Proyecto de Sistemas de Información.

Rainel Fernández Abreu C312, Lázaro Alejandro Castro Arango C311

Facultad de Matemática y Computación
Universidad de la Habana

Resumen:

Se implementaron tres modelos de Recuperación de Información programados en `Python`, con una interfaz visual diseñada en `Django`. Cada modelo consta de 4 etapas fundamentales: Procesamiento de texto, Procesamiento de consulta, Evaluación y por último la Interfaz visual, donde el primero y último de estos 4 es común para cada modelo. Con el objetivo de evaluar los modelos se implementaron los criterios de evaluación pertinentes.

Requerimientos de software.

- Python 3.7
- sklearn
- re
- ir_datasets
- nltk 3.0
- django 3.2
- numpy

Procesamiento de texto:

Para poder implementar mecanismos de recuperación sobre una colección de documentos de texto es necesario obtener una representación de los mismos. Con el objetivo de lograr dicha representación, seguimos un conjunto de pasos para dejar solo los términos de los textos que van hacer útil para la posterior recuperación de información, lo cual además, hace mucho más rápido el trabajo con cada modelo.

Haciendo uso de la librería **re** de python para el trabajo con patrones en strings, reemplazamos todos las posibles contracciones que se utilizan en el idioma inglés por su respectiva forma

correcta(Ejemplo: you're--you are, aren't-- are not), luego utilizando **nltk** eliminamos todas las stopwords(palabras que no aportan significado) y signos de puntuaciones. Para un mejor **stemming** utilizamos la clase **SnowballStemmer** de **nltk**, la cual nos reduce en gran medida la cantidad de términos con los que vamos a trabajar. Devolviendo una lista de tokens, para su posterior utilización.

Sería demasiado costoso tener que hacer todo este proceso sobre los documentos de un *corpus* cada vez que se requiera recuperar información de algún documento. Para esto guardamos información en diccionarios utilizando la librería **pickle** que nos permite almacenar información en archivos de extensión *.txt* para luego cargarla y utilizar dicha información.

Procesamiento de consulta:

Al igual que se procesan cada uno de los documentos, las consultas necesitan ser procesadas para llevar las palabras a su raíz, chequear la colocación de parentesis, así como el exceso o falta de operadores booleanos (& , |, ~), en dependencia del modelo que se desee utilizar.

Modelos implementados:

El modelo vectorial fue el primero que elegimos pues nos parecío muy ventajoso poder realizar consultas en lenguaje natural. Este modelo requería calcular *tf* e *idf*, para eso dicha información se guardaba y decidimos reutilizarla en algún otro modelo, por ello decidimos implementar el Booleano extendido, pues era sencillo reutilizar los diccionarios que ya teníamos con dicha información. La decisión de implementar el Booleano como último modelo fue por la sencillez de comprender e implementar el mismo con la información que ya se tenía.

Modelo Vectorial

Representa documentos en lenguaje natural de una manera formal mediante el uso de vectores.En esta representación vectorial de documentos el éxito o fracaso se basa en la ponderación o peso de los términos. Aunque ha habido mucha investigación sobre técnicas de ponderación de términos, en realidad no hay un concenso sobre cuál método es el mejor.

Para implementar el modelo vectorial, se necesita información como el *tf* e *idf* de los términos de la query y los documentos. Esta información se carga de los diccionarios que se guardan en el procesamiento de texto. A partir de la información necesaria se crea un ranking utilizando la función de similitud del modelo. Dicho ranking de los documentos se ordena de forma descendente y se muestra al usuario los documentos ordenados por su ranking.

Modelo Booleano

En el modelo booleano los vectores se representan de forma binaria, para cada término si se encuentra en un documento este se representará con un 1, sino, con un 0. Para cada consulta en

el modelo, se chequea que la consulta esté bien escrita y luego se pasa a intentar recuperar los documentos. Como todos nuestros modelos, este utiliza los diccionarios ya cargados para comprobar de manera eficiente si un término pertenece o no a un documento. Por cada término de la consulta se verifica la pertenencia o no al documento $i-ésimo$, si aparece este término pasa a ser True y se evalúa la expresión final, si es 1 (True), el documento es recuperado.

Modelo Booleano Extendido

El modelo booleano extendido recibe las consultas de forma similar al modelo booleno. Utiliza para crear el ranking el *tf* e *idf*, pero la función similitud se calcula en dependencia de la operacion a realizar. La busqueda de términos en los documentos y consultas sucede de manera similar a los modelos anteriores.

Interfaz visual:

Para realizar la interfaz visual se utilizó **Django** pues creemos que la posibilidad que brinda la web de acceder al sistema desde cualquier dispositivo con internet es un gran plus. La primera vista consta de un selector con el tipo de modelo a utilizar, un campo para ingresar la consulta, un botón busqueda para mostrar los resultados y un botón que permite cargar el corpus de ser necesario. Más adelante hablaremos de como añadir elementos al corpus de forma correcta. La segunda vista consta de una tabla con los resultados. Podemos ver aquí el direcctorio donde se encuentra el documento recuperado y su puntuación en el ranking. Los resultados son ordenables, se muestran paginados y un filtro permite realizar búsquedas sobre ellos.

Evaluación del sistema:

Para evaluar nuestro sistema nos apoyamos principalmente en la libreria *ir_datasets* la cual brinda varios corpus y facilidad con las medidas de evaluación. Usamos los corpus:

- "vaswani" que contiene 11k de documentos y 93 querys
- "cord19/trec-covid/round1" que contiene 51k de documentos y 30 querys

Esta librería a través del método : `ir_measures.iter_calc(args)` nos permite calcular las métricas que se especifiquen en los argumentos del mismo.

En las gráficas que se encuentran en las carpetas que hay en el directorio junto a este informe se evidencia como se comportó la **Precisión**, **Recobrado** y **Medida F** para cada uno de nuestros modelos en los corpus antes mencionados.

Las siguientes tablas revelan el comportamiento de la media y la varianza en cada uno de los modelos para los corpus utilizados.

Media

"vaswani"	Vectorial	Booleano Extendido
Precisión	0.3466666666666667	0.1978494623655914
Recobrado	0.02933390313816676	0.08212731098046903
Medida F	0.012126194340642549	0.11433019087060156
"cord19/trec-covid/round1"		
Precisión	0.3419354838709677	0.2533333333333333
Recobrado	0.11944831099382808	0.01824840421628721
Medida F	0.01875265296544419	0.09908674742341403

Varianza

"vaswani"	Vectorial	Booleano Extendido
Precisión	0.09426292056885192	0.05375881604809805
Recobrado	0.02366470211464605	0.022492843258988378
Medida F	0.0004437543138790658	0.018151456680889348
"cord19/trec-covid/round1"		
Precisión	0.08515555555555555	0.06648888888888888
Recobrado	0.0009056070322238115	0.0003859698868320344
Medida F	3.975250416657034e-05	# 0.010123500097554698

Comparación entre los modelos implementados:

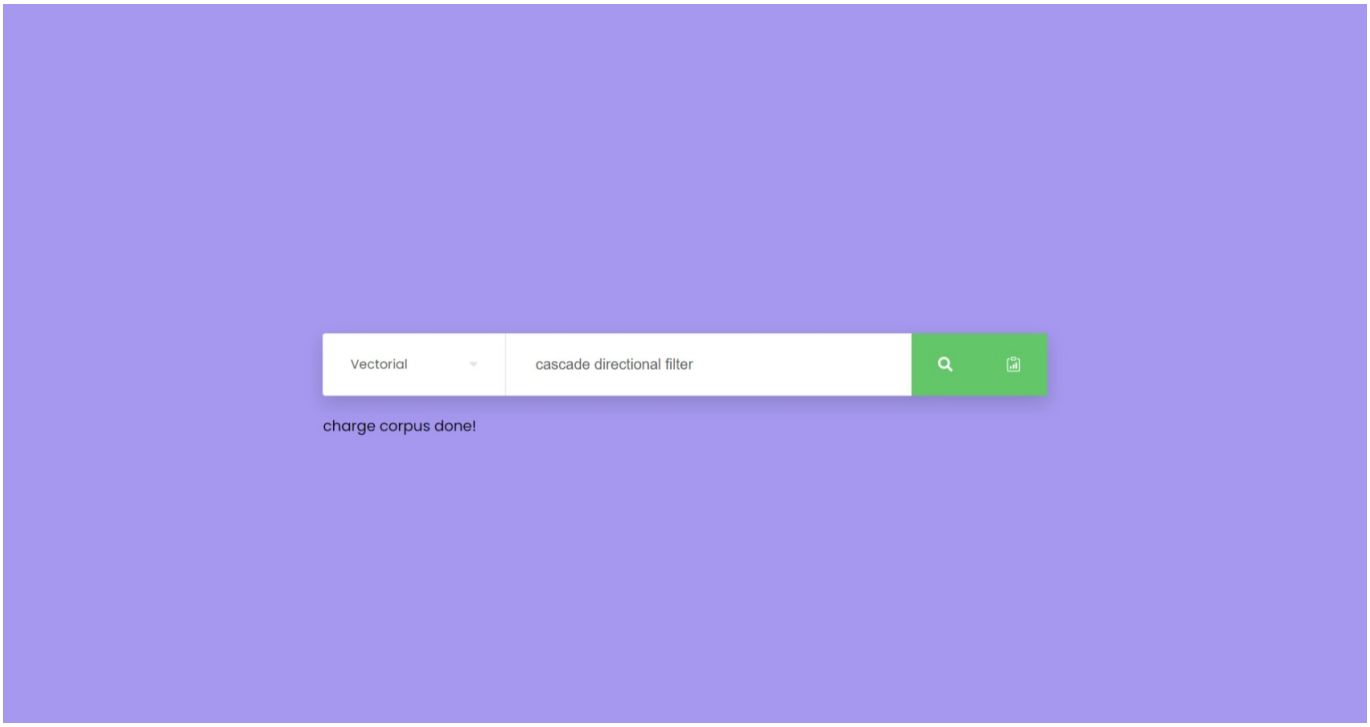
Criterios	Booleano	Vectorial	
Documentos	Vectores binarios	Vectores de pesos no binarios	Vectores de los pesos asociados a los términos de los documentos (binarios)
Consultas	Expresiones booleanas	Vectores de pesos no binarios	Vectores de los pesos asociados a los términos de consulta (binarios)
Framework	Teoría de conjuntos y algebra booleana	Espacio n-dimensional y operaciones entre vectores del álgebra lineal	Teoría de probabildades
Pesos	Binarios	Dados por tf * idf	Dados por idf
Similitud	sim = {0,1}	Varía de 0 a 1 dada por el coseno del ángulo	sim = $\frac{P(dj)}{\overline{P}(dj)}$
Dependencia entre términos	No	No	No
Correspondencia parcial entre documentos y consultas	No	Sí	Sí
Ranking	No	Sí	Sí

Consideraciones generales:

Las consultas tanto en el booleano como en el booleano extendido deben realizarse utilizando palabras que se saben tendrán peso(sustantivos y adjetivos). Una consulta como *of & computer* es invalida pues la palabra *of* no tiene peso en un texto por su clasificación semántica. Estas deben realizarse utilizando los símbolos &, |, ~. Al emplear paréntesis siempre se debe dejar un espacio antre el mismo y las palabras.

Para añadir elementos al corpus estos deben estar en formato .txt o texto sin formato como aparecen en 20 Newsgroups por ejemplo. Luego de añadir documentos es necesario volver a cargar el corpus en el botón correspondiente.

Ejemplos:



Mostrar 10 registros por página

Buscar:

Path	Score	View
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\9952.txt	1.0	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\19.txt	0.9999999999999998	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\11182.txt	0.8706901687737947	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\11310.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\424.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\4938.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\7111.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\7890.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\8447.txt	0.8706901687737945	Coming soon
C:/Users/lachy/Desktop/Nueva carpeta/static/Corpus\6768.txt	0.8648995024566963	Coming soon

Mostrando página 1 de 1,143

Anterior 1 2 3 4 5 … 1,143 Siguiente