Mejoras al algoritmo de cálculo de Deltas ($\triangle$)

Ing. Martín G. Casatti
e-mail: mcasatti@gmail.com

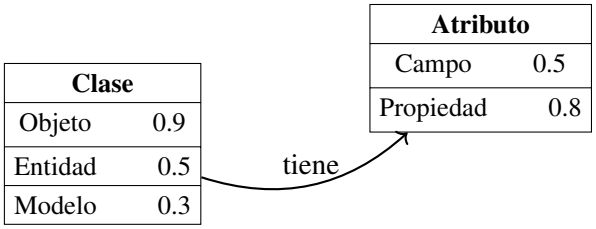
8 de julio de 2019

Se presenta el algoritmo original para el cálculo de Δ, o sea la distancia entre un concepto dado en una respuesta y el mismo concepto en la respuesta base, así como un algoritmo alternativo, más eficiente, para realizar dicha tarea.

1. Introducción

El primer paso para la interpretación de la respuesta a una pregunta de examen, provista por un alumno, es interpretar los conceptos que intervienen en la misma sin coartar su capacidad expresiva. Ahí nos encontramos con un problema omnipresente en el idioma castellano, **las equivalencias**.
Existen múltiples palabras en castellano que pueden utilizarse para expresar el mismo concepto. Y si bien las relaciones conceptuales en una materia técnica como es PPR son concretas y simples, el alumno puede utilizar diversos sinónimos y términos equivalentes para expresar una respuesta válida.
Existe además la complejidad agregada de que si bien un término equivalente puede expresar el concepto que se pide como respuesta, no necesariamente los términos tienen que ser intercambiables como iguales.
Surge de esta manera la necesidad de ponderar qué tan exacta es una equivalencia, con respecto al concepto que representa.
La gestión de equivalencia tiene entonces dos objetivos complementarios:

1. Registrar términos equivalentes asociados a un concepto dado
2. Indicar el grado de exactitud con el que representan a dicho concepto



2. Métodos

La base de datos OrientDB[2] permite almacenar, asociado a un nodo, diversos tipos de datos además de los tipos estándar string, entero, float, etc.
Dentro de los tipos compuestos que se pueden almacenar, uno en particular es adecuado para el tratamiento de las equivalencias. El tipo "Embedded Map" permite el almacenamiento de pares `<String, Object>` que es adecuado para la representación de las equivalencias, según el esquema mencionado anteriormente (`<String,Float>`).
Primeramente se deben crear las estructuras correspondientes con las siguientes sentencias (el SQL utilizado es el dialecto específico de OrientDB) tal como se ve en el Listado1.

```
-- Se crea la clase Concepto que hereda de V (Vertex. Nodo)
create class Concepto extends V;
-- Se crea la propiedad Nombre, de tipo string
create property Concepto.Nombre string;
-- Se crea la propiedad Usuario, de tipo string
create property Concepto.Usuario string;
-- Se crea la propiedad Actualidado, de tipo Date
create property Concepto.Actualizado Date;
-- Se crea la propiedad Equivalencias, un Map Embebido en el Nodo,
-- con Key = string y Value = Float
create property Concepto.Equivalencias EMBEDDEDMAP float;
```

Listing 1: Creación de estructuras

Este esquema permite la gestión de conceptos generales y de conceptos que incluyan a su vez su propia lista de equivalencias.
La búsqueda de conceptos es directa a partir de la instrucción SQL `select` que se comporta de manera muy similar a un SQL estándar (Listado 2).

```
select from Concepto where Nombre = 'POO';
```

Listing 2: Sentencia SELECT simple

Para que el esquema de equivalencias pueda ser consultado "al mismo tiempo" que los datos del nodo se debe ampliar la sintaxis del SQL[3] para que incluya una búsqueda dentro del diccionario embebido en cada nodo.
La base de datos soporta este tipo de operaciones mediante la siguiente sintaxis:

```
select from Concepto where Equivalencias containskey 'MODELO';
```

Listing 3: Sentencia SELECT con diccionario

La clave en el listado 3 radica en la palabra clave **containskey** que es la encargada, al actuar sobre la propiedad Equivalencias, de encontrar una clave en el diccionario que coincida con el parámetro, en este caso 'MODELO'.

```
+----+-------+-----------------------------------------------------------+
|#   |Nombre |Equivalencias                                              |
+----+-------+-----------------------------------------------------------+
|0   |OBJETO |{MODELO=0.5, PLANTILLA=0.6, ENTIDAD=0.9, CLASE=1.0, COSA=0.1}|
|1   |ENTIDAD|{OBJETO=0.9, MODELO=0.5}                                   |
+----+-------+-----------------------------------------------------------+
```

Combinando ambos métodos podemos incluir una sentencia que consulte tanto los conceptos puros (a partir de la propiedad Nombre) como las posibles equivalencias.

```
select Nombre,Equivalencias from Concepto
where
```

```
Equivalencias containskey 'MODELO'
or
Nombre = 'MODELO'
```

Listing 4: Sentencia SELECT compuesta

```
+----+-------+----------------------------------------------------------------+
|#   |Nombre |Equivalencias                                                   |
+----+-------+----------------------------------------------------------------+
|0   |OBJETO |{MODELO=0.5, PLANTILLA=0.6, ENTIDAD=0.9, CLASE=1.0, COSA=0.1}|
|1   |ENTIDAD|{OBJETO=0.9, MODELO=0.5}                                        |
|2   |MODELO |                                                                |
+----+-------+----------------------------------------------------------------+
```

Listing 5: Resultado de la consulta por 'MODELO'

En la figura precedente podemos ver que la consulta devolvió tres conceptos, uno con una concordancia exacta en el nombre 'MODELO' y dos que se encontraron dentro de la lista de equivalencias, una (OBJETO) con una equivalencia ponderada en 0.5 y otra (ENTIDAD) con una equivalencia ponderada en el mismo valor.

3. Resultados

Podemos apreciar que la implementación de listas de equivalencias así como las consultas asociadas a las mismas no reviste gran dificultad en un nivel básico, debido principalmente a que la base de datos incluye de manera nativa mecanismos que permiten almacenar y consultar datos con estructuras complejas anidadas.

4. Discusión

Un problema importante que tiene la búsqueda en el diccionario de equivalencias es que no permite coincidencias parciales tales como las que se logran con el operador 'LIKE' en SQL estándar.
Hay que mencionar que la base de datos si permite el uso de dicho operador sobre propiedades que no son de tipos complejos, como por ejemplo el Nombre.
Una consulta del tipo

```
select from Conceptos where Nombre like 'MODELO%'
```

es perfectamente válida.

5. Referencias y bibliografía

Referencias

[1] OrientDB, `http://orientdb.com/`

[2] OrientDB Data Types, `http://orientdb.com/docs/last/Types.html`

[3] OrientDB Select Syntax, `http://orientdb.com/docs/last/SQL-Query.html`