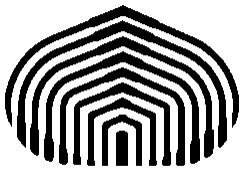


UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

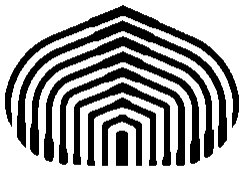
EXTRACCIÓN DE INFORMACIÓN FOCALIZADA BASADA EN RESPUESTAS INCOMPLETAS (FIE)

Por:
FRANCISCO RODRÍGUEZ DRUMOND

Realizado con la asesoría de:
PROF. SORAYA ABAD

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero de Computación

Sartenejas, Septiembre de 2012



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

ACTA FINAL PROYECTO DE GRADO

EXTRACCIÓN DE INFORMACIÓN FOCALIZADA BASADA EN RESPUESTAS INCOMPLETAS (FIE)

Presentado por:
FRANCISCO RODRÍGUEZ DRUMOND

Este Proyecto de Grado ha sido aprobado por el siguiente jurado examinador:

_____
PROF. 1

_____
PROF. 2

_____
PROF. 3

Sartenejas, X/X/12

Resumen

RESUMEN GOES HERE - To be done.

Acrónimos

FIE	Focalized Information Extraction	Extracción Focalizada de Información
DIA	Document Interrogation Architecture	Arquitectura de Interrogación de Documentos
BFR	Módulo de Búsqueda Focalizada de Información	Buscador Focalizado de Respuesta
DOI	Determinador del Origen de Incompletitud	Determinador del Origen de Incompletitud
EFI	Extractor Focalizado de Información	Extractor Focalizado de Información
NLP	Procesamiento del Lenguaje Natural	Natural Language Processing
BIE	Extracción Amplia de Información	Broad Information Extraction

DUDA: Que hago con los acrónimos que son sólo en español? Deberia poner los acronimos siempre en inglés?

Índice general

Índice de tablas

Índice de figuras

Introducción

Este trabajo consiste en el estudio de una variante del problema de enrutamiento de vehículos (VRP por las siglas de Vehicle Routing Problem) el cual se define como un problema de optimización combinatoria del área de logística y distribución de bienes.

Capítulo 1

Planteamiento del Problema

El problema a tratar en el presente trabajo de investigación consiste en diseñar, implementar y probar el proceso de Extracción Focalizada de Información *(FIE)* definida la Arquitectura para la Interrogación de Documentos *(DIA)* por Abad Mota [1]. DIA es una arquitectura que permite realizar consultas sobre un conjunto de documentos utilizando una representación basada en ontologías por medio de un lenguaje llamado *ODIL* (Ontology-based Document Interrogation Language). Para ello DIA funciona en 3 fases: preparación de datos, extracción de datos e interrogación de documento.

La primera fase consiste en la elección de un conjunto de documentos sobre el cual se realizará la interrogación y en la construcción de un extractor de información. El extractor de información puede realizarse utilizando mecanismos de procesamiento de lenguaje natural, de aprendizaje de máquina, estadísticos, entre otros posibles.

La segunda fase consiste en realizar una primera Extracción Amplia de Información *(BIE)* del conjunto de documentos utilizando el extractor ya construido. El objetivo final de esta fase es poblar una Base de Datos relacional.

En la tercera fase los usuarios, toda vez que se haya poblado la base de datos con datos, pueden realizar preguntas sobre el documento. En este punto se debe determinar la respuesta basado en la información presente en la Base de Datos y en un conjunto de documentos. El proceso de realizar una segunda extracción sobre documentos es el denominado Extracción Focalizada de Información.

El objetivo de la Extracción Focalizada de Información es encontrar información desconocida en la ontología realizando una extracción sobre un conjunto de documentos utilizando información presente en la ontología. El conjunto de documentos sobre el cual se hace la segunda extracción puede ser el mismo sobre el cual se hizo Extracción Amplia de Información o no. Una de las ventajas de esto es que se puede

explorar un conjunto de documentos de mayor tamaño al inicial, incluyendo por ejemplo la *WWW* (World Wide Web), sobre el cual es más complejo realizar una extracción inicial debido a su magnitud. La utilización de información conocida puede ser explotada por el extractor para simplificar considerablemente la búsqueda de la información faltante.

La Extracción Focalizada de Información supone a su veces tres fases similares a las que plantea DIA.

En primera instancia es necesario determinar el conjunto de documentos sobre el cual se realizará la FIE. El conjunto de documentos puede ser heterogéneo. Es decir, puede ocurrir que la información necesaria para encontrar información faltante sobre un concepto se encuentre en más de un tipo de documento. Puede pasar de igual manera que un tipo documento sirva para responder consultas sobre más de un concepto de la ontología y sobre más de una instancia. Puede ocurrir de igual manera que los documentos contengan información adicional que no está relacionada con los dominios de información de los conceptos de la ontología. En tal sentido, la primera labor necesaria para la FIE consiste en definir el conjunto de documentos y cómo acceder a ellos.

Duda: ¿Es necesario definir qué es ontología, qué es una instancia y un concepto? Algo más?

Una vez definido el conjunto de documentos, es necesario construir extractores de información que permitan encontrar la información faltante en la ontología. Igual que en el esquema general de DIA, el extractor de información puede realizarse utilizando mecanismos de procesamiento de lenguaje natural, de aprendizaje de máquina, etcétera. Lo relevante en esta fase de DIA es que el extractor de información debe construirse tomando en cuenta la información contenida en la ontología, el *contexto de extracción*. Dicho contexto se puede utilizar para facilitar la búsqueda de la información faltante.

En tercer lugar es necesario determinar los orígenes de incompletitud. Esto se debe realizar tomando en consideración dos cosas: la consulta realizada por el usuario y la información ya existente en la ontología. La incompletitud puede abarcar más que valores necesarios para responder las consultas: puede incluir información necesaria para calificar la consulta. Tomando en cuenta estos dos elementos se pueden realizar tantas búsquedas de información como sean necesarias.

La determinación de los origenes de incompletitud de información es un tema que será abordado posteriormente en profundidad. Sin embargo es importante precisar en este punto que pueden existir más de un tipo de fuentes de incompletitud:

1. Incompletitud por la ausencia de instancias (tuplas) en la ontología.

2. Incompletitud por el desconocimiento de valores de atributos.

3. Incompletitud por el desconocimiento de interrelaciones entre instancias.

Para este trabajo se ha definido el alcance de la Extracción Focalizada de Información en la determinación y resolución de incompletitud por desconocimiento de valores de atributos. El motivo de esto es que el descubrimiento de nuevas tuplas es un problema que no puede ser resuelto por Extracción Focalizada de Información, ya que la misma supone la existencia de información en la ontología sobre todas las instancias involucradas en una consulta. Por otro lado, si bien el descubrimiento de interrelaciones entre instancias puede modelarse como el descubrimiento de los valores de las claves foráneas que definen cada interrelación, se ha determinado que esta tarea es más compleja y puede involucrar tareas de extracción más sofisticadas que salen del alcance de este proyecto.

Se asume, por lo tanto, que todas las instancias concernientes a los dominios sobre los cuales se hace la búsqueda están contenidas en la ontología con un subconjunto de sus campos con valores conocidos y que las interrelaciones entre los conceptos presentes en la ontología están determinadas.

Finalmente, para probar la solución propuesta para la FIE, se trabajó con tres dominios de información relacionados con el funcionamiento interno de la Universidad Simón Bolívar: las designaciones de cargos, los ingresos y ascensos dentro del escalafón de Profesores y la designación de Jurados para Trabajos de Ascenso. Para trabajar con esos 3 dominios se trabajó con las Actas de Consejos Directivos y Actas de Consejos Académicos como conjuntos de documentos. La información concerniente a las designaciones de cargos, los ingresos y ascensos dentro del escalafón de Profesores se puede encontrar en las Actas de los Consejos Directivos. Asimismo, la información concerniente a la designación de Jurados para Trabajos de Ascenso se encuentra en las Actas de Consejos Académicos.

Duda: ¿Es pertinente definir aquí los dominios sobre los cuales se probó el sistema realizado?

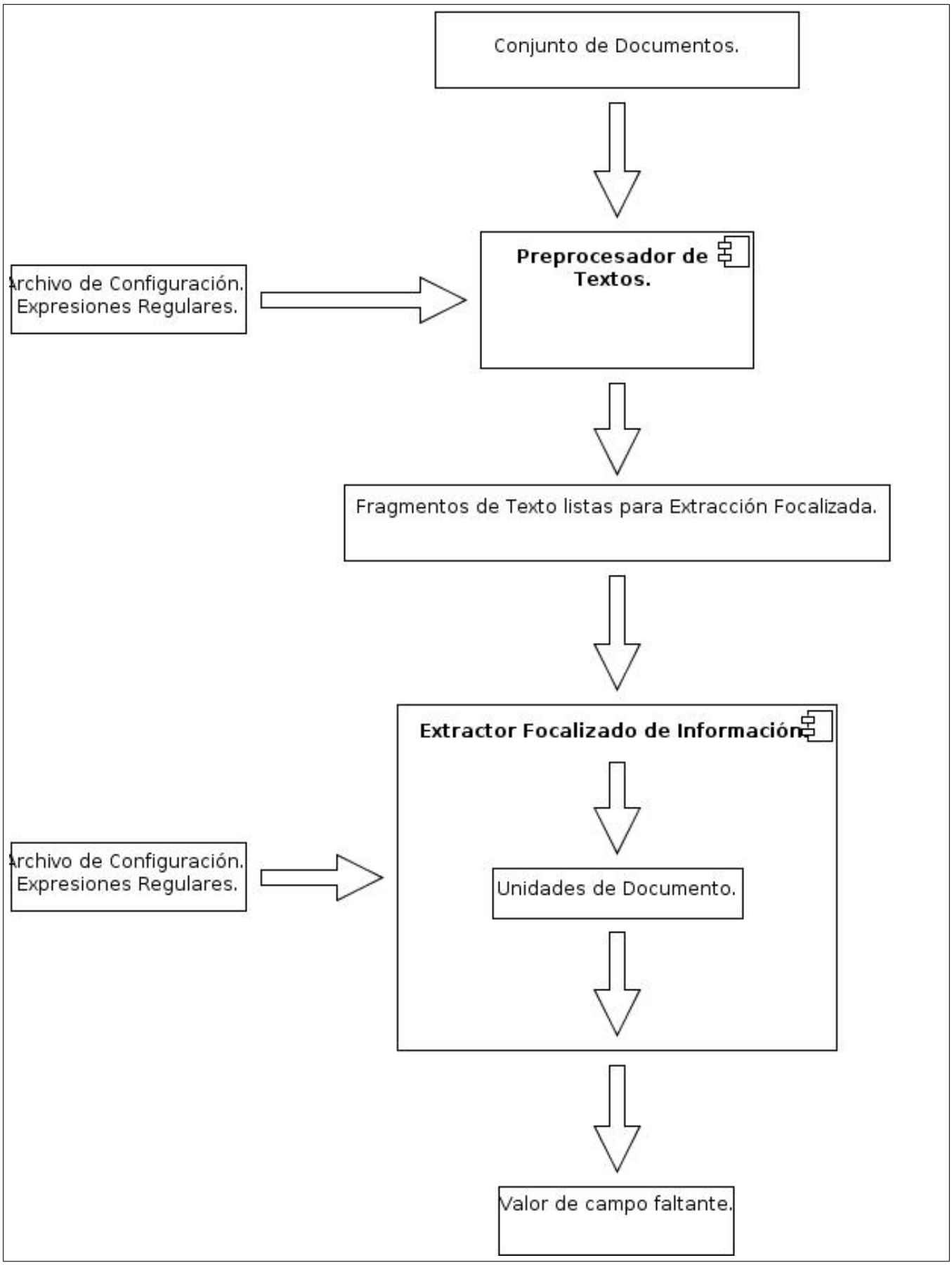


Figura 1.1: Arquitectura de Interrogación de documentos

Capítulo 2

Marco Teórico

Marco Teórico goes here.

2.1. Revisión de trabajos existentes en el área de extracción de Información

En "A survey of Uncertain Data Algorithms and Applications" (2005) [2], Aggarwal y Fellow presentan un estudio de las tecnologías que se han desarrollado para trabajar con datos inciertos. Los autores definen los datos inciertos como datos que pueden tener estar incompletos o que pueden contener errores. Por ejemplo, se puede tener una base de datos con datos de mediciones meteorológicas cuya precisión depende de los instrumentos utilizados y que por ende sus datos pueden contener errores. Puede darse también el caso en el que no se tenga la información completa, como en la base de datos un censo en la cual no se tiene la información de todos los ciudadanos de un país. En el caso particular de este proyecto de grado se trabaja con datos incompletos.

En base a la definición de datos inciertos, los autores examinan las técnicas más recientes en tres campos fundamentales: modelado de datos inciertos, manejo de datos inciertos y minería de datos inciertos. Este proyecto de grado está enmarcado en las dos primeras categorías: modelado y manejo de datos inciertos. Esto se debe a que se quiere estudiar la forma como modelar el problema así como un conjunto de mecanismos que permitan completar la información que falta mediante extracción focalizada.

Los autores definen una base de datos probabilística como un espacio de probabilidad finito en el cual los resultados son las posibles bases de datos consistentes con un esquema dado. En pocas palabras, se tiene una representación de los "posibles mundos". Existen varias soluciones para ello, como modelar la probabilidad de que una tupla esté en la base de datos o, si se quiere más detalle, la probabilidad de que un atributo de una tupla de base de datos esté presente. Sobre esta base se pueden construir técnicas para

el manejo de datos y para realizar minería sobre ellos.

Por otro lado, Aggarwal y Fellow (2006) [2] proponen utilizar una función de probabilidad sobre la correctitud de los datos presentes en la base de datos. Si bien el objetivo de este proyecto es el desarrollo de un mecanismo para contestar una consulta cuya respuesta no está en su totalidad en la ontología, la aproximación que hacen estos autores puede ser útil para definir una medida de la calidad de los datos presentes en la Base de Datos.

Es importante destacar sin embargo, una referencia que se hace al trabajo "Evaluating Probabilistic Queries over Imprecise Data" (2003), de Cheng, Kalashnikov y Prabhakar [3]. En el mismo se hace un análisis sobre las consultas con datos imprecisos o propensos a errores. Una vez más, esto es diferente de lo que se quiere en este trabajo de investigación: trabajar consultas incompletas. Sin embargo, Cheng et al proponen una clasificación de consultas probabilísticas que puede ser tomada en cuenta para clasificar las consultas incompletas.

Básicamente los autores toman en consideración dos criterios: el tipo de elemento devuelto por la consulta y el uso de funciones agregados. En pocas palabras, cuando se toma en consideración el tipo de elemento devuelto por la consulta se tiene que puede devolver un valor puntual o un conjunto de tuplas. En el contexto de las consultas probabilísticas esto puede tener implicaciones: para las consultas que buscan un valor los autores definen cotas superiores e inferiores que definen intervalos en los que los valores de la función deben estar, con una probabilidad acumulada de 1.

La segunda clasificación toma en cuenta si existen agregados o no. La presencia de agregados puede afectar como se verá la evaluación de consultas incompletas.

Además de estos dos trabajos se han examinado otros 3. Sin embargo, su aporte y utilidad para el presente de trabajo de investigación es menor. Chen, Chen y Xie proponen en "Cleaning Uncertain Data with Quality Guarentees" (2008) [4], una métrica para cuantificar la ambiguedad de una respuesta de consulta bajo semánticas de mundo posibles. Sobre esta base, se podría construir un mecanismo para "limpiar" la base de datos.

El trabajo "On databases with Incomplete Information" de Lipski (1981) [5] es muy citado por otros investigadores en el área de consultas incerteras y sin duda alguna constituye un hito muy importante en ésta area. Sin embargo, en dicho trabajo se busca tratar el tema con un formalismo matemático que va más allá de los alcances de este proyecto.

Por otro lado, Kang y Kim proponen en "Query type classification for web document retrieval"

(2003) [6] un mecanismo de clasificación de consultas para extracción de información del WEB. Sin embargo, dicha clasificación corresponde con el tipo de operación que se desea: úbicar algo por tópico, ubicar un homepage o ubicar un servicio. Dicha clasificación no está enmarcada dentro del presente trabajo de investigación y por ende tiene poca utilidad.

Por último, Rocquenco, Segoufin y Viano presentan en "Representing and querying XML with incomplete information" (2001) [7] un modelo para hacer consultas incompletas sobre XML. Existe cierto paralelismo con lo que se quiere hacer en este trabajo de investigación: realizar una segunda extracción de información cuando no se pueda contestar una consulta con lo que está en la ontología. Sin embargo, dicho trabajo no fue de utilidad para la clasificación de las consultas.

Capítulo 3

Análisis de incompletitudes en consulta por el desconocimiento de valores de atributos y propuesta para su determinación

Análisis de tipos de consulta tomando en cuenta la estructura de la ontología y de las consultas.

La incompletitud en una consulta expresada en el lenguaje ODIL, puede estar dada porque falta una tupla en la base de datos, por el desconocimiento de una relación entre dos tuplas o por el desconocimiento del valor de uno o más atributos de una tupla.

El primer caso tiene quizás la mayor dificultad a la hora de determinar con precisión que existe incompletitud ya que no hay una forma de saber, a partir de los datos que se tienen cargados en la BD y la estructura de la ontología, que falta una instancia. El segundo caso también es complejo por cuanto se necesita poder determinar si dos instancias están relacionadas siguiendo la semántica de una interrelación que se encuentre en el modelo ER o en la ontología en base al texto. El tercer caso es el menos complejo de los 3 ya que se puede trabajar con la información que ya se tiene para determinar qué atributos faltan y posteriormente realizar la búsqueda focalizada, usando potencialmente pistas o datos ya contenidos en la ontología o la BD.

Por la complejidad de los primeros casos de incompletitud, en este trabajo nos concentramos sólo en el tercer caso, la determinación de valores de atributos faltantes.

En este informe se presenta un resumen de los resultados obtenidos hasta ahora para determinar qué atributos faltan en una consulta. Para ello se presenta primeramente un ánalisis y una presentación general del mecanismo para encontrar qué valores de atributos faltan en una consulta. En una segunda instancia se presenta una clasificación de las consultas según las operaciones que se deben hacer para determinar las fuentes de incompletitud.

1. Mecanismo para encontrar qué valores de atributos faltan en una consulta.

Es conveniente recordar que la resolución de una consulta de ODIL pasa por tres pasos: en primera instancia, la consulta de ODIL es traducida a una o más consultas SQL que se ejecuta en una Base de Datos Relacional. Dependiendo de la estructura de la ontología (la presencia de agregaciones o generalización-especialización), se hace necesario combinar las respuestas a las consultas en SQL, en una sola para responder a la consulta en ODIL hechas para responder a la ontología??? Por último, se hace necesario evaluar esta respuesta a la consulta en ODIL, para determinar si está incompleta o no. (NOTA: quizas valdría la pena ponerle un nombre a cada tipo de consulta y a cada respuesta. Por ejemplo, pareciera que la respuesta en ODIL, si está incompleta, se va a desplegar un proceso de extracción de otras fuentes, para tratar de completarla, entonces la nueva respuesta, con lo que se consiga en esas otras fuentes, es una respuesta aproximada, pero también la respuesta a ODIL, combinando las respuestas a las consultas en SQL y que se determinó estar incompleta, esa también es una respuesta aproximada. Esto es fundamental para entender dónde estamos parados y hacia dónde queremos ir.)

En este contexto, el proceso de encontrar qué valores de atributos faltan en una consulta consiste en esencia en determinar, para cada una de las instancias de los conceptos que participan en una consulta, qué atributos, que son necesarios para responder la consulta, tienen el valor nulo. Podría darse el caso de que, por decisiones de diseño de la ontología o de la base de datos, sea posible asignar un valor por defecto, diferente de nulo, para señalar incompletitud; en caso de que esto ocurra la tarea de encontrar nulos se reemplaza por la tarea de inspeccionar los valores de los atributos participantes en la consulta para encontrar el valor por defecto especificado.

Para poder realizar la verificación mencionada, se pueden hacer consultas sobre cada uno de los conceptos participantes en una consulta en las que aparezcan los atributos listados en la consulta, en el LIST, SUCH THAT o en una consulta anidada. Dichas consultas pueden preguntar qué atributos de los necesarios para responder a la consulta tienen valores nulos.

Para cumplir con el requerimiento de que se evaluen todas las instancias presentes en la Base de Datos y para poder identificar las instancias con datos faltantes, se puede utilizar el concepto de clave primaria de un concepto. Si bien el concepto de clave primaria está relacionado con el Modelo Relacional, puede perfectamente adaptarse en el contexto de ontologías. En tal sentido, una clave primaria en una ontología puede verse como un conjunto de propiedades o atributos tales que juntos identifican unívocamente una instancia.

(Creo que esta sección debía comenzar por este párrafo, es decir, antes de hablar de los nulos y demás, hay que recordar cuál es la sintaxis de una consulta en ODIL.) Es conveniente recordar que la estructura general de una consulta en ODIL está dada por dos partes. En primera parte se tiene la claúsula LIST INSTANCES, en donde se colocan los atributos que son necesarios para responder a la consulta. En segunda instancia se tiene la cláusula SUCH THAT en la que se realiza la calificación de la consulta. Esto implica que para cada consulta en la que se quiera verificar incompletitud se deban hacer como mínimo dos conjuntos de consultas.

(Esta parte necesita estructura, para no perdernos. Puedes hacer una lista enumerada, donde coloques los dos tipos de consultas adicionales o dos subsecciones, para los dos tipos de consultas adicionales.) Puede verse que el primer tipo de consulta sirve para determinar un tipo de incompletitud de la Base de Datos en el cual no se puede responder la consulta del usuario porque la información que se desea no está presente en las tuplas de la bd relacional. Dado que se quiere realizar extracción focalizada de información para completar los valores faltantes, es pertinente utilizar la calificación completa de la consulta original, con todas las condiciones que usan atributos de L2 (en este documento, todavía no has dicho lo que es L2, Hay que ir mas despacio, copiar la sintaxis de odil y colocar alli qué es L2, L1, etc.). Esto es, es deseable que en las consultas adicionales para encontrar valores de un atributo que falta se mantengan las condiciones de los otros atributos pertenecientes a L2. De lo contrario se trataría de una búsqueda de atributos incompletos que se puede realizar en cualquier momento, diferente de lo que DIA define como extracción focalizada. (No se si este comentario se entiende en el contexto de este documento solamente, no estoy segura de que esté claramente explicado.)

El segundo tipo de consulta adicional está ligado a otra clase de incompletitud, en la cual es posible que la respuesta no esté completa, porque no es posible encontrar qué instancias cumplen con las condiciones del SUCH THAT (no entiendo la descripción, quizás hace falta un ejemplo, antes de seguir adelante). Este tipo de consulta se realiza para cada uno de los atributos que participa en la calificación de la consulta, utilizando el resto de las condiciones del SUCH THAT que no incluyen al atributo que se esté examinando. Al utilizar el resto de las condiciones sin incluir a uno de los atributos, se está realizando una calificación parcial de la consulta. Esto se debe a que al igual que en el primer tipo de consulta, se quiere realizar extracción focalizada y por lo tanto interesa que haya una calificación sobre los atributos siempre y cuando no sea sobre el atributo sobre el que se quieren buscar valores nulos (creo que en todo el documento, excepto en los ejemplos de consultas, se puede sustituir NULL por nulo o nulos, según corresponda). Esto último se debe a que de lo contrario podrían no encontrarse todos los valores NULL. Para realizar la calificación parcial de la consulta, es necesario eliminar exactamente las partes de la condición en las cuales se encuentra el atributo en cuestión. Dado que el SUCH THAT puede estar formada por una serie de conjunciones y disjunciones de condiciones, es necesario elegir bien qué partes de la condición se quieren eliminar. El criterio que se desea aplicar es que se reduzcan todas las ocurrencias del atributo en la calificación manteniendo el mayor número de calificaciones sobre otros atributos.

Para hacer esto, se puede suponer que el SUCH THAT se encuentra en una Forma Normal Conjuntiva (FNC) - esto es, está formado por una secuencia de conjunciones de subcondiciones formadas exclusivamente por disjunciones de átomos. En el caso en el que el SUCH THAT no se encuentre en la FNC, es posible realizar una serie de transformaciones sobre la mismas aplicando tantas veces como sea necesarios los teoremas de Morgan y otras propiedades lógicas. Una vez en la FNC, se deben eliminar las cláusulas que precisamente conciernen al atributo que se está inspeccionando. La necesidad de utilizar la FNC es que si se tiene una cláusula que involucre al atributo A en cuestión en disjunción con condiciones sobre una lista de otros atributos LA, eliminar la condición sobre A puede hacer más restrictiva la calificación

hecha en el SUCH THAT.

Por ejemplo, si se tiene la siguiente consulta:

LISTA estudiante.carnet INSTANCES SUCH THAT estudiante.carrera = 'Ingeniería de Computación' OR estudiante.indice 3

Con los siguientes datos: Carnet Carrera Indice 06-40000 Ingeniería de Computación 2 06-40001 null 2 06-40002 Ingeniería de Computación 4 06-40003 null 4

Entonces hacer la consulta:

LISTA estudiante.carnet INSTANCES SUCH THAT (estudiante.carrera is null) AND estudiante.indice 3; (en la que se eliminó la condición sobre estudiante.carrera del or)

Devuelve como resultado:

06-40001

Mientras que se esperaría que la respuesta sea

06-40001 06-40003

Que son todas las tuplas que potencialmente podrían responder a la consulta realizada (puede darse que el estudiante con carnet 06-40003 estudie computación y por lo tanto deba aparecer en el resultado de la consulta realizada por el usuario aunque su indice sea mayor o igual a 3).

Habiendo aclarado como se realiza la calificación parcial, es pertinente realizar dos comentarios adicionales sobre estos tipos de consulta explicados. En primer lugar, tiene sentido realizar todas las consultas de tipo 2 antes de realizar la consulta de tipo 1: el motivo de esto es que es necesario garantizar que exista el menor número de valores null para atributos usados para calificar la consulta 1, de forma que se disminuya el riesgo de que la respuesta esté incompleta porque no se encuentren todas las tuplas que califiquen en la consulta.

En segundo lugar, el proceso de extracción focalizada puede ser de tipo iterativo y que la búsqueda focalizada puede repetirse hasta que se cumpla que no se encuentren más valores de atributos. El motivo de esto es que si en la búsqueda focalizada se utiliza información sobre los atributos con valores diferentes de null de las instancias que están incompletas para facilitar la ubicación de los atributos faltantes, entonces la posibilidad de encontrar valores de atributos aumenta a medida que se tienen más atributos diferentes de nulls en la Base de Datos. Consecuentemente puede ser útil realizar estos tipos de consultas más de una vez .

Se hace necesario una vez explicado el mecanismo general destacar que dado que las consultas realizada tienden a ser por su naturaleza de menor complejidad que las consultas originales, es posible realizarlas como una consulta más en ODIL. Esto es, si se tiene por ejemplo una consulta con agregados (caso que se verá posteriormente), el mecanismo de determinación de la incompletitud trabajará sobre los miembros del mismo, por lo que la composición de la respuesta que realiza el procesador de consultas que hace ODIL no afecta de forma alguna la determinación del origen de incompletitud.

2. Forma canónica de los diferentes tipos de consulta que se pueden hacer en ODIL y las consultas realizadas determinar que atributos están incompletos.

Como se ha visto, el proceso de determinar la incompletitud de una respuesta por la falta de valores de atributos, incluye la realización de consultas adicionales sobre los conceptos y atributos que participan en la consulta hecha para precisar la incompletitud. A continuación se presentan varios casos diferentes para poder identificar fácilmente dada la forma general de una consulta que consulta se debe realizar.

2.1 Consulta simple.

La forma más simple la siguiente:

LIST Lista de atributos L1 INSTANCES SUCH THAT Calificación de la consulta con condiciones sobre la lista de atributos L2

Que genera dos tipos de consultas adicionales diferentes para cada uno de los conceptos involucrados, para determinar incompletitud, a saber:

1) Para cada atributo A perteneciente a L1:

LIST Clave del Concepto INSTANCES SUCH THAT A is null, Calificación de la consulta con las condiciones sobre la lista de atributos L2;

2) Para cada atributo A perteneciente a L2:

LIST Clave del Concepto INSTANCES SUCH THAT A is null, Calificación parcial de la consulta con las condiciones sobre L2-Atributo A;

Para ilustrar este caso, considérese la siguiente consulta en ODIL:

LIST pais.poblacion INSTANCES SUCH THAT pais.capital = 'Brasilia';

Si la tupla correspondiente a Brasil no tiene un valor no nulo para el atributo capital la siguiente consulta dará una lista vacía, considerando que en la Base de Datos se tiene lo siguiente: Nombre Capital Población Brasil null 5.000.000

La consulta pertinente para determinar la incompletitud en este problema sería:

LIST pais.nombre INSTANCES SUCH THAT pais.capital is null;

y

LIST pais.nombre INSTANCES SUCH THAT pais.poblacion is null pais.capital = 'Brasilia';

La primera consulta permite determinar que la capital de Brasil tiene un valor null. Análogamente, en caso de que la población tuviese un valor null, la segunda consulta determinaría que no se tiene la población de Brasil.

2.2 Uso de funciones agregadas.

Son del tipo:

LIST Función agregada sobre atributos L0, Lista de atributos L1 INSTANCES SUCH THAT Calificación de la consulta con las condiciones sobre la lista de atributos L2

Que genera las siguientes consultas adicionales para determinar incompletitud (para cada concepto que participe):

1) Para cada atributo A de L0

LIST Clave del Concepto INSTANCES SUCH THAT A is null Calificación de la consulta con las condiciones sobre la lista de atributos L2;

2) Para cada atributo A de L1:

LIST Clave del Concepto INSTANCES SUCH THAT A is null, Calificación de la consulta con las condiciones sobre la lista de atributos L2;

3) Para cada atributo A perteneciente a L2:

LIST Clave del Concepto INSTANCES SUCH THAT A is null, Calificación parcial de la consulta con condiciones sobre L2-Atributo A;

En esencia las consultas generadas son muy similares a las consultas generadas en el caso simple, sólo que se añade la verificación de nulls sobre los agregados.

Para consultas con funciones agregadas, se puede presentar este ejemplo:

LIST sum (pais.poblacion) INSTANCES SUCH THAT pais.nombre like 'A' or pais.nombre like 'B' ;

Genera las consultas:

LIST pais. nombrepais INSTANCES SUCH THAT pais.poblacion is null (pais.abreviacion like 'A

La cual permite estar seguro de tener todas las poblaciones necesarias para calcular el agregado. También se genera la consulta:

LIST nombrepais INSTANCES SUCH THAT pais.abreviacion is null;

Que permite ver que todas las abreviaciones estén completas.

Si se tienen los siguientes datos en la BD: País Población Abreviación Venezuela 30 null Argentina null Arg Colombia 40 null

La primera consulta determina que falta la población de Argentina. La segunda determina que falta la abreviación de Venezuela.

2.3 Consultas anidadas.

Son del tipo:

LIST Lista de atributos L1 INSTANCES SUCH THAT Condiciones sobre la lista de atributos L2, Subconsulta LIST Lista de atributos L3 INSTANCES SUCH THAT Condiciones sobre la lista de atributos L4

Que genera las siguientes consultas adicionales para determinar incompletitud:

1) Para cada atributo A de L1 LIST Clave del Concepto INSTANCES SUCH THAT A is null Calificación con las condiciones de consulta anidada y la lista de atributos L2;

2) Para cada atributo A perteneciente a L2:

LIST Clave del Concepto, Atributo A INSTANCES SUCH THAT A is null Calificación parcial de la consulta con las condiciones de la consulta anidada-Atributo A U L2-Atributo A;

Para cada consulta anidada:

3) Para cada atributo A de L3

LIST Clave del Concepto INSTANCES SUCH THAT A is null Calificación de la consulta con las condiciones sobre lista de atributos L4;

4) Para cada atributo A perteneciente a L4:

LIST Clave del Concepto INSTANCES SUCH THAT A is null Calificación parcial de la consulta con las condiciones sobre la lista L4-Atributo A;

El orden de resolución de estas consultas es similar al orden de resolución de consultas del caso más simple y general. En efecto, puede verse como un proceso Bottom-Up en el cual el árbol de la consulta se va resolviendo de abajo hacia arriba.

Por ejemplo, en la siguiente consulta se piden el clima de las regiones cuyas poblaciones totales son mayores a 100.000 habitantes:

LIST region.clima INSTANCES fsdfsdfsdfsdfsdf

En este caso hay que realizar consultas para determinar que atributos faltan tanto en la consulta interior como en la consulta exterior:

1) LIST region.nombre INSTANCES SUCH THAT region.clima is null;

2) LIST region.nombre INSTANCES Sdsadsadasdasdasdasdasdsa

LIST region.nombre INSTANCES SUCH THAT dasdsadasdasd

3)

LIST nombrepais, pais.poblacion INSTANCES;

Si se tienen los siguientes datos en la Base da Datos: Nombre de región Clima Continente Abreviacion Andina Tropical America And Latinoamerica null America null

Nombre Poblacion Venezuela null Brasil 50 Colombia 35

Y se tiene que la región andina está formada por Venezuela, Colombia y la región latinoamericana por Venezuela y Brasil. Entonces las consultas devuelven los siguientes resultados:

1) Latinoamerica

2) vacia.

Andina, And. Latinoamerica, null. (se determina que no se tiene la abreviacion de latinoamerica).

3) Venezuela

Así, las tres consultas realizada permiten determinar los atributos que están incompletos: la población de Venezuela (necesaria para el agregado) y el clima y abreviación de la región Lationamerica.

2.4 Consultas con agregados.

El uso de agregados de la ontología puede ser manejado de forma similar a la manera como se aborda el uso de funciones agregadas. Es necesario verificar en primer lugar que las partes del agregado que se define esté completo, esto se puede hacer con las siguientes consultas:

LIST Lista de atributos L0 de conceptos que son partes del agregado, Lista de atributos L1 INSTANCES SUCH THAT Calificación de la consulta con las condiciones sobre la lista de atributos L2

Que genera las siguientes consultas adicionales para determinar incompletitud:

1) Para cada atributo A de L0 (lista de atributos del concepto parte)

LIST Clave del Concepto parte del agregado INSTANCES SUCH THAT A is null Calificación de la consulta con condiciones sobre la lista de atributos L2;

2) Para cada atributo A de L1

LIST Clave del Concepto sobre el que se hace la consulta INSTANCES SUCH THAT A is null Calificación de la consulta con condiciones sobre la lista de atributos L2;

3) Para cada atributo A perteneciente a L2:

LIST Clave del Concepto sobre el que se hace la consulta INSTANCES SUCH THAT A is null Calificación parcial de la consulta con condiciones sobre L2-Atributo A;

Aunado a esto, existe otro origen de incompletitud para este subcaso que se debe contemplar. Puede darse el caso de que el agregado no esté completo. Es decir, puede ser que falten instancias en la lista de los miembros del agregado. Por ejemplo si se tiene un concepto "Equipo" que es un agregado de "Jugadores", puede ser que falten jugadores en un equipo.

Desafortunadamente este problema no puede ser resuelto con el mecanismo propuesto para resolver incompletitud por la falta de valores de atributos. En cierta forma, es comparable con el problema de descubrir interrelaciones entre conceptos a partir de la extracción, ya que el pertenecer a un agregado es una interrelación.

Por ejemplo se tiene el siguiente caso:

LIST region$^{p}ais.poblacionpais, region.climaINSTANCESsadasda$

Se generan las siguientes consultas:

1) LIST pais.nombre INSTANCES SUCH THAT pais.poblacionpais is null;

2) LIST region.nombre INSTANCES SUCH THAT region.clima is null region.abreviacion like 'And';

3) LIST region.nombre INSTANCES SUCH THAT region.clima is null region.abreviacion like 'And';

Si ste tienen los siguientes datos: Nombre Población Región Venezuela 35 Andina Colombia null Andina México 20 Norteamerica

Nombre Abreviacion Clima Andina And Tropical Nortamerica null null

Las consultas devuelven:

1) Colombia (falta su población). 2) Nortamerica. (falta el clima) 3) Norteamerica. (falta la abreviacion).

2.5 Consultas con relaciones de especialización-generalización.

Cómo se ha estudiado anteriormente, si se parte de la suposición de que al realizar una consulta sobre una superclase o una subclase las consultas realizadas sobre la base de datos tomen consideración la estructura jerárquica de la generalización-especialización entonces no se hace necesario tomar ningún tipo medidas adicionales para las relaciones de especialización-generalización. Esto es, si el procesador de consultas ODIL enmascara la relación, no es necesario realizar ninguna operación extra. Es importante no obstante tomar en cuenta que la clave de una subclase puede incluir atributos de la superclase.

Referencias:

El Masri Ramez y Navathe Shamkant (2003), Fundamentals of Database Systems. Cuarta edición, Editorial Pearson Education, 1030 páginas.

(NOTA: deberías poner como referencia también el artículo completo de DIA, pues de hecho, lo citas.)

Capítulo 4

Diseño

El sistema para realizar la Extracción Focalizada de Información está conformado por dos módulos: el Módulo de Determinación de la Fuente de Incompletitud y el Módulo de Búsqueda Focalizada de Información.

4.1. El Módulo de Determinación de la Fuente de Incompletitud (MDFI)

Extractor plano: no se toman en cuenta ors. Quizas se debería.

4.2. El Módulo de Búsqueda Focalizada de Información (MBFI)

El Módulo de Búsqueda Focalizada de Información es el encargado de encontrar los valores de los campos cuyos valores son desconocidos o incompletos para una consulta dada. Dichos valores son buscados en un conjunto de documentos presentes en el sistema de archivos donde se ejecute el sistema utilizando información de contexto proporcionada por el Módulo de Determinación de la Fuente de Incompletitud (MDFI).

Para llevar a cabo esta tarea, el MBFI tiene dos componentes: un preprocesador de texto y un extractor focalizado de información.

El Preprocesador de Texto tiene como objetivo preparar el conjunto de documentos sobre el cual se hará la Búsqueda Focalizada de Información y dejarlo listo para la realización de la extracción. Las tareas para ello incluyen leer el documento en su formato original y extraer el fragmento o el conjunto de fragmentos del documento que están relacionados con el dominio sobre el cual se hará la Búsqueda Focalizada de Información.

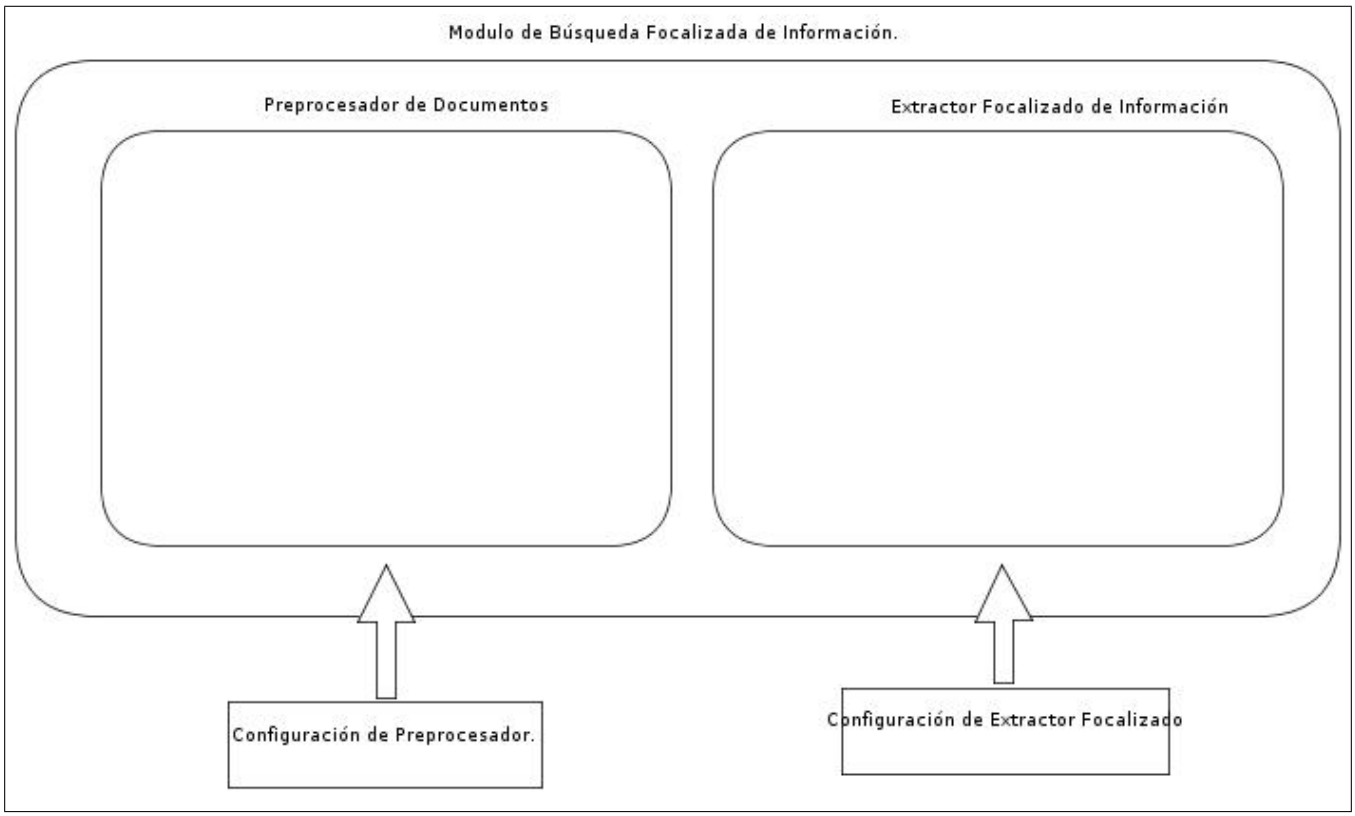


Figura 4.1: Diagrama del Módulo de Búsqueda Focalizada de Información

El preprocesamiento de texto es una tarea que se realiza *offline*. Es decir, cada conjunto de documentos puede ser preprocesado previamente de forma para que las consultas sean respondidas lo más rápidamente posible.

Por su parte, el Extractor Focalizado de Información (EFI) tiene como objetivo extraer el valor del campo faltante utilizando como materia prima el conjunto de documentos preprocesados y un contexto de extracción. El contexto de extracción consiste en toda la información que el MDFI puede recopilar sobre la consulta que se está realizando que puede ser de útil para hacer la extracción focalizada de información. Puede incluir por ejemplo valores de campos que están relacionados con el campo faltante dentro del dominio sobre el cual se hace la búsqueda.

El MDFI trabaja directamente con el Extractor Focalizado de Información para encontrar los valores faltantes. Esto es, una vez que se determinan las fuentes de incompletitud, el EFI recibe solicitudes de búsqueda para cada uno de los campos con valores faltantes. Esas solicitudes permiten descubrir y completar la ontología y dar respuesta a las consultas que tiene el usuario. El preprocesador, como se ha

dicho, se ejecuta una única vez por conjunto de documentos.

Capítulo 5

Detalles de implementación

5.1. Consideraciones generales

5.1.1. Lenguaje de programación

El lenguaje de programación utilizado para la implementación del sistema de Extracción Focalizada de Información fue Java en su versión 1.6.0_20.

5.1.2. Librerías utilizadas

Para parsear las consultas, se trabajó con las librerías elaboradas por Andueza y Arteta (año) en su proyecto de Grado (referenciar). Hablar aqui de ellas.¿

Se utilizaron ademas las siguientes librerías open source:

- Para desplegar mensajes de traza y errores se utilizó la librería Log4J. Dicha librería permite en lugar de imprimir mensajes por el Standard Output, imprimirlos dependiendo de un nivel de desarrollo espeficiado en un archivo de configuración. A la hora de desplegar un mensaje se elige un nivel de desarrolo: traza, error, error fatal, warning, entre otros. Luego con modificar el archivo de configuración de la librería se puede fácilmente apagar y prender los niveles para que se impriman o dejen de imprimir algunos mensajes y cambiar rápidamente entre etapas de desarrollo, debugging, pruebas y puesta en producción. Se utilizó la version 1.2.16.
- Para procesar archivos PDF se utilizó la librería PDFBox y Tika, en sus versiones 1.6 y 1.1.
- Para procesar archivos HTML se utilizó la librería Jericho, en su versión 3.2.
- Para procesar archivos .doc se utilizó la librería POI, en su versión 3.8.

- Para leer archivos de configuración de XML se utilizó la librería que ofrece Java para parsear y consultar XML, JAXP, en su versión 1.4.

5.2. Implementación del Módulo de Determinación de Fuente de Incompletitud (MDFI)

Dificultades: Flatter: RHS no puede tener parentesis.

5.3. Implementación del Módulo de Búsqueda Focalizada de Información (MBFI)

Como se ha explicado en 4.2, el Módulo de Búsqueda Focalizada de Información (MBFI) es el módulo que encuentra los valores de los campos cuyos valores son desconocidos o incompletos para una consulta dada. Estos valores son buscados en un conjunto de documentos definidos por el usuario y se encuentran tomando en cuenta un contexto de extracción que contiene información útil para la búsqueda. El MBFI tiene dos componentes: un preprocesador de texto y un extractor focalizado de información.

En grandes rasgos, el funcionamiento y la implementación de ambos componentes son bastantes similares. Ambos componentes operan como extractores de texto utilizando expresiones regulares definidas previamente por un usuario experto. Dichas expresiones regulares funcionan como delimitadores del fragmento de texto que interesa según la tarea que se esté realizando: en el caso del preprocesador de texto, el objetivo es extraer el fragmento de texto que está relacionados con el dominio sobre el cuál se realiza la extracción focalizada; el extractor focalizado busca encontrar el valor del campo faltante para responder una consulta.

Ambos componentes están hechos para trabajar con un archivo de configuración en XML que especifica los parámetros necesarios para realizar el preprocesamiento y la extracción. En particular los parámetros que se guardan en estos archivos de configuración incluyen las expresiones regulares, las rutas de directorio de los archivos de entrada, la ruta de los archivos de salida para el caso del preprocesador, entre otras cosas. De esta manera, el usuario del sistema debe preocuparse tan sólo por entender cómo realizar un archivo de configuración para cada uno de los dominios necesarios.

Más adelante se expondrán algunas consideraciones sobre las expresiones regulares de interés para el entendimiento de la implementación del MBFI.

5.3.1. Preprocesador de Textos

Ahondando más en el componente de pre-procesamiento, su objetivo como se ha dicho es extraer de cada documento el conjunto de fragmentos que está relacionado con el dominio sobre el cual se hace la búsqueda focalizada. En tal sentido tiene como entrada un conjunto de documentos en su formato original (PDF, Portable Readable Document; HTML , Hyper Text Markup Language; DOC y DOCX, archivos de Microsoft Word) y un archivo de configuración. Su salida es un archivo .txt en el cual se tiene el conjunto de fragmentos que están relacionados con el dominio en cuestión según lo especificado por las expresiones regulares. Para procesar los documentos se utilizaron las librerías descritas anteriormente.

Falta hablar del timing.

5.3.2. Extractor Focalizado

En cuanto al componente de extracción focalizada, su objetivo es dado un conjunto de fragmentos de documentos que puede contener valores de campos necesarios para responder una consulta, encontrar el valor de esos campos. Para ello opera en 2 fases: primero rompe cada documento en unidades relacionadas con el dominio en cuestión. Para ilustrar este punto, tomando en cuenta los 3 dominios que se consideraron para este trabajo, las unidades vienen siendo designaciones, ingresos y ascensos dentro del escalafón y designación de un jurado de ascenso, por trabajo. Este refinamiento sobre los documentos es hecho utilizando también expresiones regulares y su objetivo es aislar fragmentos de texto donde haya una cierta cohesión semántica (un fragmento de texto que se refiera a los mismo) que permita ubicar la respuesta utilizando el contexto de extracción.

Una vez separados los documentos en unidades, se utiliza el contexto de extracción para ordenar las unidades en un orden de preferencia. Dicho orden se realiza tomando un valor denominado *UnitHitMeasure* que busca medir a través de una media ponderada cuánto se aproxima o relaciona cada unidad encontrada en los documentos a la unidad correcta según lo especificado por el contexto de extracción. Por ejemplo, si se está trabajando con designaciones de cargos, se buscan ordenar las unidades tomando en cuenta el contexto de extracción y los valores de campos conocidos sobre una designación las designaciones según la presencia de los valores conocidos en cada unidad.

El UnitHitMeasure es calculado de la siguiente manera: cada campo de la ontología tiene un peso que es asignado por un usuario experto en el archivo de configuración. Cada contexto de extracción tendrá uno o más valores de campos conocidos. Para cada campo presente el contexto de extracción cuyo resultado es conocido y que está presente en la unidad se suma el peso asociado a ese campo. Luego se divide esa sumatoria entre la máxima suma posible (el valor de la suma si todos los campos cuyo valor se conocen estuviesen en la unidad con los valores conocidos). De esta manera se obtiene un número del 0

al 1 que indica el grado de relación que tiene la unidad en cuestión con el contexto de extracción focalizado.

DUDA: Profe, quiere que coloque un ejemplo de cómo se calcula esto para que quede más claro? O se entiende para un lector cualquiera?

Vale acotar que a la hora de determinar si un valor está presente en la unidad se pueden realizar modificaciones del String con el objetivo de incorporar el uso de sinónimos, traducción de idiomas, desglose de siglas, entre otros, que pueden permitir aumentar la posibilidad de que un campo tenga esté contenido dentro de una unidad de una forma equivalente en cuando a significado. Esto debe ser sin embargo implementado según el contexto y requiere una modificación en los archivos XML, en su lector y la implementación del código para realizar las modificaciones. Por ejemplo, se implementó una funcionalidad para dada una fecha en un formato cualquiera, generar las formas de fechas más comunes según lo utilizado en los documentos.

Una vez ordenadas las unidades en orden de precedencia se ubica el valor del campo que se quiere encontrar por medio de expresiones regulares. Cabe destacar que el usuario puede especificar un *Minimum HitMeasure* (valor mínimo de HitMeasure) para que una respuesta sea considerada válida: mientras más cercana a 1 será más probable que la respuesta sea correcta aunque se disminuye la posibilidad de encontrar el valor.

En la figura 5.3.2 puede apreciarse el funcionamiento interno del MBFI.

5.3.3. Diseño de las expresiones regulares para la configuración del Preprocesador de textos y del Extractor Focalizado

La elaboración de las expresiones regulares es una tarea que debe realizar un usuario experto que conozca el dominio sobre el cual se quiere realizar la extracción focalizada. En general los dominios admisibles son aquellos con textos estructurados o semi-estructurados, donde se puedan deducir patrones en la redacción de los contenidos. Aún teniendo cierta regularidad en la forma del texto, existen retos como errores de redacción, ortografía, transcripción, formato en los documentos fuente, presencia de tablas entre otros, que dificultan el uso de expresiones regulares. Es posible también por ejemplo que hayan presentes un conjunto de carácteres que parezcan espacios en blanco pero que no hagan match con una expresión regular porque no son espacios.

En todo caso, lo relevante en esta descripción de la implementación del sistema es que para atender esta dificultad los archivos de XML permiten contener varias expresiones regulares ordenadas por orden de precedencia. Los textos se exploran con las expresiones regulares según el orden de precedencia: si no se

encuentra un fragmento de texto con una expresión regular se explora posteriormente con otra de menor precedencia y así sucesivamente.

El objetivo de esta precedencia es que el usuario pueda colocar con alta precedencia expresiones regulares que sean bastante selectivas y restrictivas, con el objetivo de asegurar correctitud. Luego a medida que se desciende en la escala de precedencia se puede relajar la restrictividad de la expresión regular, con el objetivo de aumentar la probabilidad de encontrar un fragmento de texto. Relajar la restrictividad implica que la respuesta no sea exactamente la buscada y que por el contrario pueda contener caracteres de más (esto es, que el fragmento de texto obtenido contenga al fragmento de texto correcto). Esto puede ser deseable en muchos casos, dada la complejidad de utilizar expresiones regulares.

DUDA: PONER COMO SE OBTIENEN LAS REGEXPS?

En el Apéndice A se muestran fragmentos de archivos de configuracion.

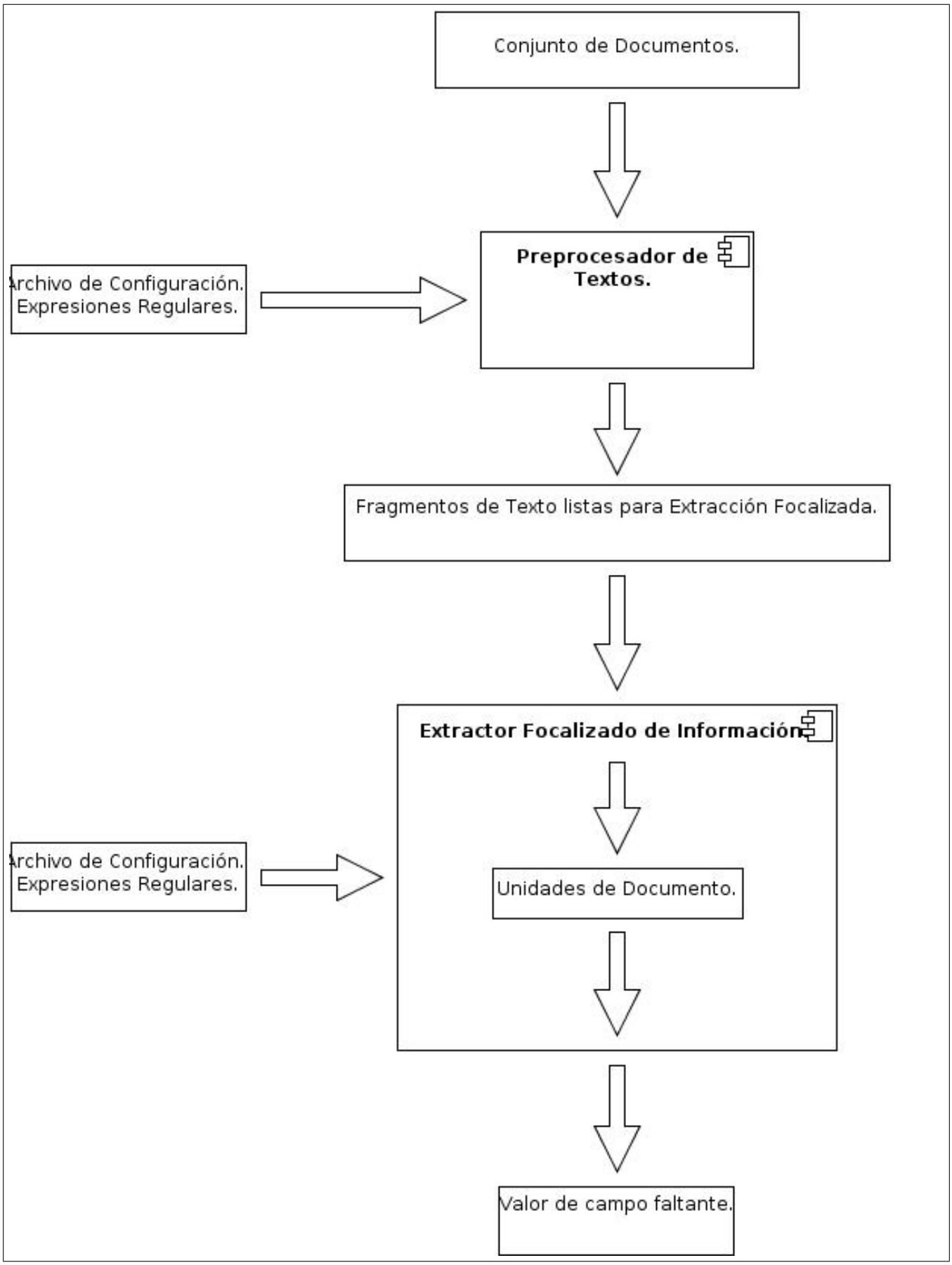


Figura 5.1: Funcionamiento interno del Módulo de Búsqueda Focalizada

Capítulo 6

Pruebas

6.1. Consideraciones Generales

Con el objetivo de realizar las pruebas sobre el sistema se trabajó como ya se ha dicho sobre tres dominios relacionados con el funcionamiento interno de la Universidad Simón Bolívar: las designaciones de cargos, los ingresos y ascensos dentro del escalafón de Profesores y la designación de Jurados para Trabajos de Ascenso. Para trabajar con esos 3 dominios se trabajó con las Actas de Consejos Directivos y Actas de Consejos Académicos. La elección de los 3 dominios se tomó buscando tener una variedad de estilos de texto semiestructurado para obtener resultados diversos sobre el funcionamiento.

La selección de los documentos se realizó sobre el conjunto de las actas de los Consejos Directivos y Académicos de la Universidad Simón Bolívar desde el año 1998 hasta el año 2012. Esto permite que las pruebas se hagan sobre un conjunto de documentos que pueden tener variabilidad en su estilo, redacción y estructura en los años. Del conjunto de actas en el intervalo de tiempo especificado, se hizo una preselección de los documentos para utilizar las actas de los Consejos Ordinarios. Adicionalmente, según cada dominio de aplicación se extrajeron las actas que sí contienen información sobre los dominios. Es decir, se utilizaron los documentos que contienen información sobre los dominios elegidos, descartando aquellas actas que no contenían texto de interés.

Una vez definidos los dominios se definieron varios conjuntos de prueba. En primera instancia se probaron los Módulos de Determinación de Fuente de Incompletitud y Módulo de Búsqueda Focalizada de Información por separado. Luego se hicieron pruebas sobre ambos compontentes integrados. A continuación se describen con mayor profundidad las pruebas realizadas sobre el sistema.

6.2. Pruebas unitarias sobre el Módulo de Determinación de Origen de la Incompletitud.

6.3. Pruebas unitarias sobre el Módulo de Búsqueda Focalizada de Información

Para probar el Módulo de Búsqueda Focalizada de Información se hicieron diseñaron 2 pruebas por dominio. Por un lado de hicieron pruebas para el Preprocesador de Texto y en segunda instancia Pruebas para el Extractor Focalizado.

6.3.1. Pruebas del Preprocesador de Texto

Para probar el preprocesador, se tomó un conjunto de archivos seleccionados aleatoriamente con uniformidad sobre los años. La muestra elegida fue de un tercio de la población total de actas existentes, para cada dominio. Para las designaciones se trabajó con 73 documentos, para las incorporaciones y ascensos dentro del escalafón se trabajó con 77 documentos y para la designación de Trabajos de Jurado de Ascenso se trabajó con 49 documentos.

Las pruebas en esencia consistieron en ver si el fragmento de texto extraído por el preprocesador coincidía exactamente, estaba contenido, contenía o era completamente disjunto con el segmento de texto que debería ser extraído. Esto es: pueden verse ambos fragmentos de textos como secuencias de caracteres y se puede examinar si ambas secuencias coinciden por completo, tienen relaciones de contención o son completamente diferentes.

Esta prueba se hizo manualmente: se examinó el resultado de la extracción realizada por el preprocesador y se buscó dentro de cada documento el fragmento que debería extraerse.

Se definió como *aprobado* que o bien el fragmento extraído sea completamente igual a la respuesta esperada (caso correcto) o si el fragmento extraído contiene a la respuseta esperada y algunos caracteres de más. Dicho caso se considera aprobado porque en esencia el que el documento preprocesado no contenga algunas líneas más de código no afecta considerablemente la extracción focalizada. Como *no aprobado* se entienden los casos en los que la respuesta está incompleta (la respuesta esperada contiene al fragmento de texto obtenido) o que los fragmentos son completamente disjuntos.

6.3.2. Pruebas de Extractor Focalizado

Para probar el extractor focalizado, se tomó al igual que en las pruebas del Preprocesador de texto un conjunto de prueba seleccionado aleatoriamente con uniformidad sobre los años. La muestra es de un tercio de la población total. Para hacer esto de cada archivo seleccionado, se tomaban hasta 3 unidades de documento según lo explicado en 5.3.2. Dichas unidades fueron vaciadas en un archivo de pruebas que posteriormente era leído por un módulo de pruebas que se encarga de probar el extractor focalizado automáticamente.

Una prueba individual consiste en hacer una búsqueda focalizada sobre un campo presente en una unidad de documento. Una unidad de documento da por lo tanto para hacer tantas pruebas como campos con valor tenga esa unidad. Si el dominio tiene por ejemplo 3 campos y se tiene una unidad de documento con 2 campos, se realizan 2 pruebas para esa unidad: una para cada campo. Las pruebas se hacen iterando entonces sobre cada uno de los campos con valores en la unidad y buscando extraer el valor del mismo utilizando un contexto de extracción dado por los valores de los campos que sí se tienen.

Para simular condiciones en las que no se tienen todos los valores de los campos relacionados con una unidad - esto es, que a pesar de que la unidad contenga valores para varios campos a la hora de hacer la búsqueda focalizada el *contexto de extracción dado por la consulta* no contenga todos esos valores -, los contextos de extracción se construyeron con un subconjunto de los valores de los campos presentes en la unidad de documento. Este subconjunto se determina aleatoriamente: se genera un número al azar y si ese número es mayor que una *probabilidad de tener cada campo* se incluye en el contexto de extracción. De esta manera se evitar suponer para estas pruebas que los contextos de extracción son completos.

Las pruebas se realizaron iterando el valor de la *probabilidad de tener cada campo*. Se tomaron como *probabilidades de tener cada campo* 1; 0.66 y 0.33. Adicionalmente, se iteró también sobre el *Minimum Hit Measure* (según lo definido en 5.3.2) con el objetivo de probar el efecto que tiene la elección de este parámetro en el funcionamiento del extractor focalizado. Los valores sobre los que se realizaron las pruebas fueron: 1; 0.66 y .33.

De esta manera, y resumiendo, las pruebas se realizaron tomando tres unidades de documento de cada documento de cada dominio. Para cada unidad se itera sobre la *probabilidad de tener cada campo* y el *Minimum Hit Measure*. Los resultados fueron posteriormente totalizados por dominio y tabulados.

Capítulo 7

Resultados

A continuación se muestran los resultados de las pruebas realizadas.

7.1. Resultados de las pruebas unitarias sobre el Módulo de Determinación de Origen de la Incompletitud.

7.2. Resultados de las pruebas unitarias sobre el Módulo de Búsqueda Focalizada de Información

Tabla 7.1: Resultados detallados la evaluación del Preprocesador de Textos

Dominio	**Aprobados**			**No aprobados**		
	Correctos	Con texto de más	**Total aprobados**	Incompletos	Incorrectos	**Total no aprobados**
Designaciones	87.69 %	7.69 %	**95.39 %**	3.07 %	1.54 %	**4.61 %**
Escalafón	80.51 %	12.98 %	**93.6 %**	6.4 %	0 %	**6.4 %**
Jurados de Ascenso	77.55 %	16.32 %	**93.88 %**	0 %	6.12 %	**6.12 %**

Por que el extractor falla con designaciones? (Observaciones hechas al hacer las pruebas que pueden ser útiles en el análisis de datos)

1. Las unidades de informacion no son precisas. A veces hay multiples designaciones en una misma linea.

2. Hay multiples designaciones que coinciden en un mismo día para una misma persona.

3. Hay ratificaciones, correcciones, posteriores a la designacoines que modifican el resultado que uno pensaria correcto.

4. Hay errores de tipeo por parte de la secretaria.

5. Hay casos "patologicos"que es imposible generalizar con expresiones regulares que no sean hechas a la medida.

6. Hay casos en los que los campos tienden a tener muchos valores null y al no encontrar la respuesta en el

mejor hit, se procede al segundo. Arreglar esto en el extractor? Hay muchos casos en los que un segundo match ayuda.

Conclusiones y recomendaciones

Destacar:

1. íngeniería del documento'. 2. Dependencia de un extractor más inteligente. 3. Que el software se hizo suponiendo la existencia de unidades de documentos. 4. Que el software (extractor) se hizo suponiendo que el numero de campos es contable.

Bibliografía

[1] Abad Mota Soraya, "Document interrogation: Architecture, information extraction and approximate answers", 2009.

[2] Aggarwal and Fellow, "A survey of uncertain data algorithms and applications", *IEEE*, vol. 21, pp. 609–623, 2009.

[3] D. KALASHNIKOV R. CHENG and S.PRABHAKAR, "Evaluating probabilistic queries over imprecise data", *Proc ACM SIGMOD*, 2003.

[4] J. CHEN R. CHENG and Xie X., "Cleaning uncertain data with quality guarantees", *Proceedings of the VLDB Endowmen*, vol. 1,1, pp. 722–735, 2008.

[5] Lipski Witold, "On databases with incomplete information", *Journal of ACM*, vol. 8,1, 1981.

[6] In-ho Kang and Kim GilChang, "Query type classification for web document retrieval", *Proc Sigir '03*, 2003.

[7] Segoufin Rocquenco and Viano, "Representing and querying xml with incomplete information", *PODS'01*, 2001.

Apéndices

Apéndice A

Ejemplos de archivos de configuración XML de Preprocesador de Texto y Extractor Focalizado

A continuación se presentan dos archivo de configuración del Preprocesador de Texto y del Extractor Focalizado del Módulo de Búsqueda Focalizada.

Código A.1: Archivo de configuración del Preprocesador de Texto con Dominio Designaciones

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

<Filesources>

    <Filesource>

        <Name>Designaciones</Name>

        <InputFilePath>/home/frandres/Eclipse/workspace/ExtractionModule/tests/
            Designaciones/rawDocuments/</InputFilePath>

        <OutputFilePath>/home/frandres/Eclipse/workspace/ExtractionModule/tests/
            Designaciones/preProcessedDocuments/</OutputFilePath>

        <RegExps>

            <RegExp>
                <Priority>1</Priority>
                <String>De acuerdo al numeral 7 del articulo 16(.*?)[\^0-9 ][23(?:IV)(?:
                    III)(?:II)][\.\t].*</String>
            </RegExp>

            <RegExp>
                <Priority>2</Priority>
                <String>.*(?:[Ii][Nn][Ff][Oo][Rr][Mm][Ee] [Dd][Ee][Ll] [Rr][Ee][Cc][Tt][
                    Oo][Rr]).*?([Dd][Ee][Ss][Ii][Gg][Nn][Aa][Cc][Ii][Oo\u00f3][Nn](?:[Ee
                    ][Ss]){0,1}.*?)[^0-9 ][23(?:IV)(?:III)(?:II)][\.\t].*</String>
            </RegExp>

        </RegExps>

        <readAllFiles>1</readAllFiles>

        <outputExtension>.txt</outputExtension>

    </Filesource>
</Filesources>
```

Código A.2: Fragmento del Archivo de configuración del Extractor Focalizado de Texto con Dominio Designaciones

```
<Configuration>
    <UnitRegExps>
        <UnitRegExp>^(.+?)$</UnitRegExp>
    </UnitRegExps>


    <MinimumHitRatio>.5</MinimumHitRatio>
    <DocumentsFilePath>/home/frandres/Eclipse/workspace/ExtractionModule/tests/
        Designaciones/preProcessedDocuments/</DocumentsFilePath>
    <FieldDescriptors>

        <FieldDescriptor>
            <FieldName>EsAsignado.calificacion</FieldName>
            <SpecificRegExps>
                <SpecificRegExp priority="1" >([Jj]ef.*?)(?:(?:a\s+partir\s+del{0,1}
                    \s)|(?:desde)|(?:hasta)|(?:por motivo)|(?:en sustitu)|(?:del\s\d)
                    |(?:al\s\d)|(?:\s+entre\s+el\s+\d)|(?:por\s+el\s+periodo\s+)).*</
                    SpecificRegExp>
                <SpecificRegExp priority="2" >([Cc]oordinador.*?)(?:(?:a\s+partir\s
                    +del{0,1}\s)|(?:desde)|(?:hasta)|(?:por motivo)|(?:en sustitu)
                    |(?:del\s\d)|(?:al\s\d)|(?:\s+entre\s+el\s+\d)|(?:por\s+el\s+
                    periodo\s+)).*</SpecificRegExp>

                <SpecificRegExp priority="13" >([Jj]ef.*)</SpecificRegExp>
                <SpecificRegExp priority="14" >([Cc]oordinador.*)</SpecificRegExp>

            </SpecificRegExps>
            <Weight>1</Weight>
            <isDate>0</isDate>
        </FieldDescriptor>
    </FieldDescriptors>

</Configuration>
```