Jacopo Farina, 3 settembre 2010

relazione sul progetto di ingegneria informatica:

Assegnamento automatico delle pagine di wikipedia alle macrocategorie

1. Descrizione del problema

Wikipedia è un'enciclopedia online il cui contenuto è modificabile da chiunque. La versione in lingua inglese, *en.wikipedia*, è la più grossa e contiene (a inizio 2010) circa 3 milioni di articoli raggruppati in 500mila categorie.
Ogni articolo è inserito in una o più categorie, per esempio l'articolo *Italy* è categorizzato in *European countries, Countries of the Mediterranean Sea, G8 nations* e altre.
Ogni categoria può appartenere a sua volta a una o più categorie, e l'appartenenza alle categorie è, come il contenuto degli articoli, modificabile in qualsiasi momento. Il risultato è un grafo di appartenenza delle categorie molto elaborato e che cambia continuamente nel corso del tempo.
Le categorie non sono organizzate ad albero: possono esserci cicli, categorie non categorizzate e categorie senza sottocategorie. L'unico vincolo imposto dal software MediaWiki, su cui si basa Wikipedia, è che una categoria non può appartenere a se stessa.
Il problema che si vuole affrontare è quello di poter stabilire qual'è la categoria principale di un articolo dato un insieme di macrocategorie in cui dividere gli argomenti.

Uno studio svolto in precedenza da Aniket Kittur e Bongwon Suh[1] mostra come sia possibile dedurre questa informazione con un margine di errore non troppo elevato basandosi sulla distanza dell'articolo dalla categoria di appartenenza nel grafo delle appartenenze.
Basandosi su questo studio e utilizzando un database a grafo è possibile replicare e estendere l'analisi su una quantità di articoli e categorie maggiore di quella analizzata dai due studiosi (*en.wikipedia* aveva allora 276mila categorie e 20 milioni di assegnamenti a categorie contro i 40 milioni del 12 marzo 2010[2]) e estrarre delle statistiche indicative della struttura semantica di wikipedia.

2. Software e dati usati

Java 1.6: Linguaggio di programmazione compilato e eseguito da un'apposita macchina virtuale (JVM). È interamente orientato agli oggetti, la tipizzazione è statica. Permette le tecniche di reflection, il multithreading e ha praticamente tutte le caratteristiche degli altri linguaggi moderni come C++ o Python.
Eclipse 3.5.2: Popolare IDE per Java e altri linguaggi, effettua la colorazione e l'indentazione automatica del codice, permette il refactoring automatico, suggerisce i nomi dei metodi e delle classi durante la digitazione e automatizza la scrittura dei blocchi try/catch e la navigazione nel codice dei progetti, espandibile con dei plug-in.
Maven 2.0: Programma che automatizza l'inclusione e l'aggiornamento di librerie all'interno di un progetto, gestendo le dipendenze e il controllo delle nuove versioni, nonché i parametri di compilazione specifici per l'utilizzo di una libreria. È stata usata l'estensione m2Eclipse 0.10 per integrarne l'utilizzo nell'IDE.
Neo4j 1.0: Database a grafo progettato primariamente per l'utilizzo all'interno di applicazioni Java. È embedded, quindi viene caricato all'interno della stessa JVM dell'applicazione che lo utilizza. Supporta l'utilizzo da parte di più thread tramite le transazioni. Integra un servizio di indicizzazione (Neo4j-index) che permette di assegnare a un nodo delle coppie chiave-valore per recuperarlo in un secondo momento.

Neo4j 1.0 graph-algo 0.3: Libreria per il calcolo del cammino minimo tra i nodi di Neo4j e altri algoritmi come l'algoritmo di Floyd per trovare tutti i cammini minimi tra tutte le coppie di nodi possibili e analisi statistiche del grafo.
Neoclipse: Programma che permette la navigazione grafica in un database di Neo4j, visualizzando i nodi direttamente collegati a un nodo di partenza prefissato e muovendosi lungo il grafo seguendo le relazioni. Non supporta la ricerca nel grafo, quindi è utile solo per controllare manualmente che non ci siano errori macroscopici nei dati.
MySQL: Database relazionale interrogabile tramite SQL. I dati inizialmente si trovavano all'interno di un database di questo tipo, il cui dump è stato scaricato dal sito web http://download.wikipedia.org.
Linguaggio R: Linguaggio di programmazione e ambiente di sviluppo usato soprattutto per l'analisi statistica, espandibile con dei pacchetti organizzati in repository.
Igraph: Pacchetto di R che permette l'analisi statistica di grafi in formato Pajek,

Il dump di wikipedia utilizzato risale al **12 marzo 2010**, ed è nella forma di un database MySQL contenente tra le altre tre tabelle che verranno usate con la seguente struttura:

ArticlePage(id, name) memorizza gli id numerici di ogni pagina. Sia l'ID che il nome della pagina sono univoci, tuttavia una pagina e una categoria possono avere lo stesso nome.

CategoryPage(id, name) memorizza gli id numerici di ogni categoria. Sia l'ID che il nome della categoria sono univoci, ma una pagina e una categoria possono avere lo stesso nome.

categorylinks(cl_to,cl_from) memorizza le appartenenze degli articoli e delle categorie alle categorie. Il campo *cl_to* contiene l'ID della categoria o della pagina, mentre *cl_from* contiene il nome della categoria di appartenenza.

Il nome della categoria in *categorylinks* utilizza degli underscore al posto degli spazi, mentre in *CategoryPage* sono utilizzati degli spazi, ogni nome è preceduto dal namespace Category e i caratteri speciali come le virgolette sono codificati in HTML (per esempio le virgolette diventano *"*). Bisogna ricordarsi di questi particolari per gestirli con delle sostituzioni di stringhe.

3. Struttura del software

Il programma è interamente scritto in Java tramite Eclipse.
Sono stati creati tanti piccoli programmi per affrontare i vari passaggi del trasferimento in un database a grafo e l'eliminazione delle categorie prive di valore semantico in quanto create ad uso interno di wikipedia.
In particolare le fasi del lavoro sono state:

1. Esportazione dei dati da MySQL a semplici file di testo
2. Lettura di questi file e generazione del grafo Neo4j
3. Esportazione del grafo in formato .net (Pajek)
4. Analisi del file .net tramite la libreria Igraph di R
5. Estrazione del sottoramo delle categorie ad uso interno di Wikipedia
6. Eliminazione di tali categorie, semanticamente non rilevanti, dal grafo
7. Calcolo delle distanze di ogni categoria dalle macrocategorie
8. Creazione delle statistiche di appartenenza delle pagine alle macrocategorie

3.1 Esportazione da MySQL a file di testo semplice

È sufficiente creare un ciclo che legga un certo numero di record, in questo caso sono stati prelevati 2000 record alla volta, con una query del tipo

```
SELECT *
FROM category link s
LIMIT ?offset, 2000
```

Dove offset verrà aumentato di 2000 ad ogni iterazione. Non si possono prelevare tutte le righe in un colpo solo perché MySQL non utilizza un recupero di tipo lazy ma tenta effettivamente di copiare tutti i valori nella RAM bloccandosi.

Le tuple sono memorizzate riga per riga usando il simbolo > come separatore per i campi, poiché questo carattere non appare mai in nessun nome di pagina o categoria.
Il formato è dunque, per le tre tabelle:

ID>nomearticolo
ID>nomecategoria
ID>categoria_di_appartenenza

È conveniente memorizzare i dati in un file di testo prima di trasferirli nel grafo, perché la lettura da file è notevolmente più veloce che la lettura da database: per trasferire il database in file di testo semplice sono state necessarie circa 30 ore, mentre per leggere questi file e creare il grafo ne sono bastate due. Vista la quantità di volte che è stato necessario ricreare il grafo a causa di errori di programmazione e la facilità con cui si possono ricercare manualmente dati al loro interno e creare file di prova è stato utilissimo se non indispensabile farvi ricorso.

3.2 Lettura dei file e generazione del grafo

I tre file (corrispondenti alle tre tabelle MySQL elencate prima) vengono parsati riga per riga. Prima vengono parsati i file degli articoli e delle categorie, in modo da creare i nodi corrispondenti.
Neo4j prevede che ad ogni nodo, così come alle relazioni, possano essere assegnate delle proprietà nella forma chiave->valore, dove la chiave è una stringa contenente il nome della proprietà e il valore è uno dei tipi di dato primitivi di Java o una stringa, nonché un array di tali elementi.
In questo caso i nodi rappresentanti le pagine hanno la proprietà *name*, contenente il nome della pagina, e la proprietà *idpag*, contenente l'id della stessa.
Similmente, le categorie hanno la proprietà *name* e *idcat*.

Neo4j mette a disposizione la *modalità batch*, che eliminando la transazionalità permette di inserire dati con una velocità circa 5 volte maggiore del solito. Questa modalità è indicata per inserire grandi quantità di dati prelevati da altri database o da file, ed è stata usata per questa operazione.
Tutti i nodi così creati vengono indicizzati, in modo da permetterne il recupero diretto in un secondo momento, tramite il componente *LuceneIndexBatchInserter*, quindi si passa alla lettura del file *links.txt* contenente le informazioni sull'appartenenza delle voci e delle pagine alle categorie.
Per ogni riga viene cercato nell'indice il nodo con l'id corrispondente e la categoria con il nome indicato, per poi creare una relazione tra di essi.

La lettura e la scrittura dei file sono state semplificate tramite la creazione di due classi chiamate *Lettore* e *Scrittore*, che fungono da Wrapper per la classe *File* semplificando la scrittura e la lettura di file di testo semplice.

In questa fase è anche necessario tenere conto della differente codifica degli spazi e dei caratteri speciali nei file estratti dalle tre tabelle, in particolare sostituendo gli underscore con degli spazi durante il prelievo da links.txt

3.3 Estrazione del grafo nel formato .net di Pajek

Il formato .net di Pajek è una rappresentazione dei grafi destinata alla loro analisi statistica e matematica, nonché alla visualizzazione grafica, grazie a una serie di strumenti come igraph.
Un file .net ha questo formato

```
*Vertices 230
1 12 “Albums”
2 5645 “India”
[...]
*Arcs
23 45
1 2345
```

Alla prima riga dopo **Vertices* viene indicato il numero di vertici, o nodi, del grafo.
Quindi per ogni vertice viene indicato il suo numero identificativo e una serie di proprietà, in questo caso l'id della pagina o della categoria e il suo nome.
Infine, dopo l'indicazione **Arcs*, vengono indicate tutte le coppie sorgente->destinazione, descritte tramite i numeri identificativi appena inseriti. Gli archi sono orientati, infatti il primo numero indica l'id del nodo di partenza e il secondo quello del nodo di arrivo.
Il risultato, ottenuto in circa mezz'ora di elaborazione, è un file .net da 1.5 GB.

3.4 Analisi del file .net tramite la libreria Igraph di R

Esiste una libreria di R chiamata **Igraph** che permette di compiere numerose analisi sui grafi in formato .net, utilizzabile da linea di comando:

carico la libreria e il file .net

```
> library('igraph')
> g <- read.graph(file='/home/farina/workspace/WikipediaCatGraph/rete.NET',
format='pajek')
```

ottengo la quantità di nodi

```
> length(V(g))
[1] 7794849
```

Nonostante Wikipedia abbia 500 mila categorie e segnali 3 milioni di pagine, esistono numerose categorie e pagine ad uso interno del progetto e non di contenuti, in particolare redirects, che portano la cifra dei nodi del grafo a quasi 8 milioni. Questa caratteristica verrà descritta più avanti.

Calcolo il diametro del grafo, ossia la massima distanza tra due nodi

```
> diameter(g)
[1] 19
```

Ottengo i due nodi alla massima distanza

```
> farthest.nodes(g)
[1] 672188 430234     19
```

La loro distanza è 19, come previsto. I due nodi sono Bowers Hill (Un villaggio della Virginia) e m,n,k-game (un gioco da tavola). Com'era intuibile, sono due argomenti appartenenti ad aree semantiche totalmente diverse.

Calcolo la distanza media fra i nodi

```
> average.path.length(g)
[1] 4.781262
```

Calcolo la densità del grafo, ossia il rapporto tra il numero di archi e il numero di possibili archi distinti

```
> graph.density(g)
[1] 1.408986e-07
```

avendo circa 8 milioni di nodi e 40 milioni di archi questa cifra è facilmente calcolabile direttamente, ottenendo lo stesso risultato.

3.5 Estrazione del sottoramo delle categorie ad uso interno di Wikipedia

Non tutte le pagine e le categorie di Wikipedia sono di contenuti: per organizzare la gestione dell'enciclopedia gli utenti hanno creato una serie di categorie per organizzare gli articoli in base alla qualità, alla presenza di fonti, alla necessità di aggiungere immagini, alla necessità di rendere più neutrale la pagina e altre caratteristiche che nulla hanno a che vedere con il campo semantico dei contenuti della voce.
Questo ovviamente è un problema perché rende inaffidabili i dati ottenuti nell'ipotesi che una vicinanza in termini di grafo delle appartenenze corrisponda a una vicinanza semantica, e l'errore non è trascurabile perché quasi tutte le pagine appartengono a almeno una categoria ad uso interno, in particolare quelle per classificare la qualità degli articoli.

Esistono inoltre numerose voci prive di contenuto, ossia le disambigue e i redirects.
Una **pagina di disambiguazione** è una pagina che contiene un elenco di link a pagine che trattano vari argomenti con lo stesso nome. Ad esempio alla pagina *Java* corrisponde la voce sull'isola di Java in Indonesia, ma nella pagina Java_(disambiguation) possiamo vedere oltre una decina di link alle voci sui vari significati del termine: il linguaggio di programmazione, una marca di sigarette russe, un tipo di caffè, un pipistrello, un tipo di pollo, un distretto della Georgia e altre cose.
Un **redirect** è una pagina che reindirizza automaticamente i visitatori su un'altra pagina con il nome corretto. Per esempio “George W.Bush” redireziona a “George W. Bush”, senza lo spazio prima del cognome. I redirect sono numerosissimi, anche perché vengono creati automaticamente dal software MediaWiki quando si sposta una pagina allo scopo di mantenere i link alla vecchia versione funzionanti.
Entrambe le voci si individuano e si rimuovono facilmente dal grafo poiché appartengono sempre a categorie il cui nome contiene le sottostringhe “isambiguation” oppure “edirects”.

Per selezionare l'elenco delle categorie ad uso interno del progetto si può sfruttare l'esistenza di una categoria di nome *Wikipedia_administration* (Immagine 1), che contiene tutte le categorie ad uso interno del progetto, a loro volta contenenti numerose sotto-categorie.
Si può quindi estrarre ricorsivamente il sotto-grafo delle categorie contenute, direttamente o indirettamente, in tale categoria, per poi eliminarle in modo che non infastidiscano la ricerca di collegamenti semantici tra le pagine.
Questo risultato è facilmente ottenibile aprendo contemporaneamente due diversi database di Neo4j, ossia il grafo delle categorie e un grafo nuovo che conterrà i nodi appena estratti e percorrendo con un algoritmo del tipo breath-first ricorsivo il primo fino a una profondità di estrazione di 10.

Tale profondità è stata scelta arbitrariamente osservando via web la struttura di questa categoria, illustrata in questo screenshot.

Percorrendo manualmente le categorie, infatti, si nota come difficilmente queste raggiungano una profondità troppo elevata, a parte i casi in cui una categoria semanticamente rilevante sia inclusa in una categoria ad uso interno.
Un esempio fra i tanti è la categoria *Very large categories* inclusa in *Wikipedia categories in need of attention* all'interno di *Wikipedia categorization* che è sotto-categoria di *Wikipedia administration*. Tale categoria, come indica il nome, contiene delle categorie molto grandi, segnalate dagli utenti in modo da procedere a una riorganizzazione più specifica dei contenuti che però sono comunque categorie semanticamente rilevanti.
Sorge quindi la necessità di escludere dall'estrazione le categorie che pur essendo sotto-categorie di W*ikipedia administration* sono in realtà significative. Si tratta di migliaia di categorie, e questo lavoro deve essere necessariamente svolto a mano selezionando quei nodi i cui sottorami sono normali categorie di contenuti.

Immagine 1: La rappresentazione web di Wikipedia_administration. Si nota la grande quantità e varietà delle sottocategorie.

Tra le categorie escluse dall'estrazione, ossia quelle che il programma ignora durante la ricorsione, sono da notare:

Parent categories: contiene tutte le categorie che non devono contenere direttamente delle pagine ma solo altre categorie. La categoria è ad uso interno, ma non il suo contenuto.
Wikipedia categories in need of attention: contiene articoli che hanno bisogno di correzioni, ma anche categorie segnalate perché troppo o poco usate, mancanti di una descrizione o oggetto di frequenti vandalismi.
Christianity Portal pages:contiene delle categorie e delle pagine del portale di Wikipedia sul Cristianesimo.
WikiProjects: elenca i progetti, ossia dei gruppi di utenti che collaborano alla gestione delle pagine su un certo argomento (es.:sport, medicina, ecc.) e queste categorie contengono a loro volta le pagine su questi argomenti, quindi non devono essere incluse nell'estrazione
Categories which are included in the JEL classification codes: Categorie organizzate in base alla classificazione JEL (Journal of Economic Literature), quasi tutte le categorie di economia rientrano indirettamente in questa.

Ce ne sono molte altre escluse, quelle appena elencate sono un esempio per mostrare una panoramica dei motivi per cui può essere necessario ignorare una categoria e i suoi contenuti in

questa fase.

3.6 Eliminazione di tali categorie, semanticamente non rilevanti, dal grafo

Una volta estratto il grafo delle categorie ad uso interno è necessario eliminarle dal grafo principale. Per semplificare l'operazione il programma oltre a estrarre il grafo genera un file di testo semplice contenente gli id delle categorie da eliminare, quindi è sufficiente iterare le righe di questo grafo e, per ognuna, ottenere dal servizio di indicizzazione il nodo della categoria corrispondente.
Una volta ottenuto il nodo è necessario eliminare tutte le sue relazioni poiché Neo4j non permette di eliminare dei nodi contenenti ancora delle relazioni. Dopo questo ultimo passaggio si può eliminare il nodo rappresentante la categoria.

I redirect e le pagine di disambiguazione rimangono così senza categoria, quindi si possono individuare grazie a questa caratteristica e eliminare facilmente.

3.7 Calcolo delle distanze di ogni categoria dalle macrocategorie

L'assegnazione delle pagine alle macrocategorie viene determinata osservando le distanze dai loro nodi di tutte le categorie a cui appartiene l'articolo, per poi calcolare le percentuali di appartenenza alle macrocategorie basandosi sul confronto di tali distanze.
Ogni categoria ha una quota uguale alle altre di punti rappresentanti il livello di correlazione, che vengono assegnati alla macrocategoria più vicina, oppure ripartiti a loro volta in sottoquote nel caso la categoria sia equidistante da più macrocategorie.
Se, per esempio, un articolo appartiene a tre categorie vicine a *Geography* e a una vicina a History allora apparterrà per il 75% a Geography e per il 25% a *History*.
Nel caso una categoria sia a distanza minima da più macrocategorie la sua quota di appartenenza viene invece ripartita tra di esse: se un articolo appartiene a una categoria vicina a *History* e a una equidistante da Geography e Science, l'articolo sarà assegnato per il 25% a Science, per un altro 25% a Geography e per il 50% a History.

Esiste un pacchetto, *Neo4j graph-algo*, che estende Neo4j permettendo tra le altre cose di determinare la distanza tra due nodi dati, ossia la lunghezza del minimo percorso. È possibile anche ottenere i nodi di tale percorso, che è calcolato con l'algoritmo di Djikstra.
Lo studio di Kittur precedentemente citato mostra come si possa determinare la macrocategoria più adatta a contenere un articolo in maniera affidabile semplicemente calcolando quale delle macrocategorie è più vicina in termini di numero di nodi da attraversare e, allo stesso tempo, ha mostrato che il miglioramento ottenuto affinando la tecnica, per esempio normalizzando in base alla profondità della tassonomia. Anche un altro studio[3] ha utilizzato con successo questo metodo.

L'operazione di calcolo della distanza tra due nodi, però, richiede un tempo compreso tra uno e una decina di secondi: avendo 3 milioni di articoli e 21 macrocategorie dovremmo effettuare 63 milioni di calcoli del percorso minimo, che richiederebbero anni per essere effettuate una ad una.

Si può ottenere un miglioramento delle prestazioni considerando che il cammino minimo verso un nodo indica la distanza non solo dal nodo di partenza ma anche dei nodi intermedi, quindi è possibile memorizzare questi dati per recuperarli velocemente quando richiesti una seconda volta senza rifare il calcolo.
Per gestire questi dati si possono usare le proprietà dei nodi, assegnando alle categorie dei nodi di nome FROMx, dove x è il nome di una macrocategoria (es.:FROMSports), che contengano il valore calcolato.

Tenendo conto di questi fatti si può implementare un algoritmo di calcolo delle distanze di di questo

tipo:

- Itera sulle categorie di appartenenza di un certo articolo
- Per ognuna di esse, itera sulle macrocategorie di arrivo
- La categoria in esame ha la proprietà FROMx, dove x è il nome della macrocategoria in esame ?
    - Se sì, la distanza cercata è quel valore, lo restituisco senza calcolare il percorso minimo
    - Se no, calcolo il percorso minimo e assegno la proprietà FROMx opportuna a tutti i nodi del percorso ottenuto
- Utilizza i valori FROMx minimi di ogni categoria per calcolare le appartenenze

In questo modo si può evitare di ricalcolare inutilmente le distanze ogni volta.

Tuttavia, **anche in questo modo il programma richiede tempi di eleborazione proibitivi** (facendo una prova, si calcola che ci possono comunque volere vari mesi).

Serve quindi precalcolare la distanza delle varie categorie dalle macrocategorie utilizzando un algoritmo simile a quello di Djikstra:

- Si preleva la prima macrocategoria, x, da un set che le contiene
- Si crea un set di nodi A contenente solo la macrocategoria, un set di nodi B vuoto e un contatore di distanza c inizialmente pari a zero
- Per ognuno dei nodi in A si assegna il valore FROMx (dove x è il nome della macrocategoria in esame), pari a c, e si ottiene il set delle categorie ad esso collegate
    - Per ognuna di queste si determina se ha già la proprietà FROMx. Se ce l'ha, la si ignora, se non l'ha la si aggiunge a B
- Si incrementa di uno il contatore di distanza
- Si svuota il set A e vi si copiano i nodi in B, poi si svuota B
- Se A è vuoto allora si azzera il contatore e si passa all'analisi della prossima macrocategoria, finché non rimangono macrocategorie di cui precalcolare le distanze

In questo modo in poche ore si assegnano ad ogni categoria tutte le proprietà FROMx, poi per analizzare la distanza dalle categorie dalle macrocategorie basta usare questi valori precalcolati.

L'operazione di assegnamento delle distanze richiede varie ore (nel caso specifico **17 ore**)**,** mentre quella di assegnamento degli articoli alle macrocategorie (Immagine 2) richiede **circa quaranta ore**.

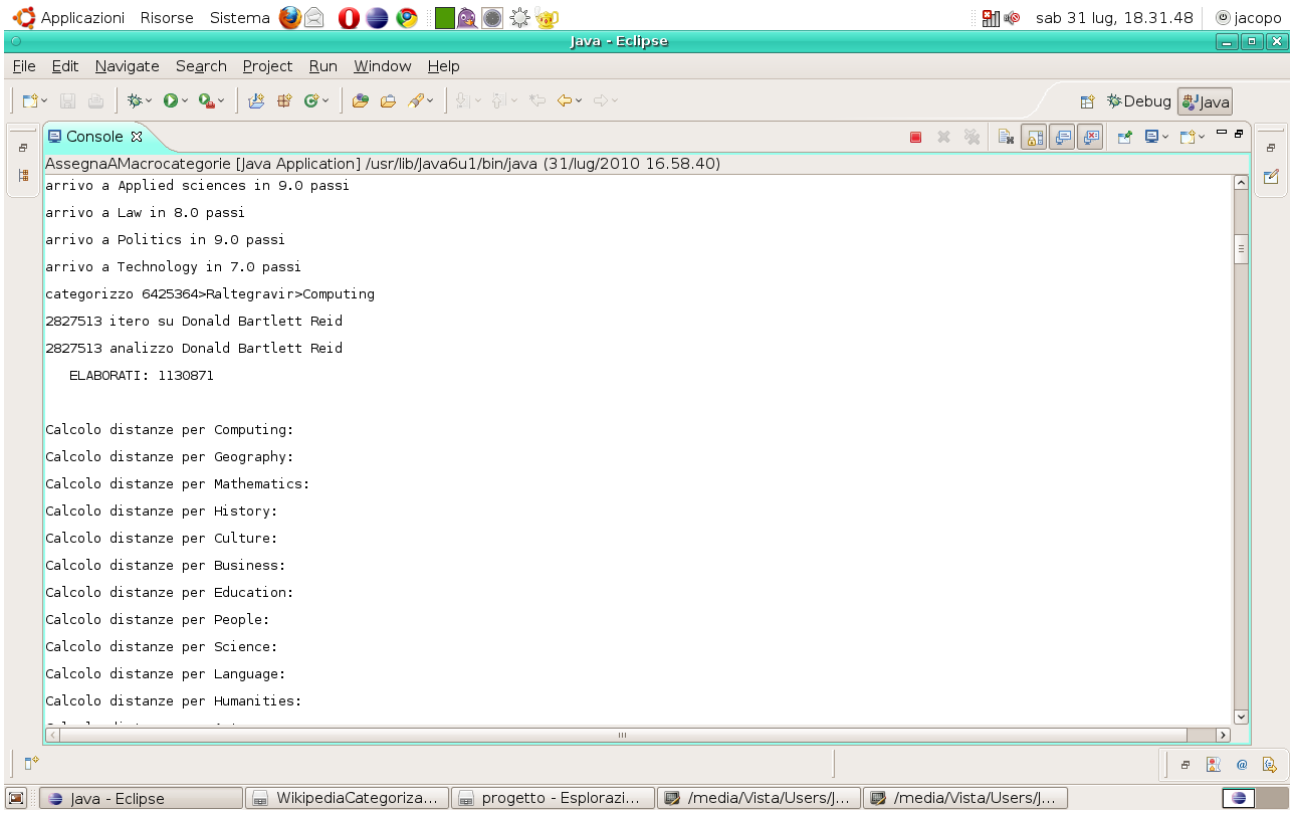


Immagine 2: Il programma che assegna le pagine alle macrocategorie durante l'esecuzione

Al termine otteniamo due file, contenenti l'elenco delle pagine con i livelli di appartenenza alle macrocategorie espressi in valori percentuali e le pagine non assegnate per l'impossibilità di trovare un percorso lungo il grafo.

Prima di avanzare, bisogna capire perchè ci sono delle pagine non classificate.
Uno dei motivi possibili è che effettivamente una pagina non ha una categoria di appartenenza o appartiene a categorie che a loro volta sono prive di categorie di appartenenza. Questo accade principalmente in due casi:
vandalismi e pagine in costruzione: pagine che non erano ancora in fase di scrittura, e che gli editori hanno salvato senza categorizzare.
azione dei bot: pagine destinate a una ricategorizzazione automatica da parte dei bot. Un bot è un programma che manipola automaticamente wikipedia, per vari motivi (aggiornamento delle biografie, correzione di piccoli errori comuni, aggiunta link a voci in altre lingue, eccetera) tra cui il riassegnamento a nuove categorie delle pagine appartenenti a categorie da eliminare.
Ad esempio la pagina *Yosemite National Park* appartiene alla categoria *Madera County, California*. Ma tale categoria non esiste, perché è stata scorporata in categorie più piccole (come *People of Madera County, California*) lasciando così la pagina appartenente a una categoria inesistente (dal lato web questo evento si manifesta con un link della categoria rosso, in attesa che un utente o un bot risolva il problema).
Questi bot, come Cydebot (http://en.wikipedia.org/wiki/User:Cydebot) sono attivati in genere dalla stessa persona che elimina la categoria, e pur agendo molto velocemente non sono istantanei, in quanto modificano le pagine come farebbe un utente, senza nessuna transazionalità su più pagine o categorie alla volta. La conseguenza è che le pagine che stavano per essere riassegnate dal bot durante il dump del database sono state “immortalate” in uno stato anomalo. Tali errori non sono correggibili se non confrontando manualmente la pagina attuale di wikipedia via web (o la sua versione precedente in cronologia) e ricategorizzando la voce manualmente.
Nel caso specifico non sono state categorizzate le pagine sulle autostrade messicane (come *Mexican*

Federal Highway 26), le autostrade della Florida precedenti il 1945 (come *Florida State Road 517 (pre-1945)*), quelle sui piccoli villaggi del Kansas,dello Iowa e del Minnesota(come *Logan Township, Gray County, Kansas*) e delle pagine su località della Russia (come *Grabownica, Świętokrzyskie Voivodeship*).

Come anticipato, il programma restituisce due file:

pagine_classificate.txt: ogni riga contiene il numero identificativo della pagina (*idpag*), il carattere >, il nome della pagina e ancora il carattere > e infine i nomi delle macrocategorie di appartenenza dell'articolo seguite dal carattere : e dal valore percentuale di appartenenza a quella macrocategoria. Contiene 2822154 articoli categorizzati

pagine_inclassificabili.txt: contiene l'elenco dei numeri identificativi e dei nomi delle pagine che non è stato possibile assegnare a nessuna macrocategoria. Contiene 127803 articoli.

3.8 Esempi di assegnamenti a macrocategorie

Esempi di assegnamento di pagine alle macrocategorie

Milan:History and events:100;
Italy:Culture:30;History and events:70;
Politecnico di Milano:Education:87,5;History and events:12,5;
Java (software platform):Computing:50;Technology and applied sciences:50;
Java:Geography and places:100;
(Quest'ultima è ovviamente l'isola)
Earth:Geography and places:100;

Nintendo:Agriculture:6,67;Culture:6,67;History and events:71,67;Geography and places:6,67;Philosophy:6,67;Technology and applied sciences:1,67;

Stupisce il fatto che *Nintendo,* rinomata azienda produttrice di videogiochi sia assegnata al 6.67% a *Agriculture* e solo al 1.67% a *Technology*.

La catena per arrivare a *Agriculture* è:

```
Nintendo -> Academy of Interactive Arts & Sciences members -> Video game
organizations -> Arts organizations -> Organizations by subject ->
Horticultural organizations -> Horticulture and gardening -> Agriculture
```

mentre quella per arrivare a Technology è

Nintendo -> Companies listed on the Tokyo Stock Exchange -> Companies by stock exchange -> Companies -> Organizations by activity -> Conventions -> Technology conventions -> Technology

L'anomalia è causata dalla distribuzione delle distanze delle categorie rispetto a *Agriculture*: è molto facile raggiungere questa macrocategoria con pochi passi, avendo delle sottocategorie immediate molto grandi e non organizzate in maniera tassonomica come avviene invece per *Geography* o *People*.

Possiamo vedere un altro esempio di assegnamento inaspettato.
Si tratta un famoso fumetto giapponese di fantascienza che viene assegnato a tante macrocategorie con la stessa percentuale.

20th Century Boys:
Environment:13,39;Society:13,39;Culture:13,39;Education:13,39;History and events:6,25;Sports:13,39;Technology and applied sciences:13,39;Science:13,39;

Ci si aspetterebbe che venga classificato in Culture, Arts o Science, ma non lo si direbbe mai legato a Sports o Environment.

La catena verso Environment è

```
20th Century Boys -> Science fiction anime and manga -> Anime and manga by genre -> Media by genre -> Television genres -> Environmental television -> Environment and society -> Environment
```

mentre verso Sports è

```
20th Century Boys -> Science fiction anime and manga -> .hack anime and manga -> Anime and manga series categories -> YuYu Hakusho -> Media franchises -> Entertainment -> Sports
```

Nel primo caso l'assegnamento è probabilmente causato come prima dalla “bassezza” della categoria *Environment,* mentre il secondo percorso segue un percorso più complesso.
Dopo essere risalito a *Science fiction anime and manga* scende alla categoria *.hack* (una serie di manga, anime e videogiochi di fantascienza), da cui risale a *Anime and manga series categories* per scendere a *YuYu Hakusho*, risalendo infine a *Sports*.

A differenza degli errori causati dalla bassa profondità tassonomica di *Agriculture* o *Environment*, che si potrebbero contrastare normalizzando i valori delle proprietà FROMx, quest'ultimo caso è di un'altra natura.
Esistono, in molte categorie, delle sottocategorie vicine a macrocategorie distanti dall'argomento trattato, che creano delle scorciatoie tali da influenzare il calcolo del percorso minimo. Per esempio se una rete televisiva appartiene a una categoria sulle reti televisive, questa categoria potrebbe avere una sottocategoria su un'azienda televisiva che produce anche video di giardinaggio, arrivando così velocemente a *Agriculture*.
Un altro esempio è:

Semiconductor fuse:Business:50;Science:50;

L'assegnamento a Business avviene per questi passaggi:

```
Semiconductor fuse-> Semiconductor devices -> Semiconductors -> Semiconductor analysis -> Analysis -> Business analysis -> Business
```

Come prima un passaggio consiste in una discesa, precisamente dalla categoria *Semiconductors* a *Semiconductor analysis*.

Questo problema potrebbe essere parzialmente risolto utilizzando i percorsi ottenuti solo con passaggi verso l'alto (ossia tutti nella direzione della relazione SUBCATEGORYOF), indipendentemente dalla loro lunghezza, dando agli assegnamenti effettuati in tale modo priorità assoluta su quelli fatti con questo metodo. La soluzione non è comunque perfetta perché moltissime pagine non sono collegabili a nessuna macrocategoria solo con passaggi che seguono le relazioni lungo la loro direzione.

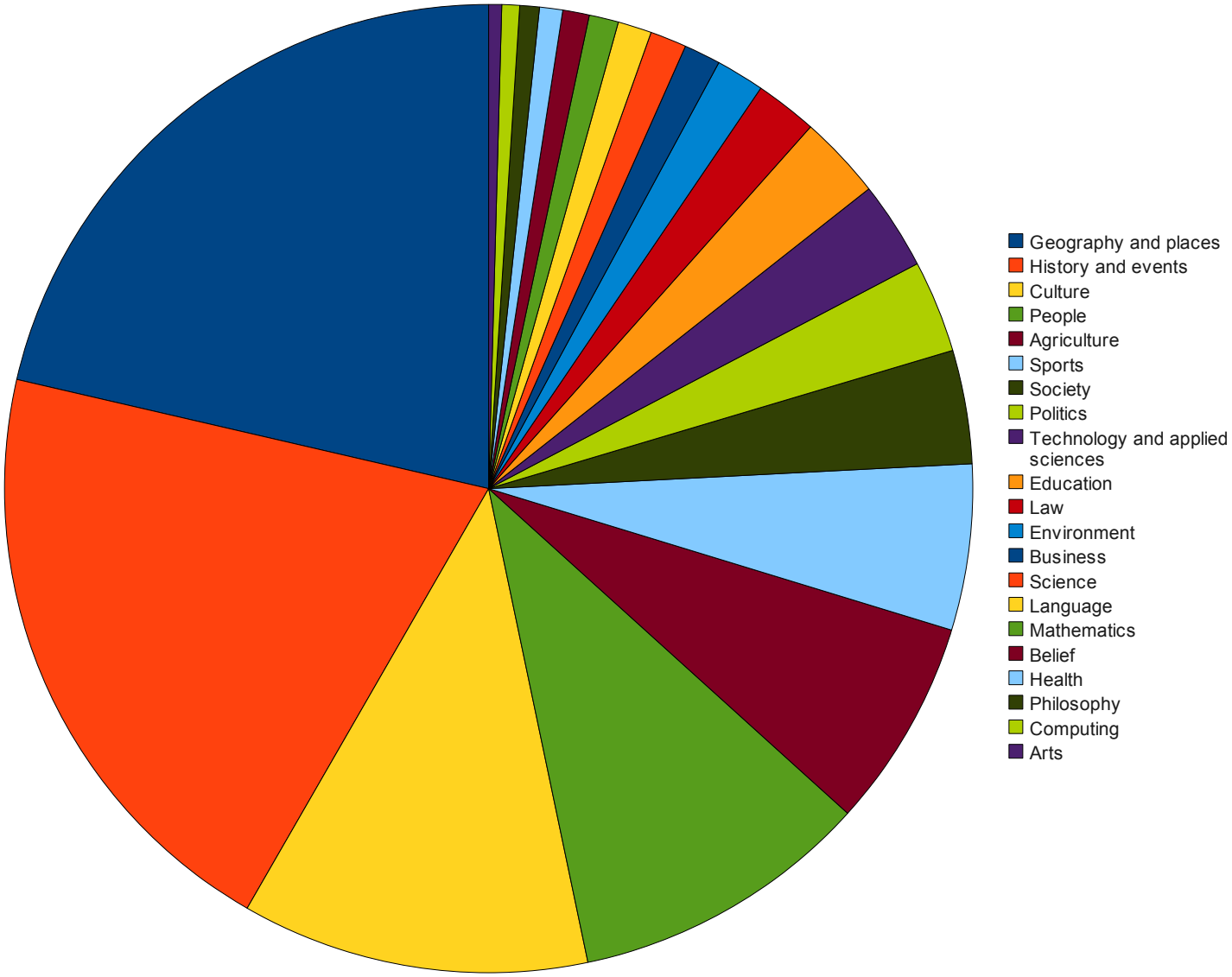
Ecco le dimensioni delle macrocategorie, ottenute sommando tutti i valori percentuali di appartenenza alle stesse assegnati ad ogni articolo.

Geography and places	60337175,28
History and events	57327498,44
Culture	32730545,69
People	28247194,2
Agriculture	19682932,19
Sports	15642414,42
Society	10686177,42
Politics	8703228
Technology and applied sciences	8290477,04
Education	7904696,77
Law	5856760,26
Environment	4541602,99
Business	3455513,21
Science	3424014,85
Language	3186144,1
Mathematics	2789931,76
Belief	2504420,2
Health	2164970,09
Philosophy	1846347,75
Computing	1662195,38
Arts	1231224,49

La macrocategoria più grossa è *Geography and places*, come era prevedibile vista la frequenza con cui si ottengono pagine su luoghi geografici utilizzando la funzione *una pagina a caso* (esistono infatti dei bot che generano rapidamente migliaia di pagine su paesini a partire da database pubblici e basandosi su un template) seguita dalle pagine di argomento storico, culturale e infine biografico. Agriculture è molto diffusa, quasi certamente a causa della bassezza tassonomica illustrata prima.

È da notare che nonostante esistano numerosissime biografie, come si può notare scorrendo il file degli assegnamenti, la macrocategoria *People* non è subito dopo Geography. La causa è probabilmente il fatto che ogni personalità enciclopedica è famosa per qualcosa, per esempio un attore rientrerà nella macrocategoria *Arts* o uno sportivo in *Sports*, rendendo molto raro che la macrocategoria *People* riceva assegnamenti con percentuali alte. Se a ciò si unisce la struttura molto tassonomica della macrocategoria, che è organizzata in maniera gerarchica distribuendo le persone in base al luogo e all'anno di nascita passando per vari livelli di categorie sempre più specifiche, il fatto che People sia quarta in classifica è comprensibile.

Si può rappresentare il risultato anche con un diagramma a torta



Un'altra possibilità interessante è cercare una stima del modo in cui le macrocategorie tendono a sovrapporsi, ossia quanto frequentemente due macrocategorie contengono percentuali alte della stessa pagina.
Per farlo si utilizza la tecnica della *cosine similarity,* che valuta la similitudine fra vettori ed è molto usato nell'analisi della vicinanza semantica dei testi.
Rappresentando ogni macrocategoria come un vettore dove ogni elemento rappresenta una pagina e vale tanto quanto la percentuale di appartenenza della pagina alla macrocategoria (che è 0 se la pagina non le appartiene affatto) e la stessa pagina occupa la stessa posizione in ogni vettore, la similarità tra due vettori di due macrocategorie A e B è data da:

$$\frac{A \cdot B}{||A|| \cdot ||B||}$$

Più il valore è alto e più due macrocategorie sono sovrapposte, ossia tendono a coprire gli stessi argomenti. Naturalmente l'operazione è commutativa quindi il coefficiente tra A e B è lo stesso tra B e A.

Vediamo dunque i valori di affinità più elevati per ogni macrocategoria

Mathematics	Science	0,34666
Technology_and_applied_sciences	Science	0,20803
Society	Health	0,20137
Culture	People	0,16715
History_and_events	Culture	0,13873
Agriculture	Culture	0,11920
Geography_and_places	Culture	0,10830
Business	Technology_and_applied_sciences	0,09900
Politics	People	0,09674
Belief	Culture	0,09335
Sports	People	0,08060
Computing	Technology_and_applied_sciences	0,07468
Philosophy	Mathematics	0,07158
Law	History_and_events	0,06995
Education	Science	0,06277
Language	Geography_and_places	0,06209
Arts	Language	0,05829
Environment	Sports	0,05422

Il coefficiente più alto è quello tra *Mathematics* and *Science*, seguito da *Science* e *Technology and applied sciences*.Questi dati sono abbastanza plausibili.
Guardando invece in fondo alla tabella, dove sono elencati i coefficienti più bassi tra le coppie si estraggono le macrocategorie più distanti tra di loro.

Society	Arts	0,00935
Environment	Politics	0,00309
Agriculture	Arts	0,00675
Culture	Computing	0,00537
Business	Belief	0,00701
Politics	Computing	0,00207
History_and_events	Health	0,00607
Belief	Health	0,00363
Health	People	0,00159
Technology_and_applied_sciences	Belief	0,01755
Philosophy	People	0,00988
Geography_and_places	Health	0,00909
Language	Politics	0,00830
Sports	Health	0,00746
Science	Arts	0,00525
Mathematics	Health	0,00281
Education	Computing	0,00226
People	Health	0,00159
Law	Arts	0,00145
Computing	Arts	0,00107

Le due macrocategorie più distanti sono *Computing* e *Arts* seguite dalla coppia *Law* e *Arts*.
La macrocategoria con il più alto minimo tra valore di similarità rispetto alle altre è *Technology and applied sciences*: si può dunque considerare tale categoria come quella più interdisciplinare, ossia con una buon indice di sovrapposizionerispetto a tutte le altre, la cui più distante è *Belief*.

Questi abbinamenti sono abbastanza plausibili.

3.9 Valutazione della qualità dei risultati

Utilizzando la tecnica della cosine similarity per confrontare le appartenenze generate dal programma e quelle assegnate da un valutatore umano a 50 articoli selezionati casualmente si ottiene un coefficiente di similitudine di 0.4.

4. Conclusioni e possibili approfondimenti

È stato empiricamente dimostrato che la tecnica basata sulla distanza topologica nel grafo delle categorie e degli articoli permette di ottenere degli assegnamenti che hanno una correlazione positiva con quelli che produrrebbe un essere umano.
La tecnica quindi produce dei dati significativi, che potrebbero essere utili in molte applicazioni come stabilire l'argomento di un testo analizzando le macrocategorie in cui vengono collocate le parole che lo compongono, o condurre delle analisi statistiche su Wikipedia.
Come si è visto è stato possibile suddividere, le pagine in macro aree per vedere come è strutturata la conoscenza all'interno di Wik ipedia.
Il coefficiente di similarità, però, potrebbe essere migliorato modificando l'algoritmo di assegnamento.
Una possibile miglioria potrebbe consistere nel normalizzare le distanze calcolate dalle macrocategorie moltiplicando i coefficienti FROMx per delle opportune costanti tali da eguagliare i baricentri delle curve di distribuzione delle distanze da tutte le macrocategorie, per provare e limitare gli errori dovuti alla differente struttura tassonomica delle categorie.
Si potrebbe anche percorrere il grafo solo nella direzione degli archi, oppure ad assegnare dei costi di attraversamento differenziati a seconda che li si attraversi nel loro orientamento o al contrario.
Tutte queste varianti verranno discusse nel successivo lavoro di tesi.

1 2009-Kittur-What's in Wikipedia? Mapping topics and conflicts using socially annotated category structure, Aniket Kittur, Ed H. Chi, Bongwon Suh - Conference on Human Factors in Computing Systems
2 Statistiche di en.wikipedia: http://en.m.wikipedia.org/wiki/Wikipedia:Statistics, visitato il 24 agosto 2010
3 2007-Strube - Wikirelate! Computing semantic relatedness using Wikipedia. Strube, M. Ponzetto -Proceedings of The Twentieth International Joint Conference for Artificial Intelligence (2007)