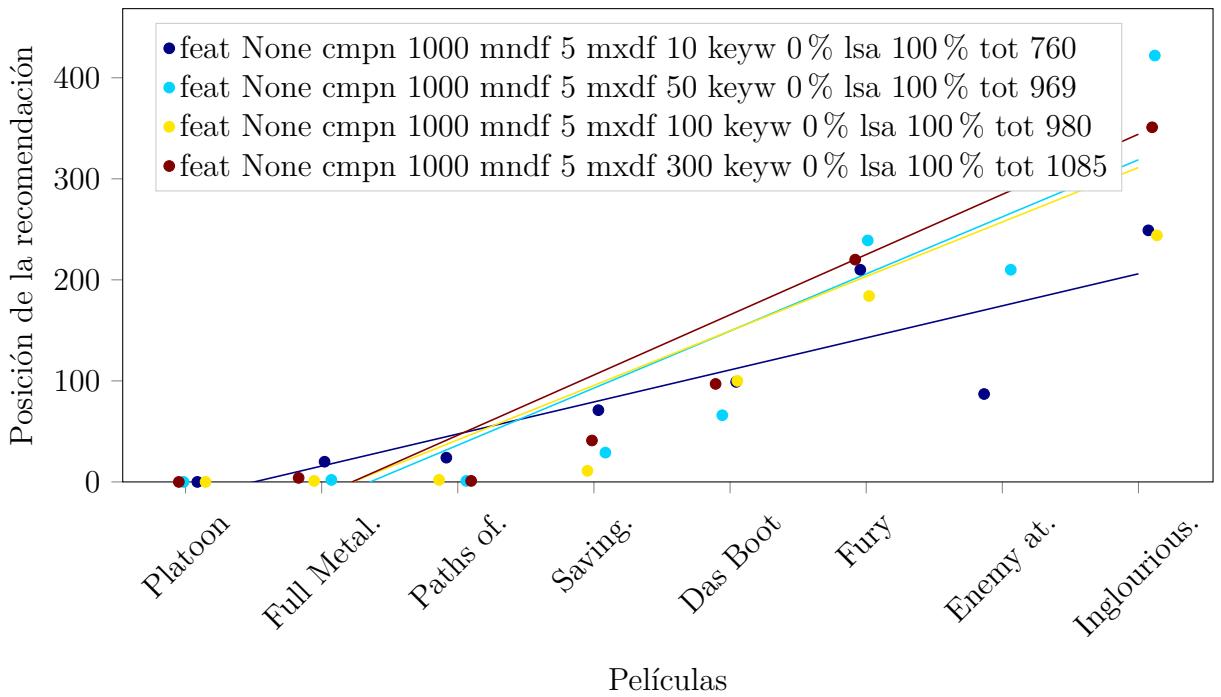


Figura 8.5: Resultados al modificar ‘Maximum Document Frequency’ (2001: A Space Odyssey)

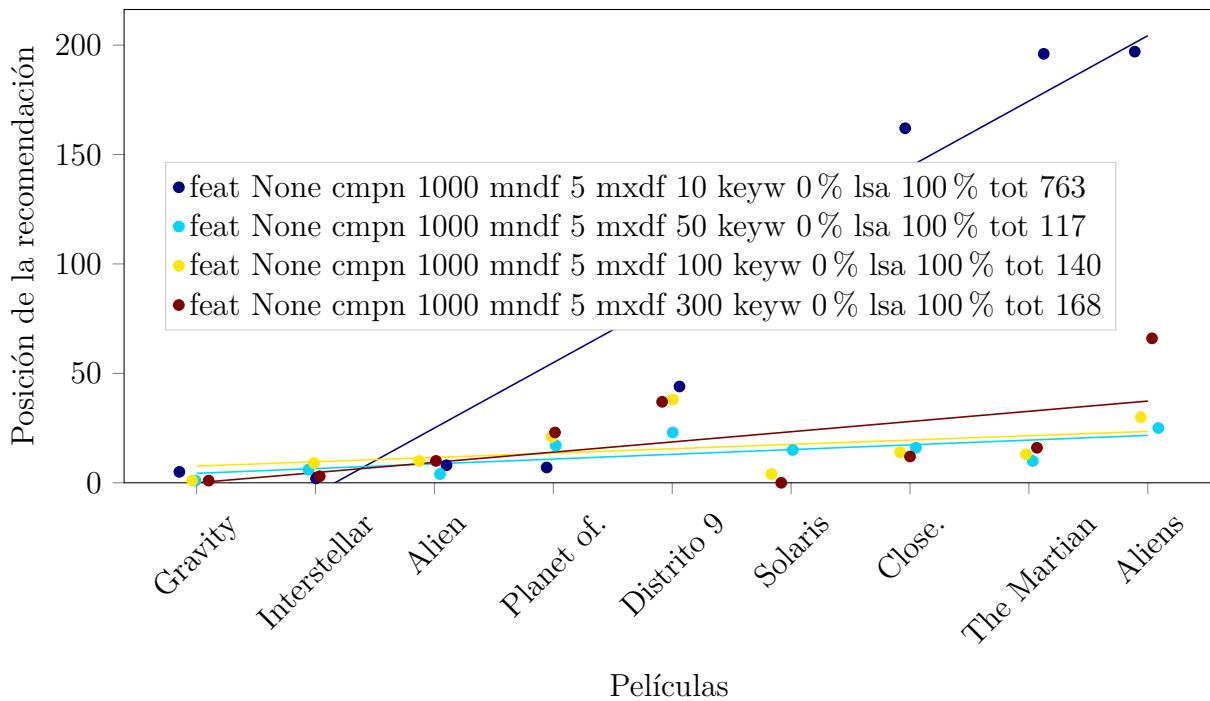


Figura 8.6: Resultados al modificar ‘Maximum Document Frequency’ (Ratatouille)

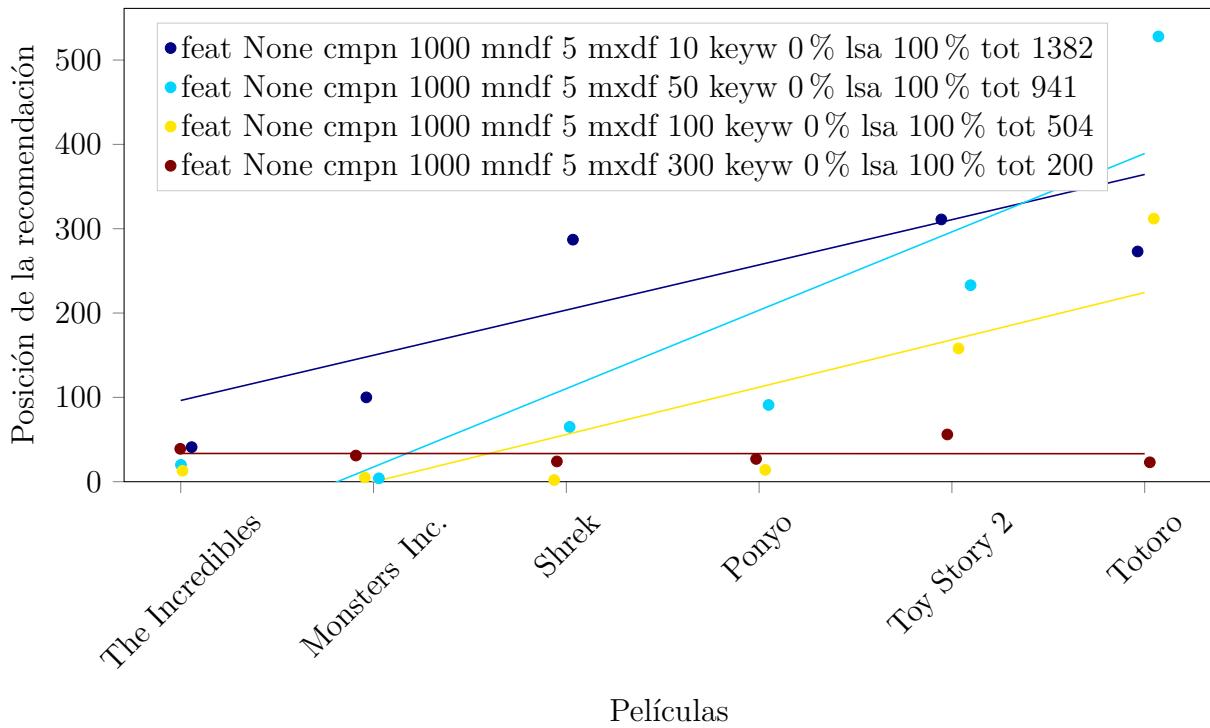


Figura 8.7: Resultados al modificar ‘Number of Components’ (Apocalypse Now)

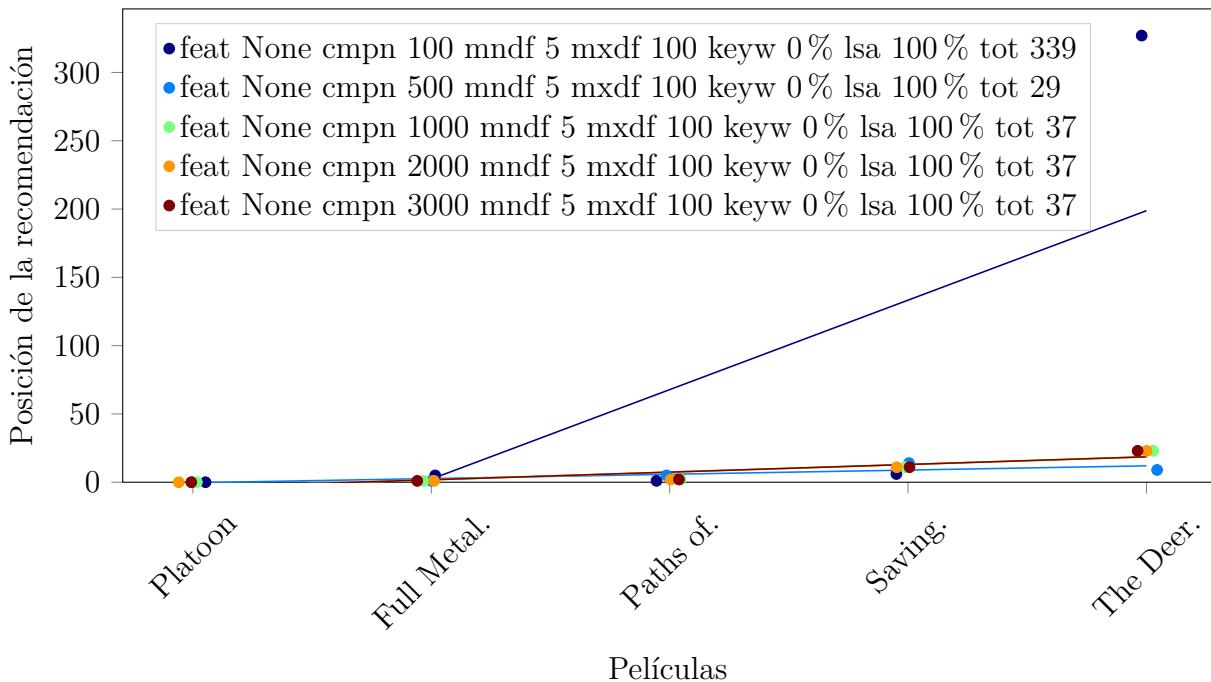


Figura 8.8: Resultados al modificar ‘Number of Components’ (2001: A Space Odyssey)

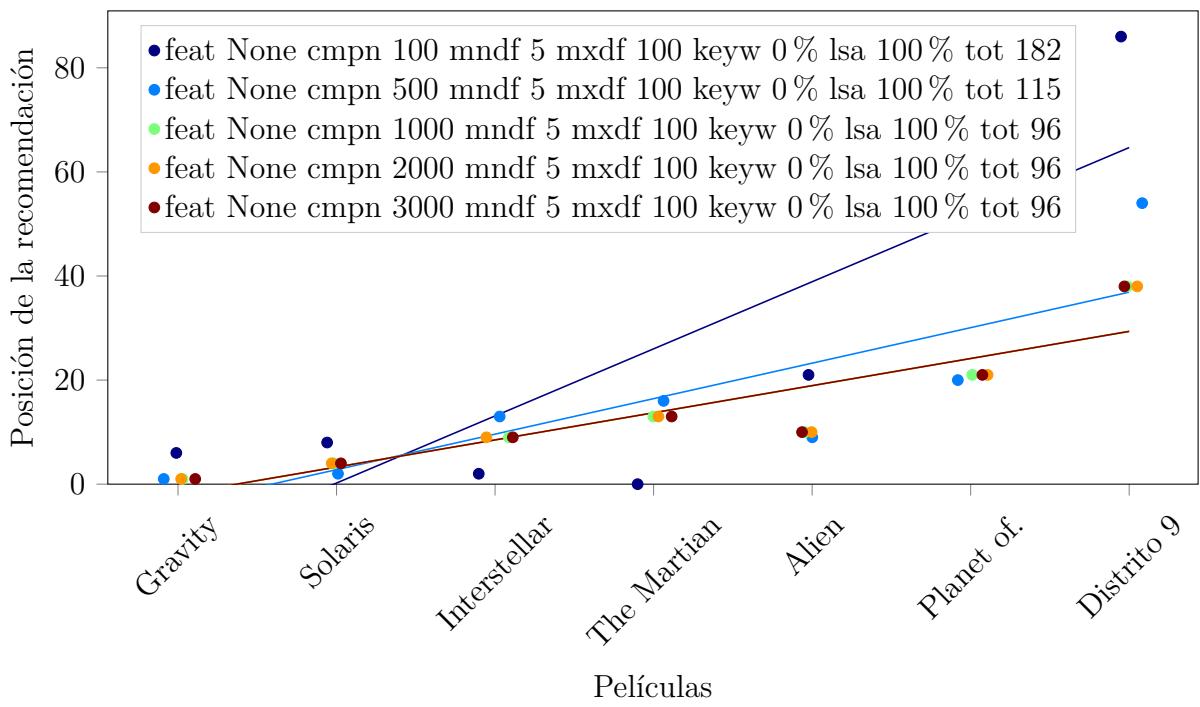


Figura 8.9: Resultados al modificar ‘Number of Components’ (Ratatouille)

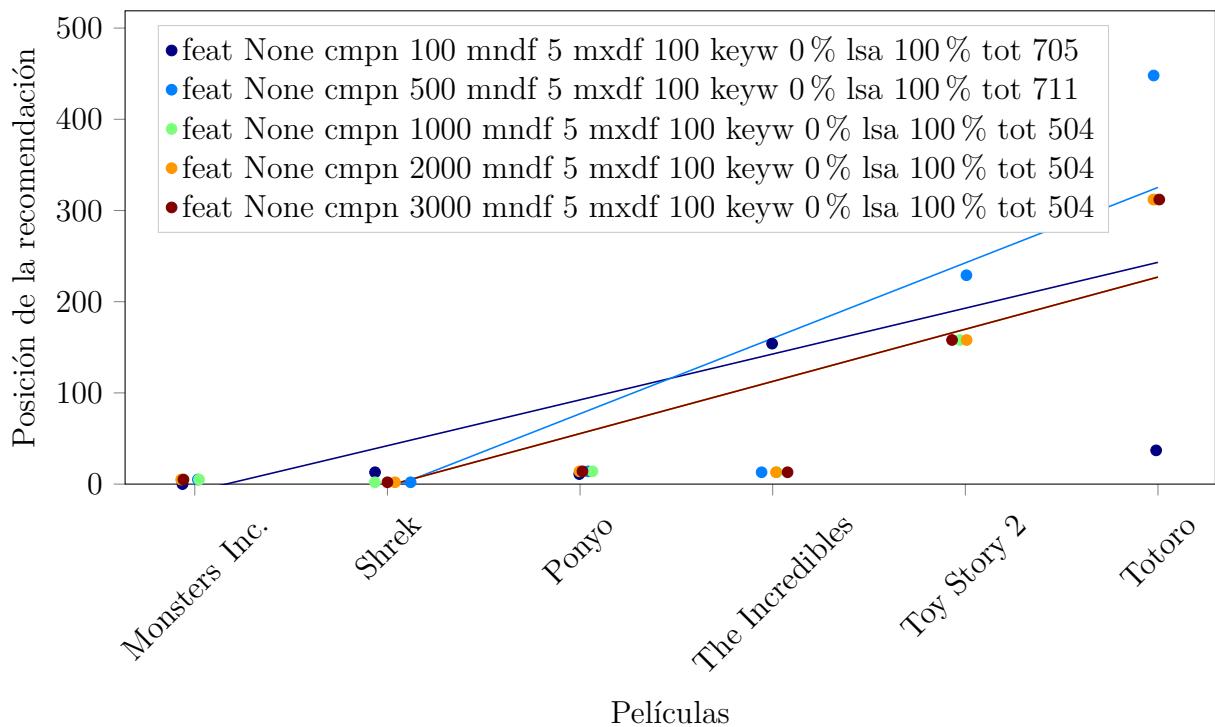


Figura 8.10: Resultados al modificar ‘Number of Features’ (Apocalypse Now)

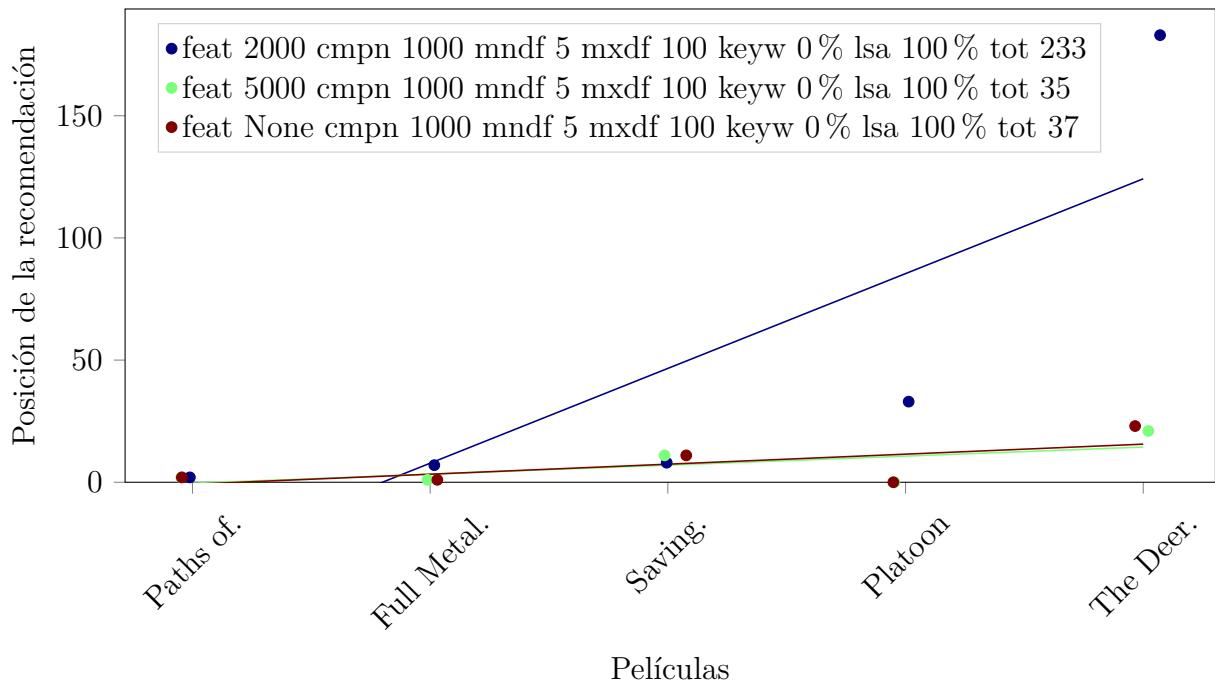


Figura 8.11: Resultados al modificar ‘Number of Features’ (2001: A Space Odyssey)

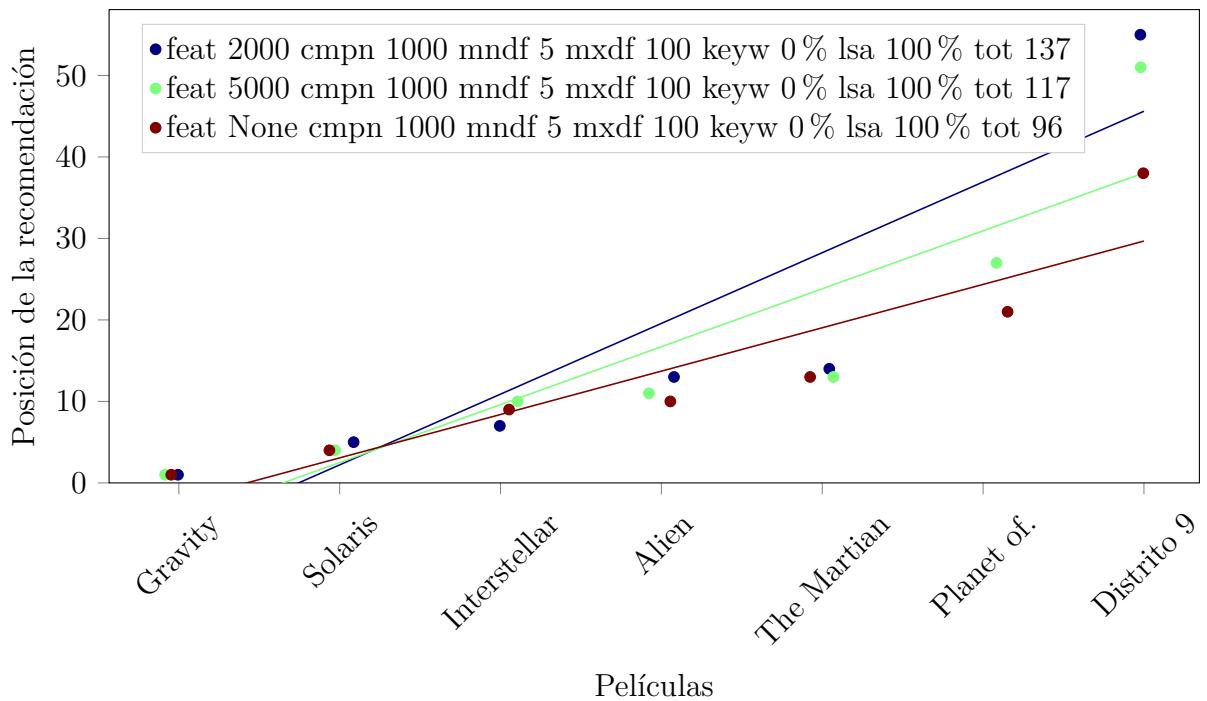
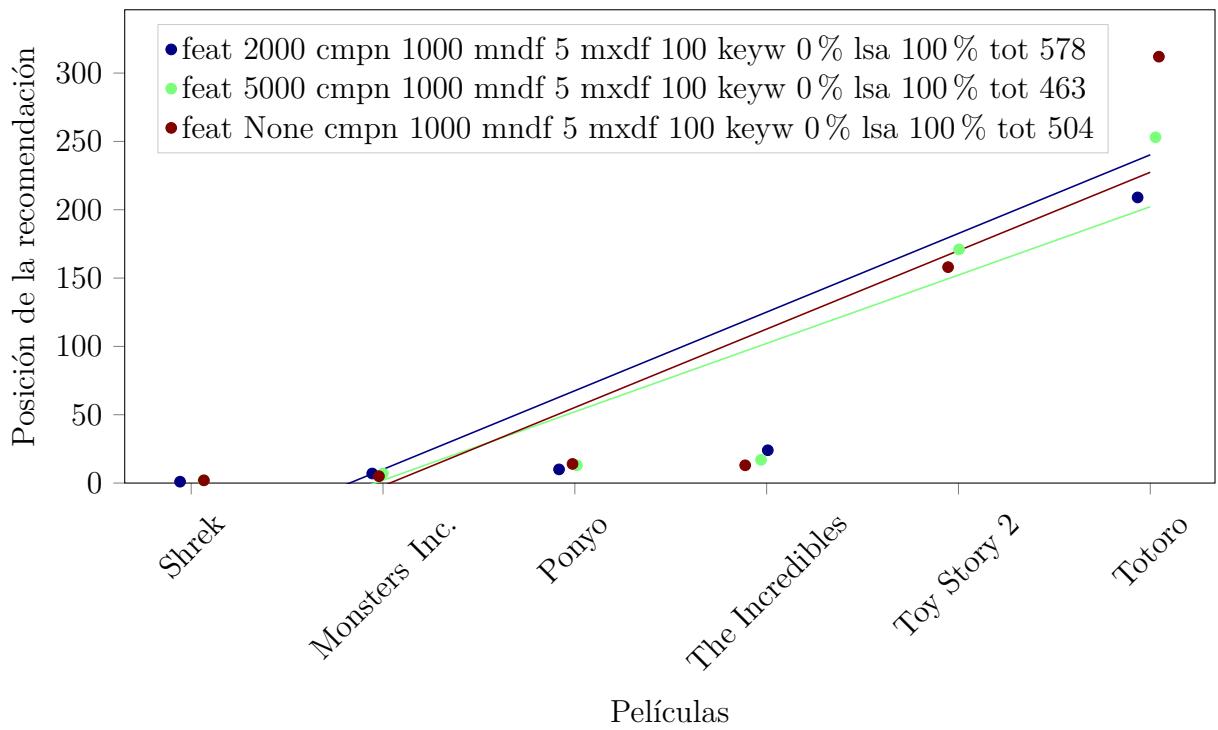


Figura 8.12: Resultados al modificar ‘Number of Features’ (Ratatouille)



8.2. Doc2Vec

Al igual que en el modelo LSA, en Doc2Vec se pueden ajustar cuatro parámetros.

- Size: Número de neuronas (o features) que aprender.
- Window: Cuantas palabras usar como ventana para relacionar palabras en una frase.
Ejemplo en la figura 6.1.
- Minimum Word Count: Cuantas veces debe aparecer una palabra en nuestro corpus para que pueda ser usada como palabra válida.
- Iterations: El número de veces que se entrenará la red neuronal.

Los parámetros elegidos han sido size 1000, window 5, minimum count 5 y número de iteraciones 20.

Figura 8.13: Resultados al modificar ‘Size’ (Apocalypse Now)

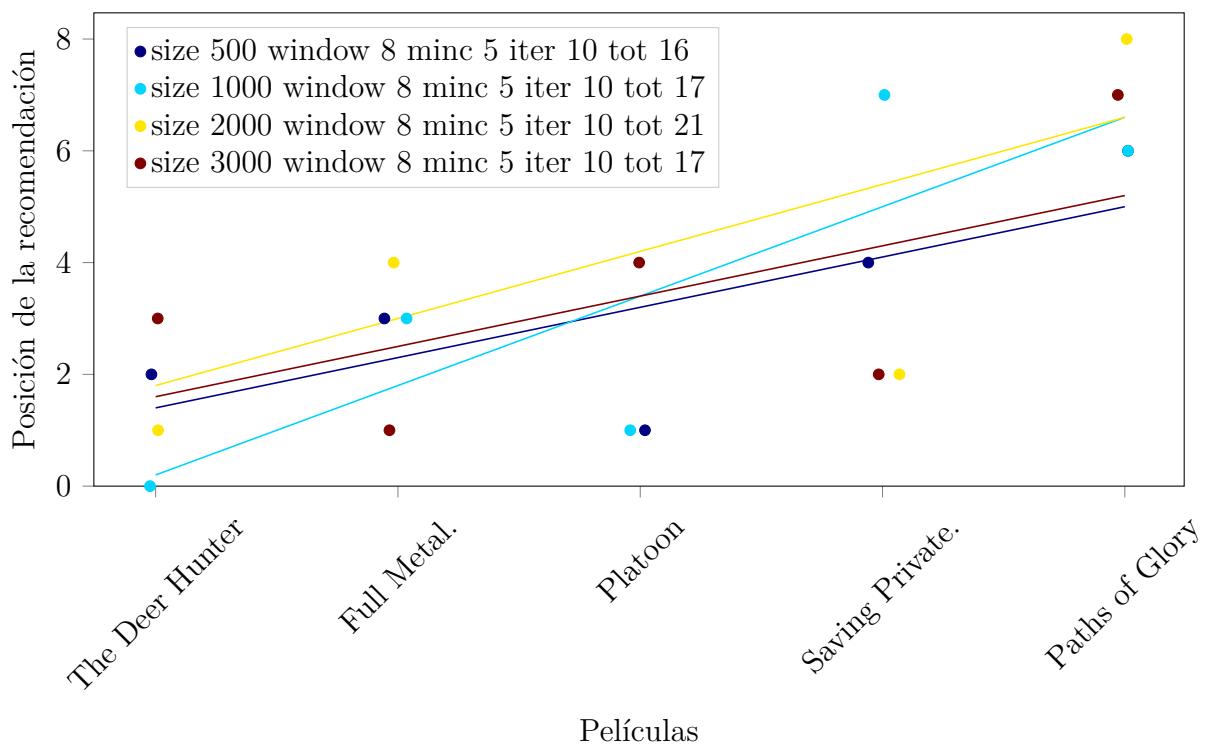


Figura 8.14: Resultados al modificar ‘Size’ (2001: A Space Odyssey)

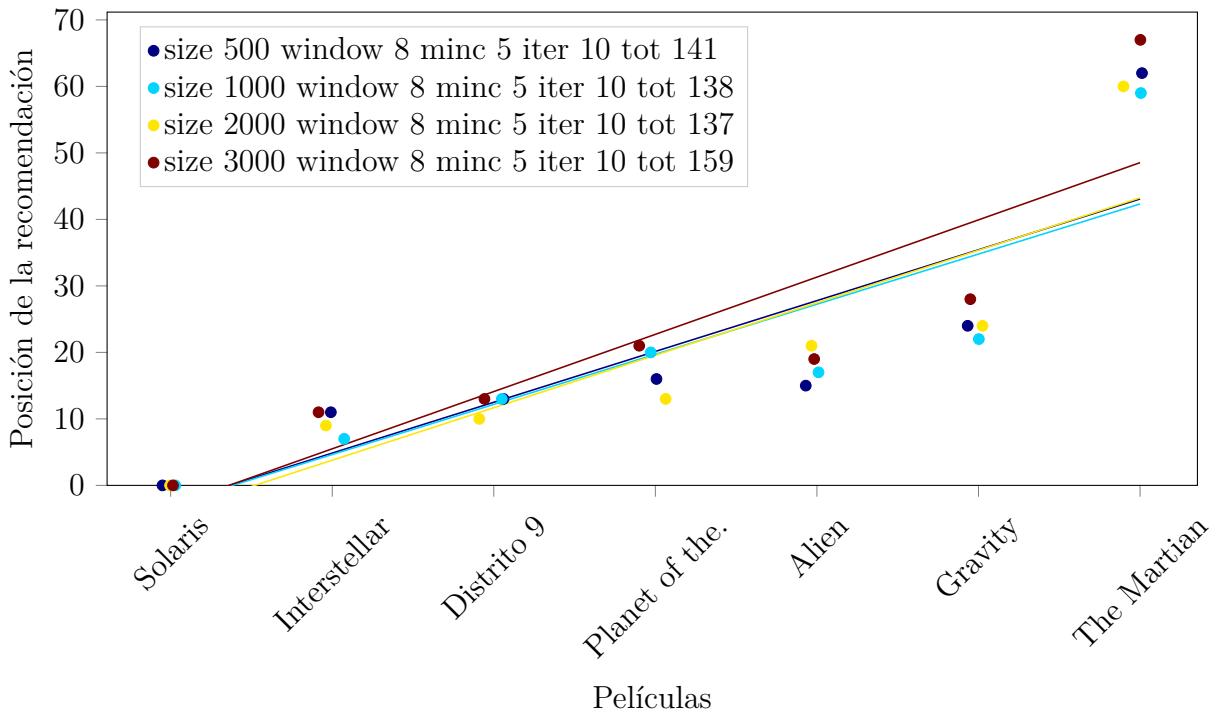


Figura 8.15: Resultados al modificar ‘Size’ (Ratatouille)

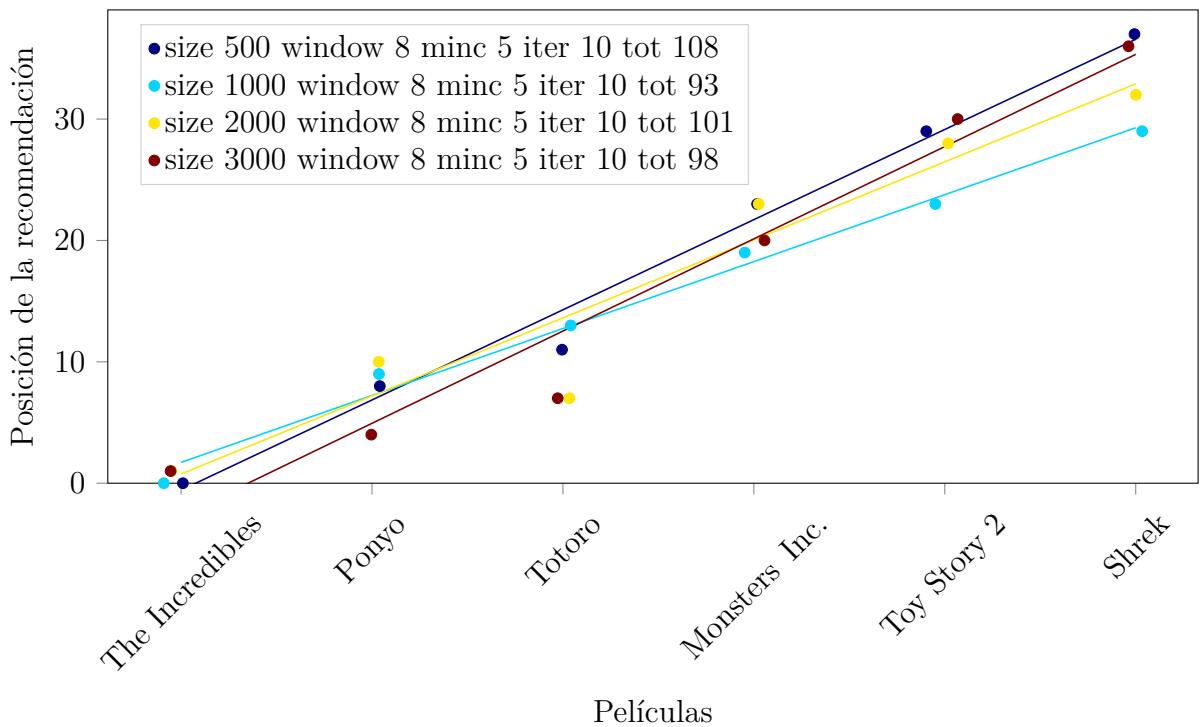


Figura 8.16: Resultados al modificar ‘Window’ (Apocalypse Now)

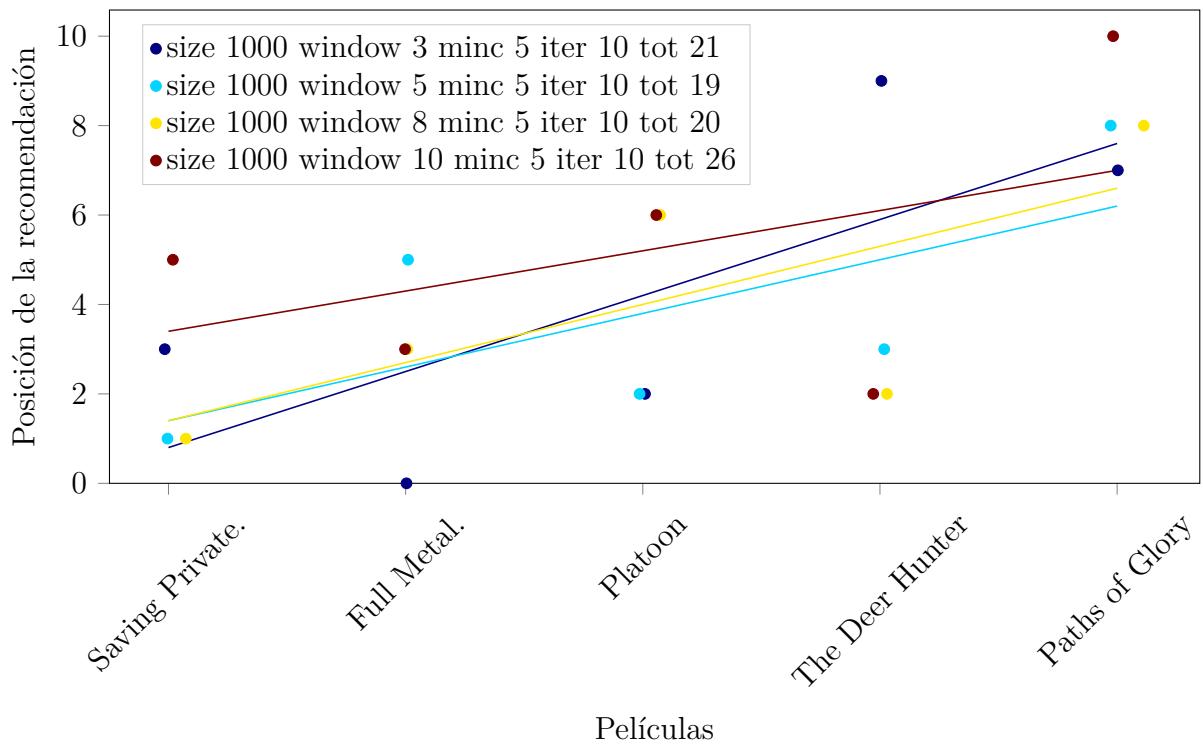


Figura 8.17: Resultados al modificar ‘Window’ (2001: A Space Odyssey)

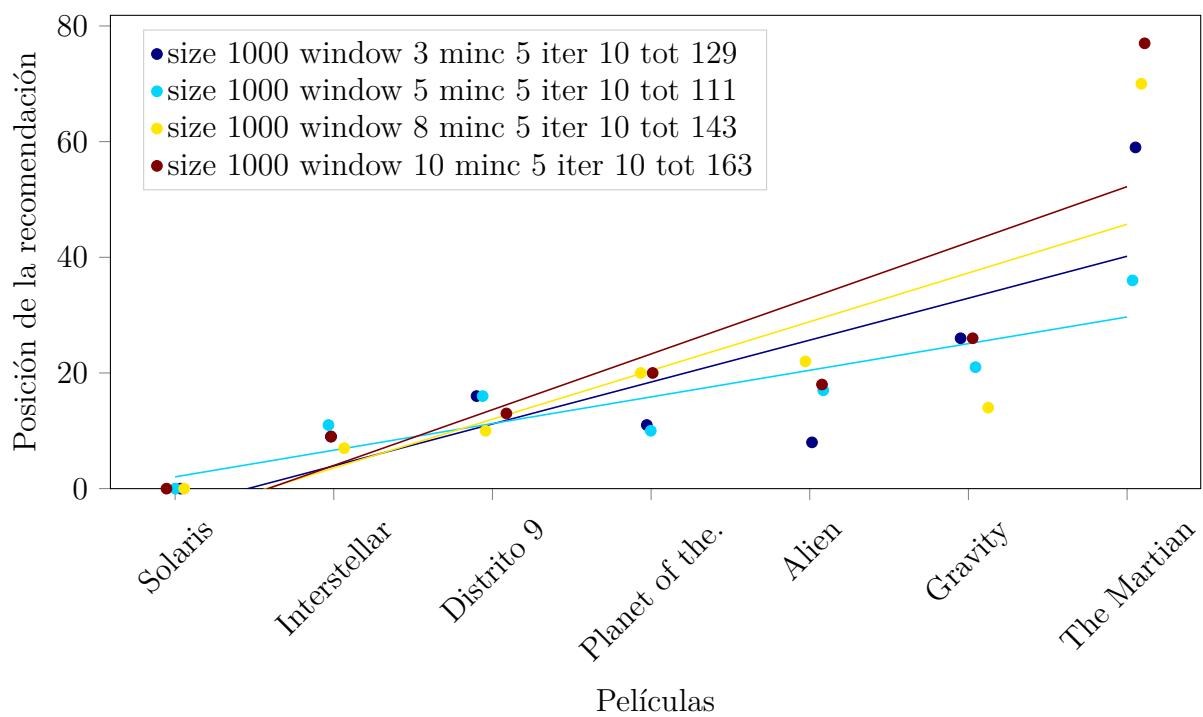


Figura 8.18: Resultados al modificar ‘Window’ (Ratatouille)

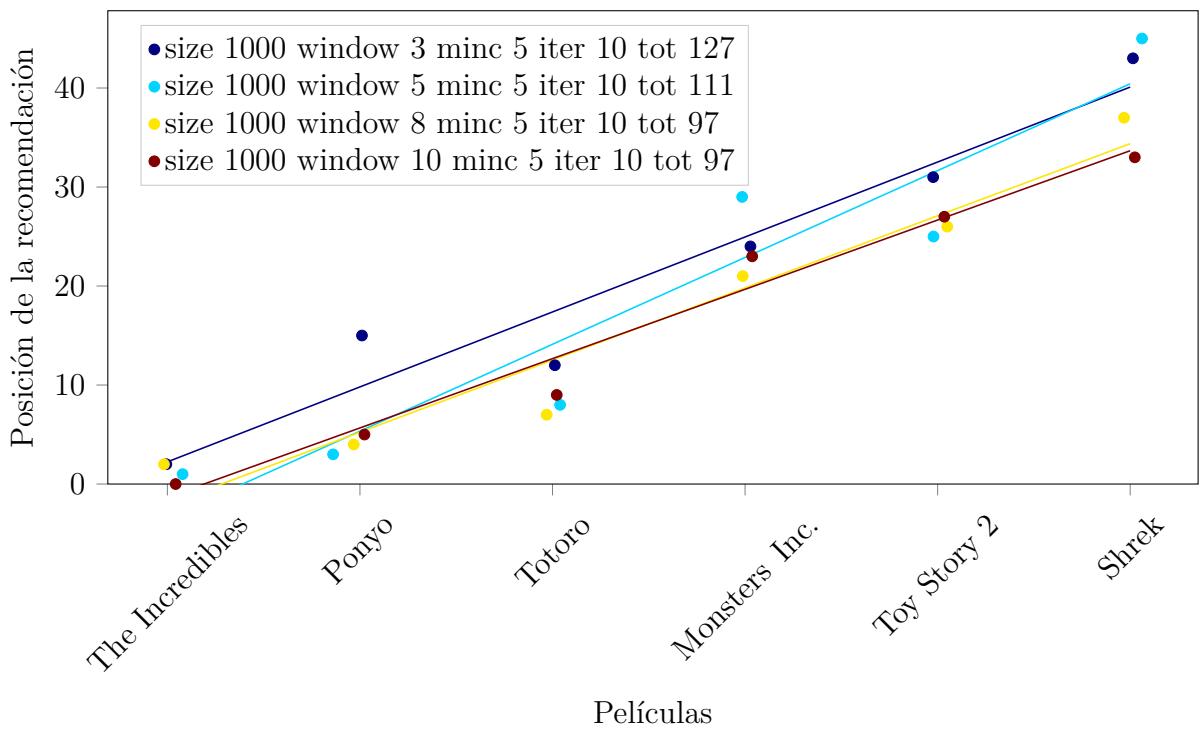


Figura 8.19: Resultados al modificar ‘Minimum count’ (Apocalypse Now)

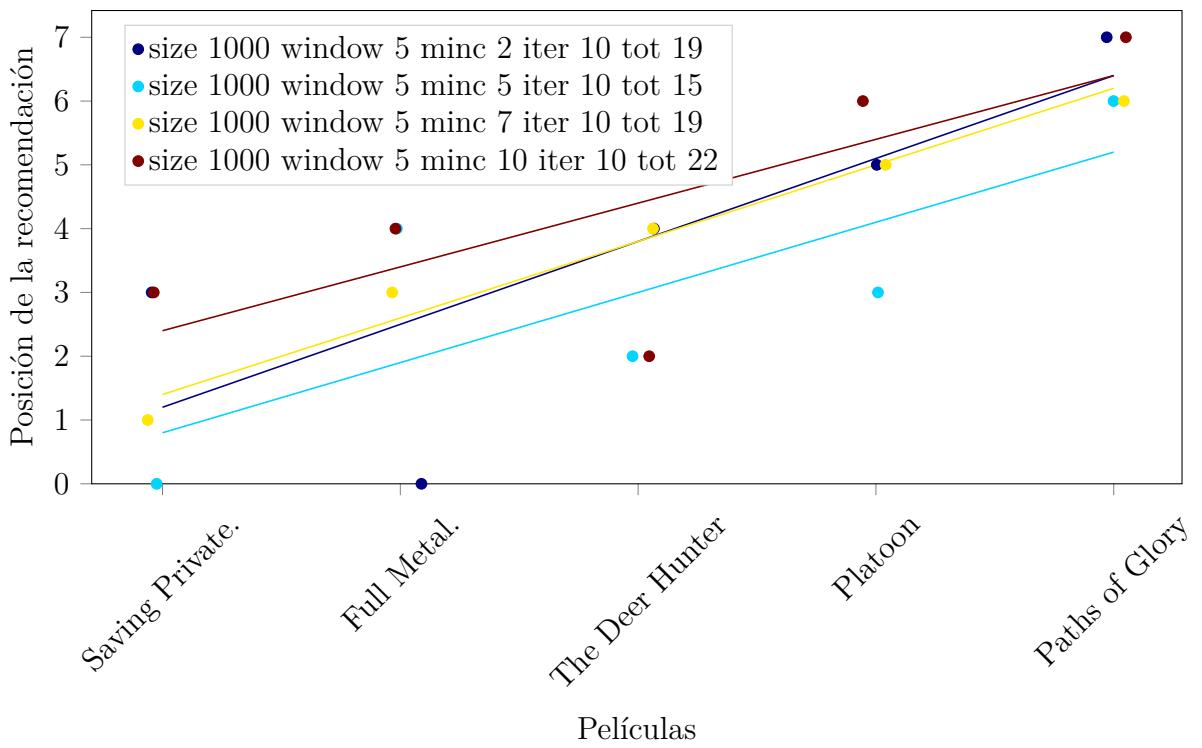


Figura 8.20: Resultados al modificar ‘Minimum count’ (2001: A Space Odyssey)

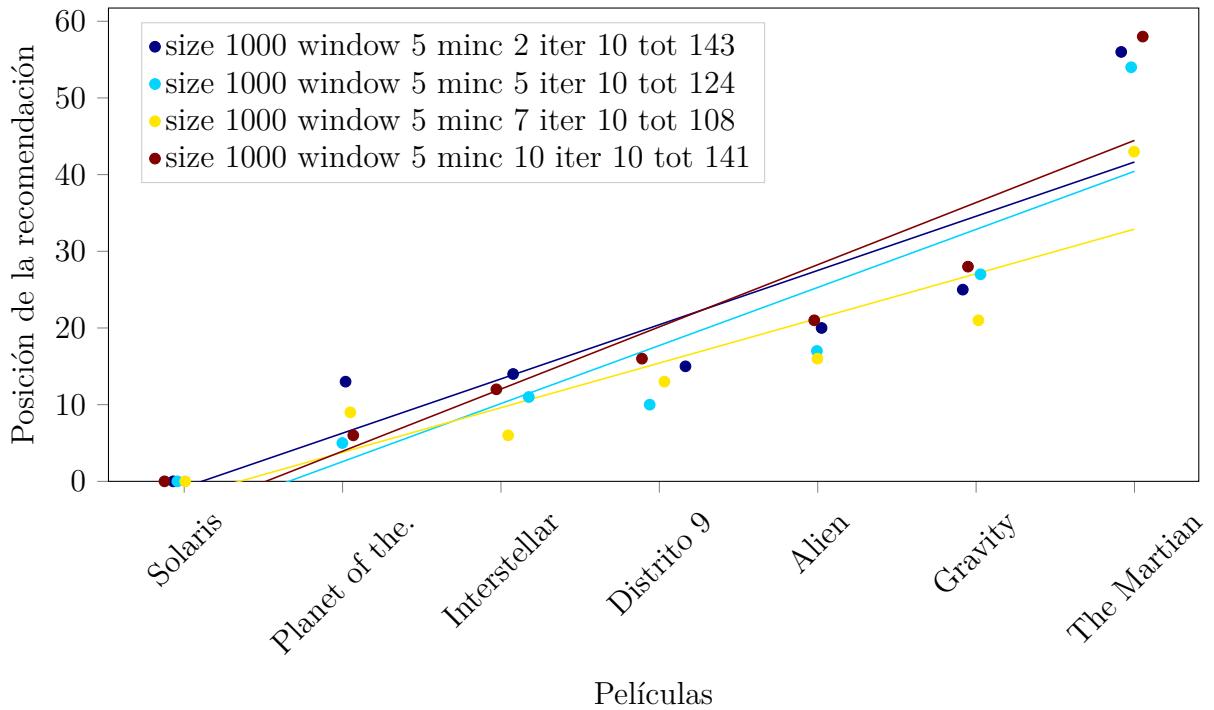


Figura 8.21: Resultados al modificar ‘Minimum count’ (Ratatouille)

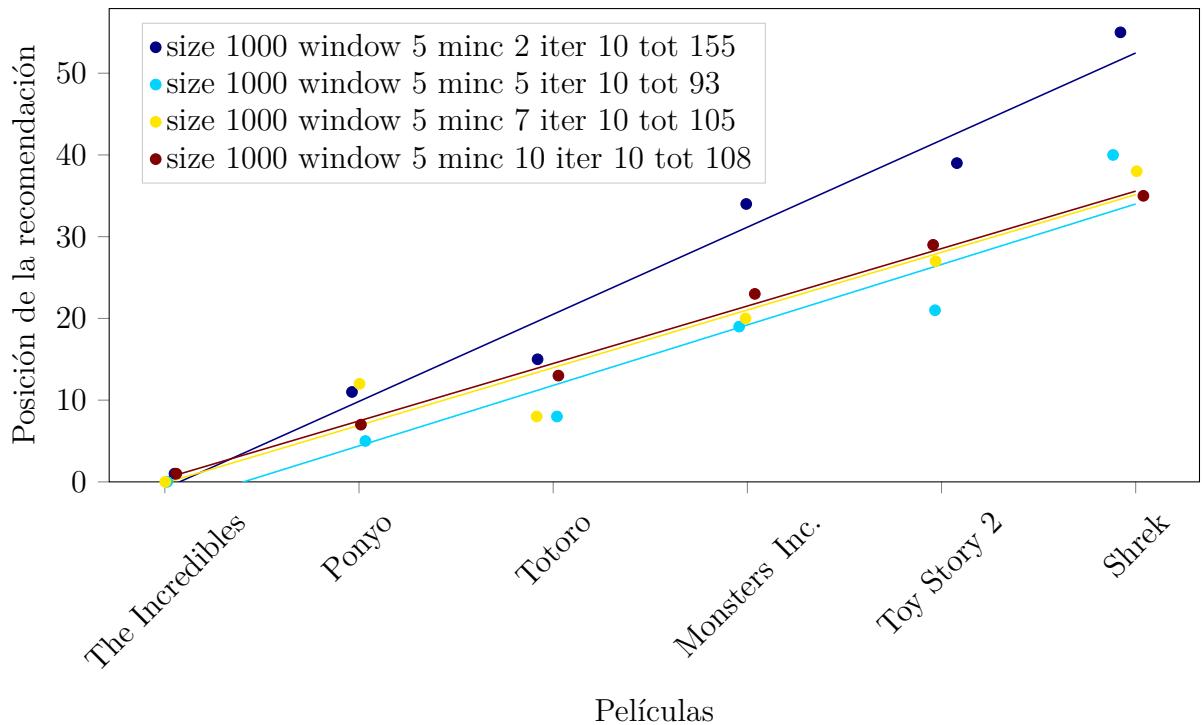


Figura 8.22: Resultados al modificar ‘Iterations’ (Apocalypse Now)

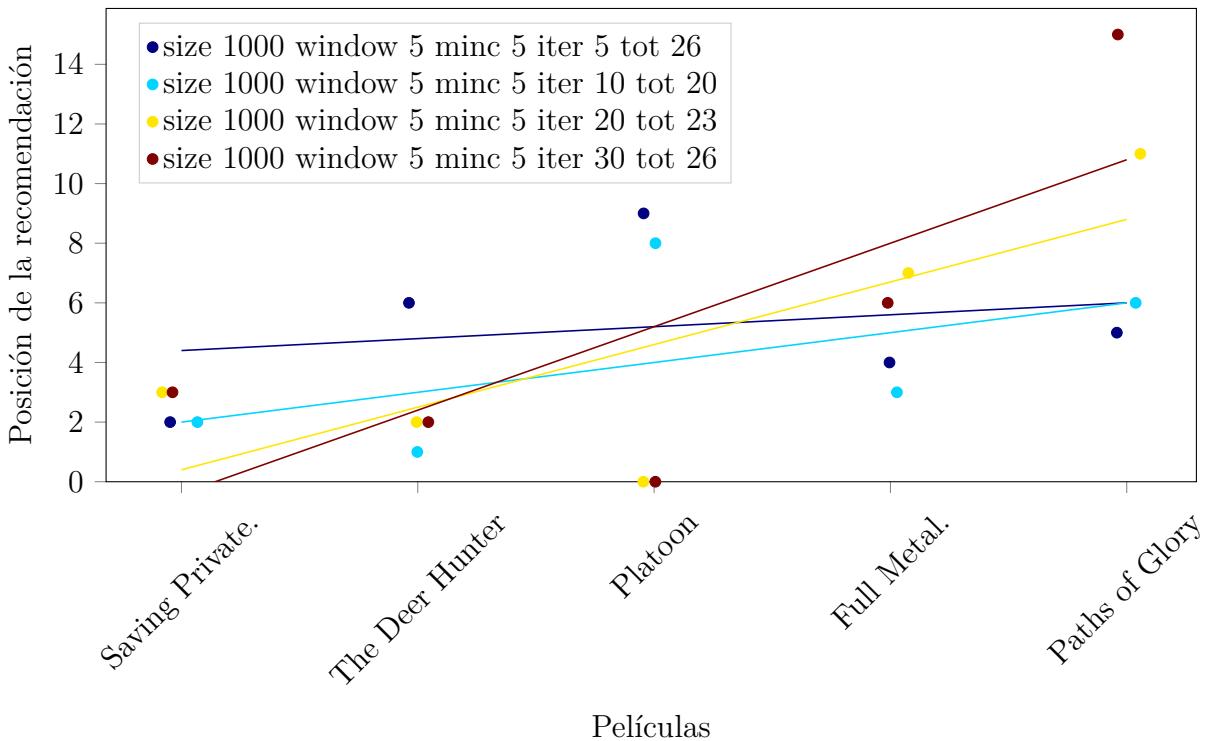


Figura 8.23: Resultados al modificar ‘Iterations’ (2001: A Space Odyssey)

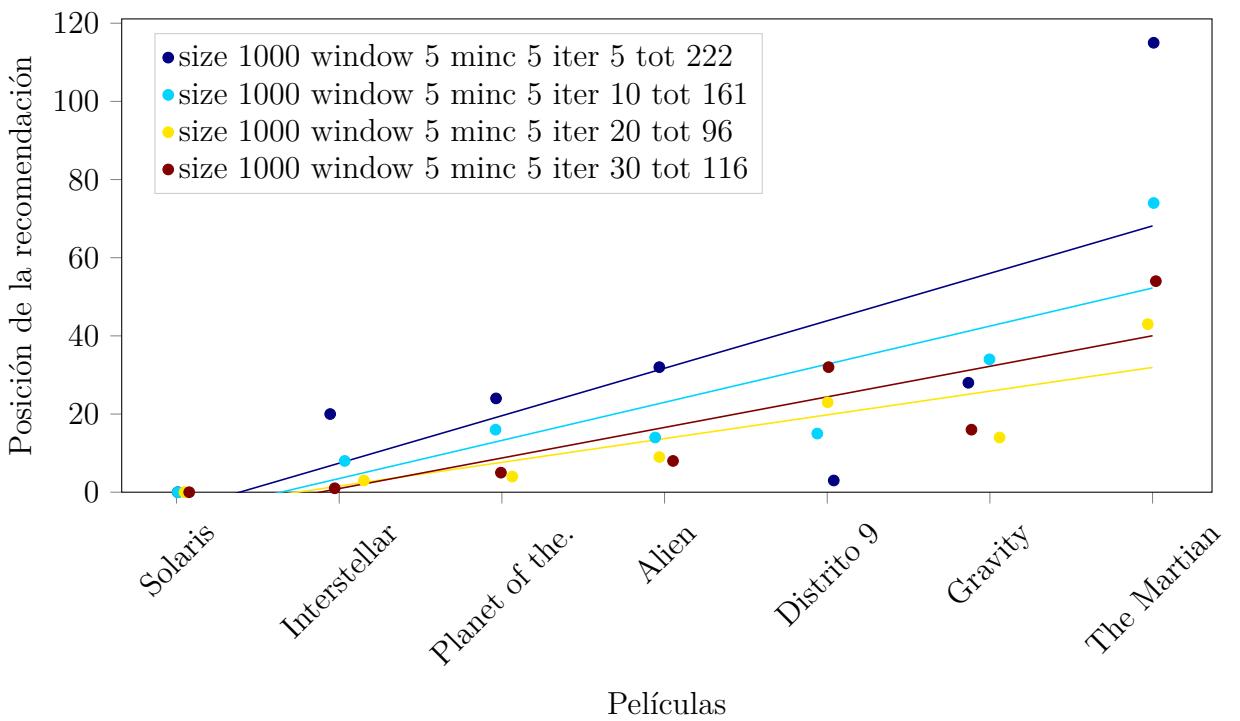
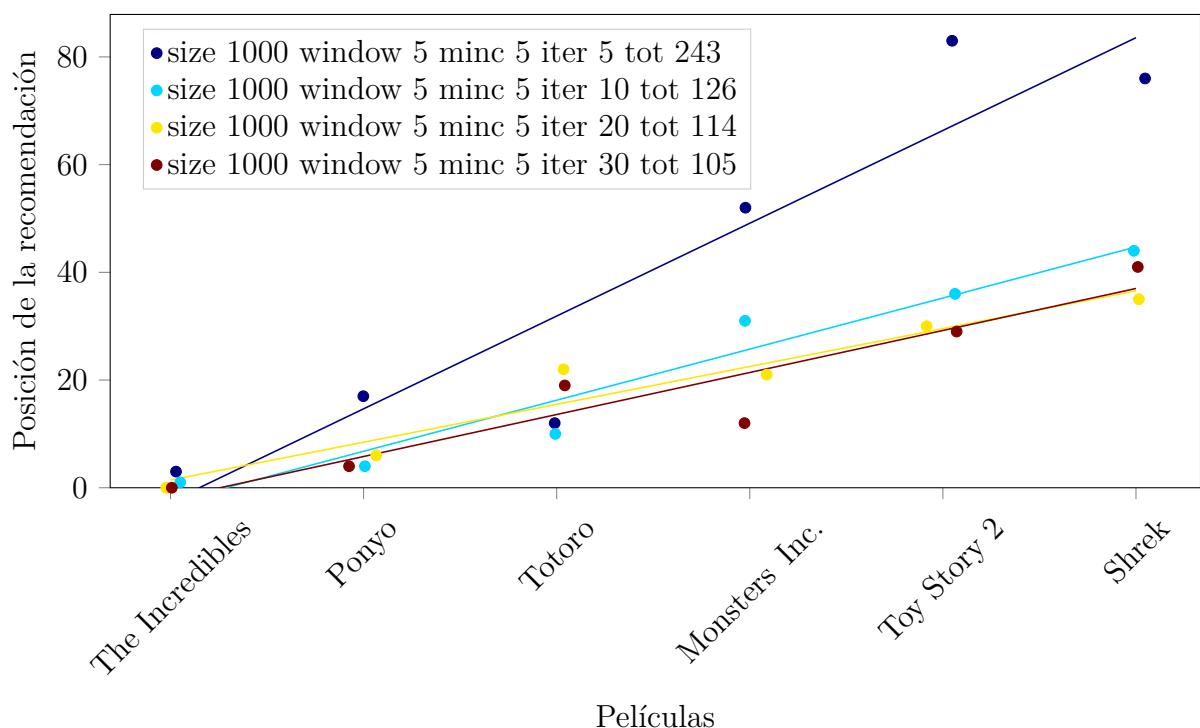


Figura 8.24: Resultados al modificar ‘Iterations’ (Ratatouille)



Capítulo 9

Resultados

Una vez se ha decidido que parámetros usar para los modelos se pueden extraer unos resultados. En la siguientes tablas se van a comparar las recomendaciones entre los dos modelos implementados, las recomendaciones de Movistar+ y las de IMDb.

Hay que tener en cuenta que las recomendaciones de un servicio como Movistar+ se ciñen a contenidos que tienen disponibles en la plataforma. También conviene recordar que el modelo implementado no está entrenado con todo el catalogo de IMDb, sino con unas 800–1000 películas con una alta popularidad.

Cuadro 9.1: Recomendaciones para *Star Wars: Episode VII – The Force Awakens*

LSA	Doc2Vec	Movistar+	IMDb
Star Wars IV	Rogue One	Star Wars I	Star Wars VI
Star Trek	Star Wars IV	Star Wars IV	Star Wars IV
Star Wars V	Star Wars V	Star Wars V	Star Wars V
Star Wars VI	Star Wars VI	Viaje al centro de la Tierra	Star Wars III
Rogue One	Star Trek	Star Wars VI	Rogue One
Star Trek Into Darkness	Star Trek Into Darkness	Star Wars III	Star Wars II
Rise of the Planet of the Apes	Guardians of the Galaxy		Star Wars I
Avatar	Serenity		Guardians of the Galaxy
Wonder Woman	Avatar		Avengers: Age of Ultron
Star Trek II The Wrath of Khan	The Hobbit: Desolation of Smaug		Deadpool

Cuadro 9.2: Recomendaciones para *Match Point*

LSA	Doc2Vec	Movistar+	IMDb
Strangers on a Train	Manhattan	The Deep Blue Sea	Vicky Cristina Barcelona
Hoosiers	Midnight in Paris	Lunas de hiel	Scoop
Dial M for Murder	Hannah and Her Sisters	El ilusionista	Blue Jasmine
The Royal Tenenbaums	The Grand Budapest Hotel	La mejor oferta	Midnight in Paris
Cast Away	Stranger Than Fiction	Tiempo de tormenta	To Rome with Love
Hannah and Her Sisters	La La Land	Lantana	Whatever Works
25th Hour	500 Days of Summer		Manhattan
Kuch Kuch Hota Hai	Tell No One		Deconstructing Harry
The Best Years of Our Lives	In Bruges		The Curse of the Jade Scorpion
Manhattan	Toni Erdmann		Melinda and Melinda

Cuadro 9.3: Recomendaciones para *Toy Story 2*

LSA	Doc2Vec	Movistar+	IMDb
Toy Story	Toy Story	Finding Nemo	Toy Story 3
Toy Story 3	Toy Story 3	Monsters, Inc.	Toy Story
The Lego Movie	Finding Nemo	Toy Story 3	The Incredibles
Monsters, Inc.	Monsters, Inc.	Las Tortugas Ninja	Ratatouille
Shrek	Shrek	Bichos	Shrek
Finding Nemo	Up	Toy Story	Monsters, Inc.
Up	The Incredibles		Shrek 2
Wreck-It Ralph	Despicable Me		Finding Nemo
Inside Out	The Lego Movie		Ice Age
How to Train Your Dragon 2	Fantastic Mr. Fox		Kung Fu Panda

Cuadro 9.4: Recomendaciones para *Ponyo*

LSA	Doc2Vec	Movistar+	IMDb
Castle in the Sky	The Secret World of Arrietty	La sirenita 2	Kiki's Delivery Service
Porco Rosso	My Neighbor Totoro	Toy Story 3	The Secret World of Arrietty
Howl's Moving Castle	Castle in the Sky	Finding Nemo	Castle in the Sky
Spirited Away	Spirited Away	The Secret World of Arrietty	My Neighbor Totoro
My Neighbor Totoro	Howl's Moving Castle	El niño y la bestia	Howl's Moving Castle
Princess Mononoke	Kubo and the Two Strings	Regal academy	Spirited Away
Finding Nemo	Princess Mononoke		From Up on Poppy Hill
Kubo and the Two Strings	Susurros del corazón		Susurros del corazón
Moana	How to Train Your Dragon		Haru en el reino de los gatos
Shrek	Moana		Pom Poko

Cuadro 9.5: Recomendaciones para *Los Otros*

LSA	Doc2Vec	Movistar+	IMDb
The Wicker Man	The Haunting	Fear of the Walking Dead	1408
Halloween	Repulsion	El orfanato	Dark Water
The Cabinet of Dr. Caligari	What Ever Happened to Baby Jane?	La bruja	The Ring
Shutter Island	The Exorcist	Cube	The Amityville Horror
Get Out	Halloween	El ilusionista	Dream House
Psycho	Dogville	La mujer de negro	The Awakening
Saw	Rosemary's Baby		Silent Hill
What Ever Happened to Baby Jane?	Coraline		Sinister
The Haunting	The Wicker Man		Whispers
Dark City	Bailar en la oscuridad		Insidious

Cuadro 9.6: Recomendaciones para *Deadpool*

LSA	Doc2Vec	Movistar+	IMDb
Iron Man	Kick-Ass	The Flash	Guardians of the Galaxy
Kick-Ass	Kingsman: The Secret Service	Las Tortugas Ninja	Avengers: Age of Ultron
The Incredibles	Guardians of the Galaxy	Powers	Captain America: Civil War
Captain America: Civil War	Big Hero 6	Batman v Superman	The Avengers
Watchmen	Logan	Guardians of the Galaxy	Doctor Strange
Logan	Batman Begins	Marvel Los Vengadores Unidos	Iron Man
Batman	Guardians of the Galaxy Vol. 2		Captain America: The Winter Soldier
X-Men: First Class	John Wick: Chapter 2		Ant-Man
The Dark Knight	Kung Fu Hustle		Rogue One
Wonder Woman	Lock, Stock and Two Smoking Barrels		Batman v Superman: Dawn of Justice

Capítulo 10

Interfaz

Una vez terminado la optimización de los modelos se pasó a programar tanto la interfaz desde donde el usuario interactúa con la herramienta (al que he llamado Movie Pepper), como el backend desde donde consultar los modelos creados.

El backend está escrito en Python y el front en Javascript (ES6). Estos dos lenguajes usan un tipado dinámico, pero usando mypy y Flow se pueden tipar estática y opcionalmente. Esto significa que contamos con más seguridad ante errores de programación.

El frontend está construido como SPA usando VueJS 2 y usando Nuxt para tener SSR. Cuando el usuario entre en la web se envía la página ya renderizada por el servidor, como en una web tradicional, mientras que en paralelo se descarga el resto de la aplicación Javascript, con la que seguirá con la navegación.

La web ofrece una funcionalidad muy sencilla. Cada página tiene 100 películas en orden alfabético, pudiendo cambiar de páginas usando los botones al final de la página. Cada película se presenta en una ficha donde se muestra su título, géneros, año, puntuación y carátula. Cada ficha tiene tres botones. El primero abre la película en IMDb. El segundo pide al recomendador las recomendaciones de la película usando LSA. El tercer botón muestra las recomendaciones usando Doc2Vec.

Las recomendaciones se presentan encima de la lista de películas con el mismo formato de fichas, pero solamente con el botón de IMDb.

También se incluye un buscador básico para filtrar películas por título. Las fichas se filtran en tiempo real mientras el usuario escribe en la barra de búsqueda.

En la figura 10.4 se muestra la arquitectura de la aplicación. La API REST solamente tiene tres rutas, una para enumerar las películas disponibles, otra para pedir recomendaciones usando LSA y otra para las recomendaciones usando Doc2Vec.

Figura 10.1: Interfaz de la aplicación

Welcome! HOME ABOUT

Movie Pepper

Movie List

Movie filter

2001: A Space Odyssey Director: Stanley Kubrick Genres: Adventure, Sci-Fi Year: 1968 Rating: 8.3 IMDB ALS DOC2VEC	21 Grams Director: Alejandro González Iñárritu Genres: Crime, Drama, Thriller Year: 2003 Rating: 7.7 IMDB ALS DOC2VEC	25th Hour Director: Spike Lee Genres: Drama Year: 2002 Rating: 7.7 IMDB ALS DOC2VEC
3 Idiots Director: Rajkumar Hirani Genres: Comedy, Drama Year: 2009 Rating: 8.4 IMDB ALS DOC2VEC	3-Iron Director: Ki-duk Kim Genres: Crime, Drama, Romance Year: 2004 Rating: 8.1 IMDB ALS DOC2VEC	300 Director: Zack Snyder Genres: Action, Fantasy Year: 2006 Rating: 7.7 IMDB ALS DOC2VEC
3:10 to Yuma Director: James Mangold Genres: Adventure, Crime, Drama, Western Year: 2007 Rating: 7.7 IMDB ALS DOC2VEC	5 Centimeters Per Second Director: Makoto Shinkai Genres: Animation, Drama, Romance Year: 2007 Rating: 7.8 IMDB ALS DOC2VEC	500 Days of Summer Director: Marc Webb Genres: Comedy, Drama, Romance Year: 2009 Rating: 7.7 IMDB ALS DOC2VEC
Boy A Director: John Crowley Genres: Drama Year: 2007 Rating: 7.7 IMDB ALS DOC2VEC	Boyhood Director: Richard Linklater Genres: Drama Year: 2014 Rating: 7.9 IMDB ALS DOC2VEC	Braveheart Director: Mel Gibson Genres: Biography, Drama, History, War Year: 1995 Rating: 8.4 IMDB ALS DOC2VEC
Brazil Director: Terry Gilliam Genres: Drama, Sci-Fi Year: 1985 Rating: 8.0 IMDB ALS DOC2VEC	Breakfast at Tiffany's Director: Blake Edwards Genres: Comedy, Drama, Romance Year: 1961 Rating: 7.7 IMDB ALS DOC2VEC	

<
1
2
3
4
5
...
8
>

Hugo Fernando Seage : bugofs.com

Figura 10.2: Interfaz al buscar

Welcome! HOME ABOUT

Movie Pepper

Movie List

Movie filter **star wars**

Star Wars: Episode IV - A New Hope Director: George Lucas Genres: Action, Adventure, Fantasy, Sci-Fi Year: 1977 Rating: 8.7 IMDB ALS DOC2VEC	Star Wars: Episode V - The Empire Strikes Back Director: Irvin Kershner Genres: Action, Adventure, Fantasy, Sci-Fi Year: 1980 Rating: 8.8 IMDB ALS DOC2VEC	Star Wars: Episode VI - Return of the Jedi Director: Richard Marquand Genres: Action, Adventure, Fantasy, Sci-Fi Year: 1983 Rating: 8.4 IMDB ALS DOC2VEC
Star Wars: The Force Awakens Director: J.J. Abrams Genres: Action, Adventure, Fantasy, Sci-Fi Year: 2015 Rating: 8.1 IMDB ALS DOC2VEC		

<
1
2
3
4
5
...
8
>

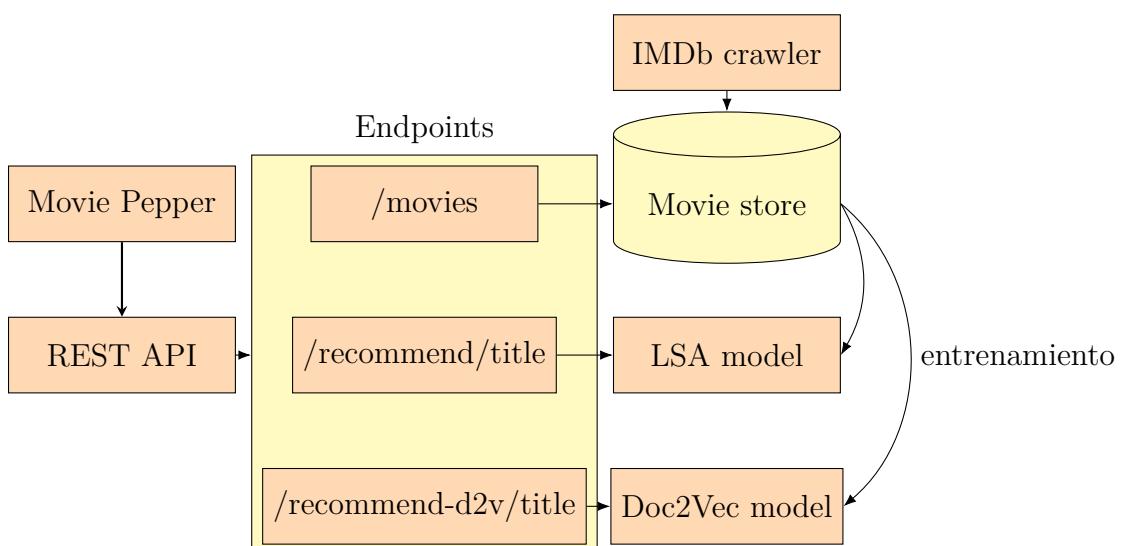
Hugo Fernando Seage : bugofs.com

Figura 10.3: Interfaz al mostrar unos resultados

The screenshot shows two main sections of the Movie Pepper application:

- Comparison Results**: A section titled "Chosen movie" displays "Star Wars: The Force Awakens" with its details (Director: J.J. Abrams, Genres: Action, Adventure, Fantasy, Sci-Fi, Year: 2015, Rating: 8.1) and a small thumbnail image. Below it is a grid of other movies:
 - Rogue One**: Director Gareth Edwards, Genres: Action, Adventure, Sci-Fi, Year: 2016, Rating: 7.9
 - Star Wars: Episode IV - A New Hope**: Director George Lucas, Genres: Action, Adventure, Fantasy, Sci-Fi, Year: 1977, Rating: 8.7
 - Star Wars: Episode V - The Empire Strikes Back**: Director Irvin Kershner, Genres: Action, Adventure, Fantasy, Sci-Fi, Year: 1980, Rating: 8.8
 - Star Wars: Episode VI - Return of the Jedi**: Director Richard Marquand, Genres: Action, Adventure, Fantasy, Sci-Fi, Year: 1983, Rating: 8.4
 - Star Trek**: Director J.J. Abrams, Genres: Action, Adventure, Sci-Fi, Year: 2009, Rating: 8.0
 - Star Trek Into Darkness**: Director J.J. Abrams, Genres: Action, Adventure, Sci-Fi, Year: 2013, Rating: 7.8
 - Guardians of the Galaxy**: Director James Gunn, Genres: Action, Adventure, Sci-Fi, Year: 2014, Rating: 8.1
 - Serenity**: Director Joss Whedon, Genres: Action, Adventure, Sci-Fi, Thriller, Year: 2005, Rating: 7.9
 - Avatar**: Director James Cameron, Genres: Action, Adventure, Fantasy, Sci-Fi, Year: 2009, Rating: 7.8
 - The Hobbit: The Desolation of Smaug**: Director Peter Jackson, Genres: Adventure, Fantasy, Year: 2013, Rating: 7.9
- Movie List**: A section titled "Movie List" with a "Movie filter" input field containing "star wars: the". It shows a single result for "Star Wars: The Force Awakens" with its details and a thumbnail image.

Figura 10.4: Arquitectura del sistema



Capítulo 11

Conclusión

Completar este proyecto me ha dado una nueva perspectiva en el potencial del procesamiento del lenguaje natural. Modelos como los usados aquí demuestran que es posible para una máquina extraer suficiente información de textos para que se puedan usar en varias aplicaciones.

Estoy más que satisfecho de los resultados que se muestran aquí, pudiendo competir fácilmente con recomendadores usados en servicios comerciales o servicios que llevan operando durante varios años en la industria.

Son herramientas útiles, con un impacto real en los usuarios de los servicios ya que mejora la experiencia y usabilidad. Sin duda, el filtrado colaborativo es una técnica indispensable a la hora de desplegar un servicio de este estilo, pero pienso que estas técnicas son un buen complemento a las recomendaciones personalizadas. Se puede usar para recomendar al terminar de ver contenidos, para nuevos usuarios que aún no han usado la plataforma lo suficiente como para poder hacer recomendaciones personalizadas de manera robusta o incluso son una buena alternativa para usuarios que desean que se respeta su privacidad.

También estoy muy contento del trabajo realizado en el E-Modelo. Aunque no se haya podido mostrar aquí creo que tiene una potencia enorme, ya que añade toda una nueva dimensión al filtrado colaborativo clásico. Una de las causas de que sus resultados no me hayan convencido como para incluirlo es posiblemente que necesite más una cantidad de datos más grande que los que estaba usando (alrededor de 100.000 textos).

Creo que otro aspecto importante es que el código sea libre para que otras personas puedan probarlo y extenderlo. Por ahora no es posible compartir el código del E-Modelo, pero es posible que en el futuro sea publicado por el grupo de Big Data Lab de la UEM.

11.1. Mejoras

Este proyecto se podría ampliar para incluir otros algoritmos de recomendación basados en distintas técnicas como LDA, cadenas de Markov o clustering. La interfaz podría ser más pulida y ofrecer más información sobre las recomendaciones. Faltan tests unitarios y la habilidad de agregar contenidos fácilmente con el crawler. Las recomendaciones se hacen en tiempo real y no se incluye ningún tipo de cache. Precalcular las recomendaciones o usar algún sistema de cacheado como Service Workers o Redis sería conveniente.

Por supuesto el modelo tiene amplio margen de mejora, ya que algunos resultados son un tanto extraños. También sería una buena opción poder ajustar los parámetros desde la página web. Esto supondría un nivel de complejidad en la arquitectura más alto, ya que el proceso de entrenamiento lleva unos minutos y consume todos los recursos disponibles de la máquina donde se ejecuta. Además habría que guardar cada modelo por separado para evitar recalcular modelos ya entrenados, pero no ocupan poco espacio, y la máquina donde se ejecutan tiene un cantidad muy limitada (20 GB) de espacio de disco.

Bibliografía

- [1] H. Borko y M. Bernick, «Automatic Document Classification», *J. ACM*, vol. 10, n.^o 2, págs. 151-162, abr. de 1963, ISSN: 0004-5411. DOI: 10.1145/321160.321165. dirección: <http://doi.acm.org/10.1145/321160.321165>.
- [2] C. van Rijsbergen, S. Robertson y M. Porter, «New models in probabilistic information retrieval», 1980.
- [3] M. Lesk, «Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone», en *Proceedings of the 5th Annual International Conference on Systems Documentation*, ép. SIGDOC '86, Toronto, Ontario, Canada: ACM, 1986, págs. 24-26, ISBN: 0-89791-224-1. DOI: 10.1145/318723.318728. dirección: <http://doi.acm.org/10.1145/318723.318728>.
- [4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer y R. Harshman, «Indexing by latent semantic analysis», *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, n.^o 6, págs. 391-407, 1990.
- [5] L. Terveen y W. Hill, «Beyond Recommender Systems: Helping People Help Each Other», en *HCI In The New Millennium*, J. Carroll, ed., Massachusetts: Addison-Wesley Verlag, 2001, págs. 1-21.
- [6] A. Felfernig, K. Isak, K. Szabo y P. Zachar, «The VITA Financial Services Sales Support Environment», en *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ép. IAAI'07, Vancouver, British Columbia, Canada: AAAI Press, 2007, págs. 1692-1699, ISBN: 978-1-57735-323-2. dirección: <http://dl.acm.org/citation.cfm?id=1620113.1620117>.
- [7] E. L. Bird Steven y E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.
- [8] N. Halko, P.-G. Martinsson y J. A. Tropp, «Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions», *ArXiv e-prints*, sep. de 2009. arXiv: 0909.4061 [math.NA].

- [9] K. Onuma, H. Tong y C. Faloutsos, «TANGENT», en *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, ACM Press, 2009. doi: 10.1145/1557019.1557093. dirección: <https://doi.org/10.1145/1557019.1557093>.
- [10] X. Su y T. M. Khoshgoftaar, «A Survey of Collaborative Filtering Techniques», *Advances in Artificial Intelligence*, vol. 1425, pág. 19, 2009. doi: 10.1155/2009/421425.
- [11] J. Leskovec y A. Rajaraman, «Stanford CS345a: DataMining 56», 2010, dirección: https://web.stanford.edu/class/cs345a/slides/11-dim_red.pdf.
- [12] R. Řehůřek y P. Sojka, «Software Framework for Topic Modelling with Large Corpora», English, en *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, <http://is.muni.cz/publication/884893/en>, Valletta, Malta: ELRA, mayo de 2010, págs. 45-50.
- [13] E. García-Cuesta, I. M. Galván y A. J. D. Castro, «RECURSIVE DISCRIMINANT REGRESSION ANALYSIS TO FIND HOMOGENEOUS GROUPS», *International Journal of Neural Systems*, vol. 21, n.º 01, págs. 95-101, feb. de 2011. doi: 10.1142/s0129065711002663. dirección: <https://doi.org/10.1142/s0129065711002663>.
- [14] S. B. Tania Farinella y L. Po, «A Non-intrusive Movie Recommendation System», 2012.
- [15] J. Beel, M. Genzmehr, S. Langer, A. Nürnberg y B. Gipp, «A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation», en *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, ép. RepSys '13, Hong Kong, China: ACM, 2013, págs. 7-14, ISBN: 978-1-4503-2465-6. doi: 10.1145/2532508.2532511. dirección: <http://doi.acm.org/10.1145/2532508.2532511>.
- [16] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang y R. Zadeh, «WTF: The Who to Follow Service at Twitter», en *Proceedings of the 22Nd International Conference on World Wide Web*, ép. WWW '13, Rio de Janeiro, Brazil: ACM, 2013, págs. 505-514, ISBN: 978-1-4503-2035-1. doi: 10.1145/2488388.2488433. dirección: <http://doi.acm.org/10.1145/2488388.2488433>.
- [17] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient Estimation of Word Representations in Vector Space», *CoRR*, vol. abs/1301.3781, 2013. dirección: <http://arxiv.org/abs/1301.3781>.
- [18] Q. V. Le y T. Mikolov, «Distributed Representations of Sentences and Documents», *CoRR*, vol. abs/1405.4053, 2014. dirección: <http://arxiv.org/abs/1405.4053>.

- [19] X. Rong, «word2vec Parameter Learning Explained», *CoRR*, vol. abs/1411.2738, 2014. dirección: <http://arxiv.org/abs/1411.2738>.
- [20] H. Chen, A. G. O. II y C. L. Giles, «ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries», *CoRR*, vol. abs/1511.02058, 2015. dirección: <http://arxiv.org/abs/1511.02058>.
- [21] J. Beel, B. Gipp, S. Langer y C. Breitinger, «Research-paper recommender systems: a literature survey», *International Journal on Digital Libraries*, vol. 17, n.º 4, págs. 305-338, nov. de 2016, ISSN: 1432-1300. DOI: 10.1007/s00799-015-0156-0. dirección: <http://dx.doi.org/10.1007/s00799-015-0156-0>.
- [22] E. García-Cuesta, D. Gómez-Vergel, L. G. Expósito y M. Vela-Pérez, «Prediction of User Opinion for Products - A Bag-of-Words and Collaborative Filtering based Approach», en *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, SCITEPRESS - Science y Technology Publications, 2017. DOI: 10.5220/0006209602330238. dirección: <http://abacus.universidadeuropea.es/handle/11268/6365>.
- [23] *IMDb*, www.imdb.com.
- [24] L. Lorenz, *Forward Propagation: Building a Skip-Gram Net From the Ground Up Part 1: Skip-gram Feedforward*. dirección: <http://blog.districtdatalabs.com/nlp-research-lab-part-3-forward-propagation-1>.
- [25] *Penn Treebank tagset*, <https://catalog.ldc.upenn.edu/ldc99t42>.
- [26] *scikit-learn*, <http://scikit-learn.org/stable/>.
- [27] *Scrapy*, <https://scrapy.org/>.
- [28] *Snowball*, <https://snowballstem.org/>.
- [29] S. Zafrany, *NLP with gensim (word2vec)*. dirección: <http://www.samyzaf.com/ML/nlp/nlp.html>.

Índice de figuras

1.1.	Sistemas de recomendación basados en filtrado colaborativo	14
1.2.	Recomendaciones por película en IMDb	15
1.3.	Recomendaciones en Twitter	16
2.1.	Diagrama de Gantt con tiempos del proyecto	20
3.1.	Diagrama de flujo de la descarga de datos	21
4.1.	Resultados del POS tagging	25
5.1.	Ejemplo matriz TF-IDF	30
5.2.	Ejemplo matriz LSA	31
5.3.	Matriz LSA palabra a categoría	31
5.4.	Ejemplo matriz Σ de LSA	32
5.5.	Matriz LSA documento a categoría	32
5.6.	Matriz LSA documento a categoría reducida	33
5.7.	Alta similitud entre Matrix y Alien	34
5.8.	Baja similitud entre Matrix y Amelie	34
5.9.	Visualización de la similitud del coseno	34
6.1.	Modelos Word2Vec	36
6.2.	Función de activación lineal en Word2Vec	37
6.3.	Representación visual del modelo Skip-Gram[24]	38
6.4.	Vectores de resultado de Word2Vec en 2D[29]	39
6.5.	Modelos Doc2Vec	40
7.1.	Matriz 3D del E-Modelo	41
7.2.	Matriz E-Modelo de 2 dimensiones	42
8.1.	Resultados al modificar ‘Minimum Document Frequency’ (Apocalypse Now) .	46
8.2.	Resultados al modificar ‘Minimum Document Frequency’ (2001: A Space Odyssey)	46
8.3.	Resultados al modificar ‘Minimum Document Frequency’ (Ratatouille) .	47

8.4. Resultados al modificar ‘Maximum Document Frequency’ (Apocalypse Now)	47
8.5. Resultados al modificar ‘Maximum Document Frequency’ (2001: A Space Odyssey)	48
8.6. Resultados al modificar ‘Maximum Document Frequency’ (Ratatouille)	48
8.7. Resultados al modificar ‘Number of Components’ (Apocalypse Now)	49
8.8. Resultados al modificar ‘Number of Components’ (2001: A Space Odyssey)	49
8.9. Resultados al modificar ‘Number of Components’ (Ratatouille)	50
8.10. Resultados al modificar ‘Number of Features’ (Apocalypse Now)	50
8.11. Resultados al modificar ‘Number of Features’ (2001: A Space Odyssey)	51
8.12. Resultados al modificar ‘Number of Features’ (Ratatouille)	51
8.13. Resultados al modificar ‘Size’ (Apocalypse Now)	52
8.14. Resultados al modificar ‘Size’ (2001: A Space Odyssey)	53
8.15. Resultados al modificar ‘Size’ (Ratatouille)	53
8.16. Resultados al modificar ‘Window’ (Apocalypse Now)	54
8.17. Resultados al modificar ‘Window’ (2001: A Space Odyssey)	54
8.18. Resultados al modificar ‘Window’ (Ratatouille)	55
8.19. Resultados al modificar ‘Minimum count’ (Apocalypse Now)	55
8.20. Resultados al modificar ‘Minimum count’ (2001: A Space Odyssey)	56
8.21. Resultados al modificar ‘Minimum count’ (Ratatouille)	56
8.22. Resultados al modificar ‘Iterations’ (Apocalypse Now)	57
8.23. Resultados al modificar ‘Iterations’ (2001: A Space Odyssey)	57
8.24. Resultados al modificar ‘Iterations’ (Ratatouille)	58
 10.1. Interfaz de la aplicación	64
10.2. Interfaz al buscar	64
10.3. Interfaz al mostrar unos resultados	65
10.4. Arquitectura del sistema	65

Índice de cuadros

6.1.	Ventana deslizante en Word2Vec	36
6.2.	Vectores one-hot	37
6.3.	Resultados Word2Vec	40
9.1.	Recomendaciones para <i>Star Wars: Episode VII – The Force Awakens</i>	60
9.2.	Recomendaciones para <i>Match Point</i>	60
9.3.	Recomendaciones para <i>Toy Story 2</i>	61
9.4.	Recomendaciones para <i>Ponyo</i>	61
9.5.	Recomendaciones para <i>Los Otros</i>	62
9.6.	Recomendaciones para <i>Deadpool</i>	62

Índice alfabético

- ALS, 41, 42
- Amazon, 41
- Apache Spark, 41
- Doc2Vec, 16, 19, 22, 35, 39, 40, 52, 63
- E-Modelo, 16, 19, 41, 42, 67
- FilmAffinity, 19
- IMDb, 14–16, 19, 22, 39, 41, 59, 63
- Javascript, 63
- JSON, 22
- LSA, 14, 16, 19, 22, 26, 31–33, 45, 63, 65
- Movistar+, 14, 15, 59
- Netflix, 14, 15
- Python, 21, 22, 63
- SVD, 14, 30, 33
- TF-IDF, 14, 29, 30, 45
- Wikipedia, 19
- Word2Vec, 35–37, 39, 40