CAIM - Session 3

User Relevance Feedback

Santiago Arxé i Carbona

Bryan Leonardo Salto Salao

8/11/2021

Implementación

Para implementar el sistema de User Relevance Feedback, se siguió el esquema de la regla de Rocchio presentado en el enunciado. Eso sí, se incorporaron algunas modificaciones.

Pseudocódigo

- Reordenar la query
    - Si una query contenía los operadores *fuzzy* y *boost*, era mejor colocar el de *boost* al final: `nyc^2~3 → nyc~3^2`
- Durante *nrounds*:
    - Computar la query y obtener los *k* documentos más relevantes
    - Computar la representación *Tf-Idf* de estos *k* documentos
    - Obtener un diccionario con la relevancia media de los términos encontrados en los *k* documentos
    - Pasar la *query* actual a diccionario (más adelante se explica por qué)
    - Aplicar Rocchio
    - "Podar" la *query* resultante para quedarse con los *R* términos más relevantes
    - Pasar la *query* de diccionario a forma "normal"
- Computar la query y obtener los *k* documentos más relevantes

Explicaciones y puntos importantes

El primer punto a comentar es por qué se reordenan los términos para tener el *boost operator* al final. Este proceso se lleva a cabo para poder extraer la importancia que le da el usuario a un término de manera más sencilla. Además, así se puede efectuar la *query* por primera vez manteniendo el término escrito por el usuario y sin tener que deshacerse del operador *fuzzy*.
Otro aspecto importante a comentar es la utilización de diccionarios. Aunque parezca estúpido pasar de *query* en modo lista a diccionario y luego de vuelta a lista, tiene una explicación. Esto se hace para, por ejemplo, poder calcular la relevancia *Tf-Idf* media de cada palabra en los documentos de manera mucho más eficiente. Al operar con todas las palabras que aparecen en cada documento, es necesario realizar una implementación óptima, y los diccionarios así lo hacen.
La manera de hacerlo es:

- Crear un diccionario vacío `dict`
- Recorrer cada documento término a término:
- Por cada término en el documento, buscarlo en `dict` y crear una entrada si no existe o sumar el valor encontrado si ya existe.

Al hacer esto, recorrer cada diccionario tiene complejidad O(n_i), y buscar cada uno de sus términos en `dict` tiene complejidad O(1), con lo que buscar todos sus términos acaba teniendo complejidad O(n_i). Aplicando esto a los *k* documentos, cada ejecución de este proceso tendrá una complejidad de $O(\sum_i^k n_i)$.

Si, por contra, se utilizasen listas, en cada iteración del proceso habría que buscar los términos de una lista en la lista ya computada y, aunque se hiciese de manera óptima y con las listas ordenadas, el coste siempre sería mayor (ya fuese este el coste de ordenar las listas $O(\sum_i^k n_i \cdot log(n_i))$ o el coste de recorrerlas todas sin utilizar tablas de *hash* como hacen los diccionarios.

Esta optimización, por tanto, es importante tanto a la hora de calcular la relevancia *Tf-Idf* media de cada palabra en los documentos como a la hora de ejecutar *Rocchio* y obtener la nueva *query*.

Una de las dudas que aparecieron al implementar el código fue el papel de la *R* cada vez que se ejecuta *Rocchio*. En la documentación, no quedaba del todo claro si la nueva *query* debía contener los términos de la *query* anterior **más** los *R* términos más relevantes encontrados por *Tf-Idf* (creciendo así la *query* en *R* términos en cada iteración) o si debía simplemente contener *R* términos. Después de experimentar un poco, se concluyó que aumentar la *query* en *R* términos en cada iteración, hacía que esta creciese de manera desproporcionada y en ocasiones acababa devolviendo resultados vacíos. De esta forma, se decidió efectuar la “poda” quedando solamente *R* términos cada vez.

Experimentación

Se han realizado 4 experimentos para estudiar la regla de *Rocchio*, cada uno enfocado en un parámetro en concreto, manteniendo los otros iguales para poder sacar conclusiones más claras. Además al momento de indexar los documentos se han utilizado *--token letter* y *--filter lowercase stop asciifolding snowball* (utilizada en sesión 2) que ha permitido ver casos más curiosos, como el experimento 3. Los documentos indexados han sido los pertenecientes al dataset *20_newsgroups* de la práctica 1. Cabe destacar que los dos vectores con los pesos están ordenados de mayor a menor peso y que corresponden a la segunda y última *queries*.

Experimento 1:

nrounds = 5	K = 5	R = 5	query inicial = murder

Tabla 1: Configuración base del experimento 1.

- Caso 1 : **α:β = 1:1**

['murder^1.39', 'moral^0.17', 'object^0.11', 'system^0.10', 'innoc^0.10']

.....

['murder^2.14', 'moral^0.94', 'object^0.75', 'system^0.67', 'innoc^0.35']

1. PATH=/alt.atheism/0000008 2. PATH=/alt.atheism/0000025 3. PATH=/alt.atheism/00000393.
4. PATH=/alt.atheism/0000332 5. PATH=/talk/talk.politics.misc/0018746

- Caso 2: **α:β = 0.8:0.2**

['murder^0.88', 'moral^0.03', 'object^0.02', 'system^0.01', 'innoc^0.02']

.....

['murder^0.57', 'moral^0.14', 'object^0.12', 'system^0.10', 'innoc^0.05']

1. PATH=/alt.atheism/0000008 2. PATH=/alt.atheism/0000039 3. PATH=/alt.atheism/0000025
4. PATH=/alt.atheism/0000332 5.PATH=/talk/talk.politics.misc/0018746

- Caso 3: **α:β = 0.2:0.8**

['murder^0.51', 'moral^0.13', 'object^0.089', 'system^0.08', 'innoc^0.07']

.....

['moral^0.33', 'murder^0.25', 'object^0.25', 'system^0.20', 'goal^0.13']

1. PATH=/alt.atheism/0000008 2. PATH=/alt.atheism/0000039 3. PATH= /alt.atheism/0000332
2. PATH=/alt.atheism/0000025 5. PATH=/alt.atheism/0000789

Con estos tres casos, puede empezar a verse la influencia de **α** y de ***β***. Se aprecia como los valores de más peso de la última *query* es diferente a los valores correspondientes en los dos primeros casos. También se ve como el resultado es diferente en los tres casos. En los dos primeros, son los mismos documentos en diferente orden, al haber en la *query* final los mismos términos pero con diferentes pesos. En el tercer caso, el documento situado en el directorio *politics* desaparece.
El caso 1 y 2 són los más parecidos, tiene sentido dado que los valores con más tiempo siempre tienen más peso de lo normal. Aún así, aumentando ***β*** se añade mas variedad al resultado. Y aumentando **α** se hace crecer la precisión.

Experimento 2

α:***β*** = 0.7:0.6	K = 5	R = 5	query inicial = author vancouver

Tabla 2: Configuración base del experimento 2.

- Caso 1: **nrounds = 3**

1. PATH=/talk.politics.guns/0016000 2. PATH=/talk.politics.guns/0016022

3. PATH=/talk.politics.guns/0015998 4. PATH=/talk.politics.guns/0016076 3. PATH=/talk.politics.guns/001664

- Caso 2: **nrounds = 10**

1. PATH=/talk.politics.guns/0015998 2. PATH=/talk.politics.guns/0016000
3. PATH=/talk.politics.guns/0016022 4. PATH=/talk.politics.guns/0016076 5. PATH=/talk.politics.guns/0016646

- Caso 3: **nrounds = 10, α:*β* = 0.2:0.8**

.....

['homicid^0.36', 'gun^0.31', 'handgun^0.26', 'vancouv^0.23', 'seattl^0.21']

1. PATH=/talk.politics.guns/0015998 2. PATH=/talk.politics.guns/0016000
3. PATH=/talk.politics.guns/0016076 4. PATH=/talk.politics.guns/0016646

En este caso puede verse como los documentos devueltos en los dos primeros casos son exactamente los mismos con mínimos cambios respecto al orden. Por lo que se deduce que aplicar *Rocchio* un gran número de veces no producirá grandes cambios; la mejora se producirá en las primeras iteraciones.
Del mismo modo, se probó la diferencia con valores de *beta* más altos para ver si más rondas provocaban más dispersión. En este caso, se ve como los términos más importantes también acaban convergiendo, desapareciendo incluso ciertos términos de la *query* inicial.

Experimento 3:

α:*β* = 1:1	nrounds = 10	R = 5	query inicial = author vancouver

Tabla 3: Configuración base del experimento 3.

- Caso 1: **k=2**

[author, vancouver, homicid, vancouv, edgeway]

.....

[author, vancouver, homicid, vancouv, edgeway]

- Caso 2: **k=10**

[author, vancouver, homicid, vancouv, seattl]

.....

[homicid, author, vancouver, vancouv, seattl]

En este experimento, parecido al 1, puede verse como en el caso 2 hay más variedad para la iteración de *queries*. En el caso 1 no ha habido ninguna variación, mientras que en el 2 sí. Es más, si se añadiese un poco más de peso a la componente ***β*** *vancouv* estaría por delante de *vancouver*(comprobado). Y es normal, porque al haber más documentos hay más palabras que no

estén relacionadas con la *query* principal, y en cada iteración cogen más peso si aparecen más. Es el caso de *author* i *vancouver*. *vancouv* és un caso especial, porque es la raíz de *vancouver*, por lo que saldrá más veces o igual. En este caso por el método de indexación más (*snowball*).

Experimento 4:

α:*β* = 1:1	nrounds = 10	K = 5	query inicial = vancouv author

Tabla 4: Configuración base del experimento 3.

- Caso 1: **R = 4** → Retorna 5 documentos, máximo según la k establecida.
- Caso 2: **R = 8** → Retorna 4 documentos.
- Caso 3: **R = 20** → No retorna ningún documento.

Este experimento muestra como al principio es muy fácil encontrar 5 documentos que satisfagan la *query*, pero al aumentar la *R* se vuelve imposible de satisfacer. Aumenta el *recall* y es imposible encontrar algo tan general.

Valoración

Los parámetros *alfa* y *beta* son los principales responsables de la progresión del algoritmo. Como puede deducirse de la regla de *Rocchio*, el parámetro *alfa* da “importancia” a los términos ya presentes en la *query* anterior, mientras que *beta* pone en valor la relevancia de los términos encontrados en los *k* documentos (parámetro *k* comentado más adelante).

Los experimentos efectuados, han confirmado esta hipótesis, viendo que, como menor fuese la proporción *alfa:beta*, más se diferenciaba la *query* final de la inicial, permitiendo más variabilidad en los resultados. Esto lleva a pensar directamente que la *beta* es el parámetro que afecta de manera más directa al *recall*, detectando los términos más importantes de los documentos más importantes. Por otro lado, el parámetro *alfa* hace que la importancia de las *queries* se vaya “acumulando” en cierto modo, haciendo que sea muy difícil desplazar de su lugar los términos obtenidos en las iteraciones anteriores (casi siempre se mantienen los términos obtenidos en la primera iteración). De este modo, *alfa* afecta de manera directa a la precisión, haciendo que con cada iteración se obtengan resultados cada vez más acotados.

En el caso del parámetro *nrounds*, su importancia no parece ser tan directa. No obstante, para proporciones bajas de *alfa:beta*, es necesario un valor relativamente alto de *nrounds* para conseguir que la búsqueda converja, encontrando los términos más relevantes en la colección.

El parámetro *k*, por su lado, se ocupa de decidir cuántos documentos se tendrán en cuenta para aplicar la regla de *Rocchio*. De manera similar al parámetro *nrounds*, cuánto mayor sea su valor, más variabilidad podrá entrar en el cálculo del resultado. De este modo, se encontrarán términos y documentos más relevantes en la colección, con el riesgo de obtener una *query* demasiado concreta que no obtenga ningún documento como respuesta.

Por último, el parámetro *R* afecta de manera directa al número de términos que se utilizarán para realizar las *queries* en cada iteración. Este es un parámetro algo complicado de gestionar, ya que valores demasiado altos generan *queries* demasiado complicadas y hacen que se acaben obteniendo resultados vacíos. Valores bajos, por contra, hacen que la *query* final sea más concreta (y más similar a la original), con lo que se consigue un resultado más amplio. Idealmente, los valores intermedios son los que permiten conseguir *queries* más depuradas y resultados más relevantes.