UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Natural Language Processing per la scoperta di discriminazione di genere

Candidato: *Claudio Parisi*

Relatore: *Franco Turini*

Correlatore: *Alessandro Lenci*

Anno Accademico 2010-2011

A mia madre

Indice Generale

1. Introduzione

1.1 Discrimination Discovery: Il problema

L'individuazione della discriminazione (di razza, di genere, etc.) in maniera automatica a partire da un insieme di dati è un problema che, se risolto, può essere di grande utilità al rispetto della Dichiarazione universale dei Diritti Umani, secondo la quale, nell'interpretazione data dalla Carta dei diritti fondamentali dell'Unione Europea all'art.21,

> "È vietata qualsiasi forma di discriminazione fondata, in particolare, sul sesso, la razza, il colore della pelle o l'origine etnica o sociale, le caratteristiche genetiche, la lingua, la religione o le convinzioni personali, le opinioni politiche o di qualsiasi altra natura, l'appartenenza ad una minoranza nazionale, il patrimonio, la nascita, gli handicap, l'età o le tendenze sessuali"

1.2 DCUBE: data mining nella scoperta di discriminazione

Il sistema DCUBE, al quale questo progetto di tesi vuole fornire supporto, implementa un metodo che permette di analizzare in maniera automatica un insieme di dati e stabilire se, nel caso di decisioni socialmente sensibili che interessano gli individui cui l'insieme di dati fa capo, è stato commesso o meno un atto discriminatorio.

Un possibile esempio, che coincide con il caso analizzato durante lo sviluppo del progetto di tesi, è l'assegnazione di fondi monetari da parte del MIUR ad un insieme di progetti di ricerca presentati. In particolare, si desidera sapere se nella decisione di negare fondi ad un determinato progetto si sia seguito un criterio discriminatorio nell'ambito del genere nei confronti del coordinatore che ha presentato il progetto stesso.

Per arrivare a tale risultato, il sistema DCUBE analizza un database contenente informazioni sul progetto in sé (costi, personale richiesto, tempo di ricerca, etc.) e sul *project coordinator* (età, sesso, pubblicazioni precedenti, etc.). Il sistema, tuttavia, non è in grado di analizzare gli *abstract*, quei dati, cioè, che descrivono i progetti e che, perciò, sono scritti in linguaggio naturale.

1.3 Natural Language Processing a supporto di DCUBE

Lo scopo del progetto di tesi, dunque, è realizzare un sistema di *Natural Language Processing* che prenda in input gli abstract e fornisca dati sui quali sia possibile utilizzare DCUBE per effettuare operazioni di *data mining*. La strada seguita prevede la scomposizione del problema in tre punti:

- *Gender Evaluator*; è un sistema che tenta di individuare il genere del candidato senza conoscerlo, prendendo come punto di partenza lo stile di scrittura adottato nell'abstract. Lo scopo è determinare se uno stile di scrittura che presenti caratteri tipici del genere femminile abbia influenzato la commissione in fase di *peer review*
- *Sector Coverage*; è un sistema che determina, attraverso il contenuto dell'abstract, se l'argomento del progetto presentato sia attinente al settore delle scienze fisiche e analitiche della chimica definito negli *Evaluator Panels* dell'ERC (European Research Council). Lo scopo è determinare se un progetto escluso dall'assegnazione dei fondi possa esserlo stato per mancanza di attinenza settoriale
- *Quality Evaluator*; è un sistema che riesce a distinguere gli abstract scritti correttamente da quelli in cui sono presenti errori morfosintattici al fine di individuare come possibile causa di negazione dei fondi una descrizione del progetto scritta in maniera superficiale o errata

I risultati ottenuti hanno raggiunto lo scopo che ci si prefiggeva, fatta eccezione per il sistema Gender Evaluator che ha rivelato una forte tendenza verso l'utilizzo di uno stile di scrittura prettamente maschile, anche nei casi in cui il project coordinator è di genere femminile [3.3].

2. Analisi e soluzione dei problemi comuni

La fase di progettazione dei tre software riassunti in [1.3] ha rivelato problemi condivisi che è stato necessario affrontare prima di passare alla fase di programmazione vera e propria.

2.1 Il Natural Language Processing oggi

Gli abstract sono stringhe il cui contenuto è scritto in linguaggio naturale. Da qui nasce il problema per DCUBE di interpretare il linguaggio umano e ricavarne informazioni utili.

Lo stato dell'arte odierno nel campo del *Natural Language Processing* non permette alcun tipo di interpretazione semantica da parte di una macchina, fatta eccezione per alcuni prototipi che presentano ampi margini di errore. Le tecnologie più collaudate si concentrano sull'analisi lessicale o sulla complessità testuale: esistono numerosi software che forniscono dati come numero di parole, ricorrenze o che individuano errori e possibili soluzioni su singoli lemmi.

Sul piano dell'analisi morfosintattica, invece, le tecnologie si stanno affinando, soprattutto nel caso della lingua inglese, ed è a queste tecnologie che si è fatto riferimento per cercare di trasformare il linguaggio naturale in numeri comprensibili per il sistema DCUBE.

In particolare, si è cercato di immaginare quali dati potessero essere utili alla scoperta di discriminazione e ci si è chiesti se fosse effettivamente possibile estrarli dagli abstract. L'analisi ha escluso dati come complessità testuale, sempre elevata dato gli argomenti trattati negli abstract, e numero di errori ortografici, vista la presenza di validi *spell checker* nei più comuni software di text editing.

Sono stati presi in considerazione, invece, i tipi di dato riassunti in [1.3] e approfonditi in seguito. I metodi automatici progettati per ottenerli si sono rivelati sufficientemente precisi, con un margine di errore accettabile.

2.2 Java

L'utilizzo di Java come linguaggio di programmazione per lo sviluppo delle tre applicazioni non è casuale. Sebbene siano stati presi in considerazione PHP e C#, il

primo per motivi di portabilità, il secondo per l'ottima velocità di elaborazione, Java ha permesso di soddisfare entrambe le esigenze.
In aggiunta a questi due fattori, Java ha consentito di integrare una libreria sviluppata dalla Stanford University (CoreNLP) che permette di effettuare il *POS tagging* degli abstract, elemento che si è rivelato di grande utilità nell'applicazione di Quality Evaluator [5.2.1].

2.3 I dati Abstract

DCUBE opera su un database Oracle di 3792 record contenenti ognuno le informazioni riguardanti un progetto sottoposto al MIUR nell'ambito delle scienze fisiche e analitiche della chimica. Per ogni progetto si hanno informazioni riguardanti il project coordinator e un abstract che descrive a grandi linee gli obiettivi che il progetto stesso si pone di raggiungere. La lingua utilizzata negli abstract è l'inglese, l'autore generalmente è italiano. La lunghezza media di ogni abstract conta circa 745 parole.
I dati abstract da analizzare sono contenuti all'interno di codice SQL per favorirne l'inserimento nel database. Ciò ha richiesto la realizzazione di un sottosistema, implementato in ognuno dei tre software, in grado di pulire gli abstract dalla sintassi SQL. In particolare è possibile individuare una struttura ricorrente che circonda ogni abstract:

```
Insert into "table_export" (CODE_A,ABSTRACT) values
(Codice,'Abstract');
```

Dove:

- `Codice` individua un intero univoco che identifica l'abstract e il progetto a cui esso appartiene
- `Abstract` è la stringa che costituisce la descrizione vera e propria del progetto
- la prima riga viene utilizzata per individuare e memorizzare il `Codice` di un nuovo abstract da analizzare
- `\d,'` è l'espressione regolare che precede `Abstract` e ne determina l'inizio
- `');` è l'espressione regolare che individua la fine dell'`Abstract`

Per ogni abstract così individuato viene effettuata la chiamata all'algoritmo che si occuperà di analizzarlo.

3. Gender Evaluator

Il primo problema affrontato riguarda l'individuazione automatica del genere dell'autore di un dato abstract. Con questo sistema si vuole capire se lo stile di scrittura tipico del genere maschile o femminile può aver influenzato gli esaminatori dei progetti nella scelta dell'assegnazione dei finanziamenti.

3.1 Stato dell'arte

I primi studi sulle differenze linguistiche fra gli stili di scrittura maschile e femminile hanno inizio negli anni '70 con le pubblicazioni di Lakoff [Lakoff, 1975] e Key [Key, 1975] che tentano di individuare caratteri dissimili fra i sessi negli scritti in inglese britannico moderno.

Ricerche successive, come quelle di Holmes o Tannel [Koppel, 2003], hanno evidenziato differenze di tipo fonologico o la maggiore tendenza da parte delle donne a trattare argomenti di relazioni interpersonali rispetto agli uomini.

È solo nel 2001, però, che Koppel pubblica un metodo automatico per l'individuazione del genere. La ricerca è fondata sull'analisi di una porzione del British National Corpus: 604 documenti, per un totale di oltre 25 milioni di parole, equamente divisi fra autori di genere maschile e femminile [Koppel, 2001]. Il corpus copre svariati settori, dai testi informali alle pubblicazioni di carattere economico o scientifico. Utilizzando una generalizzazione del *Balanced Winnow algorithm* per l'estrazione automatica di caratteristiche responsabili della differenziazione fra più testi, Koppel è riuscito ad isolare *features* tipiche dello stile maschile e femminile e a dare ad ognuna un peso ben preciso. Un algoritmo di ricerca basato sull'individuazione di queste caratteristiche permette, dunque, di supporre con un buon grado di precisione (circa 80%) quale sia il genere dell'autore di un dato testo.

L'algoritmo di Koppel è stato, poi, implementato in due applicazioni web, *Gender Genie* (2003) [Bookblog] e *Gender Guesser* (2006) [Hacker Factor], ed è a queste che il sistema Gender Evaluator sviluppato si ispira, focalizzando l'attenzione sulle features tipiche dei testi formali.

3.2 L'algoritmo

L'algoritmo di Gender Evaluator utilizza un dizionario integrato formato dalle 33 features individuate da Koppel [Fig.1]. Ognuna è sintetizzata in un oggetto *Lemma* a sua volta formato da un stringa che individua la parola in sé e da un intero che determina il peso ad essa assegnato.

Features genere Maschile		Features genere Femminile	
A	6	And	-4
Above	4	Be	-17
Are	28	Her	-9
Around	42	Hers	-3
As	23	If	-47
At	6	Me	-4
Below	4	Myself	-4
Is	8	Not	-27
It	6	She	-6
Many	6	Should	-7
More	34	Was	-1
Said	5	We	-8
The	7	When	-17
These	8	Where	-18
To	2	With	-52
What	35	Your	-17
Who	19		

Fig. 1 Elenco delle features e rispettivi pesi individuati da Kopper

Come mostrato in Fig.1, viene dato un valore positivo al peso delle keywords che individuano uno stile maschile e un valore negativo a quelle di genere femminile. I pesi sono tarati per i testi di tipo formale.

È possibile notare come i *markers* tipici dello stile maschile siano i *determiners*, cioè quelle parole che tendono a determinare o specificare i nomi comuni all'interno del testo, mentre i markers tipici dello stile femminile siano i *pronouns*, cioè tutte quelle parole che tendono a sostituire i nomi comuni [Koppel, 2003].

L'algoritmo di Gender Evaluator cerca le keywords all'interno di ogni abstract e, per ogni corrispondenza trovata, aggiorna due interi rappresentanti ognuno il peso complessivo raggiunto dallo stile dei due generi. Successivamente i due pesi vengono messi a confronto e quello maggiore determina il genere dell'autore.

Per misurare la precisione dell'algoritmo di Kopper, in Gender Evaluator è stata introdotta l'Accuracy:

$$Accuracy = \frac{|MaleWeight - FemaleWeight| * 100}{MaleWeight + FemaleWeight}$$

dove *MaleWeight* e *FemaleWeight* esprimono rispettivamente il peso complessivo raggiunto dallo stile maschile e femminile nell'ambito di un dato abstract. L'Accuracy, così calcolata, esprime una confidenza in percentuale e indica di quanto il peso del genere dominante sia prevalente sul peso del genere secondario. In particolare, per un valore di Accuracy tendente allo 0%, i due pesi tendono ad equivalersi; per un valore di Accuracy tendente al 100%, l'abstract tende a presentare keywords di un solo genere.

In aggiunta a queste operazioni, Gender Evaluator mette a confronto i risultati ottenuti attraverso l'algoritmo di Koppel con i dati reali letti, abstract per abstract, da un file CSV (*Comma Separated Values*) indicante il codice dell'abstract e il sesso del project coordinator.

3.3 Risultato e osservazioni

In output, Gender Evaluator genera due file CSV così struttuati:

gender_Evaluator.csv : `CODE_A,"GE","ACCURACY"`

comparison.csv : `CODE_A,"RealGender","GE","ACCURACY","Comparison"`

dove:

- `CODE_A` è il codice univoco dell'abstract
- `GE` è il genere trovato attraverso l'algoritmo di Koppel
- `ACCURACY` è la misura di confidenza precedentemente descritta
- `RealGender` è il sesso reale del project coordinator
- `Comparison` è il risultato, `true` o `false`, del confronto fra `RealGender` e `GE`

In seguito all'analisi dei risultati ottenuti, si è osservato che è impossibile determinare se uno stile di scrittura prettamente femminile abbia potuto influenzare la commissione durante la valutazione dei progetti sottoposti al MIUR.

Se si osserva il grafico in Fig.2, sembrerebbe che l'algoritmo, anche a livelli di accuratezza molto elevati (in ascissa), non sia in grado di distinguere correttamente i due generi: la quantità di abstract sui quali l'algoritmo di Koppel sbaglia (`false`) è quasi identica ai risultati corretti.

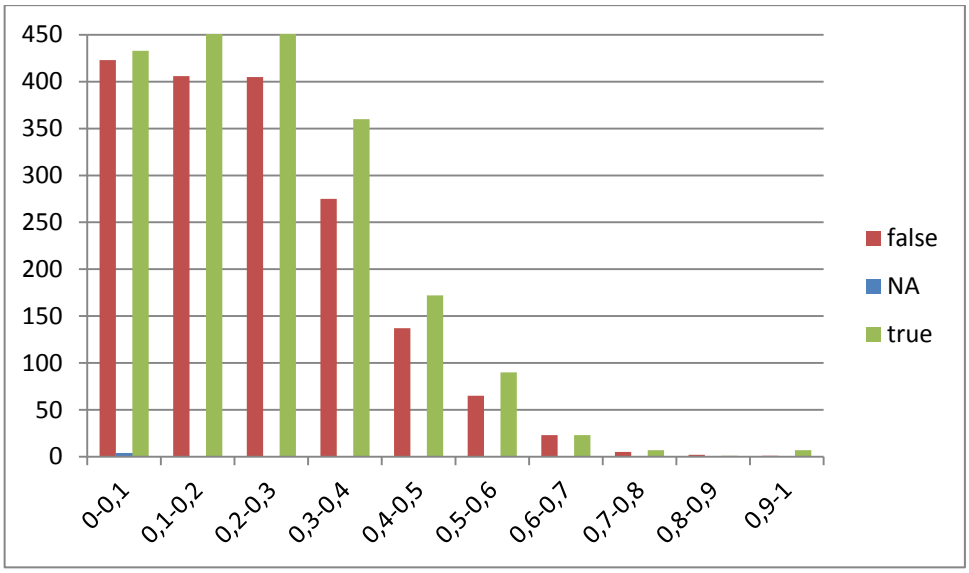

Fig. 2 Comparazione fra genere reale e calcolato, entrambi i generi

Tuttavia, filtrando i risultati in base al genere reale [Fig.3 e Fig.4] si osserva come, in realtà, l'algoritmo di Koppel generi risultati errati solo nel caso del genere femminile, risultando piuttosto preciso, invece, nel caso del genere maschile.

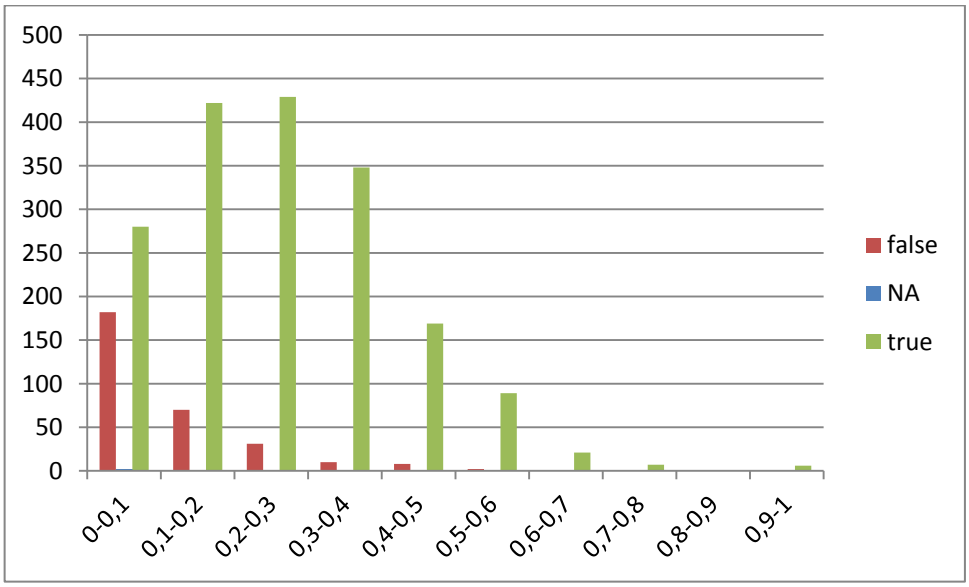

Fig. 3 Comparazione fra genere reale e calcolato, genere maschile

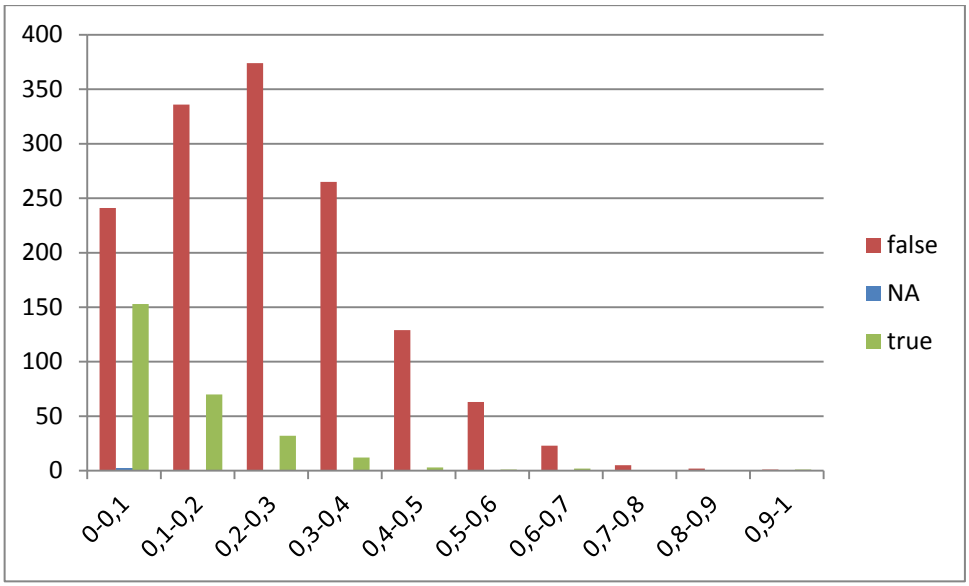

Fig. 4 Comparazione fra genere reale e calcolato, genere femminile

Tale differenza può essere dovuta a molteplici fattori come l'influenza del genere maschile su quello femminile nel caso il primo sia predominante nell'ambiente lavorativo o la necessità, visto l'argomento degli abstract, di adottare una forma di scrittura che presenti un copioso uso di *determiners*.

Ad ogni modo, sebbene il risultato non consenta di operare le osservazioni che ci si aspettava, permette di escludere lo stile di scrittura dagli elementi che possono aver portato alla discriminazione di un genere rispetto all'altro.

4. Sector Coverage

L'attinenza del contenuto di un abstract al settore per il quale il MIUR ha stanziato i finanziamenti può essere un ulteriore elemento in grado di rivelare eventuali comportamenti discriminatori. Ci si aspetta, infatti, che ad una valutazione positiva da parte degli esaminatori corrisponda altrettanta attinenza settoriale del progetto.

4.1 Keywords dall'ERC Panel

L'European Research Council è l'organismo che si occupa della valutazione dei progetti e dell'assegnazione dei fondi. Per operare fa affidamento su 25 giurie, dette *panels*, ognuna formata da scienziati e ricercatori specializzati in un determinato settore.

I progetti a cui gli abstract fanno riferimento sono individuati dal settore PE4 degli ERC Panels:

> "**Physical and Analytical Chemical sciences:** analytical chemistry, chemical theory, physical chemistry/chemical physics"

Il settore PE4 è a sua volta suddiviso in sottosezioni che descrivono più dettagliatamente gli argomenti coperti.

Ci si è posto, dunque, il problema di valutare l'attinenza di un dato abstract a questo settore. L'approccio adottato prende come base le descrizioni del settore PE4 stesso dalle quali sono state estratte *keywords* da utilizzare come chiavi di ricerca all'interno dell'abstract. La presenza e la densità di queste keywords nel testo possono costituire un indice diretto per la misurazione dell'attinenza settoriale.

Osservando la descrizione negli ERC Panels [Fig.5] è possibile notare che ci sono parole che possono esprimere un forte vincolo di attinenza, ad esempio *nanochemistry* o *spectroscopic*, mentre altre possono indicare un legame più debole ma comunque importante, come *sciences* e *techniques*.

Nel costruire il dizionario di keywords, si è scelto, perciò, di operare una distinzione delle parole in due *tier* di attinenza. Il primo con le parole che maggiormente individuano il settore PE4, il secondo con quelle che, invece, possono denotare un ambito scientifico più ampio.

Alle keywords individuate per estrazione diretta, sono state aggiunte varianti delle stesse parole al fine di coprire forme sia nominali, sia aggettivali in numero singolare e plurale [Fig.6].

PE4 Physical and Analytical Chemical sciences: analytical chemistry, chemical theory, physical chemistry/chemical physics

PE4_1 Physical chemistry
PE4_2 Nanochemistry
PE4_3 Spectroscopic and spectrometric techniques
PE4_4 Molecular architecture and Structure
PE4_5 Surface science
PE4_6 Analytical chemistry
PE4_7 Chemical physics
PE4_8 Chemical instrumentation
PE4_9 Electrochemistry, electrodialysis, microfluidics
PE4_10 Combinatorial chemistry
PE4_11 Method development in chemistry
PE4_12 Catalysis
PE4_13 Physical chemistry of biological systems
PE4_14 Chemical reactions: mechanisms, dynamics, kinetics and catalytic reactions
PE4_15 Theoretical and computational chemistry
PE4_16 Radiation chemistry
PE4_17 Nuclear chemistry

Fig. 5 Descrizione del settore PE4 degli ERC Panels

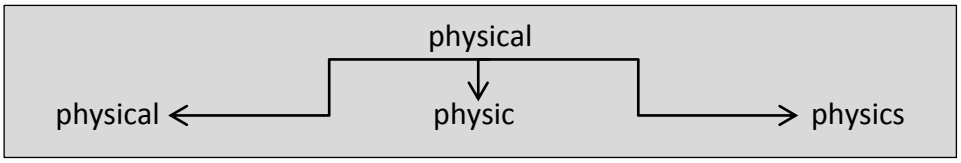

Fig. 6 Varianti a partire da una keyword

4.2 L'algoritmo

Il sistema di Sector Coverage opera in due fasi. Nella prima effettua il caricamento del dizionario di keywords da un file esterno, nella seconda ricerca le keywords all'interno di ogni abstract determinando l'attinenza settoriale.

Il dizionario di keywords è un file di testo (ERC.txt) organizzato in due settori, ognuno contenente le parole di un dato tier. Come si può notare in Fig.7, ogni parte è individuata da una riga che inizia con il carattere `#`, cui segue il tier.

Subito dopo `#1`, è inserito l'elenco delle parole che denotano un forte grado di attinenza, definito `SC_STRONG`, mentre la lista di keywords che coprono un ambito scientifico più ampio, definito `SC_WIDE`, segue la riga che inizia per `#2`.

Nel caso le descrizioni fornite dall'ERC venissero aggiornate, il dizionario può essere modificato a piacimento purché non ne venga cambiata la struttura.

```
#1 - Words with strong sector coverage (SC_STRONG)
biological
[…]
spectrometric
#2 - Words with wide sector coverage (SC_GENERIC)
analytical
[…]
theoretical
```

Fig. 7 Estratto del dizionario di keywords

Sector Coverage effettua una prima lettura del dizionario per determinare la dimensione di ogni lista e normalizzare il testo. Durante la seconda, invece, alloca due array di stringhe, ognuno contenente la lista di keywords divise in base al tier.

Al passo successivo, Sector Coverage normalizza il testo di ogni abstract e lo divide in singole parole al fine di effettuare il confronto fra esse e gli array di keywords:

```
String text = s.toLowerCase();
String regex = "[^a-zA-Z]+";
String[] textArray = text.split(regex);
```

Dove `s` è il testo grezzo di un abstract e `[^a-zA-Z]+` è l'espressione regolare che permette di individuarne le singole parole eliminando punteggiatura, parentesi o altra simbologia.

In caso di riscontri, l'algoritmo aggiorna due interi, uno per ogni tier, che tengono conto del numero di corrispondenze individuate.

4.3 Risultato e osservazioni

In output, Sector Coverage genera un file CSV così strutturato:

sector_coverage.csv : `CODE_A,"WORDS","SC_STRONG","SC_WIDE","SC_TOT"`

dove:

- `CODE_A` è il codice univoco dell'abstract
- `WORDS` è il numero totale di parole contenute nell'abstract una volta normalizzato
- `SC_STRONG` individua il numero di corrispondenze con keywords a forte attinenza trovate nell'abstract
- `SC_STRONG` individua il numero di corrispondenze con keywords a carattere scientifico più ampio trovate nell'abstract

- `SC_TOT` è la somma delle corrispondenze `SC_STRONG` e `SC_WIDE`

Una prima analisi dei risultati ottenuti attraverso l'algoritmo di Sector Coverage mostra come sia stato possibile individuare in maniera automatica abstract senza copertura di settore o con copertura al più generica [Fig.8 e Fig.9].

	SC_TOT count	Percentage SC_TOTAL
Abstracts without keywords	107	2,82%
Abstracts with keywords > 0	3685	97,18%

Fig. 8 Conteggio degli abstract con corrispondenze di qualunque tipo

	SC_STRONG count	Percentage SC_STRONG
Abstract without STRONG	1059	27,93%
Abstract with STRONG > 0	2733	72,07%

Fig. 9 Conteggio degli abstract con corrispondenze a forte attinenza

È, inoltre, possibile individuare *ouliers* definendo un relazione fra il numero di corrispondenze trovate e il numero totale di parole contenute in ogni abstract [Fig.10].

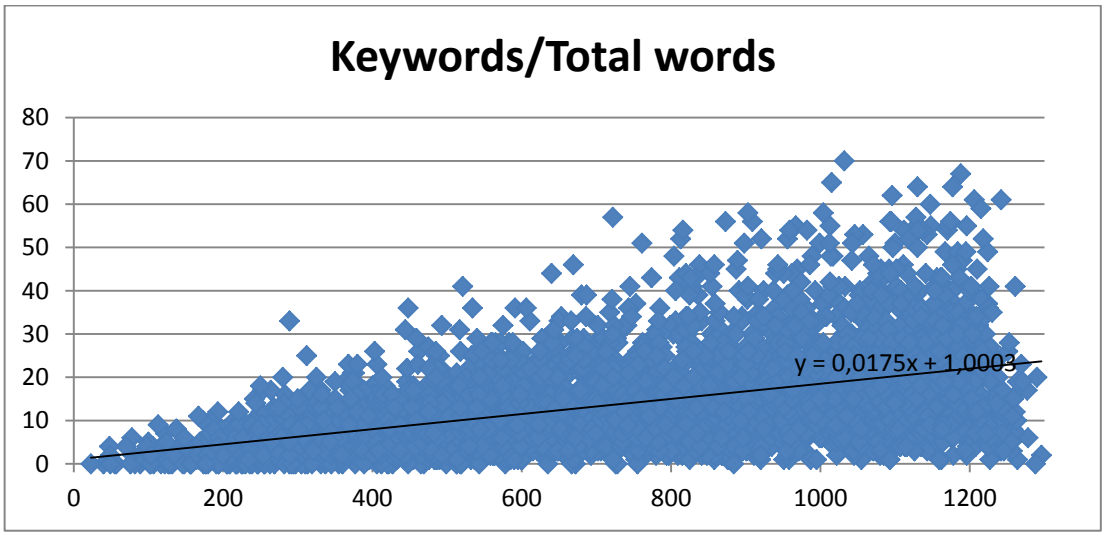

Fig. 10 Rapporto fra corrispondenze trovate e parole totali

In questo modo, il sistema DCUBE può effettuare operazioni di data mining sui risultati ottenuti per controllare se i progetti a cui è stato garantito un finanziamento abbiano un certo margine di attinenza al settore PE4. In caso contrario, si sarà ottenuto un ulteriore dato utile ad avvalorare l'ipotesi di decisione discriminatoria.

5. Quality Evaluator

Il sistema di Quality Evaluator mira ad ottenere una stima della qualità testuale di un abstract affinché si possa operare un raffronto fra i progetti a cui è stato negato il finanziamento e quelli che presentano errori di tipo morfosintattico.

5.1 Stato dell'arte

Valutare la qualità lessicale, sintattica e semantica di un documento scritto in linguaggio naturale è un problema che un essere umano può risolvere con una semplice lettura del testo. È attualmente impensabile, però, un sistema automatico in grado di svolgere lo stesso compito non solo a parità di velocità ma anche a parità di precisione.

Esistono, tuttavia, numerosi sistemi che permettono di analizzare il testo al fine di cercare errori o strutture grammaticali al suo interno, e altri che tentano di interpretarne il contenuto semantico. Ognuno di questi, però, svolge il proprio compito in maniera differente e con un grado di precisione altrettanto differente.

I più diffusi e affidabili sono gli *spell checker*, sistemi generalmente costituiti da corposi dizionari di parole che permettono di individuare errori ortografici. Sono implementati nei più comuni software di *text editing* e supportano in tempo reale l'utente nella scrittura di documenti segnalando errori e possibili correzioni. Nel progettare il sistema di Quality Evaluator si è esclusa la possibilità di utilizzare un algoritmo di spell checking proprio a causa della loro diffusione: gli abstract da analizzare sono scritti con software di text editing perciò presentano errori ortografici in quantità talmente ridotta da risultare irrilevante ai fini dell'analisi.

A seguire gli spell checker in ordine di affidabilità e diffusione, sono i *grammar checker*, sistemi che tentano di individuare errori di tipo grammaticale cercando di analizzare le strutture sintattiche di un testo per poi confrontarle con un proprio database di regole grammaticali. Gli algoritmi a loro fondamento sono quasi sempre proprietari e molto complessi. Nonostante ciò, il grado di precisione di questi sistemi è molto basso in quanto i testi scritti in linguaggio naturale presentano strutture sintattiche estremamente articolate e diversificate che spesso deviano, senza per questo commettendo errore, dalle più stringenti regole grammaticali di una data lingua. Per questa ragione i grammar checker non segnalano degli errori veri e propri ma si limitano ai cosiddetti *warnings*, cioè a segnalare delle possibilità di errore.

Implementare un algoritmo di grammar checking nel sistema di Quality Evaluator si sarebbe rivelato poco proficuo oltre che particolarmente ostico: individuare warnings non può costituire un buon indice di qualità del testo, il sistema deve individuare errori veri e propri mantenendo un opportuno grado di precisione.

Quality Evaluator, tuttavia, utilizza uno degli strumenti di analisi testuale generalmente implementati nei grammar checker: il *Part-Of-Speech tagger*. I POS tagger sono sistemi che, con buona precisione, sono in grado di classificare morfologicamente le parole presenti in un testo. L'utilizzo di un POS tagger ha permesso di aggiungere un buon numero di regole grammaticali all'interno di Quality Evaluator e di specificare ulteriormente quelle già previste per aggiungere precisione [5.2.1].

I sistemi di interpretazione semantica di un testo sono stati del tutto esclusi nella progettazione di Quality Evaluator, sia per motivi di affidabilità, estremamente bassa, sia per diffusione: gli studi nel campo sono ad uno stadio embrionale e i pochi software esistenti sono quasi sempre prototipi. Ad ogni modo, il sistema di Sector Coverage [4] può essere visto come un precursore di un sistema di interpretazione semantica poiché in grado di valutare l'attinenza settoriale del contenuto di un abstract.

5.2 Progettazione e realizzazione

Individuare errori che non siano lessicali ma morfosintattici, senza seguire la strada dei grammar checker, presuppone un'analisi eseguita su più parole che miri a rivelare strutture o forme scorrette all'interno del testo.

L'unità minima dopo la singola parola è una coppia di parole, perciò Quality Evaluator tiene conto di due parole adiacenti alla volta e controlla la loro presenza in un database di errori prestabiliti che occorrono su coppie di parole e che non ammettono eccezioni. Un'eventuale riscontro, dunque, non genera warnings ma segnala errori veri e propri.

5.2.1 Stanford CoreNLP: POS tagger

CoreNLP è una suite di software sviluppata in JAVA dall'Università di Stanford, CA. È pensata per svolgere molteplici tipi di analisi testuale.

Una parte dell'algoritmo di Quality Evaluator sfrutta una delle librerie messe a disposizione da CoreNLP. Ciò gli permette di effettuare il Part-Of-Speech tagging del testo.

Il CoreNLP: POS Tagger consente l'utilizzo di tre algoritmi di tagging, tutti sviluppati utilizzando come training corpus le sezioni 0÷18 del *British National Corpus* e come testing corpus le sezioni 19÷21. Lo schema in Fig.11 riassume il grado di correttezza su parole conosciute (CPC) e parole sconosciute (CPS) per ogni algoritmo testato.

Algoritmo	CPC	CPS
Bidirectional with Distributional Similarities	97.28%	90.46%
Left-3-Words with Distributional Similarities	97.01%	89.81%
Left-3-Words	96.97%	88.85%

Fig. 11 Precisione degli algoritmi di POS tagging

Inizialmente, Quality Evaluator si è servito dell'algoritmo bidirezionale, tuttavia, è stato necessario passare al Left-3-Words with Distributional Similarities in quanto la mole di abstract da analizzare portava il primo algoritmo a generare *memory leaks*. La suddivisione del carico di lavoro in più parti e l'ottimizzazione del codice di Quality Evaluator non sono servite a risolvere il problema ma solo a ritardarne l'occorrenza.

Per quanto riguarda il funzionamento, CoreNLP: POS Tagger prende in input una stringa e genera un array di stringhe in cui ogni elemento è formato dalla parola analizzata e dalla categoria morfologica ad essa associata secondo lo standard definito nel *Penn Treebank tagset*:

```
This is a sample sentence
```

This/DT	is/VBZ	a/DT	sample/NN	sentence/NN

Quality Evaluator modifica questo output creando per ogni elemento dell'array un oggetto *LemmaPOS* formato da due stringhe, la prima contenente la parola analizzata, la seconda contenente il POS tag. Metodi di *get* opportuni permettono di conoscere il contenuto delle stringhe. La ricerca degli errori viene effettuata su coppie di LemmaPOS.

5.2.2 Il database di errori

Il database degli errori è stato generato a partire da grammatiche e dizionari inglesi disponibili sul web [English Gratis, English for Italians, Englishpage.com]. In particolare, sono state selezionate le regole applicabili a coppie parole che non ammettono eccezioni e, a partire da esse, è stato creato un campione di 190 errori certi.

La validità di ogni errore è stata confermata attraverso una ricerca di corrispondenze nel *British National Corpus*: solo nel caso in cui non vi fossero occorrenze, l'errore è stato inserito nel database di Quality Evaluator.

Il database è contenuto in un file CSV così strutturato:

errordb.csv : `FIRST_TERM,SECOND_TERM,FIRST_POS,SECOND_POS`

dove:

- `FIRST_TERM` è il primo lemma della coppia di parole
- `SECOND_TERM` è il secondo lemma della coppia di parole, se formato da un solo carattere è l'iniziale del secondo lemma
- `FIRST_POS` è il POS tag del primo lemma della coppia di parole
- `SECOND_POS` è il POS tag del secondo lemma della coppia di parole

Si utilizza il valore 0 nel caso in cui non sia necessario specificare uno di questi valori per formalizzare l'errore.

Gli errori considerati possono essere di vari tipi, fra parentesi un esempio di formalizzazione nel database:

- Verbo modale seguito da participio presente (`0,0,MD,VBG`)
- Articolo seguito da aggettivo o pronome possessivo (`the,0,DT,PRP`)
- Aggettivo o pronome possessivo seguito da un verbo (`their,0,0,VBP`)
- Disaccordo fra la persona del pronome e quella del verbo (`it,0,PRP,VBP`)
- Disaccordo in numero fra determiner e sostantivo (`a,0,DT,NNS`)
- Disaccordo vocalico fra determiner e sostantivo (`a,e,DT,NN`)
- Disaccordo verbo più preposizione (`related,at,0,0`)

5.2.3 L'algoritmo

Le operazioni eseguite da Quality Evaluator possono essere così elencate:

1. Split degli abstract
2. Caricamento del database degli errori
3. Normalizzazione del testo degli abstract e POS tagging
4. Ricerca degli errori

Nel dettaglio, al primo punto, Quality Evaluator carica il file contenente gli abstract e, senza apportare alcuna modifica, lo divide in parti da 100 abstract l'uno. Ogni parte viene salvata in una cartella temporanea `temp` come file di testo `abstemp##.txt` dove `##` identifica numericamente la parte. Durante l'esecuzione i file temporanei già analizzati vengono eliminati e, al suo termine, viene eliminata anche la cartella.

Lo split del file principale è stato originariamente inserito per arginare il problema di memory leak generato dall'algoritmo di POS tagging bidirezionale. Con il passaggio al Left-3-Words la fase di split poteva essere eliminata ma si è deciso di mantenerla poiché, sul sistema di prova, ha permesso un risparmio costante di circa 19MB di RAM durante l'intera esecuzione.

Per quanto riguarda il caricamento del database degli errori, Quality Evaluator crea un array di oggetti *Error*. Ogni Error è formato da quattro stringhe, una per ogni elemento dell'errore, e da un intero $2 \leq errorvalue \leq 4$ che conta il numero di elementi diversi da 0.

La fase di normalizzazione e POS tagging non differisce da quelle spiegate in [4.2] e [5.2.1]. Riassumendole, il testo di ogni abstract viene privato di ogni simbolo, punteggiatura inclusa, e taggato. Poi, per ogni coppia `word/tag`, viene creato un oggetto LemmaPOS formato da due stringhe, una contenente `word` e l'altra contenente `tag`. Il testo di ogni abstract è rappresentato da un array di LemmaPOS.

La fase di analisi del testo vera e propria è realizzata attraverso due cicli `for` annidati. Il primo, caratterizzato dall'indice `i`, scorre l'array di LemmaPOS, il secondo, caratterizzato dall'indice `j`, quello di Error.

Quality Evaluator conta le corrispondenze fra gli elementi di ogni coppia di `LemmaPOS[i, i+1]` e gli elementi di ogni `Error[j]`. Il risultato di questo calcolo

viene poi confrontato con `errorvalue[j]` e, in caso di uguaglianza, viene conteggiato un errore [Fig.12].

```
error[j] : related at 0 0, errorvalue[j]=2
Se LemmaPOS[i, i+1] : related VBN to IN corrispondenze : 1
errorvalue[j] = corrispondenze? : 2≠1. Non c'è errore.
Se LemmaPOS[i, i+1] : related VBN at IN, corrispondenze : 2
errorvalue[j] = corrispondenze? : 2=2. C'è errore.
```

Fig. 12 Esempio di funzionamento dell'algoritmo di Quality Evaluator

5.3 Risultato e osservazioni

In output, Quality Evaluator genera un file CSV così strutturato:

quality_evaluation.csv : `CODE_A,"WORDS","ERRORS"`

dove:

- `CODE_A` è il codice univoco dell'abstract
- `WORDS` è il numero totale di parole contenute nell'abstract una volta normalizzato
- `ERRORS` è il numero di errori individuati nell'abstract considerato

Vanno, inoltre, menzionate le risorse richieste dalla computazione. A differenza degli altri due software realizzati, i quali hanno impiegato pochi secondi per l'esecuzione senza occupare più di 40MB di RAM sul sistema di prova, Quality Evaluator ha un consumo di risorse ben più elevato.

In particolare, Quality Evaluator ha impiegato in media 260 secondi per ogni porzione di abstract analizzata, per un totale di circa 2 ore e 40 minuti di esecuzione. Il consumo di RAM è stato di circa 380MB, costante per tutta la durata dell'esecuzione. Sul piano del multithreading, Quality Evaluator tende ad utilizzare un solo core. Se disponibile, ne dedica un altro all'esecuzione del *garbage collector*.

Il consumo di risorse così elevato è da attribuirsi al POS tagger di CoreNLP il quale, per stessa dichiarazione degli sviluppatori [Stanford CoreNLP: POS tagger, 2012], occupa circa 200MB di RAM su un sistema x64.

Per completezza è utile segnalare il fatto che l'algoritmo bidirezionale portava il sistema all'utilizzo di 1,5GB di RAM prima di generare memory leak.

Per quanto riguarda i risultati ottenuti da Quality Evaluator, la tabella sottostante [Fig.13] mostra il numero di abstract nei quali è stato individuato almeno un errore.

Number of errors detected	Abstracts	Abstract percentage
Errors = 0	**3117**	**82,20%**
Errors > 0	**675**	**17,80%**
1	567	14,95%
2	76	2,00%
3	15	0,40%
4	10	0,26%
5	4	0,11%
6	1	0,03%
8	1	0,03%
21	1	0,03%
Total abstracts	**3792**	**100,00%**

Fig. 13 Abstract per numero di errori rilevati

Il sistema ha individuato errori nel 18% degli abstract analizzati. Nella maggior parte dei casi si tratta di una sola occorrenza.

È un risultato che, a prima vista, può sembrare poco entusiasmante, tuttavia vanno fatte alcune considerazioni:

- Il corpus analizzato non è un training corpus ma un insieme di descrizioni di progetti che, considerato il loro fine, è lecito supporre abbiano un numero contenuto di errori
- Il database degli errori è un database aperto, perciò l'efficacia nell'analisi di Quality Evaluator può essere ulteriormente incrementata
- Lo scopo di Quality Evaluator non è trovare tutti gli errori presenti in un abstract ma indicare gli abstract mal formati. È possibile che in un abstract in cui sia stato rilevato un solo errore vi siano altre sviste non considerate nel database di Quality Evaluator o che occorrono su un numero di parole maggiore di due

Nella pratica, Quality Evaluator non distingue gli abstract ben scritti da quelli errati ma è in grado di indicare una parte di quelli che presentano sviste.

Questo è utile a DCUBE, in quanto consente di evidenziare i casi in cui sia stato assegnato un finanziamento a progetti che presentano errori nella loro descrizione o, se non vi fossero, permette di individuare, come possibile causa della mancata assegnazione, una scadente qualità testuale.

6. Conclusioni

I sistemi di analisi degli abstract realizzati in questo progetto hanno prodotto, come ci si prefiggeva, risultati direttamente utilizzabili da DCUBE. Tuttavia, per quanto riguarda l'utilità effettiva, i dati ottenuti non sono bastati a validare le discriminazioni scoperte dal progetto di Discrimination Discovery.

Gender Evaluator ha prodotto, come approfondito in [3.3], un risultato inaspettato, rivelando l'utilizzo di uno stile di scrittura maschile anche nei casi in cui il project coordinator fosse di genere femminile.

Sector Coverage [4.3] e Quality Evaluator [5.3] hanno, invece, prodotto risultati conformi a quelli attesi. Risultati che, però, non sono bastati in fase di validazione. Ciò è probabilmente dovuto alla quantità di testo analizzato: i dati estratti dall'analisi del solo abstract di ogni progetto si sono rivelati insufficienti.

Ad ogni modo, è possibile ipotizzare che l'analisi dell'intera descrizione di ogni progetto possa portare a risultati differenti e maggiormente robusti. I tre software realizzati possono essere facilmente adattati a questa esigenza purché si mantenga inalterata la struttura del file SQL in input.

Inoltre, Quality Evaluator costituisce una buona base di partenza nella ricerca degli errori: il database può essere ampliato e il codice opportunamente modificato affinché supporti l'analisi non solo di coppie, ma anche di gruppi formati da 3 o più parole.

7. Bibliografia

Bookblog, *The Gender Genie*

http://bookblog.net/gender/genie.php

Englishpage.com, *dizionario Verb+Preposition*

http://www.englishpage.com/prepositions/verb_preposition.html

English For Italians, *Common Mistakes*

http://www.englishforitalians.com/node/4

English Gratis, *Grammatica*

http://www.englishgratis.com/1/risorse/grammatica/gram-aggettivieipronomipossessivi.htm

Hacker Factor, *Gender Guesser*

http://www.hackerfactor.com/GenderGuesser.php

Key. (1975). *Male/Female Language.* Scarecrow Press, Metuchen.

Koppel, Argamon, Shimoni. (2001). *Automatically determining the gender of a text's author*. Bar-Ilan University Technical Report BIU-TR-01-32.

Koppel, Argamon, Fine, Shimoni (2003). *Gender, Genre, and Writing Style in Formal Written Texts.*

http://u.cs.biu.ac.il/~koppel/papers/male-female-text-final.pdf

Lakoff, R. T. (1975). *Language and Women's Place.* Harper Colophon Books, N.Y.

The Stanford Natural Language Processing Group, *CoreNLP: POS tagger*

- http://nlp.stanford.edu/software/corenlp.shtml
- http://nlp.stanford.edu/software/tagger.shtml