BibTeX Bibliography Index Maker: Meeting Notes

Ramon Xuriguera

07-04-2010

1 Validesa

Millora del sistema com es validen les referències extretes. Ara s'utilitza un número decimal que indica la validesa. Aquest número es calcula a partir d'un llistat de pesos assignats als camps que es volen validar. S'estableix un llindar a partir de la qual la referència es considera vàlida.

2 Base de dades

Simplificat la base de dades per tal de guardar només la informació necessària. A més, s'ha afegit un *timestamp* per indicar el moment d'extracció d'una referència. Això permetrà generar estadístiques sobre quins *wrappers* han deixat de funcionar.

3 Wrapper induction

Hem tingut bastants problemes a l'hora de decidir com estructurar els exemples i quins passos seguir a l'hora de generar les regles. Finalment, el flux d'informació és el següent:

- Un `ExampleManager` s'encarrega d'obtenir els exemples de la base de dades, cercar el codi font a Internet i generar objectes del tipus `Example`.
  Com que tots els exemples són del mateix domini i segurament totes les pàgines seran enviades pel mateix servidor, s'ha establert un temps d'espera entre petició i petició per evitar bloquejos. Com que l'execució de la generació de *wrappers* només cal fer-la de tant en tant, aquest temps afegit ens el podem permetre.
  De cada pàgina volem extreure diferents elements de text. Per tant, crearem un exemple per cadascun dels elements i els agruparem segons el camp de la referència que descriuen. Per exemple, si volem extreure referències de `http://portal.acm.org`, tindrem:
  - x exemples per extreure el títol.
  - y exemples per extreure la revista.
  - etc.

  Una altra millora interessant d'aquest punt és la validació de l'exemple abans d'intentar generar el *wrapper*. Comprovem que dins el document obtingut, hi ha la informació que estem buscant. Si no és així, marquem l'exemple com a invàlid a la base de dades per tal que no s'utilitzi en pròximes execucions.
  Es pot marcar tot l'exemple com a invàlid o bé només alguns camps. D'aquesta manera tindrem més exemples per generar les regles dels camps correctes.

- Una vegada tenim els conjunts d'exemples per cadascun dels camps a extreure, dividim els grups en conjunts d'entrenament i validació.

- Utilitzem el conjunt d'entrenament per crear les regles i les provem amb el conjunt de validació. Si no són correctes, podem triar conjunts entrenament/validació diferents[1]

3.1 Generació de regles

3.1.1 Ruta dins del document HTML

Tal i com ja vam comentar a la trobada anterior, la ruta està composta per cada element de l'HTML necessari per desambiguar l'element que conté la informació que volem. Per exemple:

```
[(u'table', {u'width': u'100%'}, 7), (u'tr', {}, 0)]
```

Calculem la ruta pels diferents exemples del conjunt d'entrenament. Si tot funciona com ho esperem, les rutes dels diferents exemples haurien de ser totes iguals. Si no és així, generalitzem el patró resultant per tal de cobrir tots els exemples:

- Si la ruta té el mateix número d'elements (la mateixa profunditat dins de l'arbre), deixem alguns dels paràmetres com a `None`. per exemple:

```
[[[None, {u'class': u'small-text'}, 1],
  [u'span', {}, None]]]
```

- Si les rutes tenen diferent llargada, contemplem les dues possibilitats.

```
[ 1.[[None, {u'class': u'small-text'}, 1],
     [u'span', {}, None]],

  2.[[u'span', {u'class': u'big-text'}, 5]]
]
```

3.1.2 Expressió regular dins de l'element

De la mateixa manera que WHISK (Stephen Soderland 1999), si el contingut de l'element no només conté la peça d'informació que estem buscant, l'haurem d'extreure utilitzant expressions regulars.

Per crear aquestes expressions regulars seguim un procés similar al que hem fet servir per les rutes. Agafem tots els patrons dels diferents exemples i creem l'expressió regular que els cobreixi tots. Fem ús de la biblioteca `difflib` per comparar seqüències de caràcters de dues en dues:

- Al principi, agafem qualsevol dels exemples, extraiem la informació i utilitzem el text restant com a patró *general*.

[1] Aquesta funcionalitat encara no està implementada

- Anem agafant la resta de cadenes dels altres exemples i ho anem comparant amb el patró *general*. Si:
  - La ratio[2] entre les dues seqüències és superior a un llindar esablert (e.g. 0.75), considerem que el contingut dels dos exemples és el mateix, però amb elements que varien. Per exemple:

    ```
    Volume 70, Issue 16-18; (October 2008)
    Volume 22, Issue 21-33; (January 2007)
    ```

    Suposem que extraiem la informació corresponent als anys: 2008 i 2007. Llavors ens queden els patrons (sense escapar):

    ```
    Volume 70, Issue 16-18; (October (.*))
    Volume 22, Issue 21-33; (January (.*))
    ```

    Ara, les cadenes encara no són iguals, però tenen una similaritat superior a un *threshold* de 0.75. Per tant, provem de generalitzar els patrons. Mirem els blocs que no coincideixen i els substituïm per una expressió regular. Finalment obtenim:

    ```
    Volume (?:.*), Issue (?:.*)1(?:.*)-(?:.*); ((?:.*)r
        (?:.*) (.*))
    ```

    Com es pot veure, aquest sistema té alguns problemes. A nosaltres ens interessaria que l'*1* de *16* i *21* no s'hagués comptat com una coincidencia. El mateix amb la *r* de *October* i *January*. Caldrà millorar la manera de generalitzar aquestes cadenes de caràcters, o bé anar en compte a l'hora d'escollir els exemples que s'utilitzen.
  - Si la ratio és inferior al llindar, considerem que les cadenes són massa diferents i contemplem la unió (|)

Depenent dels resultats que s'obtenen, podem millorar aquest procediment de manera que a l'hora de substituir la informació a extreure per l'expressió regular, limitem més les possibilitats. Per exemple, en el cas anterior, hauríem pogut tenir `([0-9]{4})` enlloc de `(.*)`.

[2] Número entre 0 i 1 que mostra la semblança entre les dues seqüències