UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

MANUAL DE USUARIO
SISTEMA DE RECUPERACIÓN DE INFORMACIÓN

Programado en Prolog.

Profesor: Roberto González
Alumno: Roberto Orellana

Fecha de Entrega: 10/11/2017

Santiago de Chile

10 - 11 – 2017

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCION

El siguiente documento presentará de forma explicativa y sencilla cómo utilizar la siguiente aplicación, dentro de lo que se le procede a explicar, se encuentran las siguientes interrogantes: ¿Cómo compilar la aplicación?, ¿Cómo utilizarla de manera apropiada?, ¿Cuáles son los posibles errores que se pueden encontrar durante la ejecución del problema?, ¿Qué funcionalidades cumple la aplicación?, etc. Es recomendable poseer un nivel de leguaje técnico capaz de comprender los conceptos que se mencionen, para poder hacer buen uso de esta aplicación.

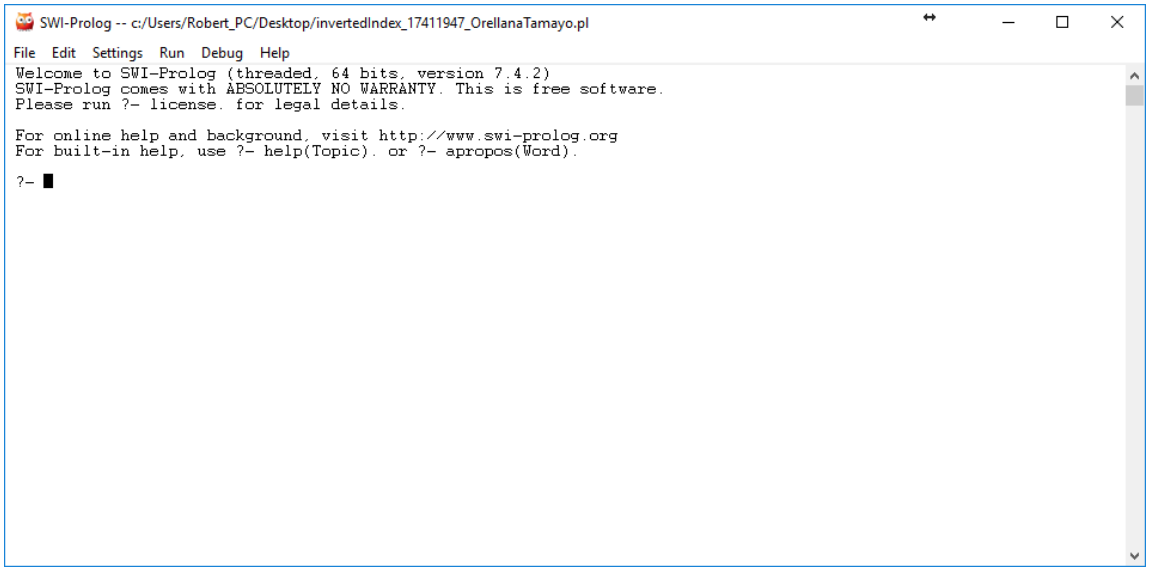
MANUAL DE USUARIO “BUSCADOR”

CAPÍTULO 2. INSTRUCCIONES DE USO

En el siguiente manual, se pretende explicar la manera de operar de las reglas generadas en el programa, además de ejemplos de cómo implementarlas.

Para su ejecución es necesaria la instalación de SWI-Prolog un intérprete de prolog, además de conocimientos básico-intermedio sobre programación.

Luego de instalar la aplicación debe abrir el archivo (invertedIndex_17411947_OrellanaTamayo.pl), que lo enviara a la siguiente ventana donde podrás ingresar las consultas.



Un resumen de las funciones que contiene el juego y que usted puede realizar.

1) Consultar por un término y documento
2) Consultar por una frase y documentos
3) Consultar por una frase y lista de documentos
4) Consultar lista re resultados y responder con lista de documentos y titulo
5) Consultar lista re resultados y cantidad de documentos
6) Consultar por un término y su frecuencia
7) Consultar por un término y responder con todos los documentos que NO contengan dicho termino

2.1 SingleTermQuery

singleTermQuery(Term, Result)

Esta función consulta por un término **Term** y por un **Result** que es un documento, si los dos valores son correctos devuelve "**true**" en caso contrario devuelve "**false**". **Term** es un término o una palabra que existe dentro de algún documento, **Result** es el documento que lo contiene, por ejemplo:

Term: corresponde al término o palabra. (vorticity)
Result: corresponde al documento. (2)

```
?- singleTermQuery(vorticity, 2).
true .
```

En Caso que desconozca el Resultado, podria realizar la consulta, ¿Qué documentos contienen la palabra vorticity?, y el programa le daria las respuesta.

```
?- singleTermQuery(vorticity, Result).
Result = 2 ;
Result = 4 .
```

2.2 BestMatch

bestMatch(Phrase, Result)

Esta función consulta por una lista de términos **Phrase** y por un **Result** que es un documento, si los dos valores son correctos devuelve "**true**" en caso contrario devuelve "**false**". **Phrase** es una lista de términos o palabras que existe dentro de algún documento, **Result** es el documento que lo contiene, por ejemplo:

Phrase: corresponde al término o palabra. ([karman, small, wing])
Result: corresponde al documento. (2)

```
?- bestMatch([karman,small,wing],2).
true .
```

En Caso que desconozca el Resultado, podria realizar la consulta, ¿Qué documentos contienen las palabras?, y el programa le daria las respuesta.

```
?- bestMatch([karman,small,wing],Result).
Result = 4 ;
Result = 2 ;
Result = 5 ;
Result = 1 .
```

2.3 ExactMatch

exactMatch(Phrase, Results)

Esta función consulta por una lista de términos **Phrase** y por **Result** que es una lista de documentos, si los dos valores son correctos devuelve "**true**" en caso contrario devuelve "**false**". **Phrase** es una lista de términos o palabras que existe dentro de algunos documentos, **Result** es la lista de documento que lo contiene, por ejemplo:

Phrase: corresponde al término o palabra. ([karman, small, wing])
Result: corresponde al documento. ([4, 2, 1])

```
?- exactMatch([karman,small,wing],[4,2,1]).
true .
```

En Caso que desconozca el Resultado, podria realizar la consulta, ¿Qué documentos contienen las palabras?, y el programa le daria las respuesta.

```
?- exactMatch([karman,small,wing],Result).
Result = [4, 2, 1] ;
Result = [4, 5, 1] .
```

2.4 Documents

documents(Results, Documents)

Esta función consulta por una lista de resultados **Result** y el programa responde con una lista de documento **Documents**, que contiene el ID y el título de cada documento, por ejemplo:

Result: corresponde al documento. ([1, 2, 3])

Como se desconoce el nombre de los titulos se pregunta, ¿Qué titulos contienen la lista documentos?, y el programa le daria las respuesta.

```
?- documents([1,2,3], Documents).
Documents = [1, "experimental investigation of the aerodynamics of a\nwing in a slipstream .
", 2, "simple shear flow past a flat plate in an incompressible fluid of small\nviscosity",
3, "the boundary layer in simple shear flow past a flat plate"].
```

2.5 NumDocuments

numDocuments(Results, Count)

Esta función consulta por una lista de documentos **Result** y por un **Count** que es la cantidad de documentos como resultado, si los dos valores son correctos devuelve "**true**" en caso contrario devuelve "**false**", por ejemplo:

Result: corresponde al documento. ([1, 3, 5])
Count: corresponde a la cantidad de documentos. (3).

```
?- numDocuments([1,3,5],3).
true.
```

En Caso que desconozca la cantidad, podria realizar la consulta, ¿Cuántos documentos contiene el Result?, y el programa le daria las respuesta.

```
?- numDocuments([1,3,5],Count).
Count = 3.
```

2.6 TermTotalCount

termTotalCount(Term,Frequency)

Esta función consulta por un término **Term** y por un **Frequency** que es una Frecuencia, si los dos valores son correctos devuelve "**true**" en caso contrario devuelve "**false**". **Term** es un término o una palabra que existe dentro de algún documento, **Frequency** es la cantidad de veces que se repite ese término en el documento, por ejemplo:

Term: corresponde al término o palabra. (wing)
Frequency: corresponde a la cantidad de veces que se repite en el documento. (4)

```
?- termTotalCount(wing, 4).
true .
```

En Caso que desconozca el Resultado, podria realizar la consulta, ¿Qué frecuencia tiene la palabra wing?, y el programa le daria las respuesta.

```
?- termTotalCount(wing,Frequency).
Frequency = 4 .
```

2.7 TermTotalCount

docsNotContain(Term, *Documents*)

Esta función consulta por un término **Term** y el programa responde con una lista de documento que NO contiene el termino consultado **Documents**, la lista contiene el ID y el título de cada documento, por ejemplo:

Term: corresponde al término o palabra. (vorticity)

Como se desconoce el nombre de los titulos se pregunta, ¿Qué titulos contienen la lista documentos?, y el programa le daria las respuesta.

```
?- docNotContain(vorticity, Documents).
Documents = [1, "experimental investigation of the aerodynamics of a\nwing in a slipstream .
", 3, "the boundary layer in simple shear flow past a flat plate", 5, "one-dimensional trans
ient heat conduction into a double-layer\nslab subjected to a linear heat input for a small
time\ninternal"].
```

CAPÍTULO 3. POSIBLES ERRORES

Se solicita que el usuario quien utilice la aplicación tenga conocimientos básicos de programación logica para utilizar cada una de las funciones, la única forma de que el programa buscador tenga errores, es ingresando datos que no correspondan a cada función.

La aplicación fue pensada y creada para que funcione con ciertos parámetros, por lo que, si estos no son recibidos, existe una gran posibilidad de que la aplicación falle.

Cualquier error por favor comunicarse al número 555-5555555 o al Mail

SeBusca@sin.soporte.cl