IEEE Std. 830-1998

Descripción del Sistema

Proyecto: snoicd-codex (Sistema de búsqueda optimizada para SNOMED CT e ICD)

Autores:

Guillermo Facundo Colunga

Jose Emilio Labra Gayo

Aquilino A. Juan Fuerte

Grupo de Investigación de Web Semántica WESO,
Universidad de Oviedo,
2019.

Revisión 1.0, 23 de febrero de 2019.

SUMARIO

1. Introducción

El documento actual recoge la descripción del sistema implementado para un sistema optimizado de búsqueda para los sistemas de clasificación de términos médicos SNOMED CT e ICD (en sus versiones 9 y 10).

Tanto SNOMEC CT como ICD son dos sistemas empleados para realizar clasificación de términos médicos.

1.1. SNOMED CT

SNOMED CT es una ontología que define un sistema de conocimiento sobre términos médicos como enfermedades, episodios o medicamentos. Es uno de los sistemas más empleados en el mundo para los sistemas electrónicos de salud. Ayuda al personal sanitario a identificar de forma inequívoca los términos que éstos quieren expresar.

En este sistema se definen relaciones entre términos, de forma que se puede recorrer la ontología como si fuese un grafo y ver la relación entre términos. Por ejemplo el término fiebre está relacionado como padre de otro término que es fiebre por dengue.

1.2. ICD

ICD por la contra no es una ontología como tal, si no que es más un sistema de clasificación de los mismos términos médicos que SNOMED CT. ICD fue ideado en un principio como un sistema de clasificación de términos médicos para ser usado en estadísticas e informes. Por lo tanto no infiere ningún conocimiento, al contrario de lo que puede hacer una ontología como SNOMED CT. Sin embargo y pese a la gran expansión de SNOMED CT muchos profesionales y sistemas aún se decantan por el sistema de clasificación ICD ya que es más lógico para ellos.

En el caso de este sistema sólo se implementará con las versiones 9 y 10 de ICD por requisito de la empresa contratante IZERTIS.

1.3. Objetivo del sistema

El sistema resultante será desplegado en un contenedor de docker y deberá de exponer un servicio unificado y optimizado de búsqueda para los sistemas de clasificación descritos a través de protocolo REST y con arquitectura de API.

Por lo tanto deberán de realizarse las siguientes tareas principales:

A) Relacionar ambos sistemas de clasificación.

B) Comprimir los sistemas de clasificación para ajustarse al tamaño de una imagen de docker.

C) Escoger el sistema de gestión de datos más adecuado para el objetivo.

D) Implementar un sistema de búsqueda unificada y optimizada.

E) Exponer los servicios de búsqueda a través de API REST.

2. Relación de los sistemas de clasificación

Para ser capaces de implementar un sistema de búsqueda unificado se deberán de relacionar los términos de ambos sistemas de clasificación ya que en ambos casos se clasifican los mismos términos pero de forma distinta.

A continuación se explican las sincronizaciones de forma individual, como tenemos dos sistemas de clasificación diferentes pero uno de ellos tiene dos versiones deberemos de emplear dos mapeos distintos para tener los sistemas sincronizados.

2.1. Mapeo de SNOMED CT a ICD 9

Para realizar la sincronización entre SNOMED CT e ICD 9 debemos de explorar cómo se representa un término en cada uno de estos sistemas. Sabemos que en el modelo de SNOMED CT tenemos un campo identificado como "concept_id" que contiene el código SNOMED de cada término. En ICD 9, sin embargo, tenemos un campo identificado como "code" que es el que contiene el código en ICD 9. Por lo tanto para sincronizar ambos sistemas debemos de usar una tabla de conversión de códigos ICD 9 a SNOMED CD y viceversa. Por suerte en la página web de UMLS podemos encontrar este archivo. También está disponible en https://github.com/thewilly/snoicd-codex/blob/master/data/snomed-icd9-map.json .

2.2. Mapeo de SNOMED CT a ICD 10

De la misma forma que en el caso anterior se empleará una tabla que relacionará ambos sistemas a través del código de SNOMED y el código ICD. En este caso la tabla también se puede encontrar en la página web de UMLS o en https://github.com/thewilly/snoicd-codex/blob/master/data/snomed-icd10-map.json .

3. Compresión y unificación de los modelos de datos

En su versión estándar SNOMED CT ocupa unos 6 Gb, mientras que ICD llega a los 2 Gb. En total serían unos 8 Gb de datos a almacenar en una imagen de docker, como se puede ver esto terminaría dando problemas debido al gran tamaño de la imagen (tardaría mucho en desplegar, se tendría que ejecutar en máquinas específicas, etc). Por lo tanto comprimir el sistema se convierte en una necesidad si queremos tener una imagen de docker "ágil" y funcional. A demás para mejorar el rendimiento del sistema de búsqueda un único modelo de datos es mejor que varios ya que evitamos tener que realizar la búsqueda en varios sitios y posteriormente agregar los resultados.

3.1. Compresión

Para realizar la compresión se han eliminado todos aquellos datos que no son relevantes para el sistema de búsqueda pedido. Por lo tanto de ambos modelos previos, SNOMED CT e ICD, se extraen solamente los códigos y descripciones de los términos. De esta forma se reduce SNOMED CT de 6 Gb a 0,5 Gb aproximadamente. E ICD de 2 Gb a 0,1 Gb. Teniendo un tamaño total de modelos de datos de 0,6 Gb.

3.2. Unificación

Una vez reducidos los modelos de datos y con los distintos archivos de sincronización obtenidos en el punto 2 se procede a unificar los dos modelos en un único modelo reducido. Dicho modelo tendrá los siguientes campos y estructura.

```
[{
  "code": String,
  "terminology_name": (snomed | icd_9 | icd_10),
  "descriptions": [
    String,
    ... ],
  "related_codes": [{
    "code": String,
    "terminology_name": (snomed | icd_9 | icd_10),
    "descriptions": [
      String,
      ... ]
    },... ]
  }, ...
]
```

De esta forma se obtiene un único modelo de datos con representación JSON dónde se encuentran los dos modelos anteriores. Dicho modelo tiene un tamaño estimado de 0,6 Gb y tiene como índices el código y los diferentes strings de las descripciones.

4. Elección del sistema de gestión de datos

Una vez tenemos nuestro dominio definido necesitamos un sistema que lo almacene y nos permita realizar consultas con un rendimiento elevado. Por ejemplo 1.0 segundos por consulta no resultaría aceptable, ni tampoco un sistema que expandiera los 0,6 Gb del modelo a 2Gb. Para realizar la elección del sistema se consideraron las siguientes opciones.

4.1. Base de datos RDF

Los modelos en RDF tienen como premisa la simplicidad y la unificación. Sin embargo después de probar el sistema BlazeGraph se concluye que no es el sistema más rápido para el tipo de consultas que se necesitan hacer, a demás tras eliminar las relaciones en el proceso de compresión ya no tiene sentido emplear un sistema de datos puramente orientado a mantener relaciones.

4.2. Base de datos en grafo

Debido a la topología de SNOMED CT se exploró también la posibilidad de emplear un sistema como Neo4J, sin embargo por los mismos motivos que la base deditos RDF, en concreto la eliminación de relaciones, este modelo de datos dejó de tener sentido.

4.3. Base de datos SQL - Relacional

Este modelo es sin duda uno de los más robustos en lo que a tecnología se refiere, sin embargo cuando se aplica a un modelo de datos dinámico, como es el caso, no es el que más flexibilidad ofrece, por eso se deja como segunda mejor opción.

4.4. Base de datos documental

Teniendo en cuenta que se han eliminado las relaciones y que cada término guarda todas sus sincronizaciones en los otros sistemas de clasificación se puede entender que un término es un documento aislado y por tanto emplear un sistema de gestión de bases de datos de tipo documental como MongoDB. En este caso se usó este sistema en concreto y los resultados fueron increíbles. El tamaño del modelo de datos no se vio incrementado en ningún caso y gracias a los múltiples índices se podían acelerar las consultas de forma exponencial sin que apuntase la latencia al añadir más documentos. Es por esto que MongoDB fue elegido como sistema de gestión de datos.

5. Sistema de búsqueda

En este punto ya tenemos un único modelo de datos, reducido e indexado para realizar búsquedas de forma optimizada. Por lo tanto lo único que resta para implementar el sistema de búsqueda es definir qué se va a poder buscar y cómo. En este caso se definen dos búsquedas principales:

A) Por código.

B) Por palabras contenidas en la descripción. Esto es que si se busca la palabra “a” deben aparecer todos aquellos que contengan la palabra “a“ y se se busca “a + b" se deberán de mostrar todos aquellos documentos que contengan “a y b”.

5.1. Búsqueda por código

Para realizar la búsqueda por código en MongoDB, teniendo en cuenta que el campo fue indexado previamente la query correspondiente sería tan sencilla como:

```
db.getCollection('concepts').find({code:"<<SNOMED_OR_ICD_CODE>>"})
```

5.2. Búsqueda por palabra/s en descripción

Para realizar la búsqueda de palabras que están en alguna definición de algún término se empleará la siguiente query:

```
db.getCollection('concepts').find({ $text: { $search: \"word1\" \"wordi\" } })
```

La query anterior da como resultado una lista de documentos que contienen las palabras dadas en un índice de texto. Como parámetros de la búsqueda (word) se pueden pasar de 1 a n, cuantas más palabras más estricta será la búsqueda.

6. Servicio API REST

Para poder usar el sistema descrito anteriormente a través de llamadas REST es necesario exponer el servicio WEB. Para la implementación de dicho servicio, por restricciones de la empresa, se usará Java 1.8 y SpringBoot.

6.1. Arquitectura del servicio

El servicio implementado en SpringBoot presenta una arquitectura MVC compuesta por los siguientes módulos.

6.1.1. Repositorios

Aquí se encuentra la clase principal ConceptsRepository, que es la encargada, a través de SpringBoot, de mantener la persistencia de los conceptos con MongoDB.

6.1.2. Tipos

Se definen como tipos todas aquellas entidades que serán empleadas como objetos POJO por el resto del sistema para transmitir o representar información. En este módulo tenemos los conceptos, los conceptos simples y las respuestas a las queries realizadas.

6.1.3. Servicios

Los servicios se encargan de manejar la lógica sobre los datos delegados por los controladores. En este caso tenemos el servicio de conceptos que nos resuelve los dos tipos de búsqueda definidos, por código y por palabras en descripción.

6.1.4. Controladores

Por último, tenemos los controladores, que se encargan e exponer los distintos servicios REST a través de endpoints configurados. En este caso tenemos el endpoint "/api/search" que con el parámetro "q" (query) y el parámetro opcional "filter" delega la búsqueda al servicio correspondiente.

6.1.5. Búsqueda

Para no cargar los servicios con lógica se crea un módulo de búsqueda que define cada uno de los tipos de búsqueda posibles para el sistema.

6.2. Diagrama UML