PROGETTO SII 2015-2016

Classificazione di post per sezione pertinente mediante features estratte da Dbpedia.

Simone Gasperoni
Antonio D'Innocente

1. Introduzione

Progetto scelto. Tecniche di filtraggio dei contenuti più rilevanti nel Web 2.0 (per 2 persone).

Ci siamo posti l'obiettivo di sperimentare un sistema di classificazione di post utente mediante features estratte da Dbpedia.

La classificazione ha lo scopo di assegnare ad ogni post la sezione del forum più pertinente in base al corpo del testo.

Abbiamo scaricato diversi archivi di post, divisi per sezione, dal dump pubblico di StackExchange ed abbiamo selezionato le cinque sezioni: *anime, beer, economics, fitness* e *robotics* per la sperimentazione.

Da questi post abbiamo quindi estratto delle feature mediante Dbpedia Spotlight ed interrogazioni SPARQL.

Infine abbiamo testato i vari modelli di classificatori utilizzando Weka.

2. Dati di partenza

Obiettivo. Ottenere più insiemi di post rilevanti per una determinata comunità di utenti (forum). Selezionare tematiche abbastanza diverse tra loro, in modo da sfruttare le caratteristiche di Dbpedia.

Inizialmente era stato creato un parser che si connetteva ad un forum, scaricando automaticamente i post per ogni sezione (progetto forumExtractor), tuttavia le sezioni del forum trattavano di diversi tipi di hardware per computer, argomenti troppo collegati tra loro per essere usati in questo progetto.

Per condurre le sperimentazioni e creare il dataset è stato necessario disporre di un ampio numero di post afferenti a topic (sezioni) abbastanza diversi tra loro.
I dati sono stati presi dal data dump degli archivi di Stack Exchange (https://archive.org/details/stackexchange).
Ciascun archivio raggruppa tutti i primi post riguardanti un determinato argomento.
Nel file Post.xml sono contenute le informazioni sui post presenti nella sezione.

```
  <row Id="2" PostTypeId="1" AcceptedAnswerId="50" CreationDate="2012-10-23T19:42:56.030"
Score="13" ViewCount="2536" Body="&lt;p&gt;I've got some hobby servos (&lt;a href=&quot;http://
www.servodatabase.com/servo/power-hd/hd-1501mg&quot;&gt;Power HD 1501MGs&lt;/a&gt;) and I'd like to
be able to control them (via an Arduino) so they will either go to the angle I set, or put them in
a 'free running' mode, where the load will take them wherever it goes.&lt;/
p&gt;&#xA;&#xA;&lt;p&gt;Is this even possible, or am I just going to end up stripping the gears?
&lt;/p&gt;&#xA;&#xA;&lt;p&gt;My first thought is to simply kill the power to the servo, but the
force required to move them in that state is more than I'd like.&lt;/p&gt;&#xA;&#xA;&lt;p&gt;If it
is possible, am I looking at a hardware change, or could I do it in software?&lt;/p&gt;&#xA;"
OwnerUserId="12" LastEditorUserId="37" LastEditDate="2013-05-20T14:52:48.300"
LastActivityDate="2013-05-20T14:52:48.300" Title="How can I modify a low cost hobby servo to run
'freely'?" Tags="&lt;control&gt;&lt;rcservo&gt;" AnswerCount="4" CommentCount="2"
FavoriteCount="1" />
  <row Id="3" PostTypeId="1" CreationDate="2012-10-23T19:43:37.690" Score="22" ViewCount="420"
Body="&lt;p&gt;&lt;a href=&quot;http://www.oricomtech.com/projects/leg-time.htm&quot;
rel=&quot;nofollow&quot;&gt;http://www.oricomtech.com/projects/leg-time.htm&lt;/a&gt; lists three
gaits:&lt;/p&gt;&#xA;&#xA;&lt;ul&gt;&#xA;&lt;li&gt;the tripod&lt;/li&gt;&#xA;&lt;li&gt;wave,
and&lt;/li&gt;&#xA;&lt;li&gt;ripple.&lt;/li&gt;&#xA;&lt;/ul&gt;&#xA;&#xA;&lt;p&gt;Can these be
improved, or can their relative pros and cons be altered, and are there other gaits worth
considering?&lt;/p&gt;&#xA;" OwnerUserId="34" LastEditorUserId="134"
LastEditDate="2014-09-03T20:22:11.577" LastActivityDate="2014-09-03T20:22:11.577" Title="What
useful gaits exist for a six legged robot, and what are their pros and cons?"
Tags="&lt;gait&gt;&lt;walk&gt;" AnswerCount="2" CommentCount="1" FavoriteCount="4" />
  <row Id="4" PostTypeId="1" AcceptedAnswerId="52" CreationDate="2012-10-23T19:43:47.140" Score="0"
ViewCount="1386" Body="&lt;p&gt;I am looking for a starting point for my project, preferably using
popular systems (ones there is a lot of support for). I have an Arduino Uno, a Raspberry Pi, and a
lot of willpower :) Anyone here built a project using the systems above?&lt;/
p&gt;&#xA;&#xA;&lt;p&gt;Observation: I'd like to start with a simple line-following vehicle and
build up afterwards.&lt;/p&gt;&#xA;" OwnerUserId="32" LastEditorUserId="37"
LastEditDate="2012-11-14T15:00:39.723" LastActivityDate="2012-11-14T15:00:39.723" Title="Good
Microcontrollers/SOCs for a Robotics Project"
Tags="&lt;microcontroller&gt;&lt;arduino&gt;&lt;raspberry-pi&gt;" AnswerCount="7" CommentCount="2"
```

L'elemento **row** identifica il post e nell'attributo **Body** è presente il testo inserito dall'utente. Sono presenti anche altri dati, come titolo del post e conteggio di risposte, che non vengono utilizzati per la classificazione.

Le cinque sezioni (forum) scelti sono stati: *anime, beer, economics, fitness, robotics.*
La classificazione dovrà prendere in input un post e decidere in quali di queste cinque sezioni il post deve essere inserito.

3. Estrazione delle features

Obiettivo. Estrazione delle features (a partire dai file xml) e creazione di un dataset per la classificazione.

È necessario scandire i file xml e, per ogni post, generare la relativa riga di features in formato .arff.
Ci avvaliamo di spotlight, estraendo le uri, quindi ricaviamo le feature da un insieme di relazioni rilevanti e da cinque insiemi di parole chiave, ciascuno contenente parole (stem) attinenti ad uno specifico topic.
Dato un post, ed un insieme di parole chiave, sintetizziamo il procedimento nei seguenti passaggi.

- Estrazione delle uri.
- Generazione delle feature ricavate dalle relazioni.
- Generazione delle feature ricavate dalle parole chiave.
- Generazione di altre feature e scrittura della linea nel file arff.

3.1 Introduzione a Dbpedia e query SPARQL

Dbpedia può essere vista come un grafo orientato. Ogni entità dbpedia è collegata ad un valore (che può essere un valore numerico, una stringa o un'altra entità) da una relazione (arco orientato).
Tramite SPARQL è possibile interrogare dbpedia fornendo la tripla:

nodo1 – relazione – nodo2.

È sufficiente omettere uno qualsiasi (o più) degli elementi che si vuole ricercare, SPARQL si occuperà di estrarre tutte le triple che possono “matchare” con la query descritta.

La posizione di nodo1 e nodo2 nella query stabilisce la direzione dell'arco (relazione), in questo caso un arco uscente da nodo1 ed entrante in nodo2. Invertire l'ordine dei nodi equivale a invertire il verso della relazione, in generale i risultati saranno diversi.

3.2 Estrazione delle uri

Prima di qualsiasi elaborazione ci siamo posti il problema di estrarre le URI (dbpedia) da ciascun post (corpo testuale), abbiamo utilizzato le funzionalità delle API REST messe a disposizione da Dbpedia Spotlight (https://github.com/dbpedia-spotlight). Dbpedia Spotlight offre un servizio web che permette di determinare le URI Dbpedia relative ad un testo con un certo livello di confidenza.
Il programma client java che richiede l'elaborazione dei testi per restituire le URI è stato realizzato da Pablo Mendes e Max Jakob, abbiamo trovato questo programma all'interno della repositoy git di Dbpedia Spotligth, abbiamo modificato il metodo evaluate in modo che prendesse in input il corpo testuale del post e restituisse un HashMap di stringhe contenente la collezione di URI ottenute.

```
public HashSet<String> evaluate(String inputFile) throws Exception {
   return evaluateManual(inputFile, 0);
}
```

Una volta ottenuta questa collezione di URI abbiamo proceduto ad ulteriori elaborazioni per la determinazione delle feature.

3.3 Features dalle relazioni.

Nel contesto del nostro lavoro abbiamo individuato delle relazioni rilevanti per determinati topic. Dati dei post attinenti a determinati topic, le entità dbpedia estratte tendono ad avere alcune relazioni significative, che possono esse stesse aiutare nella classificazione. Ad esempio da molti post della sezione anime si estraggono uri che contengono le relazioni **dbp:episodesList** e **dbp:writer**, mentre da molti post della sezione fitness si ricavano analogamente molte occorrenze della relazione **is dbp:sport of**.

Raggruppiamo quindi queste relazioni rilevanti individuate in 3 macro-gruppi: relazioni che compaiono frequentemente in *fitness*, relazioni che compaiono frequentemente in *beer* e relazioni che compaiono frequentemente in *anime*.

Quando generiamo le feature per il nostro post, per ogni uri estratta con spotlight eseguiamo delle query SPARQL per contare il numero di occorrenze di relazioni relative ad un macro-gruppo. Utilizziamo quindi questo conteggio (normalizzato in seguito) come feature per aiutarci a classificare correttamente il post.

Per i topic *economics* e *robotics* non sono state individuate relazioni particolarmente significative, pertanto non creiamo nessun gruppo di relazioni relativo a queste due classi.

3.4 Features dalle parole chiave

Dato uno qualsiasi dei topic scelti, abbiamo individuato un gruppo di relazioni “comuni”, come **dct:subject** e **is dbo:wikiPageRedirects of**, che puntano a risorse le cui uri spesso contengono molte occorrenze dello stesso set di parole.

Per ciascuno dei cinque topic scelti abbiamo individuato un insieme di parole chiave (stem) che occorrono molto spesso all'interno delle uri delle risorse puntate dalle relazioni “comuni”.

Usiamo quindi questi cinque gruppi di keywords come cinque features, contando le occorrenze delle parole chiave attinenti a ciascuno dei determinati topic.

Per ogni uri estratta da un determinato post procediamo come segue:

- Eseguiamo delle query sparql per ritrovare tutte le uri puntate da queste relazioni comuni.
- Prendiamo solamente la parte descrittiva dell'uri e le aggiungiamo ad una lista “blob di stringhe”.

Una volta ottenuto l'intero “blob” per un deterinato post, confrontiamo ogni parola chiave con ciascuna delle stringhe nel blob e, ogni volta che c'è un matching (la parola stem è contenuta in una delle uri) aumentiamo il conteggio per il gruppo di parole chiave a cui apparteneva la parola confrontata.

Le nostre cinque feature saranno quindi il numero di occorrenze (normalizzato) delle keywords relative a ciascuno dei cinque argomenti.

3.5 Altre Features, scrittura del file .arff e normalizzazione

Le altre due features che abbiamo utilizzato sono state lunghezza del post e numero di entità dbpedia rilevato (uri estratte).

Una volta che il set di feature è stato correttamente istanziato, la scrittura della linea del file .arff è banale, si scorre l'elenco ordinato di features (implementato come dizionario collegato) trascrivendo i valori sul file, quindi si inserisce la classe.

Poichè le interrogazioni Dbpedia e SPARQL possono richiedere molto tempo (circa 4 ore per la creazione di un dataset di soli 1000 post), il programma genera in formato .arff solo un conteggio non normalizzato di tutte le feature che abbiamo estratto, la normalizzazione è stata effettuata separatamente con un piccolo programma che fa un parsing che riscrive una versione normalizzata delle feature. Questo approccio ci ha garantito una maggiore flessibilità, nel caso avessimo avuto bisogno di provare diversi metodi di normalizzazione.

I valori delle feature relative al conteggio delle relazioni rilevanti per topic sono stati normalizzati in [0,1], così come i valori relativi al conteggio delle parole chiavi attinenti a ciascun topic.

4. Sperimentazioni su Weka

Obiettivo. Sperimentare le feature scelte con un algoritmo di classificazione.

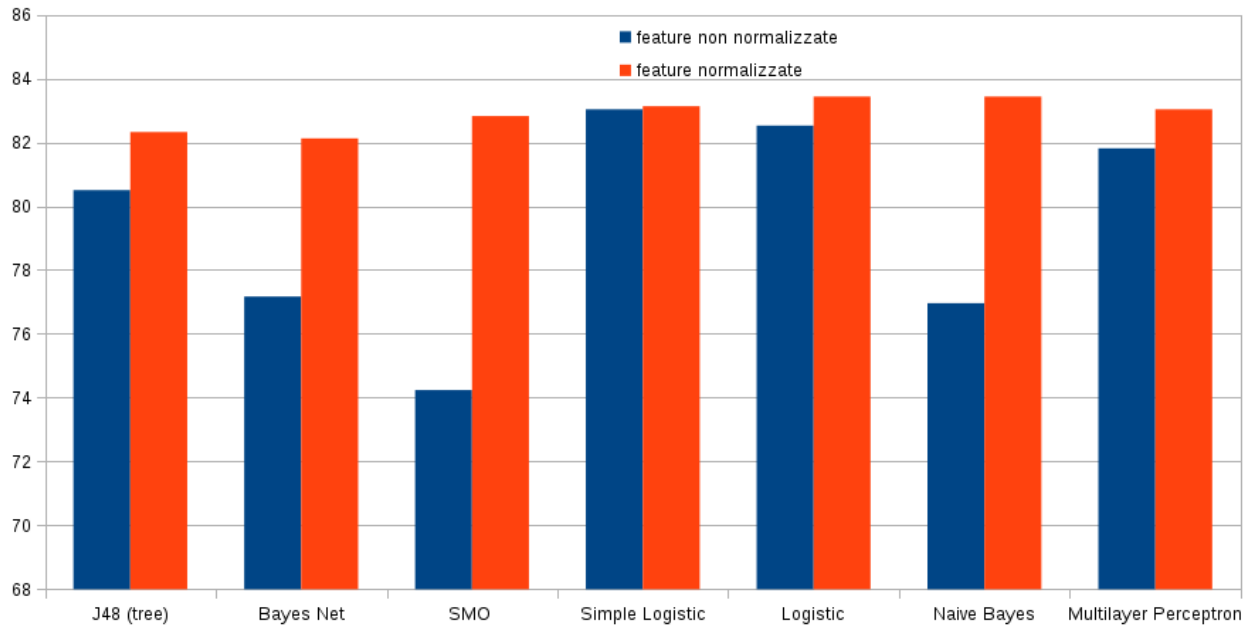
Per quanto riguarda la classificazione abbiamo eseguito alcuni k-fold-cross-validation-test (k=10) su diverse tipologie di modelli ottenendo una precisione a cavallo tra il 75% e l'84% .
Abbiamo fatto i test con il training set ed un altro training set ottenuto con i valori delle feature normalizzati tra 0 e 1. Per valutare la precisione abbiamo utilizzato Weka 3.6.0, caricando i file arff ed eseguendo gli addestramenti mediante Workbench.

I modelli che abbiamo provato sono i seguenti:

- SMO
- SIMPLE LOGISTIC
- NAIVE BAYES
- MULTILAYER PERCEPTRON
- LOGISTIC
- J48
- BAYES NET

Riportiamo di seguito i valori delle precisioni ottenuti per ciascun modello:

MODELLO	**Feature non normalizzate**	**Feature normalizzate**
SMO	74.24%	82.82%
SIMPLE LOGISTIC	83.03%	83.13%
NAIVE BAYES	76.96%	83.43%
MULTILAYER PERCEPTRON	81.81%	83.03%
LOGISTIC	82.52%	83.43%
J48	80.50%	82.32%
BAYES NET	77.17%	82.12%



Il training set che abbiamo utilizzato ha 990 esempi 198 per ogni classe di esempi (per ogni sezione del forum abbiamo creato 198 esempi). Come si può facilmente notare dal grafico, tutti i modelli di apprendimento che abbiamo usato traggono beneficio dalla normalizzazione dei valori delle feature.
Alleghiamo alla relazione i log dei test effettuati su weka (nella cartella logs weka).

5. Conclusioni

Il classificatore migliore (naive bayes) classifica con una precisione del 83.43% mediante la tecnica 10-fold cross-validation.
Le feature estratte, ad esclusione della lunghezza del post, provengono tutte da dbpedia.
Per alcuni post ci risulta difficile ottenere una classificazione accurata con le tecniche utilizzate, per esempio alcuni post in *robotics* che descrivono soccer robot vengono erroneamente classificati in *fitness,* come pure alcuni post in *anime* che descrivono robot vengono erroneamente classificati in *robotics*.