UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Ingegneria Enzo Ferrari
Laurea Magistrale in Ingegneria Informatica

Tesina di
Sistemi Informativi
A.A. 2017/2018

Data Integration: BigGorilla vs OpenRefine

Prof:
Domenico Beneventano

a cura di:
Carla Lamattina
Rosario Di Carlo

Indice

1. Introduzione

Data integration è il processo di consolidamento dei dati da un insieme di fonti di dati eterogenee in un dataset unico e uniforme.
Il set di dati integrato dovrebbe rappresentare correttamente e completamente il contenuto di tutte le fonti di dati, utilizzare un singolo modello di dati e un singolo schema, contenere solo una singola rappresentazione di ogni entità del mondo reale e non contenere dati in conflitto su singole entità.
Esistono diversi programmi che ci permettono di fare data integration, a questo scopo, la presente tesina si occupa di fare un confronto tra BigGorilla che è un ecosistema open source di integrazione e preparazione dei dati, basato su Python, e OpenRefine, precedentemente chiamato Google Refine, che è un'applicazione desktop open source autonoma per la pulitura e la trasformazione dei dati.
Si sono presi in considerazione due dataset di film che sono stati acquisiti da diverse fonti;
Il primo, "Kaggle dataset 5000 movie" è memorizzato in un file .csv che è già strutturato e pronto per l'uso. Il secondo "IMDB Plain Text Data", invece, è una raccolta di file di testo semi-strutturati che devono essere elaborati per estrarre i dati.
Prima di proseguire nell'elaborato, ci sembra opportuno parlare di Eterogeneità.
Esistono 5 tipi di eterogeneità:

1. **Technical Heterogeneity**: comprende tutte le differenze nei mezzi per accedere ai dati, non i dati stessi.
2. **Syntactical Heterogeneity**: comprende tutte le differenze nella codifica dei valori.
3. **Data Model Heterogeneity**: comprende le differenze nel modello di dati utilizzato per rappresentare i dati (Relational data model, XML data model, Object-oriented data model, RDF graph data model).
4. **Structural Heterogeneity**: comprende le differenze nel modo in cui i diversi schemi rappresentano la stessa parte della realtà.
5. **Semantic Heterogeneity**: comprende le differenze relative al significato dei dati e degli elementi dello schema.

L'obiettivo dell'integrazione dei dati è quello di superare tutti questi tipi di eterogeneità.
Nel capitolo 2 di questo elaborato spiegheremo il processo di Data Integration nelle sue 4 fasi: Data Collection, Schema Mapping Data Translation, Identity Resolution e Data Fusion.
Nel capitolo 3 inizieremo a vedere il lavoro fatto grazie all'uso di BigGorilla, riportando screenshot esplicativi della nostra analisi.
Nel capitolo 4 faremo lo stesso ma, questa volta, utilizzando OpenRefine.

Nell'ultimo capitolo faremo un confronto tra BigGorilla e OpenRefine e trarremo le conclusioni, scopo appunto di questo elaborato.

2. Data Integration process

La Data integration riguarda i processi da eseguire per estrapolare i dati da diverse sorgenti e fornire una visione globale ed unificata da poter analizzare e interrogare.
L'integrazione dei dati è generalmente implementata nei data warehouse attraverso software specializzati che ospitano archivi di dati di grandi dimensioni da risorse interne ed esterne.
ETL (estrai, trasforma e carica) è la forma più comune di data integration trovata nel data warehousing.
Extract: i dati vengono estratti dalle fonti interne e esterne sulla base di un approccio che prevede una fase iniziale nella quale il data warehouse "vuoto" viene alimentato con i dati disponibili riferiti a periodi passati e fasi successive caratterizzate da estrazioni di natura incrementale.
Transform: è costituita da due fasi: fase di pulitura e fase di trasformazione.
Nella fase di pulitura, ci si propone di migliorare la qualità dei dati estratti dalle diverse fonti mediante la correzione di inconsistenze, inesattezze, carenze.
Nella fase di trasformazione si procede ad ulteriori conversioni dei dati, che ne garantiscano l'omogeneità rispetto all'integrazione delle diverse fonti, e aggregazioni, per ottenere le sintesi necessarie a svolgere le analisi.
Load: i dati opportunamente estratti e trasformati vengono infine inseriti nelle strutture informative (tabelle) predisposte.
ETL ha degli svantaggi, quali: sono richiesti frequenti aggiornamenti e si ha la limitazione di accedere solo ad una parte dei dati, tra l'altro elaborati, invece di poter accedere alla totalità presente nelle sorgenti.
Esistono altre tecniche, tra cui data federation, database replication, data synchronization e così via. Le soluzioni basate su queste tecniche possono essere codificate a mano, effettuate attraverso dei tool o facendo un mix di entrambi.
La data integration si divide in due grandi aree: la **data integration analitica** che supporta la business intelligence e il data warehousing, e la **data integration operativa** che viene applicata al di fuori del Business intelligence/ data warehousing, per la migrazione, il consolidamento e la sincronizzazione dei database operativi, nonché allo scambio di dati in un contesto business-to-business .
Il processo della data integration si compone di 4 step:

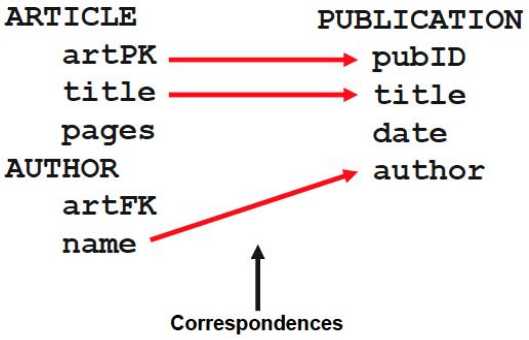
1. Data Collection
2. Schema Mapping Data Translation
3. Identity Resolution
4. Data Quality Assessment Data Fusion.

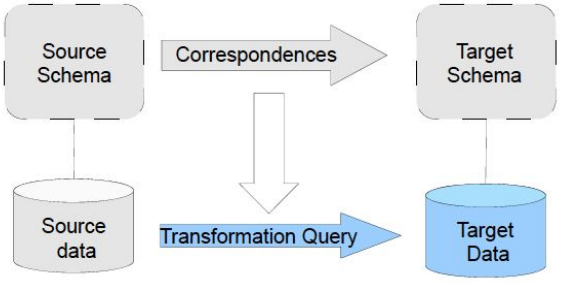
2.1 Data Collection

È la fase in cui vengono raccolti tutti i dati necessari per l'analisi da effettuare. L'obiettivo della data collection è quello di risolvere l'eterogeneità del modello tecnico e di dati in modo che i dati di tutte le fonti siano accessibili, raccolti e rappresentati nello stesso modello di dati.

2.2 Schema Mapping Data Translation

L'obiettivo dello Schema Mapping Data Translation è la risoluzione dell'eterogeneità semantica strutturale trovando le corrispondenze tra gli elementi dei diversi schemi e traducendo i dati in un singolo schema di destinazione basato su queste corrispondenze.





Esistono 4 tipi di corrispondenze:

- One-to-One: movie.title → item.name
- One-to-Many: person.name → split() → FirstName(Token1)
  → Surname(Token2)
- Many-to-One: Product.basePrice * (1+Location.taxRate) → item.price
- Higher-Order: ad esempio quando bisogna correlare diversi tipi di elementi del
  modello di dati.

2.3 Identity Resolution

L'Identity Resolution è un processo di gestione dei dati attraverso il quale un'identità viene ricercata e analizzata tra diversi insiemi di dati e database per trovare una corrispondenza e/o risolvere l'identità. L'obiettivo è quello di risolvere l'eterogeneità semantica identificando tutti i record in tutte le fonti di dati che descrivono la stessa entità del mondo reale.
Nell'identity resolution ci sono 2 sfide chiave, eliminare i duplicati, confrontando più attributi dei record e usando misure di similarità specifiche per ciascun attributo,

evitare confronti non necessari in modo da ridurre l'elevata complessità computazionale che si ottiene nel confrontare ogni coppia di record in dataset molto grandi.
Allo scopo di trovare similarità affidabili è bene che i dati prima vengono normalizzati, quindi, portare tutto in minuscolo, rimuovere la punteggiatura, rimuovere le stopwords, sistemare il formato dei valori numerici e le unità di misura, normalizzare abbreviazioni e sinonimi.

2.4 Data Quality Assessment Data Fusion

La Data Fusion è il processo di integrazione di più fonti di dati per produrre informazioni più coerenti, accurate e utili di quelle fornite da ogni singola fonte di dati. Ha come obiettivo quello di risolvere i conflitti di dati, combinando i valori degli attributi dei record duplicati in un'unica descrizione consolidata di un'entità.
La qualità dei dati è un costrutto multidimensionale che misura la "idoneità all'uso" dei dati per un'attività specifica.
Per valutare la qualità dei dati si usano metriche basate sul contenuto, come ad esempio vincoli e regole di coerenza, rilevamento di valori anomali statistici, metriche basate sulla provenienza, metriche basate sulla valutazione, ad esempio "Leggi solo articoli di notizie con almeno 100 like di Facebook", "Accetta raccomandazioni di un amico nei ristoranti, ma non si fida dei computer ", "Preferisci contenuti da siti Web con un PageRank elevato".

3. BigGorilla

BigGorilla, sviluppato dal Recruit Institute of Technology e dalla University of Wisconsin,è un ecosistema open-source per l'integrazione e la preparazione dei dati, basato su Python. I vari componenti per l'uso di BigGorilla sono scaricabili gratuitamente.
Al fine di poter utilizzare e testare BigGorilla è stato necessario scaricare Anaconda navigator, un framework Python, open source, utilizzato da data scientists e developers.
L'utilizzo di BigGorilla prevede l'esecuzione di 4 step:

1. Data Acquisition
2. Data Extraction
3. Data Profiling & cleaning
4. Data matching & merging

3.1 Data Acquisition

Come detto nell'introduzione di questa tesina, abbiamo preso in considerazione dataset diversi scaricati da diverse fonti. A tale scopo, abbiamo utilizzando **urllib**, un pacchetto python per il recupero dei dati attraverso il web.

```
In [ ]: import urllib.request
        import os
        if not os.path.exists('./data'):
            os.makedirs('./data')
        kaggle_url = 'https://github.com/sundeepblue/movie_rating_prediction/raw/master/movie_metadata.csv'
        if not os.path.exists('./data/kaggle_dataset.csv'):
            response = urllib.request.urlretrieve(kaggle_url, './data/kaggle_dataset.csv')
```

```
In [ ]: imdb_url = 'https://anaconda.org/BigGorilla/datasets/1/download/imdb_dataset.csv'
        if not os.path.exists('./data/imdb_dataset.csv'):
            response = urllib.request.urlretrieve(kaggle_url, './data/imdb_dataset.csv')
```

Kaggle_dataset.csv è un dataset che contiene 5043 film con 28 colonne.

In figura un estratto del dataset kaggle_dataset

4]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	
6	Color	Sam Raimi	392.0	156.0	0.0	4000.0	James Franco	24000.0	336530303.0	
7	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adve

plot_keywords	movie_imdb_link	num_user_for_reviews	language	country	content_rating
avatar|future|marine|native|paraplegic	http://www.imdb.com/title/tt0499549/?ref_=fn_t...	3054.0	English	USA	PG-13
goddess|marriage ceremony|marriage proposal|pi...	http://www.imdb.com/title/tt0449088/?ref_=fn_t...	1238.0	English	USA	PG-13
bomb|espionage|sequel|spy|terrorist	http://www.imdb.com/title/tt2379713/?ref_=fn_t...	994.0	English	UK	PG-13
deception|imprisonment|lawlessness|police offi...	http://www.imdb.com/title/tt1345836/?ref_=fn_t...	2701.0	English	USA	PG-13
NaN	http://www.imdb.com/title/tt5289954/?ref_=fn_t...	NaN	NaN	NaN	NaN
alien|american civil war|male nipple|mars|prin...	http://www.imdb.com/title/tt0401729/?ref_=fn_t...	738.0	English	USA	PG-13
sandman|spider man|symbiote|venom|villain	http://www.imdb.com/title/tt0413300/?ref_=fn_t...	1902.0	English	USA	PG-13

	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
)	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000
	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0
2	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000
3	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000
4	NaN	NaN	NaN	12.0	7.1	NaN	0
5	PG-13	263700000.0	2012.0	632.0	6.6	2.35	24000
6	PG-13	258000000.0	2007.0	11000.0	6.2	2.35	0
7	PG	260000000.0	2010.0	553.0	7.8	1.85	29000
8	PG-13	250000000.0	2015.0	21000.0	7.5	2.35	118000
9	PG	250000000.0	2009.0	11000.0	7.5	2.35	10000

	genres	actor_1_name	movie_title	num_voted_users	cast_total_facebook_likes	actor_3_name	facenumber_in_poster
	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	4834	Wes Studi	0.0
	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	48350	Jack Davenport	0.0
	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	11700	Stephanie Sigman	1.0
	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	106759	Joseph Gordon-Levitt	0.0
	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8	143	NaN	0.0

Imdb_dataset.csv è un dataset che contiene 869178 film con 23 colonne

	id	title	year	length	budget	rating	votes	r1	r2	r3	...	r9	r10	mpaa	Action	Animation	Comed
0	0	#	2012	15.0	10000.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
1	1	#	2014	NaN	0.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
2	2	#1	2005	30.0	5000.0	6.8	11.0	14.5	0.0	0.0	...	4.5	14.5	NaN	0	0	
3	3	#1	2009	4.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	1	
4	4	#1	2010	12.0	0.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
5	5	#1 Beauty Nail Salon	2014	5.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
6	6	#1 Serial Killer	2013	87.0	30000.0	5.8	35.0	24.5	4.5	4.5	...	4.5	24.5	NaN	0	0	
7	7	#1 With A Bullet	2011	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
8	8	#1 at the Apocalypse Box Office	2015	11.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
9	9	#10007	2015	10.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
10	10	#1137	2015	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0	0	
11	11	#137	2011	15.0	NaN	5.8	5.0	44.5	0.0	0.0	...	0.0	44.5	NaN	0	0	
12	12	#140Characters: A Documentary About Twitter	2011	30.0	NaN	7.7	9.0	14.5	0.0	0.0	...	0.0	45.5	NaN	0	0	

3.2 Data Extraction

Nel caso presentato in questa tesina, questa parte è trascurabile perchè i dataset scelti risultano in un formato utilizzabile senza particolari trasformazioni.

3.3 Data profiling & cleaning

L'obiettivo in questa fase è esaminare i dati che abbiamo acquisito ed estratto finora. Questo ci aiuta a familiarizzare con i dati, a comprendere in che modo i dati devono essere puliti o trasformati e, infine, ci consente di preparare i dati per le seguenti fasi dell'attività di integrazione dei dati.
Partendo da kaggle_dataset, iniziamo verificando la presenza di duplicati, ovvero un film che appare più di una volta nei dati.
Esistono molte strategie per gestire i duplicati. Il metodo che abbiamo scelto di adottare è quello di mantenere soltanto la prima occorrenza di ogni record duplicato.

```
# Loading the Kaggle dataset from the .csv file (kaggle_dataset.csv)
kaggle_data = pd.read_csv('./data/kaggle_dataset.csv')
kaggle_data = kaggle_data.drop_duplicates(subset=['movie_title', 'title_year'], keep='first').copy()
```

Dopo aver eliminato i duplicati, abbiamo normalizzato gli attributi che rappresentano titolo e anno di ogni film.

Il seguente frammento di codice opera le seguenti trasformazioni:

- trasforma tutti i titoli dei film in minuscolo,
- rimuove alcuni caratteri come il simbolo apice e il simbolo "?"
- sostituisce alcuni caratteri speciali come "&" con "and".

```
def preprocess_title(title):
    title = title.lower()
    title = title.replace(',', ' ')
    title = title.replace("'", '')
    title = title.replace('&', 'and')
    title = title.replace('?', '')
    return title.strip()

kaggle_data['norm_movie_title'] = kaggle_data['movie_title'].map(preprocess_title)
kaggle_data.sample(3, random_state=0)

def preprocess_year(year):
    if pd.isnull(year):
        return '?'
    else:
        return str(int(year))

kaggle_data['norm_title_year'] = kaggle_data['title_year'].map(preprocess_year)
kaggle_data.head()
```

Nelle figure seguenti vediamo il risultato della trasformazione

genres	actor_1_name	movie_title	num_voted_users	budget	title_year	actor_2_facebook_likes
Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	237000000.0	2009.0	936.0
Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	300000000.0	2007.0	5000.0
Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	245000000.0	2015.0	393.0
Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	250000000.0	2012.0	23000.0
Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8	NaN	NaN	12.0
				263700000.0	2012.0	632.0
				258000000.0	2007.0	11000.0
				260000000.0	2010.0	553.0

aspect_ratio	movie_facebook_likes	norm_movie_title	norm_title_year
1.78	33000	avatar	2009
2.35	0	pirates of the caribbean: at worlds end	2007
2.35	85000	spectre	2015
2.35	164000	the dark knight rises	2012
NaN	0	star wars: episode vii - the force awaken	?

Abbiamo poi effettuato lo stesso tipo di normalizzazione sul secondo dataset imdb_dataset.csv

3.4 Data matching & merging

L'obiettivo principale di questa fase è trovare delle corrispondenze tra i dati acquisiti da diverse fonti per unificarli in un unico dataset.
Dato che i dataset presi in considerazione sono raccolti da fonti diverse, è probabile che il nome di uno stesso film compaia in modo leggermente diverso.
Per essere in grado di trovare tali corrispondenze, si può guardare la somiglianza dei titoli dei film e considerare stessa entità i titoli con un'elevata somiglianza.
A tale scopo, BigGorilla fornisce un package Python chiamato **py_stringsimjoin** che effettua un join di similarità su due dataset.
Il seguente frammento di codice usa py_stringsimjoin per abbinare tutti i titoli che hanno una distanza di modifica, identificata con threshold, pari a 1, cioè, al massimo c'è un carattere che deve essere modificato, aggiunto o rimosso per rendere identici entrambi i titoli.
Una volta completata l'unione di similarità, vengono selezionate solo le coppie di titoli prodotte nello stesso anno.

```
import py_stringsimjoin as ssj
import py_stringmatching as sm

imdb_data['id'] = range(imdb_data.shape[0])
kaggle_data['id'] = range(kaggle_data.shape[0])
similar_titles = ssj.edit_distance_join(imdb_data, kaggle_data, 'id', 'id', 'norm_title',
                                        'norm_movie_title', l_out_attrs=['norm_title', 'norm_year'],
                                         r_out_attrs=['norm_movie_title', 'norm_title_year'], threshold=1)

data_attempt2 = similar_titles[similar_titles.r_norm_title_year == similar_titles.l_norm_year]
data_attempt2.shape
```

```
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 00:04:42
```

```
(4689, 8)
```

Il matching effettuato in questo modo trova 4689 corrispondenze.

Per ottenere un risultato migliore abbiamo ripetuto il processo utilizzando Magellan che è uno strumento disponibile pubblicamente in Python nel package py_entitymatching, sviluppato dalla University of Wisconsin.
Magellan consente di abbinare due tabelle (o una tabella con se stessa) usando tecniche di apprendimento supervisionate.
Abbiamo creato una nuova colonna in ogni dataset in modo da combinare i valori di attributi importanti in una singola stringa, che abbiamo chiamato mixiture.
Quindi, come prima, usiamo py_stringsimjoin per trovare un insieme di entità che si sovrappongono nei valori delle colonne.

```
import py_stringsimjoin as ssj
import py_stringmatching as sm

imdb_data['id'] = range(imdb_data.shape[0])
kaggle_data['id'] = range(kaggle_data.shape[0])


# transforming the "budget" column into string and creating a new **mixture** column
ssj.utils.converter.dataframe_column_to_str(imdb_data, 'budget', inplace=True)
imdb_data['mixture'] = imdb_data['norm_title'] + ' ' + imdb_data['norm_year'] + ' ' + imdb_data['budget']

# repeating the same thing for the Kaggle dataset
ssj.utils.converter.dataframe_column_to_str(kaggle_data, 'budget', inplace=True)
kaggle_data['mixture'] = kaggle_data['norm_movie_title'] + ' ' + kaggle_data['norm_title_year'] + \
                         ' ' + kaggle_data['budget']



C = ssj.overlap_coefficient_join(kaggle_data, imdb_data, 'id', 'id', 'mixture', 'mixture', sm.WhitespaceTokenizer(),
                                 l_out_attrs=['norm_movie_title', 'norm_title_year', 'duration',
                                              'budget', 'content_rating'],
                                 r_out_attrs=['norm_title', 'norm_year', 'length', 'budget', 'mpaa'],
                                 threshold=0.65)
C.shape
```

```
C:\Users\rsdic\Anaconda3\lib\site-packages\py_stringsimjoin\utils\converter.py:115: FutureWarning: Conversion of the
second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float
64 == np.dtype(float).type`.
  elif pd.np.issubdtype(col_type, pd.np.float):
C:\Users\rsdic\Anaconda3\lib\site-packages\py_stringsimjoin\utils\converter.py:115: FutureWarning: Conversion of the
second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float
64 == np.dtype(float).type`.
  elif pd.np.issubdtype(col_type, pd.np.float):
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 00:01:30
```

```
(18317, 14)
```

Alla fine di questo processo otteniamo un numero di corrispondenze **pari a 18317**, significativamente superiore al caso precedente.
Il dataset candidato, chiamato C, formato, si compone dei seguenti attributi:
_id
l_id
r_id

l_norm_movie_title
l_norm_title_year
l_duration
l_budget
l_content_rating
r_norm_title
r_norm_year
r_length
r_budget
r_mpaa
_sim_score

In cui il prefisso “l” indica le features della tabella sinistra (kaggle_dataset) e il prefisso “r” quelle della tabella destra (imdb_dataset).

Tramite le funzioni del pacchetto py_entitymatching specifichiamo quali attributi corrispondono alle chiavi in ciascun dataframe.
Inoltre, dobbiamo specificare quali attributi corrispondono alle chiavi esterne dei due dataframes in C.

```
import py_entitymatching as em

em.set_key(kaggle_data, 'id')   # specifying the key column in the kaggle dataset
em.set_key(imdb_data, 'id')     # specifying the key column in the imdb dataset
em.set_key(C, '_id')            # specifying the key in the candidate set
em.set_ltable(C, kaggle_data)   # specifying the left table
em.set_rtable(C, imdb_data)     # specifying the right table
em.set_fk_rtable(C, 'r_id')     # specifying the column that matches the key in the right table
em.set_fk_ltable(C, 'l_id')     # specifying the column that matches the key in the left table
```

```
True
```

```
C[['l_norm_movie_title', 'r_norm_title', 'l_norm_title_year', 'r_norm_year',
   'l_budget', 'r_budget', 'l_content_rating', 'r_mpaa']].head()
```

	l_norm_movie_title	r_norm_title	l_norm_title_year	r_norm_year	l_budget	r_budget	l_content_rating	r_mpaa
0	dude wheres my dog!	#hacked	2014	2014	20000	20000	PG	NaN
1	road hard	#horror	2015	2015	1500000	1500000	NaN	NaN
2	#horror	#horror	2015	2015	1500000	1500000	Not Rated	NaN
3	me you and five bucks	#horror	2015	2015	1500000	1500000	NaN	NaN
4	checkmate	#horror	2015	2015	1500000	1500000	NaN	NaN

Lo step successivo è quello di selezionare tra le coppie candidate quali di queste è una corrispondenza corretta o meno.

Innanzitutto abbiamo considerato una semplificazione del dataset C, formato da 500 elementi, che abbiamo chiamato sampled e che abbiamo convertito in .csv (sampled.csv).
Abbiamo poi scaricato dal sito Anaconda.org lo stesso subset classificato manualmente, labeled.csv e che rappresenta il “gold standard” per questo dataset.
La colonna label contiene il valore 1 se la coppia è una corrispondenza corretta e 0 altrimenti.

```
# Sampling 500 pairs and writing this sample into a .csv file
sampled = C.sample(500, random_state=0)
sampled.to_csv('./data/sampled.csv', encoding='utf-8')
```

```
labeled = em.read_csv_metadata('data/labeled.csv', ltable=kaggle_data, rtable=imdb_data,
                               fk_ltable='l_id', fk_rtable='r_id', key='_id')
labeled.head()
```

```
Metadata file is not present in the given path; proceeding to read the csv file.
```

	Unnamed: 0	_id	l_id	r_id	l_norm_movie_title	l_norm_title_year	l_duration	l_budget	l_content_rating	r_norm_title	r_norm_year	r_length	r_budget	r
0	4771	4771	2639	235925	eye of the beholder	1999	109.0	15000000	R	eye of the beholder	1999	109.0	35000000	
1	11478	11478	2001	600301	rocky balboa	2006	139.0	24000000	PG	rocky balboa	2006	139.0	24000000	
2	13630	13630	4160	691766	from russia with love	1963	115.0	2000000	Approved	the aeolians: from russia with love	2012	NaN	20000	
3	1972	1972	1248	101029	sex tape	2014	94.0	40000000	R	blended	2014	117.0	40000000	
4	15903	15903	722	758133	the scorch trials	2015	132.0	61000000	PG-13	the scorch trials	2015	132.0	61000000	

r_norm_year	r_length	r_budget	r_mpaa	_sim_score	label
1999	109.0	35000000	R	0.833333	1
2006	139.0	24000000	PG	1.000000	1
2012	NaN	20000	NaN	0.666667	0
2014	117.0	40000000	PG-13	0.666667	0
2015	132.0	61000000	PG-13	1.000000	1

Dal gold standard si evince che **soltanto 113 match dei 500** candidati sono dei match reali, ovvero **circa il 23%** di quelli abbinati automaticamente tramite Magellan.
Tramite gli algoritmi di machine learning vogliamo allenare un classificatore a riconoscere i match corretti tra quelli candidati.

Per fare ciò, abbiamo diviso il dataset labeled in test e train e abbiamo addestrato un classificatore di tipo Random Forest a riconoscere la label fornita dal gold standard sulla base di features estratte dai seguenti attributi:

- Titolo
- Anno
- Valutazione
- Budget

Per poter estrarre le features ci viene in aiuto il pacchetto py_entitymatching che automatizza il processo di estrazione semplicemente specificando gli attributi e la loro corrispondenza nei due dataset.
Il seguente frammento di codice utilizza inoltre il pacchetto py_entitymatching per determinare il tipo di ciascuna colonna. Considerando i tipi di colonne in ogni set di dati (memorizzati nelle variabili l_attr_types e r_attr_types) e utilizzando le funzioni di tokenizer e similarity suggerite dal pacchetto, possiamo estrarre un set di istruzioni per l'estrazione di features.
Una volta creato l'insieme delle features desiderate, sostituiamo i valori mancanti nei nostri dati con la media della colonna.
Dopo la fase di training, l'accuratezza del nostro classificatore risulta molto alta, pari a 98.08%.
Ciò significa che il nostro classificatore può discriminare,molto bene, quali sono i match reali tra quelli candidati trovati.

```
Precision : 96.23% (51/53)
Recall : 100.0% (51/51)
F1 : 98.08%
False positives : 2 (out of 53 positive predictions)
False negatives : 0 (out of 197 negative predictions)
```

Come ultimo step utilizziamo il nostro classificatore sul dataset completo per tenere soltanto i match reali.
Eliminiamo inoltre gli attributi superflui.
Il risultato finale è la seguente tabella

	_id	l_id	r_id	l_norm_movie_title	l_norm_title_year	l_budget	l_content_rating	r_norm_title	r_norm_year	r_budget	r_mpaa
0	2	4352	106	#horror	2015	1500000	Not Rated	#horror	2015	1500000	NaN
1	8	2726	450	crocodile dundee ii	1988	15800000	PG	crocodile dundee ii	1988	14000000	NaN
2	11	3406	838	500 days of summer	2009	7500000	PG-13	(500) days of summer	2009	7500000	PG-13
3	24	3631	1872	10 cloverfield lane	2016	15000000	PG-13	10 cloverfield lane	2016	15000000	PG-13
4	27	2965	1883	10 days in a madhouse	2015	12000000	R	10 days in a madhouse	2015	12000000	R

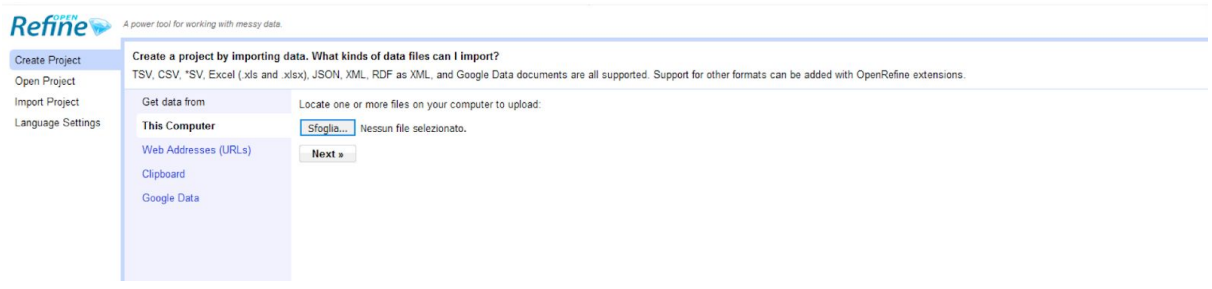
Contenente **3963 valori** matchati tra i due dataset.
Abbiamo reso disponibile il dataset integrato all'indirizzo https://goo.gl/jEVZ9v

4. OpenRefine

OpenRefine è un software desktop open source e multipiattaforma sviluppato da Google.
Consente la trasformazione e la pulitura dei dati tramite una comoda interfaccia web avviata dal servizio web-server di OpenRefine operante sulla macchina locale.
Le espressioni di trasformazione possono essere scritte utilizzando il linguaggio GREL (General Refine Expression Language) o in Python.

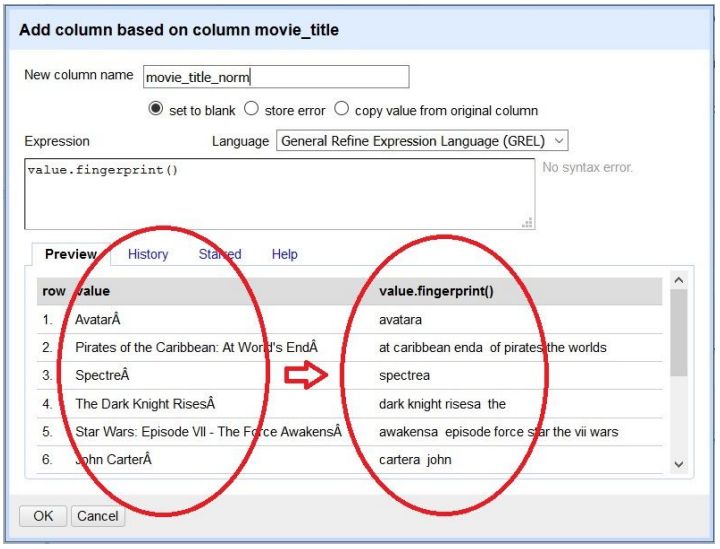
4.1. Data Acquisition

OpenRefine supporta l'acquisizione dei dati da tutti i formati più comuni: CSV, XML, JSON, RDF, Text.
Utilizziamo semplicemente l'interfaccia web per aprire i dataset "kaggle_dataset" e "imdb_dataset" in due istanze separate di OpenRefine.
Accediamo ad Open Refine e selezioniamo la voce Create Project dal menu sulla sinistra. Poi selezioniamo dal computer il file che vogliamo importare e premiamo il pulsante next. A questo punto compare un'anteprima del progetto. Nella parte bassa, appare un altro menu, da cui è possibile selezionare alcune opzioni, come ad esempio il carattere di separazione dei campi del CSV ed altre opzioni sulla destra. Un'altra opzione, situata in alto a sinistra, da la possibilità di settare la codifica. A questo punto, nella parte in alto a destra della pagina possiamo cambiare il nome del progetto e poi premere il tasto Create Project.



4.2. Data profiling & cleaning

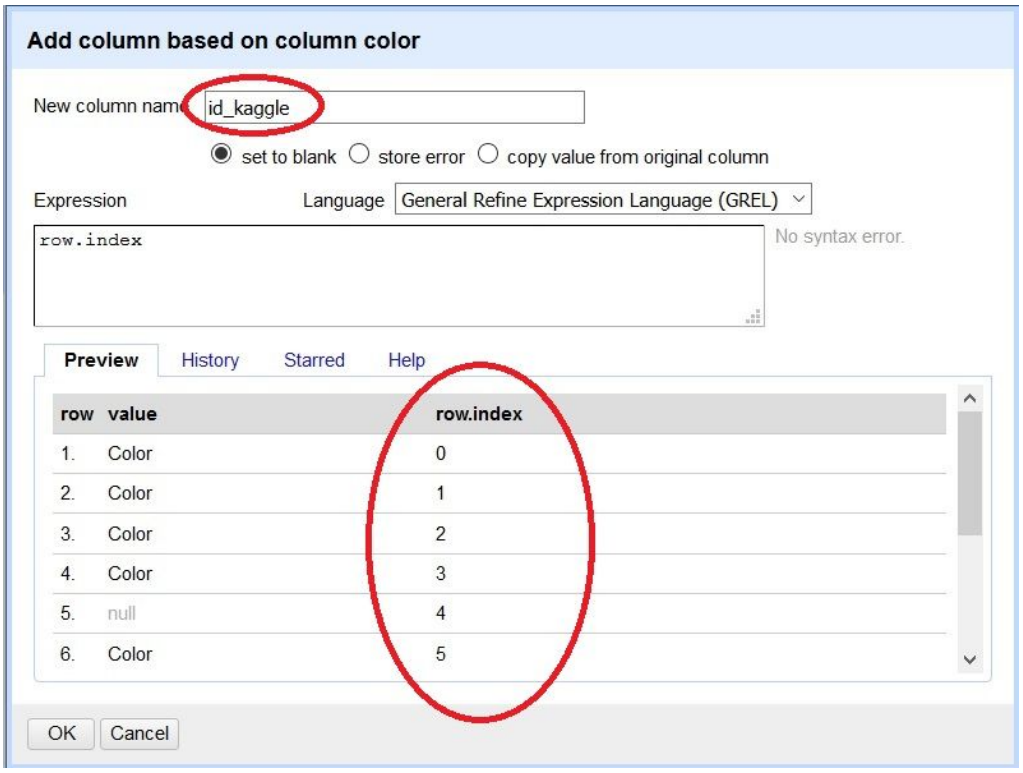
Come precedentemente visto con BigGorilla vogliamo esaminare i dati che abbiamo acquisito e prepararli per le seguenti fasi dell'attività di integrazione dei dati.
Partendo da kaggle_dataset utilizziamo la funzione fingerprint, messa a disposizione dal linguaggio GREL per normalizzare i valori della colonna che contiene il titolo del film.



La funzione fingerprint opera le seguenti trasformazioni:

- Rimuove gli spazi bianchi iniziali e finali
- Cambia tutti i caratteri nella loro rappresentazione in minuscolo
- Rimuove tutti i caratteri di punteggiatura e controllo
- Normalizza i caratteri occidentali estesi nella loro rappresentazione ASCII
- Divide la stringa in token separati da spazi bianchi
- Ordina i token e rimuove i duplicati
- Unisce i token di nuovo insieme

Notiamo la mancanza di un attributo id per identificare univocamente le righe, dunque lo aggiungiamo tramite la funzione index.



Ripetiamo poi le stesse trasformazioni sul dataset “imdb_dataset”.

4.3. Data matching & merging

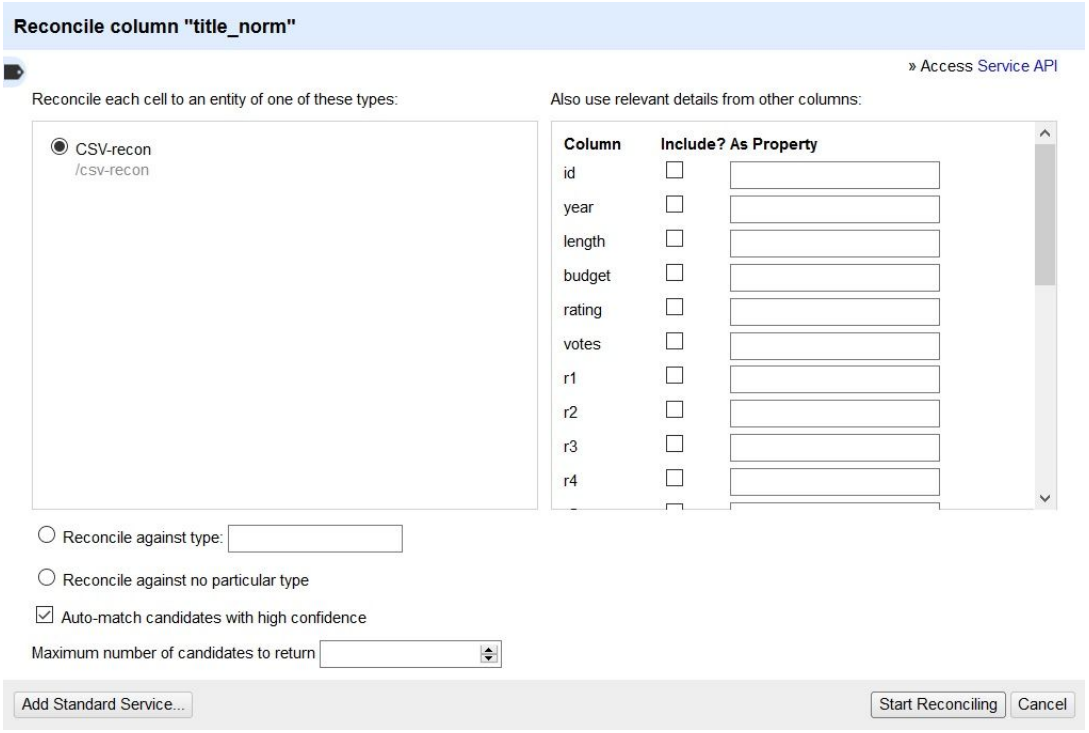
Iniziamo adesso la fase più importante di trovare delle corrispondenze tra i due dataset acquisiti.
Anche questa volta adottiamo la strategia di associare i dati in funzione della similarità tra i titoli dei film.
A tale scopo, OpenRefine fornisce un servizio chiamato “Reconcile Csv” che effettua il matching utilizzando l’algoritmo di similarità di Dice tra stringhe.
Lanciamo il servizio Reconcile Csv con i seguenti parametri:

- Nome del file csv (kaggle_dataset)
- Nome della colonna sulla quale fare il merge (titolo)
- Nome della colonna che contiene la chiave (id)

Il servizio è ora attivo sul dataset “kaggle_dataset”.

Il dataset “imdb_dataset” contiene più di 800.000 righe il che rende l’operazione di matching estremamente lenta.
Per velocizzare il processo lo riduciamo ad un subset di circa 200.000 righe.

Avviamo il processo tramite interfaccia Renconcile - Start reconcile.



Al termine dell’operazione ogni riga è stata matchata con il suo miglior candidato.

Tramite le seguenti istruzioni in linguaggio GREL:

- Cell.recon.best.id
- Cell.recon.best.score
- Cell.recon.best.name

Possiamo generare le colonne:

- **Match id:** contenente l’id del record matchato
- **Match score**: contenente il valore di similarità tra i record
- **Match title:** contenente il titolo del film matchato

208649 rows

Show as: **rows** records Show: 5 10 25 **50** rows

All		id	title_norm	match_score	true_match	match_title	match_id
			avatara (0) Create new item Search for match				
40.	39		actorslife lifea (0.429) Create new item Search for match	0.4285714285714286		lifea	543
41.	40		alleman allegianta (0.5) Create new item Search for match	0.5		allegianta	264
42.	41		american americana the (0.7) Create new item Search for match	0.7	2201	americana the	2201
43.	42		amish amistada (0.5) Create new item Search for match	0.5		amistada	1248
44.	43		anatomy comedy of romantic the a broken club comedya hearts romantic the (0.548) Create new item Search for match	0.5483870967741935		a broken club comedya hearts romantic the	4567
45.	44		a andjusticeforall of postracial rhetoric the us and deceptive jaya mentors mysteries of practice ricky the (0.444) Create new item Search for match	0.4444444444444444		and deceptive jaya mentors mysteries of practice ricky the	4746
46.	45		anniehall annie halla (0.737) Create new item Search for match	0.7368421052631579	3889	annie halla	3889

Decidiamo di tenere solamente i valori che hanno un **Match score** che supera il valore di soglia = 0.6 .
Le corrispondenze trovate imponendo questa condizione risultano **17105**.

Lo step successivo è quello di selezionare tra le coppie candidate quali di queste è una corrispondenza corretta o meno.
Per fare ciò possiamo utilizzare ancora una volta gli algoritmi di machine learning.
Modifichiamo la struttura del dataset, in modo che corrisponda a quella utilizzata precedentemente in BigGorilla, tramite le seguenti trasformazioni.
Rinominare gli attributi:

- id -> r_id
- title --> r_norm_title
- year - r_norm_year
- length - r_norm_length
- budget - r_budget
- mpaa - r_mpaa

- match_id - l_id
- match_title - l_norm_movie_title

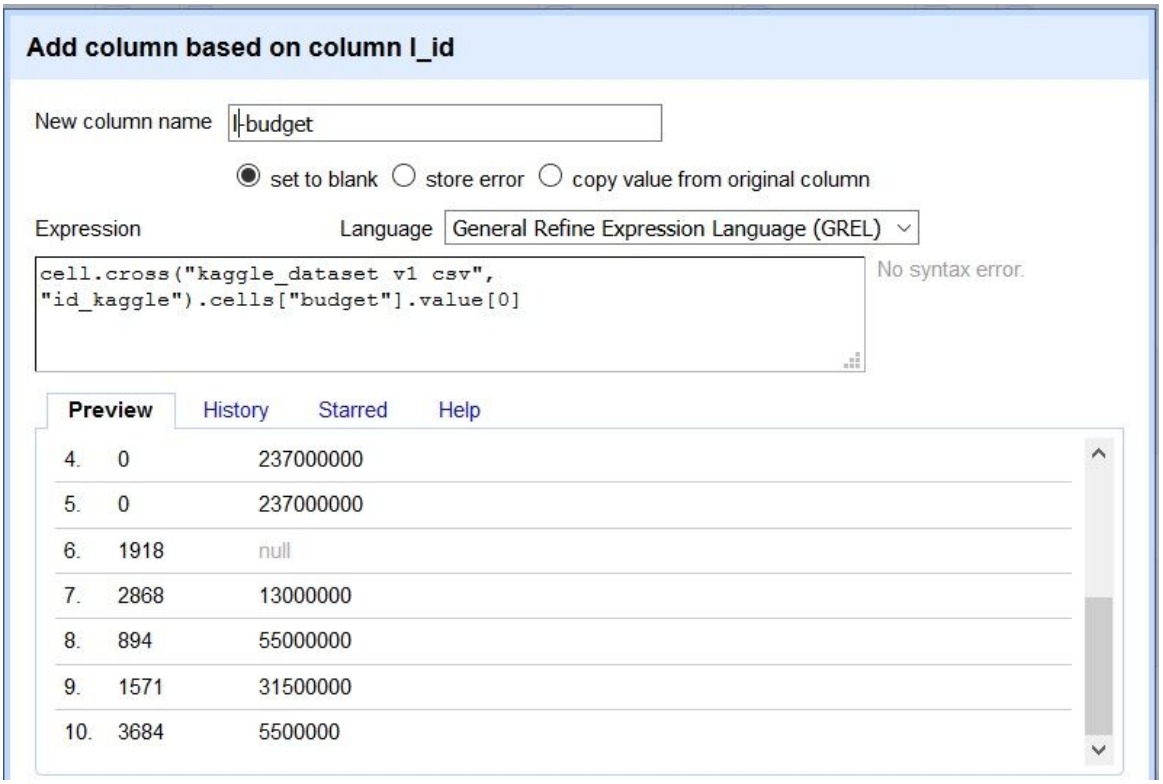
- match_score - _sim_score

Aggiungere i seguenti attributi mancanti dal dataset kaggle tramite join sull'attributo l_id:

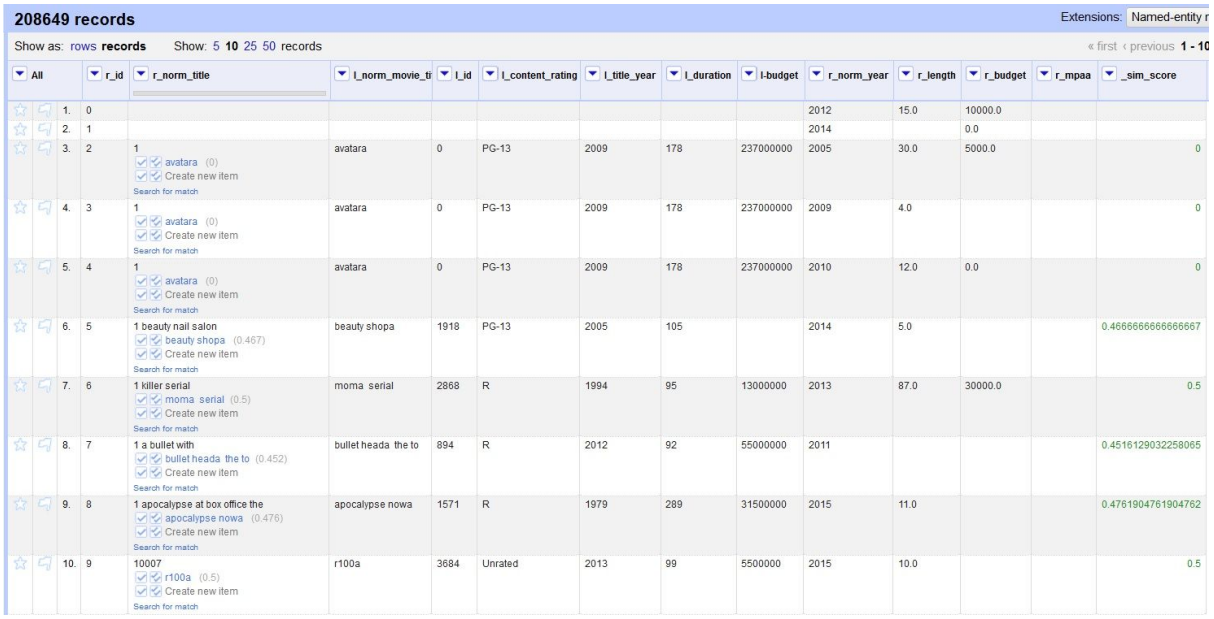
- l_norm_title_year
- l_budget
- L_content_rating

Per effettuare il join utilizziamo la seguente sintassi GREL:

cell.cross("kaggle_dataset csv", "id_kaggle").cells[“budget"].value[0]



Il dataset ottenuto è il seguente:



208649 records

Extensions: Named-entity r

Show as: rows **records** Show: 5 **10** 25 50 records « first ‹ previous **1 - 10**

All	r_id	r_norm_title	l_norm_movie_ti	l_id	l_content_rating	l_title_year	l_duration	l-budget	r_norm_year	r_length	r_budget	r_mpaa	_sim_score
1.	0								2012	15.0	10000.0		
2.	1								2014		0.0		
3.	2	1 avatara (0) Create new item Search for match	avatara	0	PG-13	2009	178	237000000	2005	30.0	5000.0		0
4.	3	1 avatara (0) Create new item Search for match	avatara	0	PG-13	2009	178	237000000	2009	4.0			0
5.	4	1 avatara (0) Create new item Search for match	avatara	0	PG-13	2009	178	237000000	2010	12.0	0.0		0
6.	5	1 beauty nail salon beauty shopa (0.467) Create new item Search for match	beauty shopa	1918	PG-13	2005	105		2014	5.0			0.4666666666666667
7.	6	1 killer serial moma serial (0.5) Create new item Search for match	moma serial	2868	R	1994	95	13000000	2013	87.0	30000.0		0.5
8.	7	1 a bullet with bullet heada the to (0.452) Create new item Search for match	bullet heada the to	894	R	2012	92	55000000	2011				0.4516129032258065
9.	8	1 apocalypse at box office the apocalypse nowa (0.476) Create new item Search for match	apocalypse nowa	1571	R	1979	289	31500000	2015	11.0			0.4761904761904762
10.	9	10007 r100a (0.5) Create new item Search for match	r100a	3684	Unrated	2013	99	5500000	2015	10.0			0.5

Importiamo ora il dataset in Python.
Utilizziamo la libreria Scikit per allenare un classificatore di tipo Random Forest, sul dataset classificato manualmente (gold standard), per riconoscere i match reali tra quelli candidati.
Utilizziamo come features i seguenti attributi:

- Title_year
- Duration
- Budget
- Length
- Sim_score

Come nel caso di BigGorilla, otteniamo una accuracy pari a 98% per il nostro classificatore.

```
In [86]: openrefine = pd.read_csv("data2/openrefine_merge.csv",',')
         openrefine = openrefine.fillna(openrefine.mean())
         openrefine = openrefine[openrefine._sim_score > 0.6]

         openrefine2 = openrefine.drop(columns=['l_id','r_id','l_norm_movie_title','l_content_rating','r_norm_title','r_mpaa'])
         openrefine2.describe()
```

Out[86]:

	l_title_year	l_duration	l-budget	r_norm_year	r_length	r_budget	_sim_score
count	17105.000000	17105.000000	1.710500e+04	17105.000000	17105.000000	1.710500e+04	17105.000000
mean	2002.470649	107.776290	3.831783e+07	1994.402865	46.663985	3.782222e+06	0.696722
std	12.633484	27.831713	1.440514e+08	30.312526	36.855282	1.132553e+07	0.091685
min	1927.000000	7.000000	2.180000e+02	1894.000000	1.000000	0.000000e+00	0.603175
25%	1999.000000	94.000000	8.000000e+06	1991.000000	13.000000	2.339565e+06	0.628571
50%	2005.000000	104.000000	2.400000e+07	2009.000000	47.029846	2.339565e+06	0.666667
75%	2011.000000	119.000000	4.000000e+07	2013.000000	73.000000	2.339565e+06	0.733333
max	2016.000000	511.000000	4.200000e+09	2023.000000	1044.000000	2.500000e+08	0.985507

```
In [87]: openrefine_labels = clf.predict(openrefine2)
         openrefine["label"] = openrefine_labels
         openrefine= openrefine[openrefine["label"] == 1]
         openrefine.describe()
```

Out[87]:

	l_id	l_title_year	l_duration	l-budget	r_norm_year	r_length	r_budget	_sim_score	label
count	1153.000000	1153.000000	1153.000000	1.153000e+03	1153.000000	1153.000000	1.153000e+03	1153.000000	1153.0

Il risultato finale è la seguente tabella contenente **1196 valori** matchati tra i due dataset.

r_id	r_norm_title	l_norm_movie_title	l_id	l_content_rating	l_title_year	l_duration	l-budget	r_norm_year	r_length	r_budget	r_mpaa	_sim_score	label
450	450	crocodile dundee ii	crocodile dundee iiaÂ	2766.0	PG	1988.0	108.0	15800000.0	1988.0	108.0	14000000.0		0.9444444444444444	1
838	838	500 days of summer	500 days of summeraÂ	3468.0	PG-13	2009.0	95.0	7500000.0	2009.0	95.0	7500000.0	PG-13	0.9444444444444444	1
1137	1137	at first sight	at first sightaÂ	1286.0	PG-13	1999.0	128.0	60000000.0	2000.0	47.02984636682366	2339564.7796096266		0.9230769230769232	1
1287	1287	a thousand words	a thousand wordsaÂ	1315.0	PG-13	2012.0	91.0	40000000.0	2010.0	9.0	2339564.7796096266		0.967741935483871	1
1872	1872	10 cloverfield lane	10 cloverfield laneaÂ	3698.0	PG-13	2016.0	104.0	15000000.0	2016.0	104.0	15000000.0	PG-13	0.9473684210526316	1
1883	1883	10 a days in madhouse	10 a days in madhouseaÂ	3015.0	R	2015.0	111.0	12000000.0	2015.0	111.0	12000000.0	R	0.9523809523809524	1
1964	1964	10 about hate i things you	10 about hate i things youaÂ	2845.0	PG-13	1999.0	97.0	16000000.0	1999.0	97.0	16000000.0	PG-13	0.96	1
2352	2352	102 dalmatians	102 dalmatiansaÂ	406.0	G	2000.0	100.0	85000000.0	2000.0	100.0	85000000.0		0.9285714285714286	1
2430	2430	10th wolf	10th wolfaÂ	3420.0	R	2006.0	107.0	8000000.0	2006.0	107.0	8000000.0	R	0.8888888888888891	1
2609	2609	12 angry men	12 angry menaÂ	4822.0	Not Rated	1957.0	96.0	350000.0	1957.0	96.0	350000.0		0.9166666666666669	1
2671	2671	12 rounds	12 roundsaÂ	2281.0	PG-13	2009.0	108.0	22000000.0	2009.0	108.0	22000000.0	PG-13	0.8888888888888891	1
2694	2694	12 a slave years	12 a slaveaÂ years	2174.0	R	2013.0	134.0	20000000.0	2013.0	134.0	20000000.0	R	0.9032258064516128	1
3179	3179	15 minutes	15 minutesaÂ	1168.0	R	2001.0	120.0	42000000.0	1999.0	25.0	2339564.7796096266		0.9	1
4087	4087	2 fast furious	2 fast furiousaÂ	515.0	PG-13	2003.0	107.0	76000000.0	2003.0	107.0	76000000.0	PG-13	0.9230769230769232	1
4459	4459	000 20 leagues sea the under	20000 leagues seaaÂ the under	3711.0	Approved	1954.0	127.0	5000000.0	1954.0	127.0	5000000.0		0.8979591836734694	1
4480	4480	20000 leagues sea the under	20000 leagues seaaÂ the under	3711.0	Approved	1954.0	127.0	5000000.0	2012.0	12.0	2339564.7796096266		0.9166666666666669	1
4497	4497	200 cigarettes	200 cigarettesaÂ	3612.0	R	1999.0	101.0	6000000.0	1999.0	101.0	6000000.0	R	0.9285714285714286	1
4544	4544	2001 a odyssey space	2001 a odysseyaÂ space	3079.0	G	1968.0	161.0	12000000.0	1968.0	149.0	12000000.0		0.9	1
4656	4656	2016 america obamas	2016 americaaÂ obamas	4181.0	PG	2012.0	87.0	2500000.0	2012.0	87.0	2500000.0	PG	0.8888888888888891	1
4789	4789	21 jump street	21 jump streetaÂ	1154.0	R	2012.0	109.0	42000000.0	2012.0	109.0	42000000.0	R	0.9285714285714286	1
4875	4875	22 jump street	22 jump streetaÂ	939.0	R	2014.0	112.0	50000000.0	2014.0	112.0	50000000.0	R	0.9285714285714286	1
5008	5008	24 7 four seven twenty	24 7 four sevenaÂ twenty	4302.0	R	1997.0	96.0	38074653.44875128	1997.0	96.0	2339564.7796096266	R	0.9047619047619048	1
5282	5282	25th hour	25th houraÂ	3846.0	R	2002.0	108.0	15000000.0	2002.0	108.0	15000000.0	R	0.8888888888888891	1
5329	5329	27 dresses	27 dresesaÂ	1607.0	PG-13	2008.0	111.0	30000000.0	2008.0	111.0	30000000.0	PG-13	0.8888888888888891	1
5370	5370	28 days later	28 days lateraÂ	3363.0	R	2002.0	113.0	8000000.0	2002.0	113.0	8000000.0	R	0.9230769230769232	1
5597	5597	3 backyards	3 backyardsaÂ	4834.0	R	2010.0	88.0	300000.0	2010.0	88.0	300000.0	R	0.9090909090909092	1
5732	5732	3 a and baby men	3 a and babyaÂ men	2602.0	PG	1987.0	102.0	11000000.0	1987.0	102.0	11000000.0		0.8666666666666667	1
5775	5775	3 back kick ninjas	3 backaÂ kick ninjas	2283.0	PG	1994.0	93.0	20000000.0	1994.0	99.0	20000000.0	PG	0.9090909090909092	1
6088	6088	30 less minutes or	30 lessaÂ minutes or	1773.0	R	2011.0	83.0	28000000.0	2011.0	83.0	28000000.0	R	0.9090909090909092	1
6094	6094	30 activity devil dragon girl insi	30 activity devil dragon girl inside	4133.0	R	2013.0	80.0	3000000.0	2013.0	80.0	3000000.0	R	0.9682539682539684	1
6222	6222	3000 graceland miles to	3000 gracelandaÂ miles to	705.0	R	2001.0	125.0	42000000.0	2001.0	125.0	62000000.0	R	0.9090909090909092	1
6231	6231	300 an empire of rise	300 an empireaÂ of rise	261.0	R	2014.0	102.0	110000000.0	2014.0	102.0	110000000.0	R	0.9047619047619048	1
6599	6599	310 to yuma	310 to yumaaÂ	1052.0	R	2007.0	122.0	55000000.0	1957.0	92.0	2339564.7796096266		0.9090909090909092	1
6600	6600	310 to yuma	310 to yumaaÂ	1052.0	R	2007.0	122.0	55000000.0	2007.0	122.0	55000000.0	R	0.9090909090909092	1

Abbiamo reso disponibile il dataset integrato al seguente indirizzo https://goo.gl/xKGBzf

5. Risultati

Il seguente report riporta i risultati ottenuti.

Dataset	Kaggle	Imdb
Record	5043	869178
BigGorilla	Match Candidati	Match Reali
n°	18317	**3963**

Per il test con OpenRefine abbiamo dovuto ridurre il dataset imdb per accelerare il processo.

Dataset	Kaggle	Imdb
Record	5043	208649
OpenRefine	Match Candidati	Match Reali
n°	17105	**1153**

Sia i match trovati con BigGorilla che con OpenRefine hanno una probabilità di essere corretti pari al 98%, in quanto abbiamo utilizzato lo stesso tipo di classificatore per selezionarli.

6. Conclusioni

Dall'analisi di questi due ambienti di integrazione dati ci è stato possibile trarre alcune considerazioni.

BigGorilla si rivela uno strumento efficace per l'integrazione, dotato di una documentazione molto chiara e ricca di esempi pratici, utile per chi si approccia per la prima volta a questa libreria di funzioni.
Non è dotato di un'interfaccia grafica e richiede una conoscenza preliminare di Python e la scrittura manuale del codice.
Le funzioni di Magellan sono molto ottimizzate per lo scopo e permettono l'integrazione di grandi quantità di dati in tempi brevi.
Si rivela molto utile anche l'integrazione all'interno di questo ecosistema degli algoritmi di machine learning, seppur facilmente sostituibili da librerie esterne come Scikit.

OpenRefine è uno strumento che non teme il confronto con software simili.
A differenza di BigGorilla, fornisce un'interfaccia web intuitiva anche per chi non ha esperienza pregressa con linguaggi di programmazione.
Le sue numerose funzioni sono anche estendibili tramite plug-in di terze parti.
Ad affiancare l'interfaccia grafica vi è la possibilità di utilizzare script sia in Python che in un linguaggio proprietario chiamato GREL (*General Refine Expression Language*).
Inoltre, OpenRefine, è un software open-source in continuo sviluppo, supportato da una vasta comunità di data scientist.
Tra gli aspetti negativi riscontrati vi è la non chiara documentazione dei servizi esterni che si integrano ad OpenRefine per estendere le sue funzioni e soprattutto dei tempi di computazione più lenti rispetto a BigGorilla durante la fase di matching.

I risultati sono invece comparabili e di alto livello in entrambi gli ambienti di integrazione dati.

In conclusione riteniamo questa esperienza utile per aver averci aperto la possibilità di familiarizzare con dei software che rappresentano lo stato dell'arte nell'ambito di data integration. Per averci fatto scoprire OpenRefine che è uno strumento molto utile di cui non avevamo sentito parlare prima di iniziare questo lavoro. Per la possibilità di poter unire competenze diverse quali l'utilizzo di linguaggi di programmazione come Python e GREL a tecniche di Machine Learning, fondamentali nelle fasi finali di integrazione per discriminare i falsi positivi dai match reali.