Universidad de la Habana, MATCOM
Sistemas de Recuperación de Información (2022)

Informe de Entrega Final

Samuel David Suárez Rodríguez C512
Enmanuel Verdesia Suárez C511
Gabriel Fernando Martín Fernández C511

1. Descripción General

1.1. Resumen

Se desarrolló un Sistema de Recuperación de Información (SRI) empleando dos modelos vectoriales para la recuperación de datos, uno de ellos el empleando una bolsa de palabras el otro usando doc2vec. Dado un conjunto de varios documentos de diversos temas la aplicación realiza una búsqueda que obtiene resultados relevantes y acordes a una consulta presentada por el usuario. Además se emplearon métodos para clasificar los documentos y extraer sus características más relevantes usando algoritmos supervisados para la clasificación y selección de características.

1.2. Estructura del proyecto

El proyecto presenta tres componentes principales: el backend o componente lógica de la aplicación, junto con la API, y el frontend en forma de un servicio web.

Está desarrollado en GitHub bajo la organización Gologle. Ahí pueden ser encontrados los dos repositorios principales. Además están hospedados el cliente del sistema y la API en internet.

- Web: https://gologle.vercel.app
- API: https://gologle-api.herokuapp.com/

Detalles de implementación

2. Procesamiento de los sets de datos

Para cada set de datos empleado fue implementado un parser para extraer su información. Este proceso no se pudo realizar de forma común pues cada set contenía los documentos de forma particular. Una vez *parseados* los sets de datos se puede delimitar e identificar cada documento que contiene de forma única por un ID y así guardar el texto de los documentos en una base de datos SQLite. Dicha base de datos facilita obtener un menor tiempo de respuesta ante un pedido de parte del cliente.

Los sets de datos con los que se trabajó fueron Cranfield (1400 documentos), Newsgroups (18828 documentos) y Reuters (10788 documentos). Se contaba con consultas y resultados relevantes para Cranfield. Para Newsgroups se tenían clasificaciones exclusivas de los documentos con cierto carácter jerárquico que se pueden considerar de cierta forma como clústers. En Reuters se contaban con clasificaciones no exclusivas para los documentos. Estos últimos sets fueron empleados para implemtentar algoritmos de clasificación de documentos.

3. Modelo Vectorial (TF-IDF) [1]

Este modelo esta basado en la representación de los documentos mediante *bag-of-words*. Es decir, se transforma cada documento a una vector de tamaño fijo que solo contiene información de la cantidad de veces que aparece cada palabra por componente. Este modelo a pesar de ser efectivo tiene como debilidad que se pierde la información respecto al orden de las palabras, además, como no aprende el significado de las palabras, la distancia entre vectores no siempre representa una diferencia en el significado.

3.1. Preprocesamiento

Dado que ha sido implementado el modelo vectorial basado en TF-IDF (del inglés, *term frequency* e *inverse document frequency*), haciendo uso de la biblioteca `sklearn` nos ajustamos a la forma

que esta requiere para representar los documentos. Dicha biblioteca se usó para *tokenizar* los documentos y extraer como *features* los términos que estos contienen. El texto de los documentos es llevado a minúscula, además los términos de un simple carácter son eliminados. De igual forma fueron eliminadas las *stopwords* del idioma inglés, conjunto de palabras comunes que no son de relevancia, lo cual permite al modelo enfocarse en palabras más relevantes. Finalmente se realizo lematización sobre el texto usando la biblioteca `nltk`. Así se obtiene cada documento como un vector de términos (palabras) con una dimensión determinada.

3.2. Construcción del índice

A partir del conjunto de documentos se obtiene una matriz esparcida que representa la cantidad de veces que aparece el término i en el documento j. Sobre esta matriz nos apoyamos para calcular la frecuencia de documentos inversa (idf) para cada término.

$$idf_i = \log \frac{N}{n_i}$$

n_i: cantidad de documentos en los que aparece el término i.

Luego haciendo uso de la clase `TfidfTransformer` que se encuentra en el módulo `sklearn.feature_extraction.text` calculamos los pesos para cada término en cada documento, de acuerdo a la expresión:

$$w_{ij} = tf_{ij} * idf_i$$

donde t_{ij} es la frecuencia normalizada del término i en el document j,

$$tf_{ij} = \frac{freq_{ij}}{max_l freq_{lj}}$$

Hecho esto, se tiene la matriz esparcida de los pesos, la cual es guardada en la base de datos para agilizar el tiempo de respuesta.

La base de datos creada a partir de los documentos, los términos, la idf para cada término y los pesos es considerada el índice del sistema para satisfacer las consultas de la forma más rápida posible.

3.3. Recuperación de Información

Cuando es recibida una consulta q, esta es procesada de forma similar a como se procesa un documento. Se calcula el peso para cada término (w_{iq}), mediante la siguiente expresión

$$w_{iq} = (\alpha + (1 - \alpha)\frac{freq_{lq}}{max_l freq_{lq}}) * idf_i$$

donde α es un valor de suavizado que permite amortiguar la contribución de la frecuencia de los términos, se le asignó un valor de 0.5.

La similitud entre el vector obtenido y los documentos de la base de datos es calculada usando como valor de referencia el coseno del ángulo entre estos. Los documentos con mayor similitud son dados como resultado de forma ordenada.

4. Modelo Vectorial (Doc2Vec)

Para desarrollar el modelo nos apoyamos en la biblioteca `gensim`, que ofrece una implementación del modelo `Doc2Vec`, que se encuentra en el módulo `gensim.models.doc2vec`. Esta esta basada en *Paragraph Vector* [2], un algoritmo no supervisado que es capaz de aprender *features* de tamaño fijo de textos de tamaño variable, como oraciones, párrafos y documentos.

4.1. Preprocesamiento

De igual forma al modelo basado en TF-IDF, aquí se llevo el texto de los documentos a minúscula, fueron removidas las *stopwords*, palabras de una sola letra y se realizó la lematización. El texto obtenido para cada documento fue empleado para crear una instancia de `TaggedDocument`, del módulo `gensim.models.doc2vec`.

4.2. Entrenamiento

Con el conjunto de `TaggedDocument`, cada uno con el ID del documento relacionado, se realizó el entrenamiento del modelo `Doc2Vec`.

Este se realizo con vectores de tamaño 50, por 200 epochs. Se probaron vectores de mayor tamaño y número de epochs mayor pero no mejoraron los resultados prácticos.

EL modelo entrenado es persistido para en futuros inicios del sistema solo tener que cargarlo.

4.3. Construcción del índice

Dado que este modelo realiza la inferencia a través del modelo, solo es necesario guardar en el índice los documentos que van a ser devueltos al usuario.

4.4. Feedback

En la base da datos (índice) también fue creada una tabla para registrar el feedback de los usuarios. Para cada documento obtenido de una consulta se puede dar una valoración de si fue relevante o no (1 o -1). Esta valoración es almacenada en el índice para en futuras consultas similares usarla y mejorar la cantidad de documentos relevantes devueltos.

Para calcular el feedback fue empleado el algoritmo de Rocchio [3], el cual tiene como premisa hallar el centroide de los puntos relevantes y tratar de alejarse de los no relevantes.

$$q_m = \alpha q_0 + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} d_j - \frac{\gamma}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j$$

donde:

D_r: conjunto conocido de documentos relevantes

D_{nr}: conjunto conocido de documentos no relevantes

α, β, γ: pesos para consulta, documentos relevantes y documentos no relevantes respectivamente

Se tomaron como valores de referencia $\alpha = 0{,}97$, $\beta = 0{,}4$ y $\gamma = 0{,}15$.

4.5. Recuperación de Información

Cuando se recibe una query antes de hallar la similitud con los documentos del modelo se le aplica Rocchio de acuerdo a lo descrito

anteriormente. El vector resultante es el empleado para calcular la similitud con los documentos del modelo. Esta es calculada usando el coseno del ángulo entre los vectores [4], de forma similar al primer modelo.

Los resultados son devueltos en un ranking, ordenados por relevancia, nuevas actualizaciones por feedback son procesadas y añadidas a la base de datos.

5. Evaluación de los modelos

5.1. Métricas objetivas

Dado que se tenía las consultas y sus relevancias para el set de Cranfield, se condujo una evaluación de los modelos con la métrica F. Los resultados no fueron buenos al promediar los resultados para las aproximadamente 150 consultas pues muchas no estaban devolviendo resultados relevantes.

Motivo de esto es que algunas consultas tenían pocos resultados relevantes en el set de prueba. En el caso del modelo basado en TF-IDF también las consultas al tener tantos términos el tiempo de respuesta se hacía grande. Doc2Vec, mostró un tiempo de respuesta rápido pero lo antes mencionado sobre la poca cantidad de relevantes en el set de prueba, sumado a la alta especificidad del dataset que causa que documentos relevantes y no relevantes no sean ubicados cerca en el espacio no ofrece resultados satisfactorios.

Para consultas más simples Doc2Vec presenta mejores resultados que el vectorial al detectar la semántica latente debido a su representación.

Para algunas consultas de Cranfield se obtienen los siguientes resultados:

TF-IDF

Query: what similarity laws must be obeyed when constructingaeroelastic models of heated high speed aircraft . Relevant: [12, 13, 14, 15, 29, 30, 31, 37, 51, 52, 56, 57, 66, 95, 102, 142, 184, 185, 195, 378, 462, 497, 858, 859, 875, 876, 879, 880] Retrieved: [51, 12, 13, 56, 141, 102, 14, 78, 57, 100, 25, 29, 142, 5, 158, 180, 101, 114, 120, 52,

27, 41, 84, 160, 45, 47, 62, 95, 144, 123, 33, 36, 50, 92, 145, 135, 49, 6, 98, 21, 156, 181, 35, 69, 55, 82, 75, 159, 24, 37, 154, 81, 119, 172, 28, 124, 30, 166, 121, 77, 53, 23, 129, 20, 89, 80, 168, 110, 176, 104, 165, 169, 157, 164, 140, 66, 38, 60, 40, 170, 174, 122, 22, 85] f1 score: 0.41666666666666663 precision: 0.16666666666666666 recall: 0.5

```
Query: what are the structural and aeroelastic problems
associated with flight of high speed aircraft .
Relevant: [12, 14, 15, 51, 52, 102, 184, 202, 285, 380, 390,
391, 442, 497, 643, 658, 746, 856, 857, 858, 859, 864, 877,
948]
Retrieved: [12, 51, 141, 14, 100, 75, 47, 78, 172, 92, 114,
101, 52, 77, 76, 41, 33, 142, 102, 70, 29, 36, 2, 46, 156,
69, 58, 181, 93, 82, 11, 60, 129, 25, 85, 116, 176, 38, 110,
40, 166, 16, 108, 27, 21, 24, 122, 107, 80, 83, 162, 139, 44,
73, 160, 87, 97, 143, 163, 67, 169, 81, 149, 99, 103, 37, 155,
19, 179, 49, 98, 4]
f1 score: 0.1736111111111111
precision: 0.06944444444444445    recall: 0.20833333333333334
```

Doc2Vec

```
Query: what similarity laws must be obeyed when constructing
aeroelastic models of heated high speed aircraft .
Relevant: [12, 13, 14, 15, 29, 30, 31, 37, 51, 52, 56, 57, 66,
95, 102, 142, 184, 185, 195, 378, 462, 497, 858, 859, 875, 876,
879, 880]
Retrieved: [367, 835, 184, 1037, 486, 518, 1207, 790, 1008, 858,
875, 78, 948, 51, 585, 195, 643, 1052, 286, 425, 91, 13, 1292,
1027, 886, 31, 303, 1400, 532, 395, 158, 915, 969, 859, 1350,
102, 350, 1022, 860, 579, 277, 462, 80, 284, 1032, 955, 911,
1277, 236, 1268, 766, 342, 1173, 582, 951, 970, 781, 1016, 865,
824, 520, 1017, 914, 353, 833, 5, 1362, 1177, 84, 418, 6, 1073,
762, 15, 1170, 368, 1335, 151, 1220, 1185, 1030, 87, 500, 944]
f1 score: 0.32738095238095244
precision: 0.13095238095238096    recall: 0.39285714285714285

Query: what are the structural and aeroelastic problems
associated with flight of high speed aircraft .
```

```
Relevant: [12, 14, 15, 51, 52, 102, 184, 202, 285, 380, 390, 391,
442, 497, 643, 658, 746, 856, 857, 858, 859, 864, 877, 948]
Retrieved: [13, 46, 12, 1294, 835, 756, 833, 955, 1178, 1052,
585, 436, 91, 817, 1362, 724, 1068, 726, 368, 486, 51, 658, 1013,
639, 950, 286, 831, 640, 875, 886, 448, 715, 723, 1122, 367, 911,
1015, 746, 220, 1037, 532, 506, 1134, 846, 642, 828, 92, 598,
355, 1369, 100, 102, 1148, 399, 728, 281, 518, 137, 1012, 285, 75,
1177, 1174, 865, 480, 195, 1205, 1379, 1042, 113, 509, 957]
f1 score: 0.20833333333333331
precision: 0.08333333333333333    recall: 0.25

Query: can a criterion be developed to show empirically the
validity of flow solutions for chemically reacting gas mixtures
based on the simplifying assumption of instantaneous local
chemical equilibrium .
Relevant: [20, 48, 58, 122, 196, 197, 354, 360, 999, 1005, 1112]
Retrieved: [309, 1255, 507, 319, 873, 1123, 1275, 1244, 669, 456,
299, 348, 493, 1301, 21, 531, 629, 625, 1026, 539, 1224, 17, 436,
449, 361, 236, 1037, 703, 976, 437, 716, 618, 784]
f1 score: 0 precision: 0.0    recall: 0.0
```

5.2. Métricas subjetivas

El sistema ofrece una interfaz web que permite el acceso a múltiples usuarios de forma concurrente. El tiempo demorado en dar respuesta a una consulta es mostrado en la cabecera de los resultados junto con la cantidad de documentos recuperados.

Las consultas del modelo basado en Doc2Vec son considerablemente más rápidas que el modelo TF-IDF. El tiempo de respuesta en TF-IDF aumenta considerablemente con el tamaño de la consulta, no así con el otro modelo.

En base a esto el usuario puede escoger que modelo utilizar para realizar sus consultas y sobre que set de datos realizarlo.

También se ofrece la funcionalidad de dar retroalimentación. Cada modelo se puede beneficiar del feedback proveído para responder las consultas a todos los usuarios.

6. Ventajas y Desventajas del Sistema

El Sistema de Recuperación de Información presentado presenta un conjunto de ventajas que permiten una buena aproximación a la búsqueda de resultados en distintos datasets, pues permite utilizar dos modelos con características diferentes en dependencia del tipo de búsqueda que se quiera realizar, en caso de búsquedas para palabras clave exactas o semiexactas (lemmatize) se puede utilizar el modelo vectorial, y para consultas en donde no importa mucho el matching exacto de las palabras sino su significado semántico se puede utilizar el modelo doc2vec, esto permite obtener resultados diferentes y abarcar documentos relevantes de distinto tipo. Otra ventaja es el soporte de retroalimentación para el modelo doc2vec, pudiendo usarla para la mejora de las respuestas, lo que permite que el SRI se adapte a las necesidades del usuario. Además, como detalles técnicos, proponemos una arquitectura escalable, a través de una API, que permite tener varias instancias del servidor corriendo de manera horizontal.

Como desventajas tenemos la incapacidad de procesar consultas demasiado largas con el modelo doc2vec, ya que la representación espacial para vectores de varias palabras puede resultar en un vector de embeddings que no se encuentre cercano a ningún grupo de datos específicos en el espacio, esto pasa también al usar Rocchio para la retroalimentación, pues la sobrecarga de retroalimentación, sobre todo negativa y hacia documentos no muy relacionados en la representación de embeddings, provoca que no exista un centroide bien definido y por tanto el nuevo vector consulta tenga documentos irrelevantes cercanos.

7. Clasificación

La clasificación de documentos puede ser de utilidad para mejorar los resultados de las búsquedas o incluso simplemente para mostrar más información de los documentos retornados. Para los documentos de los datasets de Reuters y Newsgroups se implementaron 6 tipos distintos de clasificaciones y se evaluaron los resultados. Los clasificadores empleados fueron: Multinomial Naive Bayes, Logistic Regression, K Neighbors Classifier (con k = 5), Decision Tree Classifier

y Random Forest Classifier. Para los dos conjuntos de documentos se asumió que se tenía un problema de clasificación múltiple aunque en el dataset de Newsgroups no es el caso. Para evaluar los resultados se calcularon varias de las medidas del promedio f1-score, la micro, macro, teniendo en cuenta los pesos y el de las muestras. En vez de emplear todos los datos de los datasets se restringió la cantidad de documentos a alrededor de dos mil por cada juego de datos. Se obtuvieron los siguientes resultados:

```
NEWSGROUP

Multinomial Naive Bayes
(avg : f1-score)
micro avg: 0.6420189274447949
macro avg: 0.6229742698146286
weighted avg: 0.6272770879929649
samples avg: 0.5847037688536247

Logistic Regression
(avg : f1-score)
micro avg: 0.4841445427728614
macro avg: 0.4705289626711132
weighted avg: 0.47566567952073086
samples avg: 0.3318400139725788

K Neighbors Classifier
(avg : f1-score)
micro avg: 0.11060130403284231
macro avg: 0.10710245171380031
weighted avg: 0.10866395001592159
samples avg: 0.059994760282944724

Decision Tree Classifier
(avg : f1-score)
micro avg: 0.5923186344238976
macro avg: 0.5793549946253852
weighted avg: 0.5872601387474548
samples avg: 0.49708384794840127
```

Random Forest Classifier
(avg : f1-score)
micro avg: 0.2637063887630267
macro avg: 0.24092971766745883
weighted avg: 0.2447020478086619
samples avg: 0.15238843769103133

documentos de entrenamiento: 953
documentos de prueba: 952
número de clases: 20

REUTERS

Multinomial Naive Bayes
(avg : f1-score)
micro avg: 0.684402332361516
macro avg: 0.3646946771616699
weighted avg: 0.6643401399876091
samples avg: 0.7387600920321509

Logistic Regression
(avg : f1-score)
micro avg: 0.7393617021276596
macro avg: 0.33672957157668737
weighted avg: 0.6915224193979003
samples avg: 0.7123448828948828

K Neighbors Classifier
(avg : f1-score)
micro avg: 0.5998089780324738
macro avg: 0.20311362021332247
weighted avg: 0.4962913170860935
samples avg: 0.5927269841269841

Decision Tree Classifier

```
(avg : f1-score)
micro avg: 0.7359617682198326
macro avg: 0.436536897803069
weighted avg: 0.7068268351350262
samples avg: 0.7160689824895197

Random Forest Classifier
(avg : f1-score)
micro avg: 0.6653164556962026
macro avg: 0.21820108986022607
samples avg: 0.6127802308802309


documentos de entrenamiento: 1000
documentos de prueba: 1000
número de clases: 66
```

Para ambos casos el Naive Bayes multinomial obtiene buenos resultados en comparación con el resto teniendo en cuenta la relativa sencillez del algoritmo. El decision Tree Classifier obtiene buenos resultados en ambos casos, mejores que el Naive Bayes en el set de Reuters y similares a este en el de Newsgroup. Otro algoritmo con relativamente buenos resultados fue el de regresión logística, más en el caso del set de Reuters. La similitud entre los resultados de la puntuación media micro y macro de f1 en el set de Newsgroups se debe a que las clasificaciones en dicho set son únicas. Además en este set los resultados son en general peores que en el de Reuters porque los clasificadores dados son realmente complejos y con cierta jerarquía algo que no se tuvo en cuenta al aplicar las clasificaciones.

Recomendaciones

Existen varios acercamientos a los SRI que podrían ser de utilidad para mejorar la calidad de los resultados obtenidos por los modelos propuestos y su rendimiento. Una posibilidad es agregar un sistema de recomendaciones basado en proponer documentos con características similares a los de los resultados de búsqueda. En este sentido se pueden recomendar aquellos documentos más relevantes

que sean miembros de los clústeres a los que pertenecen los resultados. Estas ideas se basan en emplear algoritmos de clasificación y agrupamiento para proponer posibles resultados que escapen a los obtenidos por los modelos implementados. Además para mejorar la velocidad de búsqueda es posible añadir filtros basados en las características de los documentos para así restringir el campo de búsqueda. Esta idea puede mejorar de manera considerable el rendimiento a cambio de cierta pérdida de precisión que dependerá de la calidad de la clasificación obtenida y de la elección de filtro del usuario.

Referencias

1. "Conferencia de Sistemas de Recuperación de Información - 2022." , June 2022.
2. Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.
3. "Rocchio algorithm." `https://en.wikipedia.org/wiki/Rocchio_algorithm`, June 2022.
4. "Method to calculate similarity." `https://radimrehurek.com/gensim/models/keyedvectors.html#gensim.models.keyedvectors.KeyedVectors.most_similar`, June 2022.