UNIVERSITAT OBERTA DE CATALUNYA

COLLECTIVE INTELLIGENCE

MEMORIA PROYECTO FIN DE CARRERA

CARLOTA VIÑA SIRGO

10/06/2011

TITULACIÓN : INGENIERÍA EN INFORMÁTICA

CONSULTOR: DAVID MASIP RODO

TABLA DE ILUSTRACIONES

RESUMEN

Antes de describir este proyecto, se hará un breve estudio de **Collective Intelligence**. Podriamos creer que Collective Intelligence es un concepto nuevo, sin embargo, ha acompañado a la humanidad desde siempre.

¿Cuál es la definición de Collective Intelligence? Podemos definirla como:

Grupos de individuos haciendo cosas colectivamente que parecen inteligentes (http://scripts.mit.edu/~cci/HCI/index.php?title=Main_Page)

Veamos algunos ejemplos de Collective Intelligence relacionados con Informática e Inteligencia Artificial:

- Inspección de imágenes. Neufeld et al. (2003) adoptan el paradigma de una asociación entre inteligencias artificiales y humana, resolviendo el problema de detección de bordes.
- Colaboradores de la NASA, donde una gran cantidad de voluntarios trabajan para identificar imágenes de Marte, produciendo resultados que son indistinguibles de aquellos producidos por estudiantes de doctorado trabajando dentro de la NASA.

Se debe citar la relación entre Collective Intelligence e Internet. Internet es una nueva fuente de puntos de vista y gustos personales. De la explotación de estos datos y su relación con la Inteligencia Artificial se encarga este proyecto.

Este proyecto se centra en el desarrollo de los algoritmos de recomendación, ranking y aprendizaje.

Recomendación se basa en la explotación de los datos acerca de los preferencias personales almacenados en Internet para hacer recomendaciones a otras personas.

Otro punto que se desarrolla en este proyecto, es la búsqueda y el ranking. Estos algoritmos permiten hacer búsquedas de palabras en documentos, haciendo posteriormente un ranking.

Por último está el algoritmo de aprendizaje. En este algoritmo se implementará una red neuronal y se verá como las distintas elecciones de los usuarios influyen en el ranking.

INTRODUCCIÓN

Con la expansión de Internet y la aparición de las redes sociales, son necesarias nuevas herramientas para la explotación de la información almacenada en la red.

Este proyecto implementa algunas de estas herramientas como son la recomendación, búsqueda, ranking y aprendizaje.

Para implementar estos algoritmos es necesario nuevas técnicas debido a las características de la información a tratar.

Las aplicaciones del algoritmo de recomendación son múltiples. Se puede emplear para recomendar películas, programas TV, vídeos, música, libros …

Un sistema de recomendación compara un perfil de usuario con las características de referencia y trata de predecir que ítem elegiría. La predicción se elabora en base a ítems ya elegidos o al entorno social del usuario.

Para construir un perfil de usuarios, nos basamos en lo siguiente:

- Pedir al usuario puntuar un ítem
- Pedir a un usuario puntuar una colección de ítems por orden de preferencia
- Pedir a un usuario que elija entre dos ítems
- Pedir al usuario que cree una lista de ítems a su criterio personal
- Tratamiento de los ítems que un usuario ve en un almacenamiento on line
- Guardar un registro de los ítems adquiridos
- Obtener una lista de usuario locales
- Analizar la red social del usuario y descubrir sus gustos

Se estudia los datos recolectados en los apartados anteriores y que sirven para establecer el perfil de usuario al que se quiere hacer la recomendación. Se compara este perfil con perfiles similares de usuarios que ya han elaborado una determinada recomendación y se muestra esta recomendación.

Los algoritmos de búsqueda y ranking son sistemas empleados por Google que permite a los usuarios buscar documentos en World Wide Web y USENET newsgroups.

Los algoritmos de búsqueda usan un programa que recupera páginas de forma automatizada. A estos programas se denominan spiders. Los spiders

surten de páginas a las máquinas de búsqueda. Debido a que la mayoría de las páginas Web contienen links a otras páginas, un spider puede comenzar en cualquier sitio. Tan pronto como detecta un link a otra página, la recupera. Máquinas de búsqueda como Google tienen muchos spiders trabajando en paralelo.

Las máquinas de búsqueda nos permiten localizar cualquier cosa en Internet cuando no conocemos la dirección URL concreta.

Hay tres tipos de máquinas de búsqueda, aquellas que son manejadas por robots (llamadas crawlers, ants o spiders) ; aquellas que son manejadas por peticiones humanas; y las híbridas.

Los crawler usan agentes automáticos de software que visitan una portal, buscan su información, leen sus meta tags y se conectan con todos los links encontrados. El crawler retorna toda la información a un registro central, donde los datos son indexados. El crawler retornará periódicamente a los sitios para chequear si la información ha cambiado. La frecuencia con la que esto ocurre es determinada por los administradores de la máquina de búsqueda.

Las máquinas manejadas por humanos, suben la información que es posteriormente indexada y catalogada.

En ambos casos, debe quedar claro que las busquesas no se realizan directamente en internet sino sobre los índices creados anteriormente por el crawler. Estos índices son bases de datos de información de gran tamaño. Si los índices no son actualizados, puede ocurrir que las búsquedas sean incorrectas.

¿Por qué la misma búsqueda sobre diferentes máquinas de búsqueda produce diferentes resultados? En primer lugar, es porque no todos los índices son los mismos. Éstos dependen de lo que los spiders encuentren o los que los humanos lancen. Pero más importante, porque no todas las máquinas de búsqueda usan el mismo algoritmo para buscar a través de los índices. El algoritmo que usa la máquina de búsqueda determina la relevancia de la información en el índice.

Uno de los elementos que el algoritmo de la máquina de búsqueda utiliza es la frecuencia y la localización de las palabras clave en la página Web. Las de más alta frecuencia son típicamente consideradas de más relevancia.

Algoritmo de ranking

La importancia de la página Web es subjetiva, depende de los intereses de los lectores, conocimiento y aptitudes. PageRank es un método para valorar páginas Web objetivamente y mecánicamente. PageRank es un método para calcular la valoración de cada página web basado en el grafo de la web. PageRank tiene aplicaciones en búsqueda, navegación y estimación.

Aunque las estimaciones varían, el grafo de la web tiene alrededor de 150 millón de nodos (páginas) y aproximadamente 1700 millones de links. Cada página tiene un número de enlaces hacia delante (links a otras páginas) y líneas hacia atrás (links a esta página desde otras webs). Dada una pagina es trivial conocer todos los links hacia delante, sin embargo la obtención de los links hacia atrás no es trivial. El número de líneas hacia atrás varían mucho entre páginas Web. Por ejemplo, la página Netscape tiene 62.804 líneas hacia atrás. Generalmente, las páginas con muchos links son más importantes que las páginas con pocos links. PageRank provee un método sofisticado para este cálculo. Nuestro sentido de importancia de una web no coincide con el que nos da PageRank. Por ejemplo, una página web tcon un único link a la página principal de Google, será más valorada que una página con dos links a pàginas menos importantes que Google.

Artificial Neural Network

Existen redes que ven los nodos como “neuronas artificiales”. Son las llamadas redes de neuronas artificiales (ANNs). Una neurona artificial es un modelo computacional inspirado en las neuronas naturales. Las neuronas naturales reciben señales a través de la sinapsis, a través de las dendritas o membranas de las neuronas. Cuando la señales recibidas son bastante fuertes (superan un cierto umbral), la neurona es activada y emite una señal a través del axón. Esta señal puede ser enviada a otra synapse, y puede activar otras neuronas.

La complejidad de las neuronas reales se abstrae cuando se están modelando las neuronas artificiales. Éstas básicamente constan de inputs (como synapses), las cuales son multiplicadas por pesos (fortaleza de las señales), y calculadas por una función matemática la cual determina la activación de la neurona. Otra función calcula la salida de la neurona artificial (algunas veces en dependencia con cierta fortaleza). ANNs combina neuronas artificiales para procesar información.

El peso más alto de una neurona artificial, es la más fuerte de las entradas. Los pesos también pueden ser negativos, esto quiere decir que la señal es

inhibida por el peso negativo. Dependiendo de los pesos, el cálculo de la neurona será diferente. Ajustando los pesos de una neurona pueden obtener la salida que queremos para específicas inputs. Pero cuando tenemos una ANN con cientos de miles de neuronas, es bastante complicado encontrar todos los pesos necesarios. Pero podemos encontrar algoritmos que pueden ajustar los pesos de una ANN para obtener la deseada salida de la red. Este proceso de ajustar los pesos es llamado aprendizaje.

Hay ANNs que pueden usar para ingeniería, como reconocimiento de patrones, previsión y compresión de datos.

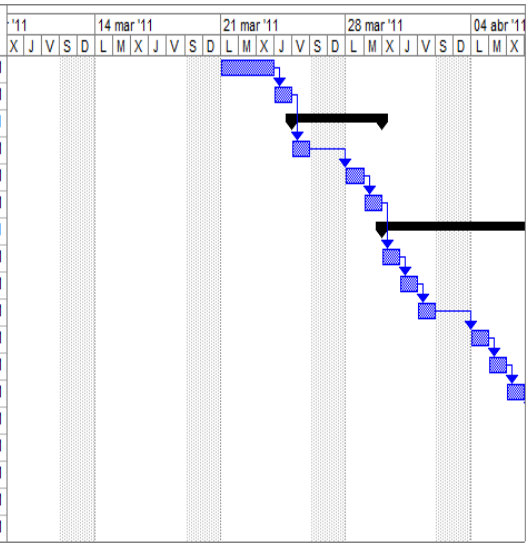
El algoritmo de backpropagation es usada en capas alimentación hacia delante ANNs. Esto significa que las neuronas están organizadas en capas, y envían sus señales hacia delante, y entonces los errores son propagados hacia atrás. La red recibe entradas por las neuronas en la capa de entrada, y la salida de la red es dada por las neuronas por la capa de salida. Puede haber una o más capas intermedias ocultas. El algoritmo backpropagation usa aprendizaje supervisado, lo cual significa que suministramos al algoritmo ejemplos de las entradas y salidas que queremos, y entonces el error (diferencia entre el resultado real y esperado) es calculado. La idea del algoritmo de backpropagation es reducir este error, hasta que la ANN aprende del entrenamiento de datos. El entrenamiento comienza con pesos random, y la metas es ajustarlos hasta que el error sea mínimo.

La definición de estos algoritmos es el punto de partida del PFC, los objetivos son que aunque hay varias implementaciones de los mismos, se implementarán en Python, un lenguaje optimizado para el tratamiento de textos.

Se mostrará cómo nos podemos conectar a del.icio.us y hacer recomendaciones con los datos extraídos.

PLANIFICACIÓN DEL PROYECTO

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Definición campos trabajo	3 días?	lun 21/03/11	mié 23/03/11
2	Normalización de los datos	1 día?	jue 24/03/11	jue 24/03/11
3	**Implementación del algorimo de recomendación**	**3 días?**	**vie 25/03/11**	**mar 29/03/11**
4	Establecer criterios búsqueda	1 día	vie 25/03/11	vie 25/03/11
5	Calcular distancia Pearson	1 día?	lun 28/03/11	lun 28/03/11
6	Buscar coincidentes	1 día?	mar 29/03/11	mar 29/03/11
7	**Implementación algoritmo de Búsqueda y Ranking**	**12 días?**	**mié 30/03/11**	**jue 14/04/11**
8	Crear rastreador base datos	1 día?	mié 30/03/11	mié 30/03/11
9	Crear base datos	1 día?	jue 31/03/11	jue 31/03/11
10	Crear buscador base datos	1 día?	vie 01/04/11	vie 01/04/11
11	Crear esquema base de datos	1 día?	lun 04/04/11	lun 04/04/11
12	Extraer texto páginas	1 día?	mar 05/04/11	mar 05/04/11
13	Buscar el texto páginas	1 día?	mié 06/04/11	mié 06/04/11
14	Indexar documentos	1 día?	jue 07/04/11	jue 07/04/11
15	Motor de búqueda	1 día?	vie 08/04/11	vie 08/04/11
16	Formatear query	1 día?	lun 11/04/11	lun 11/04/11
17	Normalizar diccionario	1 día?	mar 12/04/11	mar 12/04/11
18	Elegir métrica	2 días?	mié 13/04/11	jue 14/04/11



PRODUCTOS OBTENIDOS

De la implantación de este proyecto se han obtenido tres ejecutables:

- pearson.py

Implementa el algoritmo de recomendación

- searchengine.py

Implementa el algoritmo de búsqueda/ranking

- nn.py

Implementa el algoritmo de aprendizaje

ALGORITMO DE RECOMENDACIÓN

Después de recolectar datos acerca de los gustos de las personas, se necesita un modo para determinar cómo de similares son las personas en sus gustos. Pero necesitamos una medida para que nos mida esta similitud.

Pearson Correlation Score se usa para definir similitudes entre dos ítems. La fórmula es:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

Este cálculo devuelve un valor entre -1 y 1. Dos usuarios que tengan un valor de 1, son considerados idénticos. A diferencia que la distancia euclídea esta fórmula no necesita ser normalizada. Pearson Correlation score, también cuenta la calificaciones medias para cada usuario, un usuario que tasa todas las cosas en 5 y un usuario que tasa todas las cosas en 1 tendrán una similitud de 1. Esto puede ser o no ser el comportamiento que quieres dependiendo de tu situación.

La utilidad de la correlación depende de su tamaño y significado. Si r difiere fiablemente de 0, el valor de r será estadísticamente significativo. Si r es significativo la relación entre las dos palabras no es debido a la casualidad.

El valor de r no cambia si los valores de cada variable son convertidos a una diferente escala. Por ejemplo, si los pesos de los estudiantes son cambiados de libras a kilogramos, el valor de r no cambiará.

Que una variable se relacione con otra, no significa que los cambios en una causa cambios en la otra. Correlación no prueba causa.

La conclusión de no significativa correlación linear no significa que X e Y no estén relacionados en algún modo. Un resultado de r=0 indicada correlación no linear entre las dos variables. La correlación de Pearson se aplica sólo a datos lineales.

ALGORITMO DE APRENDIZAJE

Otro algoritmo importante en Collective Intelligence es el algoritmo de aprendizaje. Después de una búsqueda, cada usuario puede elegir los resultados que prefiera. Se buscará un modo de grabar las elecciones del usuario y utilizarlos para mejorar los rankings de los resultados. Para ello se construirá una red artificial de neuronas. Esta red se entrenará dándole las palabras de la query

En el algoritmo de aprendizaje se establecen las siguientes etapas:

- Diseñar la red

Todas las redes de neuronas están formadas por nodos (neuronas) y conexiones entre ellas. El tipo de red que se implementará será multilayer perceptron (MLP). En esta red hay varias capas de neuronas, la primera de ellas se encargará de tomar la entrada (palabras que introduce el usuario). La última capa da la salida. Se implementará una red con una sola capa intermedia. Los nodos de la input layer estarán conectados con los nodos de la capa oculta y éstos con los nodos de la capa salida.

Los nodos entrada que representan las palabras de la query se inicializan a 1. Las salidas de estos nodos son encendidos e intentan activar la capa hidden. Además los nodos de la capa hidden que tienen una entrada lo bastante fuerte encenderán sus salidas y tratarán de activar los nodos en la capa salida.

Esto depende de las fortalezas de las conexiones. Esto se logra entrenando la red cada vez que alguien realice una búsqueda y elige una de las conexiones.

Las ventajas de las redes neuronales es que se pueden hacer conjeturas acerca de los resultados de queries que nunca ha tratado basándose en la similaridad con otras queries.

- Creación de la base de datos

Trataremos de almacenar una representación de la red en la base de datos. Las tablas de esta base de datos son:

Hiddennode tabla para la capa oculta

Tablas de conexiones

De la capa de entrada a la capa oculta

Capa oculta a la capa salida

- Alimentación de la red

Ahora tenemos que crear las funciones que toman las palabras como entradas, activan las conexiones en la red y dan un conjunto de salidas para las URLs.

El primer paso es determinar una función que indicar cómo ha de responder un nodo a la entrada. En las redes de neuronas será la tangente hiperbólica (tanh)

El eje X representa la entrada total para el nodo. Cuando la entrada vale 0, la salida comienza a subir rápidamente. Con una entrada de 2, la salida es casi 1. Este es un tipo de función sigmoidea. Las redes neuronales casi siempre usan funciones sigmoideas para calcular las salidas de las neuronas.

- Entrenamiento con Backpropagation

Ahora implementaremos el algoritmo de entrenamiento de la red. Para ello necesitaremos un algoritmo que modifique las conexiones entre los nodos para reflejar mejor la respuesta correcta de la red. Los pesos se ajustarán lentamente porque no se puede asumir que cada usuario elegirá la respuesta que es correcta para todo el mundo.

El algoritmo que se usará es el de backpropagation, se mueve hacia atrás en la red ajustando los pesos.

En el aprendizaje de una red, siempre se conoce la deseada salida de cada nodo en la capa de salida. Si el usuario elige el resultado se inicializa a 1, en caso contrario 0. La única manera de cambiar la salida de un nodo es cambiar la entrada total de este nodo.

Para determinar cuánto se debería cambiar la entrada total, el algoritmo de entrenamiento tiene que conocer la pendiente de la función tanh.

El algoritmo de backpropagation realice los siguientes pasos. Por cada nodo en la capa de Salida:

- Calcula la diferencia entre la salida del actual nodo y la que debería ser
- Usa la función dtanh para determinar cuánto tiene que cambiar la entrada total del nodo
- Cambia la fortaleza de cada conexión entrante en proporción con la fortaleza de la actual conexión y la tasa de aprendizaje

Por cada nodo en la capa oculta

- Cambia la salida del nodo por la suma de las fortalezas de cada conexión de salida multiplicada por lo que el nodo destino tiene que cambiar
- Usa la función dtanh para determinar cuánto tiene que cambiar el total de la entrada del nodo
- Cambiar la fortaleza de cada conexión de entrada en proporción a la fortaleza de la conexión actual y la tasa de aprendizaje

Este algoritmo calcula todos los errores de antemano y ajusta los pesos.

ANÁLISIS UML

Diagrama de casos de uso

A continuación se adjunta el diagrama de casos de uso. Se consideran las siguientes agrupaciones de casos de uso. Una asociada al algoritmo de recomendación, otra al algoritmo de búsqueda, el algoritmo de ranking y al algoritmo de aprendizaje.

Respecto a las relaciones entre casos de uso:

- Se ha utilizado include cuando dos casos de uso compartían una cierta parte que no tenía sentido por sí sola como caso de uso. Así:
  - topmatches incluye sim_pearson

Cada caso de uso está asociado a una función de los algoritmos. En los apartados siguientes se explican cada una de estas funciones.

El usuario podrá ejecutar los siguientes casos de uso:

-Hacer recomendaciones

-Crawler

-Searcher

-Aprendizaje

Casos de uso algoritmo recomendaciones

Este caso de uso se ha dividido en los siguientes elementos:

- **sim_pearson**. Como se ha indicado anteriormente el método elegido para determinar la similar dad entre los intereses de las personas es el coeficiente de Pearson. Este caso de uso se puede ejecutar individualmente o desde el caso de uso topMatches.

- **topMatches**. Una vez que hemos calculado la distancia de Pearson, crearemos una función que dada una persona encuentre los más cercanos coincidentes.

- **getRecommendations**. En función de los valores de la distancia del Pearson se hace la recomendación de perfiles.

- **transformPrefs**. Una vez que hemos encontrado las personas de gustos similares y hemos recomendado un perfil para una persona determinada, podemos saber qué perfiles son similares.

Casos de uso crawler

- **createindextables**. Este caso de uso prepara la base de datos para funciones posteriores.

- **Crawl.** Ese caso de uso desarrolla un modo de recuperar los documentos.

Casos de uso searcher

Este caso de uso es el rastreador de la base de datos.

- **geturlname.** Obtenemos el nombre de la url
- **getscoredlist**. Ejecutando otros casos de uso obtenemos las páginas que recupera la query, pero el orden es simplemente el orden del rastreador. Debemos calcular la prioridad de cada página. Hay varias métricas para calcular la prioridad y están basadas en:
    - Frecuencia de palabras

El número de veces que las palabras que se usan en la query aparecen el documento puede ayudar a determinar que relevante es el documento

- Localización en el documento

El asunto principal de un documento aparece cerca del comienzo del documento

- Distancia entre palabras

Si hay múltiples palabras en la query, deberán aparecer juntas en el documento.

Este caso de uso implementa una de estas métricas.

- **normalizescores.** Cada una de los caso de uso de scoring (word frequency, document location y word distance) , llama a esta función que devuelve un valor entre 0 y 1.
- **frequencyscore** Este caso de uso crea un diccionario con una entrada por cada único URL y cuenta cuántas veces aparece cada ítem.
- **locationscore**. Este caso de uso determina la relevancia de una página para una consulta. Si el término que se busca es importante en una página, éste aparecerá al principio de la página, por ejemplo en el título.
- **getmatchrows**. Este caso de uso construye la query, busca las URL y buscan las diferentes palabras. Este caso de uso incluye geturlname, getscoredlist, normalizescore, frequencyscore, locationscore.
- **Calculatepagerank**. Este caso de uso implementa el algoritmo PageRank. Este algoritmo asigna a cada página una prioridad que indica cómo de importante es la página. La importancia de la página es calculada en función de otras páginas que se unan a ella y el número de links que tiene cada página.

Este caso de uso inicializa el peso de cada página a 1.0 . Para cada URL se obtiene el valor PageRank y el número total de links para cada link.

Una vez que calculamos la puntuación PageRank, tenemos que recuperar las URL de la base de datos y normalizar las puntuaciones.

Casos de uso aprendizaje

Este caso de uso incluye los siguientes casos de uso.

- **getstrength**. Determina la fortaleza de una conexión.Si no hay conexiones retorna un valor por defecto. Para las conexiones desde la capa de entrada a la capa oculta el valor por defecto es -0.2. Para las

conexiones desde la capa oculta a las URLs, el método retornará un valor por defecto de 0.

- **getallhiddenids**. Genera un nodo nuevo en la capa oculta cada vez que se encuentra una combinación de palabras que nunca se habían visto juntas antes. Crea una conexión con una ponderación por defecto entre las palabras y los nodos ocultos y entre los nodos query y las URLs retornados por esta query.

- **setupnetwork.** Construye la red con los pesos a partir de la base de datos. Esta función establece una gran cantidad de variables(la lista de palabras, los nodos de la query y URLs, el nivel de salida de cada nodo, y los pesos de cada conexión entre nodos. Los pesos se toman de la base de datos. Incluye los casos de uso getallhiddenids, getstrength.

- **feedforward**. Este caso de uso implementa el algoritmo feedforward. Este algoritmo toma una lista de entradas, las introduce en la red, y retorna la salida de todos los nodos en la capa salida.

El algoritmo recorre en un bucle todos los nodos en la capa oculta y añade todas las salidas calculadas como los nodos entrada multiplicadas por los pesos de las conexiones. La salida de cada nodo es la función tanh de la suma de todas las entradas, que es pasada a la capa de salida.

- **getresult.** Este caso de uso incluye el caso de uso setupnetwork y feedforward.

- **updatedatabase.** Guarda los nuevos pesos en la base de datos. Para ello se utilizan los variables wi y wo

- **backpropagation.** Implementa el algoritmo de entrenamiento de la red.

- **trainquery**. Este caso de uso toma la lista de palabras, id de url y seleccionadas URL. Ejecuta la alimentación y la backpropagation. Incluye los casos de uso backPropagate, updatedatabase.

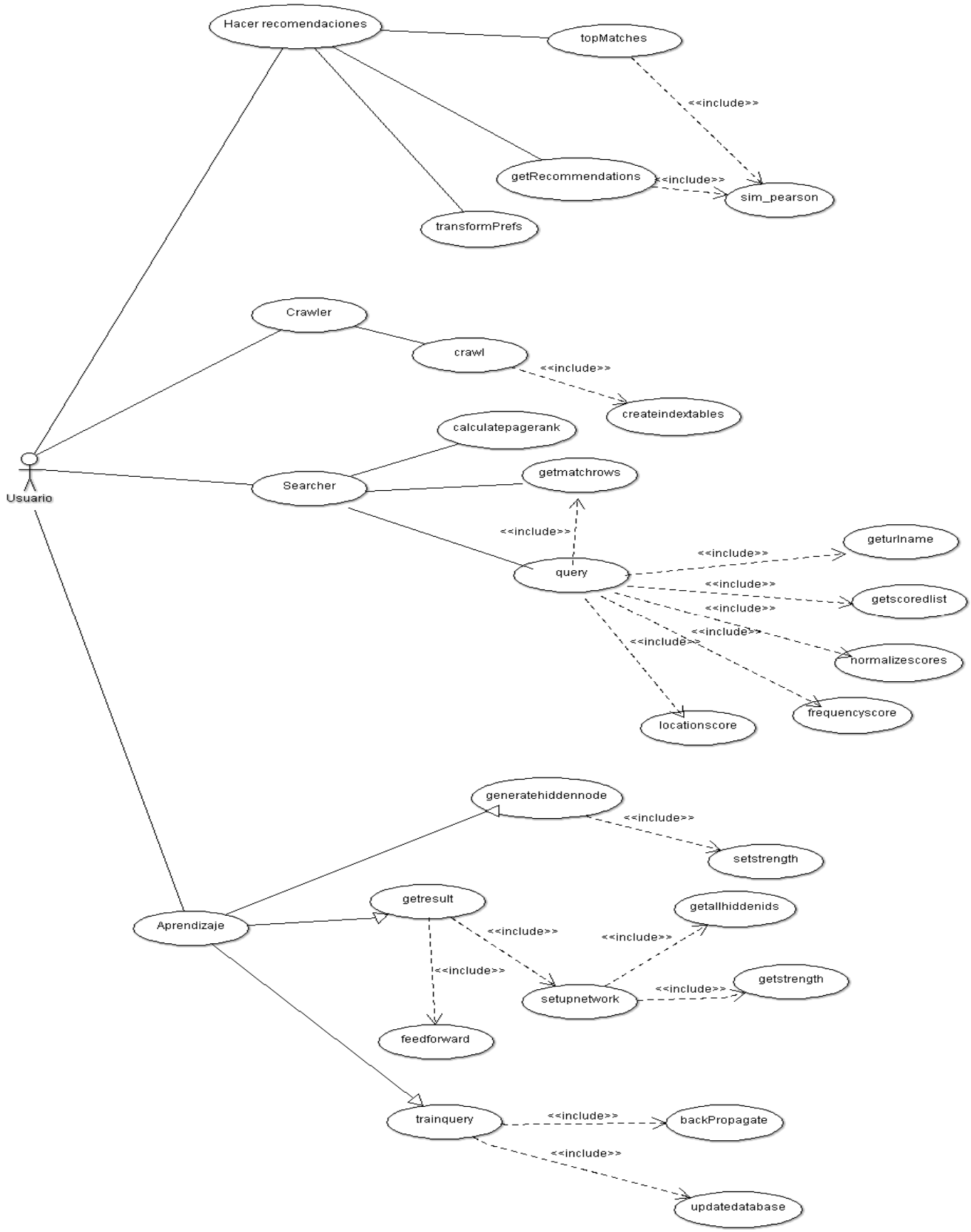


Ilustración 1.- Caso de uso

Diagrama de clases

Se han implementado tres clases que implementan los métodos necesarios para los algorimos de búsqueda, ranking y aprendizaje.

La clase crawler implementa los métodos del crawl

La clase searcher implementa los métodos del buscador (searcher)

La clase searchnet implementa los métodos necesarios para el algoritmo de aprendizaje.

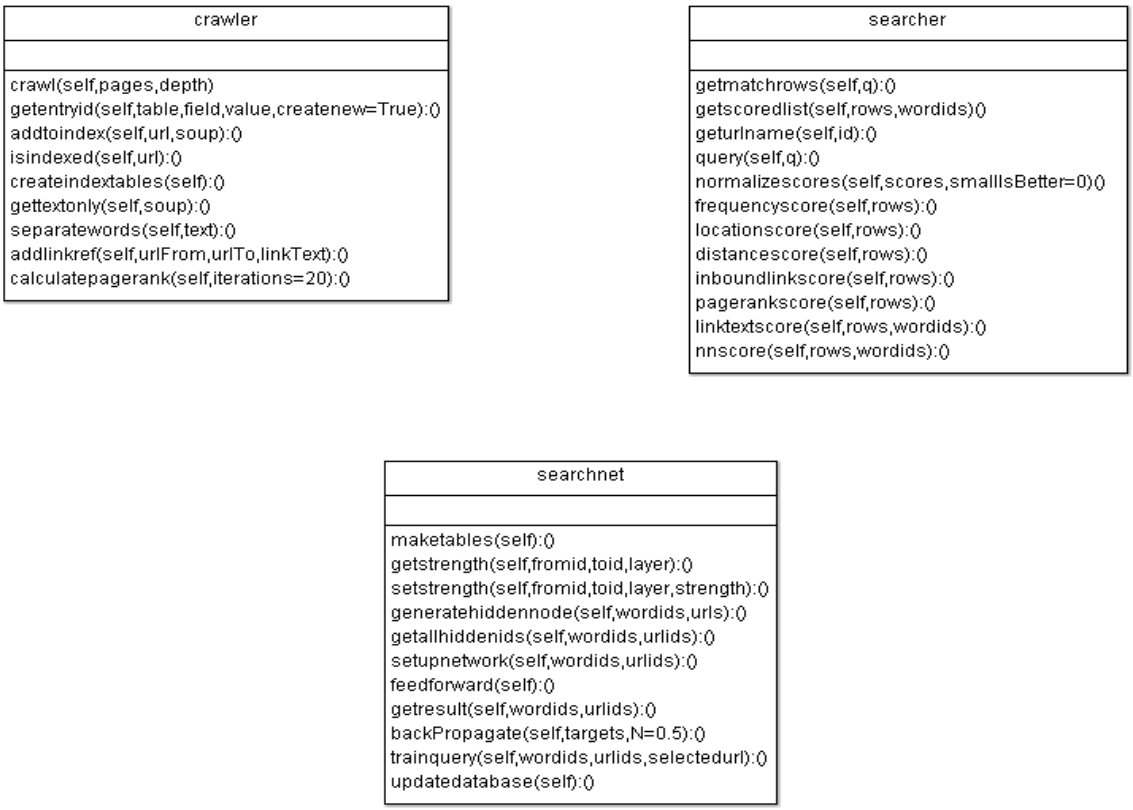


Ilustración 2.- Diagrama de clases

Diagramas de secuencia

Diagrama de secuencia algoritmo de recomendación

- Diagrama de secuencia sim_pearson

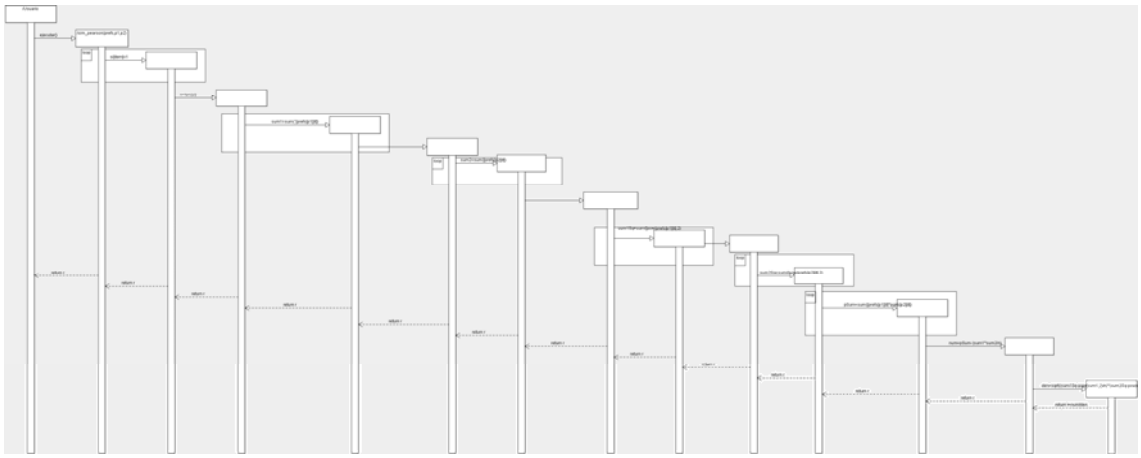


Ilustración 3.- Diagrama secuencia sim_pearson

- Diagrama de secuencia topMatches

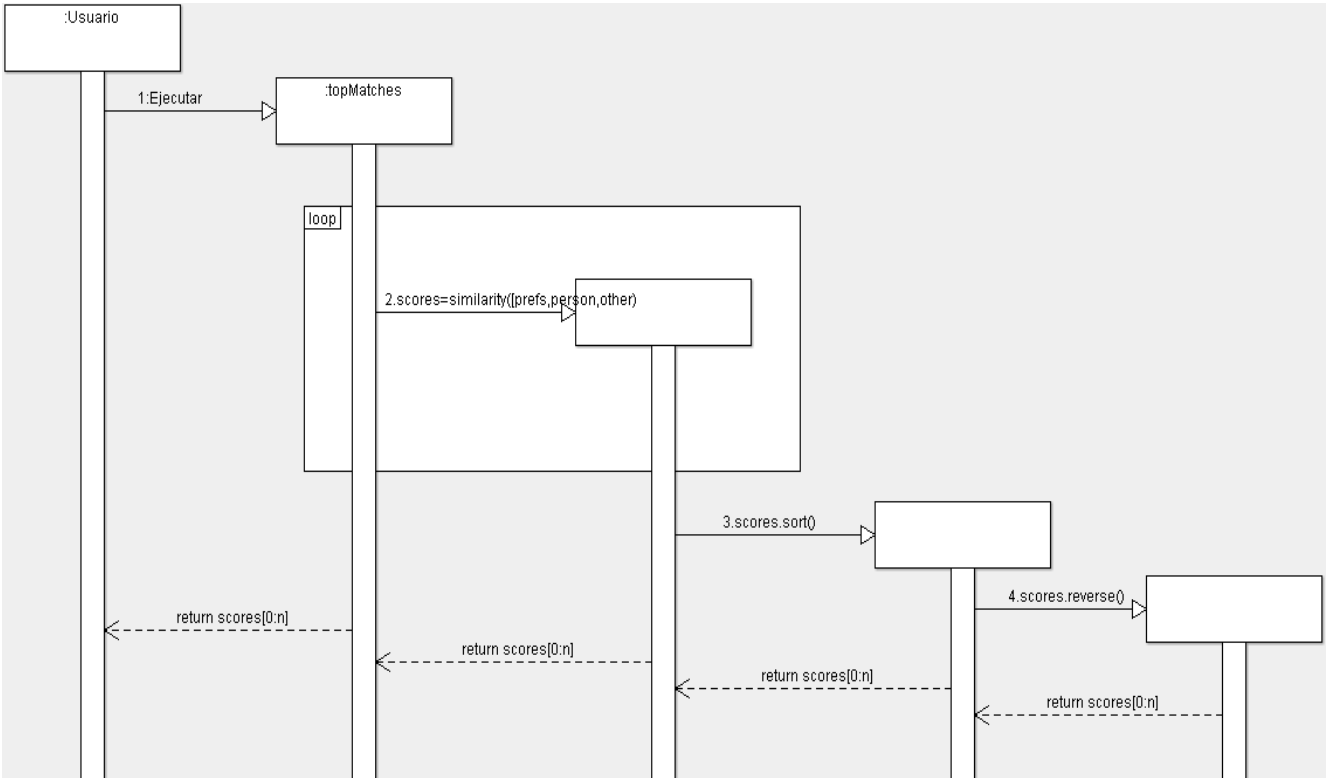


Ilustración 4.- Diagrama secuencia topMatches

- Diagrama de secuencia getRecommendations

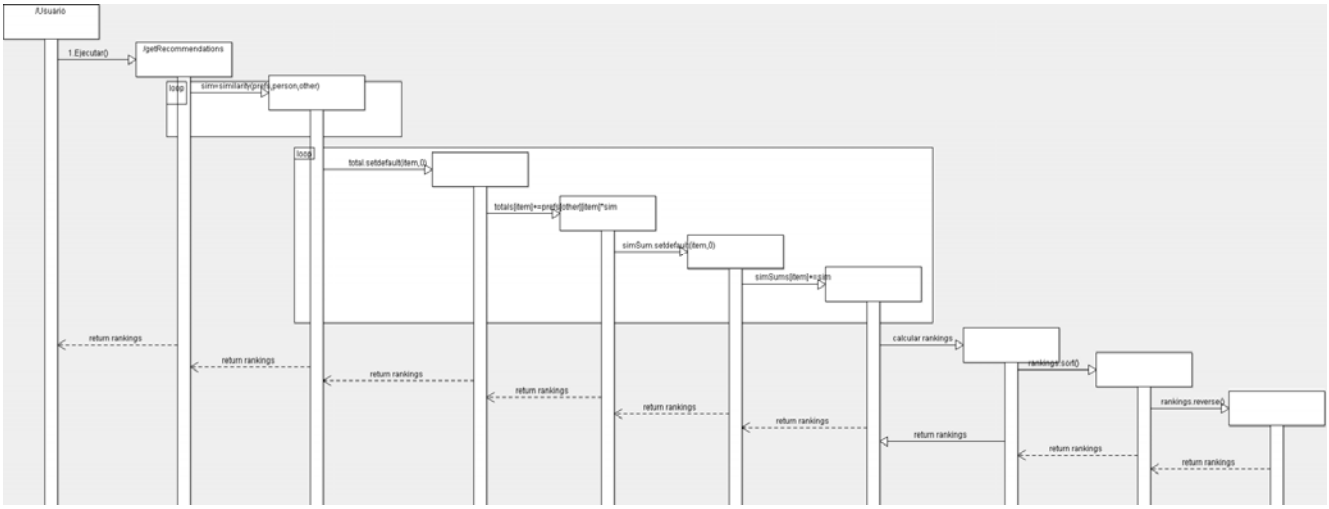


Ilustración 5.- Diagrama secuencia getRecommendations

- Diagrama de secuencia transformPrefs

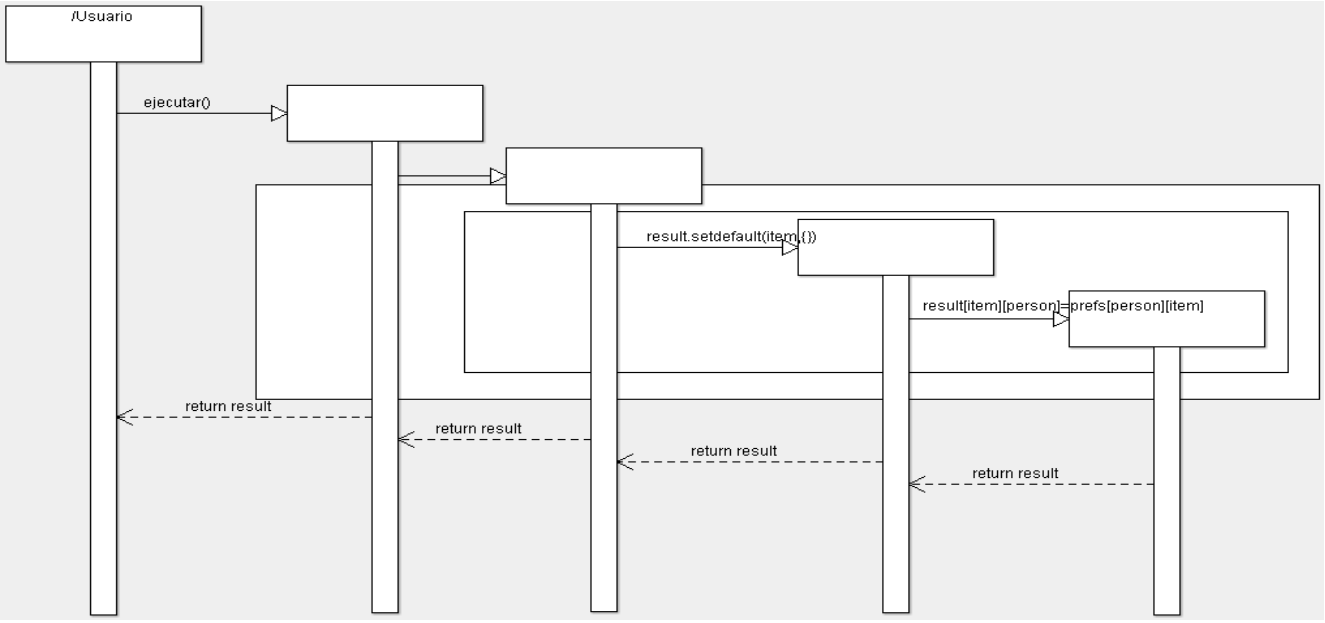


Ilustración 6.- Diagrama secuencia transformPrefs

Diagrama de secuencia del algoritmo búsqueda y ranking

- Diagrama de secuencia crawl

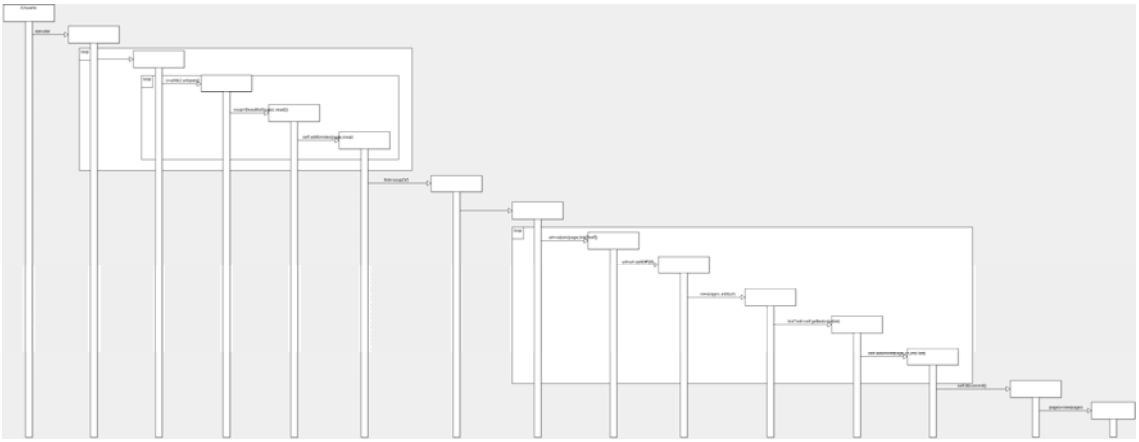


Ilustración 7.- Diagrama secuencia crawl

- Diagrama de secuencia gettextonly

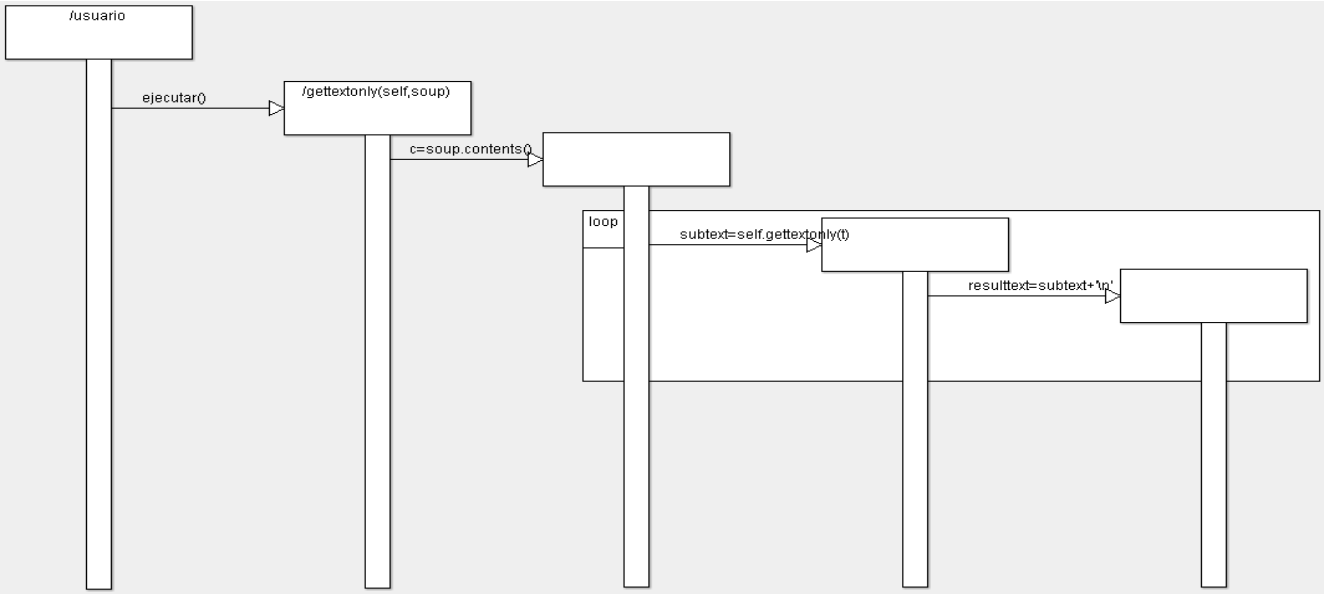


Ilustración 8.- Diagrama de secuencia gettextonly

- Diagrama de secuencia addtoindex

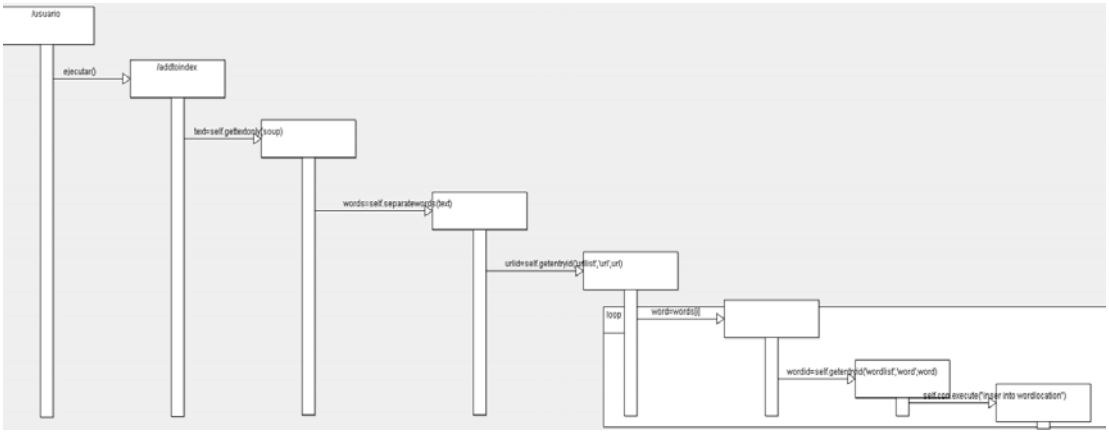


Ilustración 9.- Diagrama de secuencia addtoindex

- Diagrama de secuencia getentryid

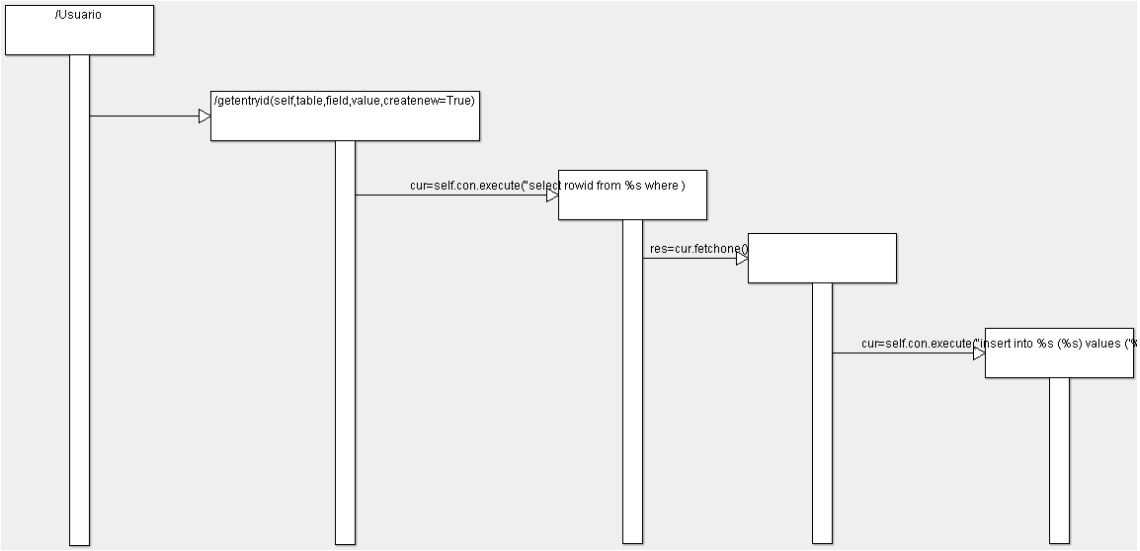


Ilustración 10.- Diagrama de secuencia getentryid

- Diagrama de secuencia addlinkref

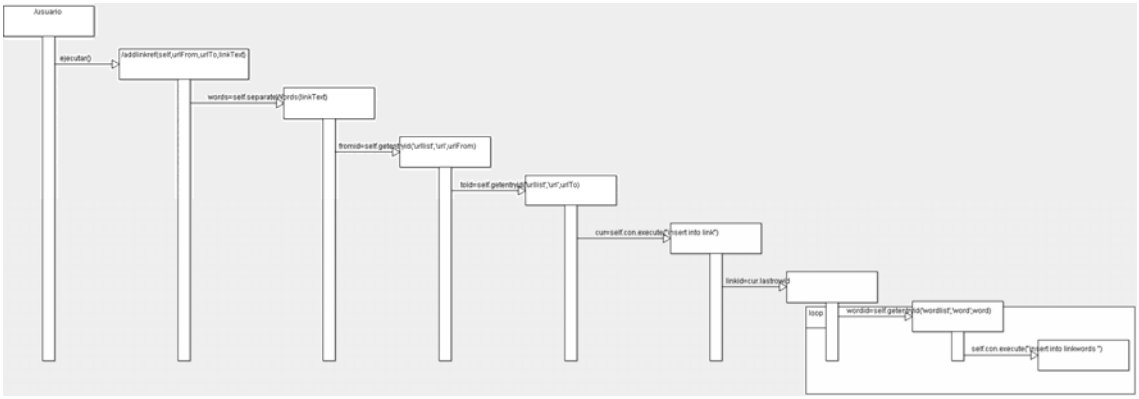


Ilustración 11.- Diagrama de secuencia addlinkref

- Diagrama de secuencia isindexed

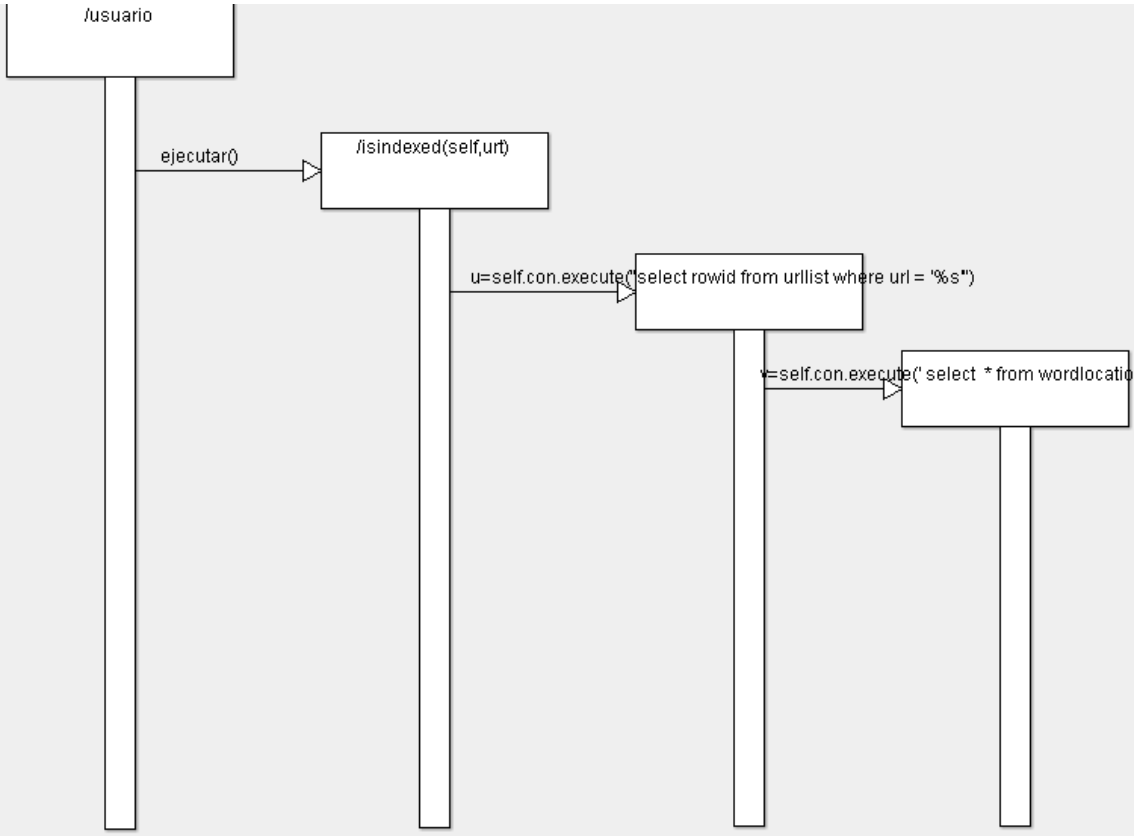


Ilustración 12.- Diagrama de secuencia isindexed

- Diagrama de secuencia getmatchrows

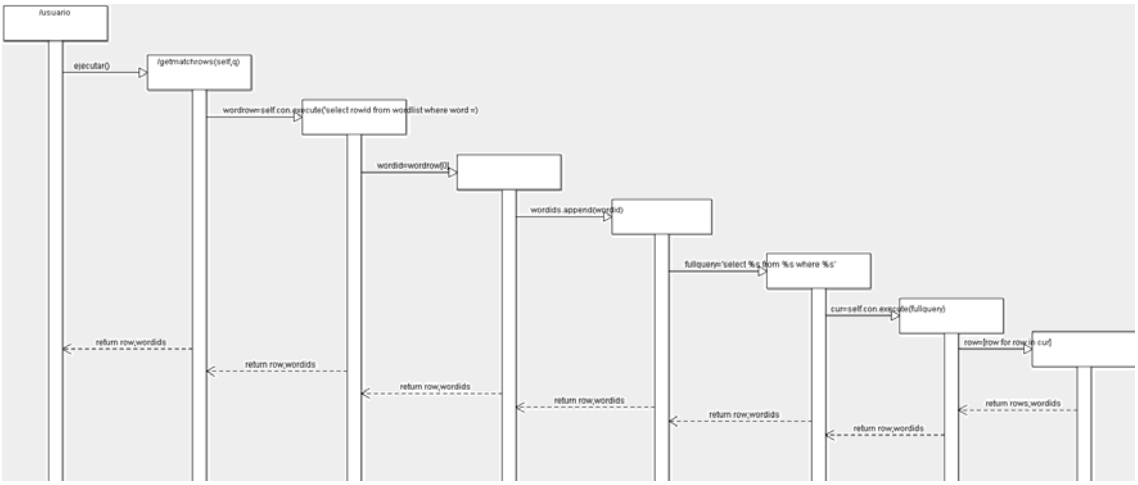


Ilustración 13.- Diagrama de secuencia getmatchrows

- Diagrama de secuencia getscoredlist

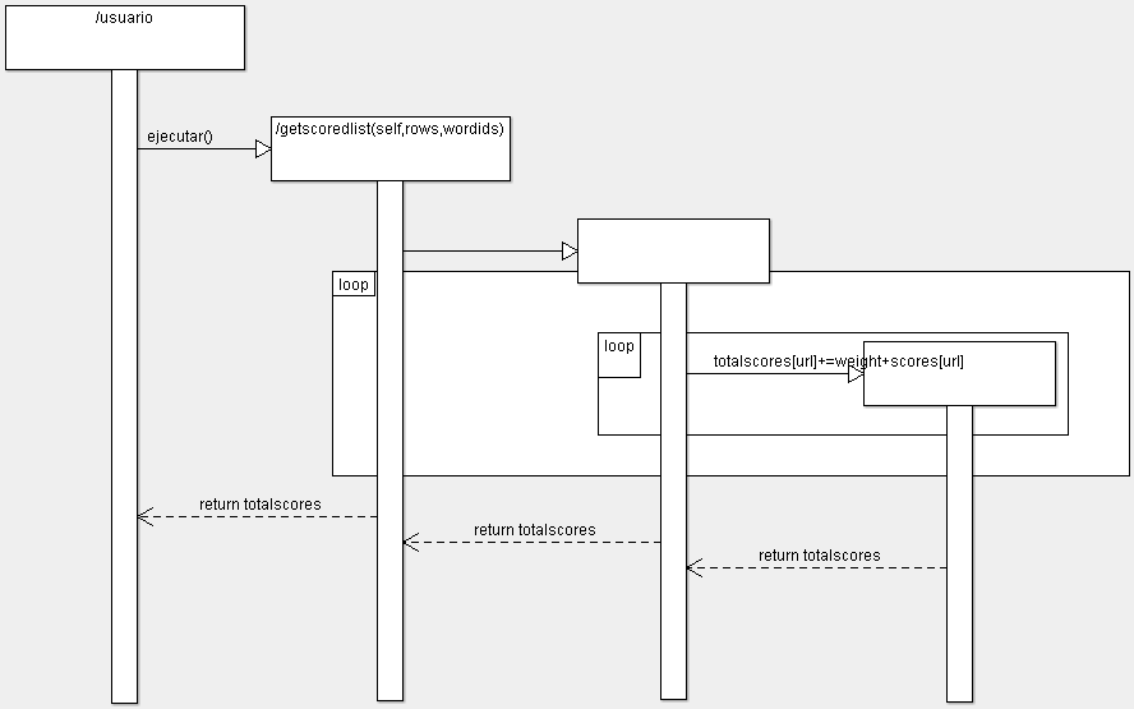


Ilustración 14.- Diagrama de secuencia getscoredlist

- Diagrama de secuencia query

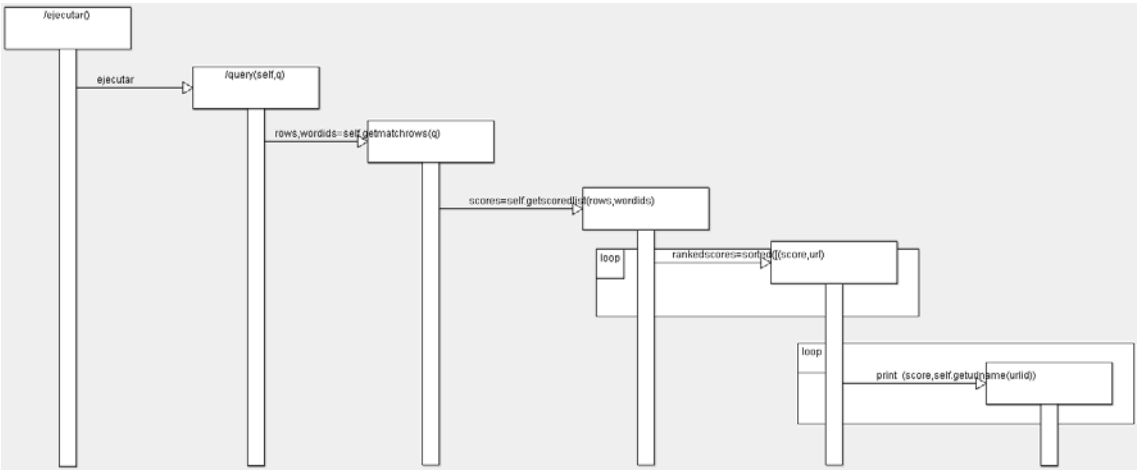


Ilustración 15.- Diagrama de secuencia query

- Diagrama de secuencia normalizedscores

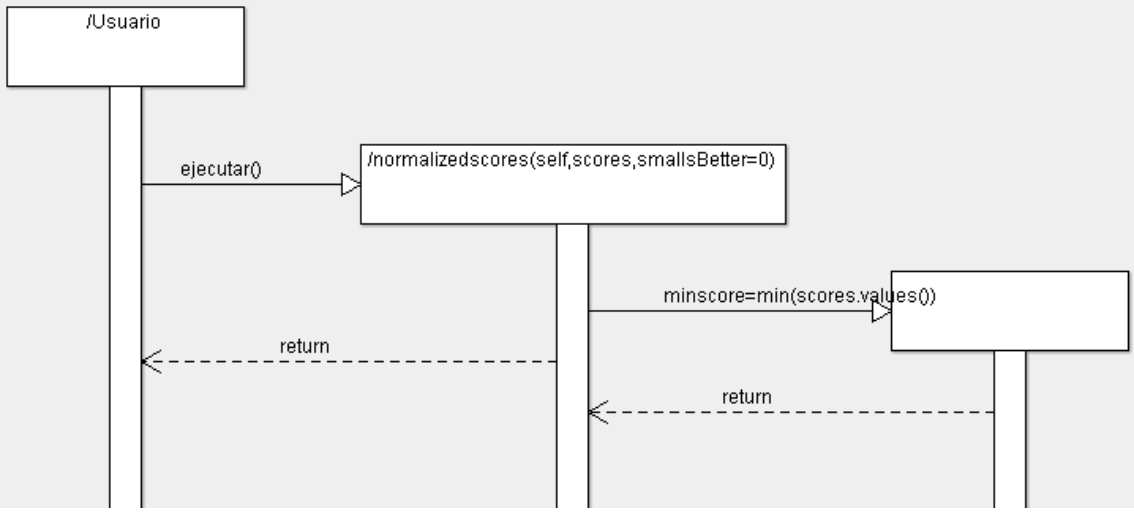


Ilustración 16.- Diagrama de secuencia normalizescores

Diagramas de secuencia algoritmo de aprendizaje

- Diagrama de secuencia getstrength

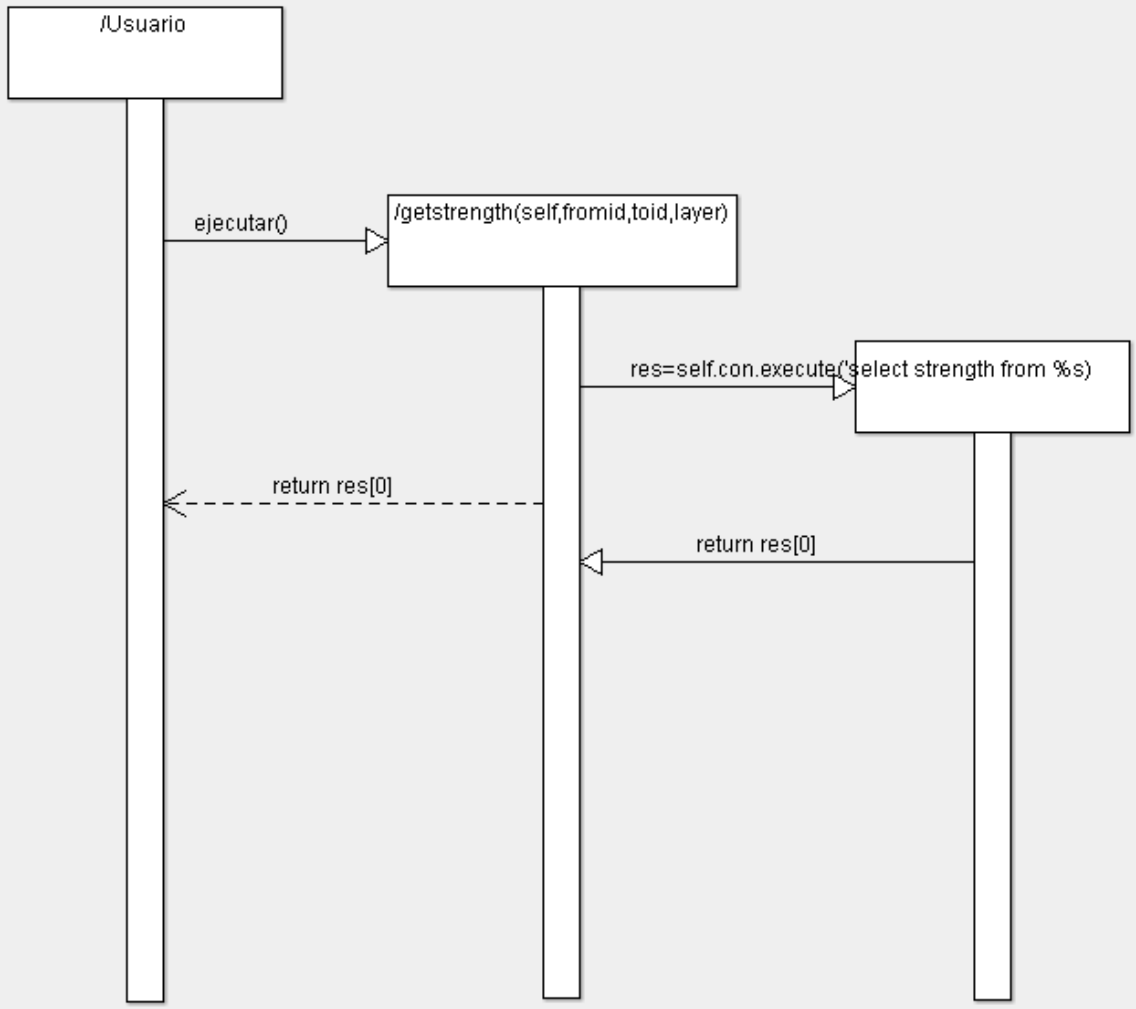


Ilustración 17.- Diagrama de secuencia getstrength

- Diagrama de secuencia setstrength

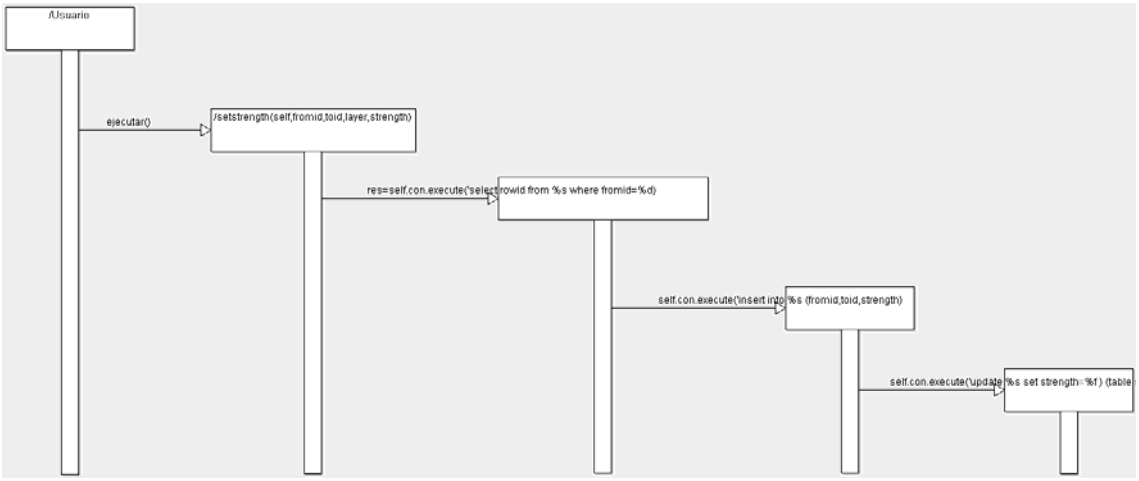


Ilustración 18.- Diagrama de secuencia setstrength

➢ Diagrama de secuencia generatehiddennode

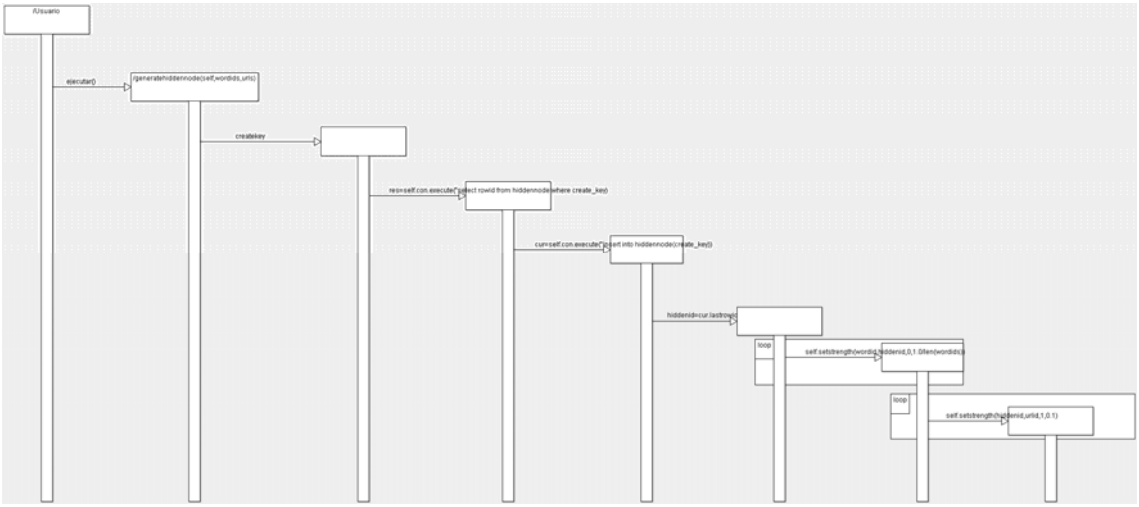


Ilustración 19.- Diagrama de secuencia generatehiddennode

APLICACIÓN PRÁCTICA

En este apartado se indica como ejecutar los diferentes métodos y los resultados.

En Python es importante el concepto de diccionario que nos sirve para almacenar datos.

El diccionario que emplearemos para las diferentes etapas de la implementación del algoritmo de recomendación, es el siguiente:

```
critics={'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
 'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
 'The Night Listener': 3.0},
'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
 'Just My Luck': 1.5, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.0,
 'The Night Listener': 3.5},
'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
 'Superman Returns': 3.5, 'The Night Listener': 4.0},
'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
 'The Night Listener': 4.5, 'Superman Returns': 4.0,
 'You, Me and Dupree': 2.5},
'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'Just My Luck': 2.0, 'Superman Returns': 3.0, 'The Night Listener': 3.0,
 'You, Me and Dupree': 2.0},
'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'The Night Listener': 3.0, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.5},
'Toby': {'Snakes on a Plane': 4.5, 'You, Me and Dupree':1.0, 'Superman Returns': 4.0}}
```

Aplicación práctica del Algoritmo de recomendación

A partir del diccionario de datos anterior se mostrará como hacer recomendaciones. En el primer punto , se indica como cuantifica la similitud entre dos ítems cualesquiera del diccionario. En este caso se ha elegido 'Lisa Rose' y 'Gene Seymour', obtenemos el siguiente valor 0.495073771488

sim_pearson

```
C:\uoc\collective_intelligence\pearson\src>c:\python26\python

Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit
(Intel)] on

win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import pearson

>>> print pearson.sim_pearson(pearson.critics,

... 'Lisa Rose','Gene Seymour')

<p>text

0.495073771488</p>

<p>

0.495073771488

>>>
```

Buscar coincidentes

El siguiente paso es comparar un ítem con todos los demás para encontrar el más similar. Encontraremos los n más similares. Este paso se basa en los cálculos que se obtienen al aplicar el Peason Correlation Score

```
C:\uoc\collective_intelligence\pearson\src>c:\python26\python

Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit
(Intel)] on

win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import pearson

>>> pearson.topMatches(pearson.critics,'Toby',n=3)

<p>text

0.66284898036</p>

<p>

<p>text

0.924473451642</p>

<p>

<p>text

0.893405147442</p>

<p>

<p>text

0.991240707162</p>

<p>

<p>text

0.592136908213</p>

<p>

<p>text

-1.0</p>

<p>

<p>text

(0.99124070716192991,    'Lisa    Rose')(0.99124070716192991,    'Lisa
Rose')</p>

<p>text

(0.92447345164190486,   'Mick   LaSalle')(0.92447345164190486,   'Mick
LaSalle')</p>
```

```
<p>text

(0.89340514744156474, 'Claudia Puig')(0.89340514744156474, 'Claudia
Puig')</p>

<p>

[(0.99124070716192991, 'Lisa Rose'), (0.92447345164190486, 'Mick
LaSalle'), (0.8

9340514744156474, 'Claudia Puig')]

>>>
```

De la observación de estos resultados vemos cuál es el más similar a nosotros, en este caso buscamos los más similares a:

'Toby': {'Snakes on a Plane': 4.5, 'You, Me and Dupree':1.0, 'Superman Returns': 4.0}}

Los más similares son:

'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,

'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,

'The Night Listener': 3.0},

Efectivamente vemos que Toby y Lisa Rose coinciden con distinta valoración en 'Snakes on a Plane',' Me and Dupree' y 'Superman Returns'

Recomendar ítems

Ahora se verá el modo de hacer recomendaciones. Nos basaremos en ver lo similar que son los ítems incluidos en el diccionario a un determinado item

```
C:\uoc\collective_intelligence\pearson\src>c:\python26\python
```

```
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit
(Intel)] on

win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import pearson

>>> pearson.getRecommendations(pearson.critics,'Toby')

<p>text

0.66284898036</p>

<p>

<p>text

(3.0, 'The Night Listener')</p>

<p>

<p>text

(3.0, 'The Night Listener')</p>

<p>

<p>text

(3.0, 'Lady in the Water')</p>

<p>

[(3.0, 'The Night Listener'), (3.0, 'Lady in the Water')]

>>>
```

Aplicación práctica del Algoritmo de Search y Ranking

En este apartado crearemos una máquina de búsqueda con el fin de encontrar el documento más importante en el que aparece una palabra.

Se describirán cuáles son los pasos necesarios y que se obtiene en cada paso. Como ya se ha mencionado anteriormente una máquina de búsqueda necesita recolectar documentos.

En esta máquina de búsqueda utilizaremos los datos de del.icio.us correspondientes a la URL http://www.delicious.com/popular/python.

Primero obtendremos todos los links de esta url entrando recursivamente en sus links. Se adjuntan partes de los resultados obtenidos porque este paso tarda algunos minutos. Hay que mencionar que es necesario crear y

almacenar los resultados en la base de datos para manipulaciones posteriores.

```
>>> import searchengine

>>> crawler=searchengine.crawler('searchindex.db')

>>> crawler.createindextables()

>>> pages=\

... ['http://www.delicious.com/popular/python']

>>> crawler.crawl(pages)

Indexing  http://www.delicious.com/popular/python

Indexing  http://blog.madoro.org/mn/90

Indexing
http://www.delicious.com/save?new=1&url=http%3A%2F%2Fpackages.python.org%2FGit
Python%2F0.3.1%2Findex.html&title=GitPython%20Documentation%20%E2%80%94%20GitP
ython%20v0.3.1%20documentation&type=URL&jump=http://www.delicious.com%2Fpopula
r%2Fpython&key=lrHycjxVqTdfQlPTFFSAUVpWqn4-&via=popular

Indexing  http://www.delicious.com/url/29a58e81d84b002ada7ce6e08c17f5ab

Indexing  http://www.delicious.com/popular/super

Indexing  http://www.lastday.jp/2011/05/30/python-tutorial

Indexing  http://www.michielovertoom.com/python/pastebin-abused/

Indexing  http://jeetworks.org/node/99

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Fwww.python.org%2Fdev%2Fpeps%2F
pep-
0008%2F&hash=eba24ab3aa904703f7fb51f1913a6093&type=URL&key=AYi6WNyzWSl1Gew8GLz
i455idbA-&title=PEP%208%20--
%20Style%20Guide%20for%20Python%20Code&jump=http://www.delicious.com%2Fpopular
%2Fpython

Indexing  http://www.delicious.com/bcachet

Indexing  http://www.delicious.com/fjgcdv

Indexing  http://libcloud.apache.org/

Indexing  http://www.delicious.com/url/094c47ed69ab3602453ed9c5e6ca40b4

Indexing  http://www.delicious.com/lluis_ballester

Indexing  http://aminsblog.wordpress.com/2011/05/29/n-queen-problem-python-2-
6-5-vs-pypy-1-5-0/
```

```
Indexing  http://www.delicious.com/recent/

Indexing  http://docs.notmyidea.org/alexis/pelican/

Indexing  http://www.delicious.com/url/c5d6286e805e211df254bfbd1b031af8

Indexing  http://www.delicious.com/popular/niho

Indexing  http://www.delicious.com/popular/html

Indexing  http://blog.delicious.com/

Indexing
http://www.delicious.com/save?new=1&url=http%3A%2F%2Flibcloud.apache.org%2F&ti
tle=libcloud%20python%20library%20-
%20libcloud%20is%20a%20standard%20client%20library%20for%20many%20popular%20cl
oud%20providers%2C%20written%20in%20python%20and%20java&type=URL&jump=http://w
ww.delicious.com%2Fpopular%2Fpython&key=lrHycjxVqTdfQlPTFFSAUVpWqn4-
&via=popular

Indexing  http://www.delicious.com/pushou

Indexing  http://m.delicious.com/settings/switch/mobile

Indexing  http://www.delicious.com/groups

Indexing  http://www.delicious.com/modon5

Indexing  http://www.delicious.com/popular/django

Indexing  http://dabeaz.blogspot.com/2011/05/class-decorators-might-also-be-
super.html

Indexing
http://www.delicious.com/save?new=1&url=http%3A%2F%2Fwww.pythonchallenge.com%2
F&title=The%20Python%20Challenge&type=URL&jump=http://www.delicious.com%2Fpopu
lar%2Fpython&key=lrHycjxVqTdfQlPTFFSAUVpWqn4-&via=popular

Indexing  http://www.delicious.com/erandras

Indexing  http://www.delicious.com/tag/

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Fwww.michielovertoom.com%2Fpyth
on%2Fpastebin-
abused%2F&hash=8df0f93f03217c8ce44a412bcb4a80c8&type=URL&key=AYi6WNyzWSl1Gew8G
Lzi455idbA-
&title=Pastebin%20abused&jump=http://www.delicious.com%2Fpopular%2Fpython

Indexing
http://www.delicious.com/save?new=1&url=https%3A%2F%2Fgithub.com%2Fccarpenterg
%2Ftodolist%2Fwiki&title=Home%20-
%20GitHub&type=URL&jump=http://www.delicious.com%2Fpopular%2Fpython&key=lrHycj
xVqTdfQlPTFFSAUVpWqn4-&via=popular

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Fdabeaz.blogspot.com%2F2011%2F0
5%2Fclass-decorators-might-also-be-
super.html&hash=56489f3ecc3963cc21ac8d1eb89e5ae5&type=URL&key=AYi6WNyzWSl1Gew8
GLzi455idbA-
&title=Dabeaz%3A%20Class%20decorators%20might%20also%20be%20super%21&jump=http
://www.delicious.com%2Fpopular%2Fpython
```

```
Indexing  http://www.delicious.com/jproy

Indexing  http://www.delicious.com/popular/guidelines

Indexing  http://www.delicious.com/popular/projects

Indexing  http://www.delicious.com/help/tools

Indexing  http://www.delicious.com/popular/soft

Indexing  http://bellm.org/blog/2011/05/27/why-astronomers-should-program-in-
python/

Indexing  http://www.delicious.com/url/c50e8687e169ccda9c7e8b5b32843500

Indexing  http://www.delicious.com/url/47fd171fcff8708c83e7335dd831ae9c

Indexing  http://www.delicious.com/popular/code

Indexing  http://www.delicious.com/url/f4752e3a4d628a951448e94de5b490bc

Indexing  http://www.delicious.com/drawkbox

Indexing  http://www.delicious.com/popular/open_source

Indexing  http://www.delicious.com/popular/problems

Indexing  http://info.yahoo.com/relevantads/

Indexing  http://www.delicious.com/popular/Algorithm

Indexing  http://www.delicious.com/popular/backbone

Indexing  http://www.pythonchallenge.com/

Indexing  http://www.delicious.com/popular/pypy

Indexing  https://github.com/ccarpenterg/todolist/wiki

Indexing  http://www.delicious.com/popular/download

Indexing  http://www.delicious.com/url/9a2645e44cbf87fdd68bad891618b8a3

Indexing  http://www.delicious.com/help/terms

Indexing  http://www.delicious.com/popular/blog

Indexing  http://www.delicious.com/user/

Indexing  http://www.aosabook.org/en/packaging.html

Indexing  http://docs.python-requests.org/en/latest/index.html

Indexing  http://www.delicious.com/popular/puzzles

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Flearnpythonthehardway.org%2Fin
dex&hash=ab31087e9621ac64cc2c83f1b512a2eb&type=URL&key=AYi6WNyzWSl1Gew8GLzi455
idbA-
&title=Learn%20Python%20The%20Hard%20Way%3A%20Learn%20Python%20The%20Hard%20Wa
y&jump=http://www.delicious.com%2Fpopular%2Fpython

Indexing  http://www.delicious.com/popular/webdev
```

```
Indexing  http://www.delicious.com/popular/python?removetag=python

Indexing  http://www.delicious.com/popular/git

Indexing  http://www.delicious.com/popular/djangodash2010

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Fblog.madoro.org%2Fmn%2F90&hash
=81ab234e24b046e43119adcbc2d610d0&type=URL&key=AYi6WNyzWSl1Gew8GLzi455idbA-
&title=Google%20App%20Engine%20%2F%20Python%20%E4%B8%8A%E3%81%A7%E3%81%AE%E9%9
6%8B%E7%99%BA%E3%81%A7%E6%9C%80%E5%88%9D%E3%81%8B%E3%82%89%E7%9F%A5%E3%81%A3%E
3%81%A6%E3%82%8C%E3%81%B0%E3%82%88%E3%81%8B%E3%81%A3%E3%81%9F%E3%80%81%E3%81%A
3%E3%81%A6%E3%81%93%E3%81%A8%E3%82%92%E3%81%84%E3%81%8F%E3%81%A4%E3%81%8B%20-
%20Masatomo%20Nakano%20Blog&jump=http://www.delicious.com%2Fpopular%2Fpython

Indexing  http://learnpythonthehardway.org/index

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Frhettinger.wordpress.com%2F201
1%2F05%2F26%2Fsuper-considered-
super%2F&hash=cdcb6fc595e19ed6b5a00869be5f3603&type=URL&key=AYi6WNyzWSl1Gew8GL
zi455idbA-
&title=Python%E2%80%99s%20super%28%29%20considered%20super%21%20%C2%AB%20Deep%
20Thoughts%20by%20Raymond%20Hettinger&jump=http://www.delicious.com%2Fpopular%
2Fpython

Indexing
http://www.delicious.com/share?url=http%3A%2F%2Fjeetworks.org%2Fnode%2F99&hash
=9a2645e44cbf87fdd68bad891618b8a3&type=URL&key=AYi6WNyzWSl1Gew8GLzi455idbA-
&title=Quick%20and%20Easy%20Debugging%20in%20Python%20%7C%20Jeet%20Sukumaran&j
ump=http://www.delicious.com%2Fpopular%2Fpython

Indexing  http://www.delicious.com/popular/

Indexing  http://www.delicious.com/tag/python

Indexing  http://www.delicious.com/popular/googleappengine

Indexing  http://info.yahoo.com/copyright/details.html
```

Consulta de los resultados en la base de datos

Se ejecutarán ahora algunas consultas sobre las tablas de la base de datos creada anteriormente para almacenar los resultados del algoritmo de búsqueda.

En la table urllist se almacena la lista de url que se han indexado(obtenido en el paso anterior). Se mostrarán las 5 primeras url

```
C:\uoc\collective_intelligence\searchengine\src>c:\python26\pyth
on
```

```
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32
bit (Intel)] on

win32

Type "help", "copyright", "credits" or "license" for more
information.

>>> import searchengine

>>> crawler=searchengine.crawler('searchindex.db')


>>> [row for row in crawler.con.execute(

... 'select * from urllist where rowid >=1 and rowid<=5')]

[(u'http://www.delicious.com/popular/python',),
(u'https://secure.delicious.com/

register',), (u'http://www.delicious.com/help/whatsnew',),
(u'http://www.delicio

us.com/help/learn',), (u'http://www.delicious.com/help',)]

>>>
```

Se visualizarán cuáles son las palabras almacenadas en la tabla wordlist. Esta tabla almacena las palabras encontradas.

```
>>> [row for row in crawler.con.execute(

... 'select * from wordlist where rowid>=1 and rowid<=5')]

[(u'doctype',), (u'html',), (u'public',), (u'w3c',), (u'dtd',)]

>>>
```

Veamos ahora los links entre las distintas url

```
>>> [row for row in crawler.con.execute(

... 'select * from link where rowid >=1 and rowid <= 5')]

[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6)]

>>>
```

Se buscarán ahora los id de las palabras almacenadas para la url número uno veamos los id de las palabras que se han almacenado

```
>>> [row for row in crawler.con.execute(
... 'select * from linkwords where linkid = 1')]
[(50, 1), (51, 1)]
```

Estos id de palabras se corresponden a las palabras

```
>>> [row for row in crawler.con.execute(
... 'select * from wordlist where rowid in (50,51)')]
[(u'join',), (u'now',)]
>>>
```

Ahora se mirará a qué url corresponden los id de url 1 y 2

```
>>> [row for row in crawler.con.execute(
... 'select * from urllist where rowid in (1,2)')]
[(u'http://www.delicious.com/popular/python',), (u'https://secure.delicious.com/
register',)]
>>>
```

Y por último la localización de las palabras en las url. Esto se ve en la tabla wordlocation

```
>>> [row for row in crawler.con.execute(
... 'select * from wordlocation where urlid in (1,2) and rowid
>=1 and rowid <=5
')]
[(1, 1, 0), (1, 2, 1), (1, 3, 2), (1, 4, 3), (1, 5, 4)]
>>>
```

```
>>> [row for row in crawler.con.execute(
... 'select * from wordlist where rowid in (1,2,3,4,5)')]
```

[(u'doctype',), (u'html',), (u'public',), (u'w3c',), (u'dtd',)]

>>>

Calcular pesos de las páginas recuperadas

Una vez que ya tenemos las url, las palabras y las localizaciones de la palabras en las url , podemos buscar una determinada palabra. En este caso buscaremos la palabra Python.

En la lista siguiente obtendremos una lista de las url ordenadas según unos pesos que nos indican cuanta importancia tiene la palabra en el documento

>>> e=searchengine.searcher('searchindex.db')

>>> e.getmatchrows('python')

>>> e.query('python')

```
3.009174     http://www.aosabook.org/en/packaging.html

2.513307     http://www.delicious.com/url/026a0ac2d52af30f131db65a80c21ca7

2.513307     http://www.delicious.com/url/eba24ab3aa904703f7fb51f1913a6093

2.438926     http://www.delicious.com/url/ab31087e9621ac64cc2c83f1b512a2eb

2.422397     http://www.delicious.com/url/cdcb6fc595e19ed6b5a00869be5f3603

2.405869     http://bellm.org/blog/2011/05/27/why-astronomers-should-program-
in-python/

2.397604     http://www.python.org/dev/peps/pep-0008/

2.331488     http://www.delicious.com/url/81ab234e24b046e43119adcbc2d610d0

2.331488     http://www.genbetadev.com/herramientas/herramientas-
imprescindibles-para-un-desarrollador-de-python

2.323224     http://feeds.delicious.com/v2/rss/popular/python?count=15
```

Establecer la importancia de la página

Ahora veremos la importancia de una página sobre las demás, es decir, la probabilidad que tiene una persona de llegar a una determinada página. Este proceso se basa en iteraciones. Después de cada iteración el algoritmo PageRank para cada página obtiene el valor más cercano al valor inicial PageRank, este valor se ha establecido en 1.0. También se inicializarán el número de iteraciones en 20.

```
>>> crawler=searchengine.crawler('searchindex.db')

>>> crawler.calculatepagerank()

Iteration 0

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Iteration 6

Iteration 7

Iteration 8

Iteration 9

Iteration 10

Iteration 11

Iteration 12

Iteration 13

Iteration 14

Iteration 15

Iteration 16

Iteration 17

Iteration 18

Iteration 19
```

Después de estas 20 iteraciones se obtienen los siguientes resultados,. Vemos que se ha alcanzado el valor PageRank inicial de 1.

```
>>> cur=crawler.con.execute('select * from pagerank order by score desc')
>>> for i in range(3): print cur.next()
...
(1, 1.0)
(2, 1.0)
(3, 1.0)
```

Para obtener el nombre de las url a las que más fácilmente llegará el usuario, se ejecutará lo siguiente:

```
>>> e.geturlname(1)
u'http://www.delicious.com/popular/python'

>>> e.geturlname(2)
u'https://secure.delicious.com/register'

>>> e.geturlname(3)
u'http://www.delicious.com/help/whatsnew'

>>>
```

Aplicación práctica del algoritmo de aprendizaje

Creación de base de datos y generación de un nodo oculto

```
C:\uoc\collective_intelligence\nn\src>c:\python26\python
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)] o
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nn
>>> mynet=nn.searchnet('nn.db')
>>> mynet.deletetables()
>>> mynet.maketables()
>>> wWorld,WRiver,wBank=101,102,103
>>> uWorldBank,uRiver,uEarth=201,202,203
>>> mynet.generatehiddennode([wWorld,wBank],[uWorldBank,uRiver,uEarth])
>>> for c in mynet.con.execute('select * from wordhidden'): print c
...
(101, 1, 0.5)
(103, 1, 0.5)
>>> for c in mynet.con.execute('select * from hiddenurl'): print c
...
(1, 201, 0.10000000000000001)
(1, 202, 0.10000000000000001)
(1, 203, 0.10000000000000001)
```

Alimentación de la red

```
>>> mynet.getresult([wWorld,wBank],[uWorldBank,uRiver,uEarth])

[0.076012508375416149, 0.076012508375416149, 0.076012508375416149]
```

Training Test

Como conclusión, en el siguiente ejemplo vemos como funciona el aprendizaje de la red. Primero fijamos los pesos. Vemos que para la URL World Bank da mejor puntuación que para River URL.

La red no sólo ha aprendido que URLs están relacionadas con qué queries, sino que ha aprendido que palabras son importantes en una determinada query. Esto no se puede alcanzar con la simple query-URL correlación

```
>>> mynet.trainquery([wWorld,wBank],[uWorldBank,uRiver,uEarth],uWorldBank)

>>> mynet.getresult([wWorld,wBank],[uWorldBank,uRiver,uEarth])

[0.33506324671253318, 0.055127057492088002, 0.055127057492088002]

C:\uoc\collective_intelligence\nn\src>c:\python26\python

Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)] on

win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import nn

>>> mynet=nn.searchnet('nn.db')

>>> mynet.deletetables()

>>> mynet.maketables()


>>> wWorld,wRiver,wBank=101,102,103

>>> uWorldBank,uRiver,uEarth=201,202,203

>>> allurls=[uWorldBank,uRiver,uEarth]

>>> for i in range(3):

...     mynet.trainquery([wWorld,wBank],allurls,uWorldBank)

...     mynet.trainquery([wRiver,wBank],allurls,uRiver)
```

```
...      mynet.trainquery([wWorld],allurls,uEarth)

...

>>> mynet.getresult([wWorld,wBank],allurls)

[0.52302176714224557, 0.26745323560082235, 0.48117924291762332]

>>> mynet.getresult([wRiver,wBank],allurls)

[0.15232293236570446, 0.57594750530570937, -0.29647136772475513]

>>> mynet.getresult([wBank],allurls)

[0.31967653022886289, 0.42866766884209107, -0.073545137568642394]

>>>
```

Integración algoritmo de recomendación y del.icio.us

Como aplicación práctica del algoritmo de recomendación se establece una integración con datos obtenidos de del.icio.us . Para esta integración se utilizaran la librerías

```
import pydelicious
import xml.dom.minidom
```

El proceso es el siguiente:

Para una determinada palabra clave, en este caso python, se buscarán todas las entradas en del.icio.us utilizando la instrucción

```
rec=pydelicious.get_popular(tag='python')
```

Devuelve un xml con las entradas solicitadas.

Para tratar este xml es necesario obtener el dom. Para ello utilizamos la librería xml.dom.minidom

```
dom = xml.dom.minidom.parseString(rec)
```

Dada la gran cantidad de datos que se pueden obtener en este proceso, se va a limitar la extracción de datos a 8 creadores.

Con la instrucción:

```
us1 = dom.getElementsByTagName("dc:creator")[y1]
```

Se obtendrán los 8 primeros usuarios que han introducido sus preferencias en del.icio.us .

Para cada uno de ellos se obtendrán sus posts

```
  postsusers = get_userposts(getText(us1.childNodes))
```

Se mirarán los títulos que contengan la palabra python. Se guardará en un array, el usuario, el título y un peso que en este caso se generará aleatoriamente con un valor entre 0 y 1.

Una vez que tenemos este array se pueden aplicar las funciones de **topMatches** y **getRecomendations**. Con estos datos no obtendremos recomendaciones, pero esto no quiere decir que con otros datos no se obtengan.

La función que ejecuta este algoritmo es:

def prueba1():

Ejemplo

```
rec <?xml version="1.0" encoding="UTF-8"?>

<rdf:RDF                                xmlns="http://purl.org/rss/1.0/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:cc="http://web.resource.org/cc/"
xmlns:syn="http://purl.org/rss/1.0/modules/syndication/"
xmlns:admin="http://webns.net/mvcb/">
```

```
  <channel rdf:about="http://feeds.delicious.com">

    <title>Popular python Bookmarks on Delicious</title>

    <link>http://www.delicious.com/popular/python</link>

    <description>the latest popular bookmarks</description>

    <items>

      <rdf:Seq>

        <rdf:li
rdf:resource="http://j2labs.tumblr.com/post/4369280869/some-black-
magic-python-for-n00bs"/>

        <rdf:li rdf:resource="http://babbledrive.appspot.com/"/>

        <rdf:li
rdf:resource="http://learnpythonthehardway.org/index"/>

        <rdf:li rdf:resource="http://ziutek.github.com/web_bench/"/>

        <rdf:li rdf:resource="http://diveintopython.org/index.html"/>

        <rdf:li rdf:resource="http://tornadogists.org/"/>

        <rdf:li      rdf:resource="http://niche-canada.org/programming-
historian"/>

        <rdf:li rdf:resource="http://www.datasciencetoolkit.org/"/>

        <rdf:li rdf:resource="http://graphite.wikidot.com/"/>

        <rdf:li
rdf:resource="http://www.pythonanywhere.com/pythonanywhere/"/>

        <rdf:li
rdf:resource="http://www.readwriteweb.com/hack/2011/03/python-is-an-
increasingly-popu.php"/>

      </rdf:Seq>

    </items>

  </channel>

  <item       rdf:about="http://j2labs.tumblr.com/post/4369280869/some-
black-magic-python-for-n00bs">

    <title>J2 Labs - Some Black Magic Python for n00bs</title>

    <dc:date>2011-04-05T21:17:53Z</dc:date>

    <link>http://j2labs.tumblr.com/post/4369280869/some-black-magic-
python-for-n00bs</link>

    <dc:creator>bobertlo</dc:creator>
```

```
    <dc:subject>programming python</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/programming"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://babbledrive.appspot.com/">
    <title>Welcome - babbledrive</title>
    <dc:date>2011-04-09T21:42:14Z</dc:date>
    <link>http://babbledrive.appspot.com/</link>
    <dc:creator>timewaves</dc:creator>
    <dc:subject>python documentation reference programming
search</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/documentation"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/reference"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/programming"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/search"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://learnpythonthehardway.org/index">
    <title>Learn Python The Hard Way: Learn Python The Hard
Way</title>
    <dc:date>2010-04-27T15:32:08Z</dc:date>
```

```
    <link>http://learnpythonthehardway.org/index</link>
    <dc:creator>berberich</dc:creator>
    <dc:subject>learn tutorial learning python</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/learn"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/tutorial"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/learning"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://ziutek.github.com/web_bench/">
    <title>Benchmarking Go and Python Web servers</title>
    <dc:date>2011-02-23T16:05:29Z</dc:date>
    <link>http://ziutek.github.com/web_bench/</link>
    <dc:creator>a13x.net</dc:creator>
    <dc:subject>python benchmark Go</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/benchmark"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/Go"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://diveintopython.org/index.html">
    <title>Dive Into Python</title>
    <dc:date>2003-01-10T20:14:18Z</dc:date>
    <link>http://diveintopython.org/index.html</link>
```

```
    <dc:creator>metaquasi</dc:creator>
    <dc:subject>python book tutorial</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/book"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/tutorial"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://tornadogists.org/">
    <title>Tornado Gists - explaining Tornado on gist at a
time</title>
    <dc:date>2011-04-07T06:31:23Z</dc:date>
    <link>http://tornadogists.org/</link>
    <dc:creator>unbotan</dc:creator>
    <dc:subject>tornado python</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/tornado"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://niche-canada.org/programming-historian">
    <title>The Programming Historian | NiCHE</title>
    <dc:date>2009-08-20T21:12:04Z</dc:date>
    <link>http://niche-canada.org/programming-historian</link>
    <dc:creator>dbietila</dc:creator>
    <dc:subject>python programming introduction</dc:subject>
    <taxo:topics>
```

```
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/programming"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/introduction"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://www.datasciencetoolkit.org/">
    <title>Data Science Toolkit</title>
    <dc:date>2011-03-23T19:29:32Z</dc:date>
    <link>http://www.datasciencetoolkit.org/</link>
    <dc:creator>SFdragon</dc:creator>
    <dc:subject>data datamining science api opensource tools
python</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/data"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/datamining"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/science"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/api"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/opensource"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/tools"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://graphite.wikidot.com/">
    <title>Graphite - Graphing library for Python</title>
    <dc:date>2008-06-27T21:47:45Z</dc:date>
```

```
    <link>http://graphite.wikidot.com/</link>
    <dc:creator>jaywgraves</dc:creator>
    <dc:subject>python dataviz</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/dataviz"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item rdf:about="http://www.pythonanywhere.com/pythonanywhere/">
    <title>pythonanywhere</title>
    <dc:date>2011-04-07T11:28:01Z</dc:date>
    <link>http://www.pythonanywhere.com/pythonanywhere/</link>
    <dc:creator>ihuston</dc:creator>
    <dc:subject>android:bookmarks python programming
cloud</dc:subject>
    <taxo:topics>
      <rdf:Bag>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/android%3Abookmarks"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>
        <rdf:li
rdf:resource="http://www.delicious.com/tag/programming"/>
        <rdf:li rdf:resource="http://www.delicious.com/tag/cloud"/>
      </rdf:Bag>
    </taxo:topics>
  </item>
  <item  rdf:about="http://www.readwriteweb.com/hack/2011/03/python-is-
an-increasingly-popu.php">
    <title>6 Free E-Books on Learning to Program with Python</title>
    <dc:date>2011-03-26T01:17:31Z</dc:date>
```

```
    <link>http://www.readwriteweb.com/hack/2011/03/python-is-an-
increasingly-popu.php</link>

    <dc:creator>mythicflow</dc:creator>

    <dc:subject>python programming ebooks tutorial</dc:subject>

    <taxo:topics>

      <rdf:Bag>

        <rdf:li rdf:resource="http://www.delicious.com/tag/python"/>

        <rdf:li
rdf:resource="http://www.delicious.com/tag/programming"/>

        <rdf:li rdf:resource="http://www.delicious.com/tag/ebooks"/>

        <rdf:li rdf:resource="http://www.delicious.com/tag/tutorial"/>

      </rdf:Bag>

    </taxo:topics>

  </item>

</rdf:RDF>
```

```
(u'timewaves', '=>', {u'Babbledrive python quick search':
0.60920251843587425, u'Python Module of the Week - Python Module of
the Week': 0.0, u"Generating PDFs on App Engine Python, and
introducing Mapvelopes - Nick's Blog": 0.0, u'python black magic
intro': 0.0, u'pyreport: generate reports out of python scripts':
0.0})

(u'unbotan', '=>', {u'Python Module of the Week - Python Module of the
Week': 0.06885657015087765, u'Babbledrive python quick search': 0.0,
u"Generating PDFs on App Engine Python, and introducing Mapvelopes -
Nick's Blog": 0.90053977382276451, u'python black magic intro': 0.0,
u'pyreport: generate reports out of python scripts':
0.20932120841296598})

(u'bobertlo', '=>', {u'Babbledrive python quick search': 0.0, u'Python
Module of the Week - Python Module of the Week': 0.0, u"Generating
PDFs on App Engine Python, and introducing Mapvelopes - Nick's Blog":
0.0, u'python black magic intro': 0.36082288232950543, u'pyreport:
generate reports out of python scripts': 0.0})

<p>text

-0.345591134276</p>

<p>

<p>text
```

```
-0.25</p>

<p>

<p>text

(-0.25000000000000006,              u'bobertlo')(-0.25000000000000006,
u'bobertlo')</p>

<p>text

(-0.34559113427603477,               u'unbotan')(-0.34559113427603477,
u'unbotan')</p>

<p>

<p>text

-0.345591134276</p>

<p>

<p>text

-0.25</p>

<p>

None  NO HAY RECOMENDACIONES
```

LIBRERÍAS PYTHON

Trataremos a continuación las librerías de python utilizadas en la implementación de los algoritmos.

En la clase crawler utilizada en la implementación del algoritmo de Search y Ranking se utilizará el API Beautiful Soup. Esta librería construye una representación de la estructura de las páginas web. Otra librería utilizada es urllib2 que permite bajas páginas.

Para crear los esquemas de base de datos utilizadas en los algoritmos se utilizará SQLite.

- Instalación de Beautiful Soup

Beautiful Soup es un parseador de Python para documentos HTML y XML. La página home de esta librería es http://www.crummy.com/software/BeautifulSoup. Se debe descargar el fichero BeautifulSoup.py y situarlo en el directorio de trabajo o en el directorio Python/Lib

- Instalación de pysqlite

pysqlite es un interface Python para SQLite embebido. La página home para pysqlite es http://www.initd.org/tracker/pysqlite/wiki/pysqlite

Para instalarlo en Windows se descarga el instalador binario para Windows y se ejecuta

- pydelicious

pydelicious es una librería para recuperar datos desde el sitio del.icio.us. Su sitio es http://oreilly.com /catalog/9780596529321. Se descarga y se coloca en el path de las librerías Python

- Otras librerías

Otras librerías usadas son html 1.13. El sitio es http://pypi.python.org/pypi/html#downloads

CONCLUSIONES

Tras el estudio de este proyecto se pueden establecer las siguientes conclusiones:

- La importancia de los algoritmos de recomendación, algoritmo de búsqueda , ranking y aprendizaje. Con la expansión de internet y los buscadores aparecen nuevas maneras de explotar la información almacenada en internet. Una de las aplicaciones es la búsqueda de información, establecer perfiles
- la recomendación de estos perfiles. Otro algoritmo es importante es el aprendizaje. Este algoritmo se basa en redes neuronales.

- La implementación de estos algoritmos en Python ofrece beneficios en el tratamiento de gran cantidad de texto.

- A la vista de la aplicación práctica adjuntada vemos la utilidad de estos algoritmos cuando se quieren clasificar páginas web para hacer recomendaciones. Se clasifican los links y se les asocian pesos.

ÍNDICE

BIBLIOGRAFÍA

http://arxiv.org/ftp/cs/papers/0308/0308031.pdf. (s.f.).

http://en.wikipedia.org/wiki/Recommendation_system. (s.f.).

http://ilpubs.stanford.edu:8090/422/. (s.f.).

http://nakkaya.com/2009/11/13/pearson-correlation-score/. (s.f.).

http://scripts.mit.edu/~cci/HCI/index.php?title=Main_Page. (s.f.).

http://scripts.mit.edu/~cci/HCI/index.php?title=Main_Page. (s.f.).

http://scripts.mit.edu/~cci/HCI/index.php?title=Main_Page. (s.f.).

http://scripts.mit.edu/~cci/HCI/index.php?title=Main_Page. (s.f.).

http://www.acm.org/conferences/sac/sac2010/T7.pdf. (s.f.).

http://www.acm.org/conferences/sac/sac2010/T7.pdf. (s.f.).

http://www.springer.com/computer/ai/book/978-0-387-85819-7. (s.f.).

http://www.webopedia.com/TERM/S/search_engine.html. (s.f.).

Segaran, T. (2007). *Programming Collective Intelligence.* O'REILLY.

www.mit.edu/~georg/papers/lecture6.ppt . (s.f.).

www.umich.edu/~exphysio/MVS250/PearsonCorr.doc - Similar. (s.f.).