



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica

Sperimentazione Record Linkage

Alberici Federico - 808058

Bettini Ivo Junior - 806878

Cocca Umberto - 807191

Traversa Silvia - 816435

Anno Accademico 2019 - 2020

Indice

1	Ricerca preliminare	2
1.1	Data Quality	2
1.1.1	Metodologia Data Quality	5
1.1.2	Miglioramento	6
1.1.3	Standardizzazione	6
1.1.4	Comparazione stringhe	7
1.2	Record Linkage	8
1.2.1	Metodologie record linkage	9
1.3	Tools	11
1.3.1	Python Record Linkage Toolkit	11
1.3.2	Pandas	12
1.3.3	Fuzzy Matcher	12
2	Sperimentazione	13
2.1	Dataset	13
2.2	Analisi dei dati	13
2.3	Data preprocessing	15
2.4	Google Maps	15
2.4.1	Risultati	17
2.5	Flusso di lavoro per l'Object Identification	19
2.5.1	Metodo 1: Raggruppamenti per attributi simili	20
2.5.2	Metodo 2: Record linkage multiplo	26

1 Ricerca preliminare

Il progetto che verrà esposto è consultabile alla seguente **repository**.

1.1 Data Quality

La consapevolezza del peso che dati di alta qualità hanno nel supportare decisioni informate e, viceversa, delle conseguenze disastrose cui dati inaccurati possono portare, è cresciuta di pari passo con il diffondersi delle fonti informative a disposizione delle organizzazioni, creando sempre più forte l'esigenza di una gestione adeguata della qualità dei dati aziendali. La ricerca sulla qualità dei dati è iniziata correttamente negli anni '90 e varie definizioni di ciò sono state date nel corso degli anni.

Un gruppo di ricerca del MIT, guidato dal professor Wang, ha definito la qualità dei dati come condizione per il loro utilizzo e ha proposto il loro giudizio dipendentemente dai consumatori finali. Allo stesso tempo, hanno definito una "dimensione della qualità dei dati" come un insieme di attributi che rappresentano un singolo aspetto o costruito della qualità dei dati.

Sono necessarie tecniche di misurazione completa per consentire alle organizzazioni di valutare lo stato della qualità delle informazioni organizzative e monitorarne il miglioramento.

Ma cosa si intende quando si parla di qualità dei dati e come si misura? La qualità dei dati è una caratteristica che ha a che fare con la loro abilità di soddisfare le esigenze e le aspettative implicite o esplicite dell'utente.

Le best practices in questo ambito suggeriscono l'utilizzo di opportune metriche per la definizione e la misurazione della qualità dei dati.

Per quanto riguarda la definizione di metrica ci riferiamo alle definizioni all'interno dello standard ISO 9126-1 e framework ISM3:

- una procedura (o metodo) di misurazione, cioè un algoritmo che prende l'elemento per misurare e lo associa a misura (sia esso valore ordinale o intervallo);
- una corretta unità di misura (o scala), ovvero di dominio di valori restituiti dalla procedura di misurazione. In generale, è possibile associare diverse metriche a cias-

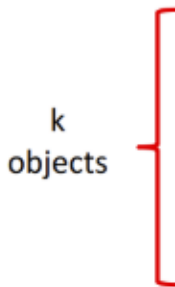
cuna dimensione di qualità;

La data quality può essere espressa attraverso molteplici dimensioni. Le dimensioni dei dati nel modello relazionale sono:

- **Accuracy:** l'accuracy di un valore v è definita come vicinanza tra v e un valore v' considerato come corretta rappresentazione del fenomeno del mondo reale. La metrica per l'accuracy sintattica è quella di adottare una funzione di distanza, per esempio la edit distance non normalizzata $UED(v1, v2)$ che rappresenta il numero di inserimenti, cancellazioni e sostituzioni di simboli alfanumerici necessari per trasformare $v1$ in $v2$. Oppure abbiamo la edit distance normalizzata $EDnorm(v1, v2)$ dove $v1$ e $v2$ sono valori nel dominio D , in cui il massimo numero di simboli è n :

$$EDnorm(v1, v2) = 1 - ED(v1, v2)/n$$

- **Completeness:** rappresenta la misura in cui i dati sono di sufficiente ampiezza, profondità e portata ai fini dell'attività in corso. La completezza nel modello relazionale può essere caratterizzata rispetto a presenza, assenza e significato di valori nulli, e validità di una delle due ipotesi dette assunzione a mondo aperto e assunzione a mondo chiuso. Data:



Attributes → Tuples	Attribute 1	Attribute 2	Attribute n
Tuple 1				
Tuple 2				
...				
Tuple m				

Fig. 1: *tabella T che rappresenta un universo U di k oggetti*

Le metriche sono le seguenti:

Type of completeness	Metrics
Object completeness (T,U)	m/k
Tuple completeness (ti)	# of null values in ti /n
Attribute completeness (Aj)	# of null values in the column of Aj /m
Table completeness (T)	# of null values in T /m * n

Fig. 2: *metriche principali completezza*

Le misure della completezza sono

- **Currency:** è la dimensione che rappresenta la rapidità con cui i dati sono aggiornati rispetto a quando avviene il fenomeno. Per esempio il voto di un esame nella base dati studenti corsi-esami. Una prima misura della currency è il ritardo temporale tra il tempo t_1 dell'evento del mondo reale che ha provocato la variazione del dato, e l'istante t_2 della sua registrazione nel sistema informativo. Questa misura è costosa perché in genere l'evento non è noto. Un'altra metrica è la currency come differenza tra tempo di arrivo alla organizzazione e tempo in cui è effettuato l'aggiornamento. Misurabile se c'è un log degli arrivi degli update.
- **Tempestività:** misura quanto i dati sono aggiornati rispetto a un processo che li utilizza. La tempestività, al contrario della currency, è dipendente dal processo, ed è associata al momento temporale in cui deve essere disponibile per il processo che utilizza il dato.
- **Consistency:** assume due significati, uno legato alla consistenza dei dati con i vincoli d'integrità definiti sullo schema e un altro legato sulle diverse rappresentazioni di uno stesso oggetto della realtà presenti nella base di dati.

Il punto focale rimane sempre il dominio sul quale si vuole effettuare un processo di miglioramento dei dati. In base alla finalità delle informazioni e alle caratteristiche dei loro consumatori l'attenzione si rivolge ad un sottoinsieme di tale metriche.

In tempi attuali, è emerso un altro tipo problema: i big data. Analisi e ricerca complete di

standard di qualità e metodi di valutazione della qualità per questo tipo di informazioni attualmente è assente o non completa. Questo topic pone una serie di nuove sfide, dettate dalle caratteristiche intrinseche dei big data, riassumibili in quelle che sono chiamate "le 5 V":

- **Volume:** ingente massa di informazioni, in crescita vertiginosa, che non è possibile raccogliere con tecnologie tradizionali;
- **Velocity:** i dati nascono e vengono acquisiti sempre più rapidamente, con necessità di analisi in tempo reale;
- **Variety:** differenti tipologie di dati disponibili, provenienti da un numero crescente di fonti eterogenee;
- **Veracity:** i dati devono essere affidabili, raccontare il vero;
- **Value:** abilità di trasformare una grande mole di dati in business.

1.1.1 Metodologia Data Quality

Il professor Batini definisce la metodologia di qualità dei dati come un insieme di linee guida e tecniche che, a partire dalle informazioni di input che descrivono un determinato contesto applicativo, ne deriva un processo razionale per valutare e migliorare la qualità dei dati [1]. Ci sono tre fasi principali per tale attività:

- **ricostruzione dello stato**, al fine di ottenere due informazioni contestuali, facoltative se sono già disponibili per l'uso;
- **valutazione e misurazione**, misurazione della qualità lungo dimensioni della qualità pertinenti o valutazione, quando tali misurazioni vengono confrontate con i valori di riferimento;
- **miglioramento**, attività che mirano a raggiungere nuovi obiettivi di qualità dei dati.

1.1.2 Miglioramento

Il miglioramento della qualità dei dati può essere effettuato attraverso strategie data-driven o su process-driven.

Nel primo caso, l'obiettivo è quello di migliorare la qualità dei dati direttamente modificando il valore del dato attraverso la comparazione con altri dati considerati di buona qualità. Le strategie più diffuse del data-driven sono quelle di:

- **acquisizione del nuovo dato**, che migliora il dato acquisendo dato di alta qualità per rimpiazzare il valore che causa problemi di qualità;
- **record linkage**, che compara i dataset con valori sporchi con una fonte di dati di certificata, identificando tuple nei due dataset che potrebbero rappresentare lo stesso oggetto nel mondo reale;
- **affidabilità della fonte**, seleziona le origini dei dati sulla base della qualità delle fonti.

Nel process-driven l'obiettivo è quello di migliorare la qualità ridisegnando i processi che creano o modificano i dati, cioè andata a risolvere il problema alla radice. Le strategie più diffuse del process-driven sono quelle di:

- **processo di controllo**, che inserisce verifiche e procedure di controllo nel processo di produzione dei dati quando i dati sono creati, aggiornati o nuovi insieme dei dati sono accessibili dal processo.
- **ridisegnamento del processo**, per rimuovere le causa della qualità scarsa. Molto più efficace ma anche più costoso.

Nello studio effettuato non sono state adottate strategie di process-driven perché non c'è un'applicazione che esegue il CRUD sui dati, ma sono forniti un insieme di file di testo che rappresentano delle istantanee di diversi datasets che indicano lo stesso concetto.

1.1.3 Standardizzazione

Questo processo, chiamato anche *normalizzazione*, sostituisce o integra i valori dei dati con i valori corrispondenti conformi allo standard (ad esempio le abbreviazioni vengono

sostituite con i nomi completi corrispondenti).

1.1.4 Comparazione stringhe

Gli errori tipografici rendono impossibile confrontare esattamente tra di loro le stringhe. Per poter fare ciò, quindi, serve una funzione che cerca di trovare un punto di accordo tra i dati. Ci sono stati diversi tentativi di fornire questa funzione:

- Jaro [3] nel 1976 ha proposto un comparatore di stringhe che tiene conto di inserimenti, eliminazioni e trasposizioni necessarie per abbinare le due stringhe;
- Winkler [4] nel 1990 ha proposto una variante della distanza Jaro (Jaro-Winkler);
- la distanza q-gram conta il numero di q caratteri consecutivi che concordano tra due corde;
- la distanza di edit classica, che conta il numero di operazioni (inserimenti, eliminazioni, modifiche) necessarie per abbinare le due stringhe

1.2 Record Linkage

Il record linkage (conosciuto anche come data matching) è l'operazione che consiste nel trovare "records" che si riferiscono alla stessa entità, in dataset presi da differenti risorse (come ad esempio file, libri, siti e database).

Questa operazione diventa necessaria quando vogliamo unire dei dataset differenti basati su dati simili che potrebbero avere o non avere lo stesso identificativo.

L'idea moderna di record linkage nasce alla fine degli anni cinquanta e viene formalizzata qualche tempo dopo da Ivan Fellegi e Alan Sunter [2] che, attraverso il loro lavoro, hanno dimostrato che le regole di decisione probabilistiche sono ottimali quando i dati che vengono confrontati sono condizionatamente indipendenti.

A partire dalla fine degli anni novanta, differenti tecniche di machine learning sono state sviluppate per poter capire, con condizioni favorevoli, la probabilità condizionata richiesta dalla teoria Fellegi-Sunter.

Il record linkage può essere interamente eseguito senza l'aiuto di un computer, ma il motivo principale per cui esso viene utilizzato è perchè si vuole ridurre o eliminare le modifiche "fatte a mano" e per rendere più facile l'ottenimento del risultato.

Il record linkage si divide generalmente nei seguenti step:

1. vengono dati in input dei dataset;
2. viene definito uno spazio iniziale di ricerca;
3. si cerca di ridurre lo spazio di ricerca tramite un processo di "blocking" ;
4. viene definito lo spazio ridotto di ricerca;
5. vengono comparati i dati e viene presa una decisione;
6. vengono definite le regole di matching, possibile matching o non-match;
7. secondo le regole definite viene generato il dataset di output.

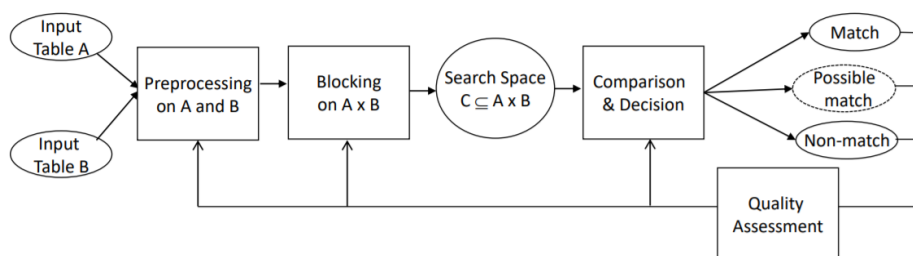


Fig. 3: *Step principali del record linkage*

1.2.1 Metodologie record linkage

Data preprocessing Il record linkage è molto sensibile alla qualità dei dati che devono essere collegati, quindi idealmente prima di svolgere questa operazione ogni dataset deve essere controllato affinché la qualità sia delle migliori. Avvengono delle operazioni di standardizzazione, che consistono nel trasformare i dati o procedure più complesse, come ad esempio la tokenizzazione.

Entity resolution L'entity resolution è un processo di operazioni intelligenti che permettono alle organizzazioni di connettere i dati più disparati attraverso la possibilità di capire i matches tra le entità e le relazioni non ovvie fra i diversi dati.

Essa analizza tutte le informazioni collegate ad una entità prese da diverse sorgenti e cerca, attraverso un calcolo probabilistico, di determinare quali entità sono collegate e quali collegamenti (non ovvi) esistono fra loro.

Deterministic record linkage La metodologia più semplice di record linkage è chia-

mata "deterministica".

Essa genera collegamenti basati sul numero di singoli identificatori che hanno una corrispondenza fra i dati dei dataset.

Due record si dicono *collegati* con una procedura di record linkage deterministico se tutti o alcuni degli identificatori sono identici.

Questo metodo è una buona opzione se si stanno utilizzando dei dataset con delle entità che sono identificate da un id comune.

Probabilistic record linkage Il record linkage probabilistico, chiamato anche *fuzzy matching*, utilizza un approccio differente per poter collegare i dati. Viene tenuto conto di una gamma più ampia di potenziali identificatori, calcolando i pesi per ciascun identificatore in base alla sua capacità stimata di identificare correttamente una corrispondenza o una non corrispondenza. Questi pesi sono dunque usati per calcolare la probabilità che due dati registrati si riferiscano alla stessa entità. Le coppie di record con probabilità al di sopra di una determinata soglia sono considerate corrispondenze, viceversa le altre sono considerate non corrispondenze. Le coppie che rientrano tra queste due soglie sono considerate "possibili corrispondenze" e possono essere trattate di conseguenza (ad esempio, revisioni umane, collegate o non collegate, a seconda dei requisiti). Mentre il collegamento deterministico dei record richiede una serie di regole potenzialmente complesse da programmare in anticipo, i metodi probabilistici di collegamento dei record possono essere "addestrati" per funzionare bene con un intervento molto meno umano.

Machine learning Negli ultimi anni, sono state utilizzate varie tecniche di apprendimento automatico per collegamento automatico. È stato riconosciuto che l'algoritmo classico per il collegamento probabilistico dei record sopra descritto è equivalente all'algoritmo Naive Bayes nel campo dell'apprendimento automatico, e utilizza la stessa assunzione dell'indipendenza delle sue caratteristiche (un presupposto che in genere non è vero).

È possibile ottenere una maggiore precisione utilizzando varie altre tecniche di apprendimento automatico, incluso un perceptrone a strato singolo. Insieme alle tecnologie distribuite, l'accuratezza e la scala per il collegamento dei record possono

essere ulteriormente migliorate.

1.3 Tools

Di seguito riportiamo i tools e le principali librerie utilizzate in questa analisi per processare i dati.

1.3.1 Python Record Linkage Toolkit

Python Record Linkage Toolkit è una libreria che permette di effettuare record linkage sia in una sola fonte di dati che in molteplici. Il toolkit fornisce la maggior parte degli strumenti necessari per il collegamento e la deduplicazione dei record. Il package contiene metodi di indexing, come blocking e sorted neighborhood indexing, funzioni per il confronto con diverse misure di similarità possibili e diversi algoritmi di classificazione, sia supervisionati che non. Uno degli obiettivi principali è proprio creare un framework estendibile per il collegamento dei record, per fare questo il tool è in grado di:

-
- pulire e standardizzare i dati
 - creare coppie di record con metodi di indicizzazione intelligente
 - confrontare i record con un gran numero di misure e di comparazione e somiglianza per diversi tipi di variabili
 - utilizzare diversi algoritmi di classificazione
 - valutare, attraverso strumenti comuni, i collegamenti dei record

Per raggiungere l'obiettivo del nostro progetto è stato fondamentale poichè ci permette di svolgere tutte le operazioni del record linkage senza dover ricorrere all'utilizzo di ulteriori tool.

1.3.2 Pandas

Pandas è una libreria software scritta in linguaggio python/C che permette la manipolazione e l'analisi dei dati. L'obiettivo di questa libreria è diventare il blocco di base di alto livello fondamentale per eseguire analisi dei dati reali. Inoltre, ha l'obiettivo più ampio di diventare lo strumento di analisi e manipolazione dei dati open source più potente e flessibile disponibile per qualsiasi linguaggio.

Questa libreria è in grado di pulire, esplorare e elaborare i dati. Gli oggetti Pandas (es. DataFrame, Series, Index,...) sono lo standard utilizzato da ogni altra libreria di processing dati utilizzata in questo studio.

1.3.3 Fuzzy Matcher

Fuzzy Matcher è un pacchetto python che permette all'utente di collegare due dataframe pandas basati su uno o più campi in comune. Fuzzy Matcher usa la ricerca del testo completa sqlite3 per trovare dei match potenziali. Nonostante sia stato provato, non è stato poi incluso nel progetto poiché questa libreria non permette le stesse possibilità di sperimentazioni, ottimizzazione dei parametri e utilizzo di comparatori possibili con Record Linkage Toolkit.

2 Sperimentazione

2.1 Dataset

Il dataset su cui sono stati effettuati gli esperimenti è un elenco di ristoranti di Manhattan, estratti settimanalmente da Gennaio a Marzo 2009 da 12 siti web. L'unico attributo comune in tutti i dataset è il nome del ristorante, informazioni aggiuntive (come l'indirizzo o il quartiere) non sono presenti in modo uniforme. Insieme ai dataset era fornito un golden standard, ossia un elenco di 467 ristoranti rimossi da alcuni siti e il loro stato ("Y" = aperto, "N" = chiuso). Nell'ottica del progetto queste informazioni non sono risultate utili e per questo motivo non sono state sfruttate per analisi aggiuntive.

2.2 Analisi dei dati

I dati sono messi a disposizione in sette file di testo, uno per ogni settimana considerata, ciascuno contenente informazioni appartenenti a tutti i siti web. In totale sono presenti 215555 record, suddivisi come illustrato nella seguente tabella:

file	records
restaurants_2009_1_22.txt	30401
restaurants_2009_1_29.txt	30775
restaurants_2009_2_05.txt	30805
restaurants_2009_2_12.txt	30863
restaurants_2009_2_19.txt	30876
restaurants_2009_2_26.txt	30898
restaurants_2009_3_12.txt	30937

Table 1: Record presenti nei file txt forniti

In particolare, per ogni ristorante è presente il seguente numero di record:

restaurant	records
ActiveDiner	6184
DiningGuide	814
FoodBuzz	2079
MenuPages	13143
NewYork	1774
NYPmag	5124
NYTimes	3095
OpenTable	1539
SavoryCities	4536
TasteSpace	3635
TimeOut	14007
VillageVoice	2684

Table 2: Record per ristorante

Sono presenti le seguenti informazioni:

- **ActiveDiner**: nome ristorante, indirizzo, paese
- **DiningGuide**: nome ristorante, indirizzo
- **FoodBuzz**: nome ristorante, indirizzo, paese, codice paese, quartiere, tipo, costo
- **MenuPages**: nome ristorante, indirizzo1, indirizzo2
- **NewYork**: nome ristorante
- **NYPmag**: nome ristorante, quartiere
- **NYTimes**: nome ristorante, quartiere
- **OpenTable**: nome ristorante, quartiere
- **SavoryCities**: nome ristorante, indirizzo, quartiere
- **TasteSpace**: nome ristorante, indirizzo, paese
- **TimeOut**: nome ristorante, posizione, indirizzo1, indirizzo2, indirizzo3, quartiere, numero di telefono, tipo
- **VillageVoice**: nome ristorante, indirizzo, quartiere

Il dataset "NewYork" non è stato utilizzato poichè non porta informazione utile.

2.3 Data preprocessing

Una prima analisi mostra come le modalità di recupero dei dati da parte dell'autore abbiano generato dei dataset differenti per schema e per frammentazione verticale, anche sulle stesse fonti. Dunque, prima di applicare le tecniche di record linkage, i dati sono stati standardizzati, eseguendo operazioni di riallineamento dello schema, rimozione dei duplicati ed infine join dei dataset per fonte.

La fase di preprocessing, nella quale sono incluse la pulizia dei dati e la standardizzazione, sono importanti poiché potrebbero aumentare l'accuratezza del record linkage.

In dettaglio le fasi del preprocessing sono state:

1. **Separazione e raggruppamento dei dati per fonte.**

Ciò ha mostrato le differenze sullo schema, utilizzate nelle fasi successive per migliorare il raggruppamento dei dati.

2. **Separazione dei dati per fonte e data.**

Utilizzando regex ad hoc per ogni fonte i dati sono stati separati per fonte e data, con drop dei duplicati, standardizzazione dell'encoding del testo (da *windows-1252* a *UTF-8*) e formato csv.

3. **Merge dei dataset per fonte.**

I vari dataset risultanti dalla fase 2 sono stati riuniti per fonte con un join applicato su colonne definite a priori.

4. **Pulizia finale.**

I dataset dunque sono stati ripuliti utilizzando delle funzioni specifiche per ogni fonte, con il compito di riallineare lo schema, rimuovere duplicati e caratteri superflui dai valori (es spazi e tabulazioni in testa e in coda, caratteri speciali ecc.).

2.4 Google Maps

Nei vari dataset le informazioni riguardanti l'indirizzo non erano riportate in maniera uniforme (es. *Street* in un record e *St.* in un altro), quindi abbiamo deciso di standardizzare questo parametro ricorrendo alle API di Google Maps.

Come prima cosa abbiamo creato una API key per poter dialogare con le **API di Google**.

Dalla documentazione individuiamo *Text Search*, molto simile alla funzione di autocompletamento presente su Google Maps. Dunque ci permette di ricavare indirizzo completi partendo da informazioni parziali.

Pertanto è stato costruito uno script che esegue una GET al seguente indirizzo, passando come parametro la stringa *address* del record del csv considerato. L'output ottenuto è un json, come si può osservare nel seguente esempio presente nella documentazione ufficiale.

```
{
  "candidates" : [
    {
      "formatted_address" : "140 George St, The Rocks NSW 2000, Australia",
      "geometry" : {
        "location" : {
          "lat" : -33.8599358,
          "lng" : 151.2090295
        },
        "viewport" : {
          "northeast" : {
            "lat" : -33.85824767010727,
            "lng" : 151.2102470798928
          },
          "southwest" : {
            "lat" : -33.86094732989272,
            "lng" : 151.2075474201073
          }
        }
      },
      "name" : "Museum of Contemporary Art Australia",
      "opening_hours" : {
        "open_now" : false,
        "weekday_text" : []
      },
      "photos" : [
        {
          "height" : 2268,
          "html_attributions" : [
            "\u003ca href=\"https://maps.google.com/maps/contrib/113202928073475129"
```

Fig. 4: *Dati dopo il preprocessing di MenuPages*

Il campo da noi considerato è *formatted_address*. In caso di indirizzo scritto male o con contenuto informativo ambiguo venivano restituiti più *formatted_address*, pertanto si è deciso che se il numero di risultati ottenuto era maggiore di 5 allora non si è applicata tale trasformazione, poiché probabilmente non porterebbe reali benefici. Negli altri casi si è preso solo il primo risultato.

Nello specifico, dato che tutti i ristoranti sono a New York, abbiamo splittato la stringa

recuperando solo la prima parte (fino alla prima virgola), in quanto l'informazione sulla città è superflua.

Considerando l'esempio riportato nell'immagine sopra, da un indirizzo del tipo *140 George St, The Rocks NSW 2000, Australia* manteniamo solo la parte *140 George St*

In questo modo facendo i confronti il contenuto informativo era maggiore e non degradato da New York, NY, dati che sarebbero stati comuni a tutti.

2.4.1 Risultati

Nel caso di MenuPages, per esempio, inizialmente i dati erano presenti in un txt in questo formato ed erano uniti ai dati di altri siti web:

Fig. 5: Dati di partenza di MenuPages

A seguito del preprocessing invece otteniamo informazioni più chiare, aggiungendo l'informazione sull'indirizzo proveniente da Google come segue:

Home

Insert

Formulas

Revisioni

Visualizza

Guida

Cerca

Generatore

Formatizzatore Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Formattazione Formattori come condizionale

Colore collettore

Colore da input

Nota

Output

Fig. 6: *Dati dopo il preprocessing di MenuPages*

2.5 Flusso di lavoro per l'Object Identification

Dalla fase di preprocessing si ottengono in output 12 dataset, uno per ogni sito web presente nei file originali. Si deve affrontare dunque il problema di record linkage multiplo, approcciato in due modi differenti, al fine di cercare di comprendere quale metodo sia migliore in termini di semplicità d'implementazione o di risultati ottenuti. I due approcci differiscono principalmente per come è stato risolto il **problema della molteplicità del linkaggio**, tema poco trattato in letteratura e che manca di uno stato dell'arte ben definito.

In entrambi i metodi, però, si seguono i passi principali definiti nella libreria python utilizzata, che sono i seguenti:

- **Indexing**: permette di creare coppie di record, denominate *candidate links*. Nel nostro algoritmo abbiamo implementato tre tecniche di indexing:
 - *full*, che crea le coppie effettuando il prodotto cartesiano dei due dataset, motivo per il quale è molto lungo e sconsigliato.
 - *blocking*, che permette di creare le coppie basandosi su una o più variabili uguali.
 - *sortedneighbourhood*, da usare nel momento in presenza di dataset con un grande numero di errori di spelling nei valori.
- **Comparing** - viene usato per confrontare le coppie di record create nella fase di indexing, sfruttando diversi metodi di similarità. Nel nostro caso abbiamo effettuato un confronto fra stringhe testando alcuni dei vari metodi disponibili ('jaro', 'jarowinkler', 'levenshtein', 'lcs') e impostando un valore di soglia (tutti i confronti approssimativi di stringhe più alti o uguali a questa soglia valgono 1) oppure usato un confronto esatto nel momento in cui trattavamo l'attributo *addressGoogle*, che sappiamo essere consistente.
- **Classification** - dove le coppie vengono classificate in matches, non-matches e possible matches. In particolare abbiamo deciso di applicare due algoritmi di apprendimento non supervisionato, non essendo in possesso di training data. Gli algoritmi

adottati sono:

- *ECM Classifier* o *Expectation/Conditional Maximisation classifier*, un algoritmo probabilistico dove viene trovata iterativamente la massima probabilità (locale) o la massima stima a posteriori (MAP) dei parametri nei modelli statistici. L'iterazione EM si alterna tra l'esecuzione di un passaggio di aspettativa (E), che crea una funzione per l'aspettativa della verosimiglianza, e un passaggio di massimizzazione (M).
- *KMeans Classifier*, algoritmo che suddivide le coppie di record in match e non-match ed ogni vettore di confronto appartiene al cluster con la media più vicina. L'algoritmo è calibrato per due cluster: un cluster di corrispondenza e un cluster di non corrispondenza.
- **Evaluation** - permette di verificare la qualità del linkage in termini di accuratezza, recall e F-score. Purtroppo non ci è stato possibile eseguire questa fase in quanto sprovvisti di *veri positivi*.

2.5.1 Metodo 1: Raggruppamenti per attributi simili

In questo approccio ci si è ricondotti al problema di record linkage fra due dataset passando dunque dai 12 dataset iniziali a due finali.

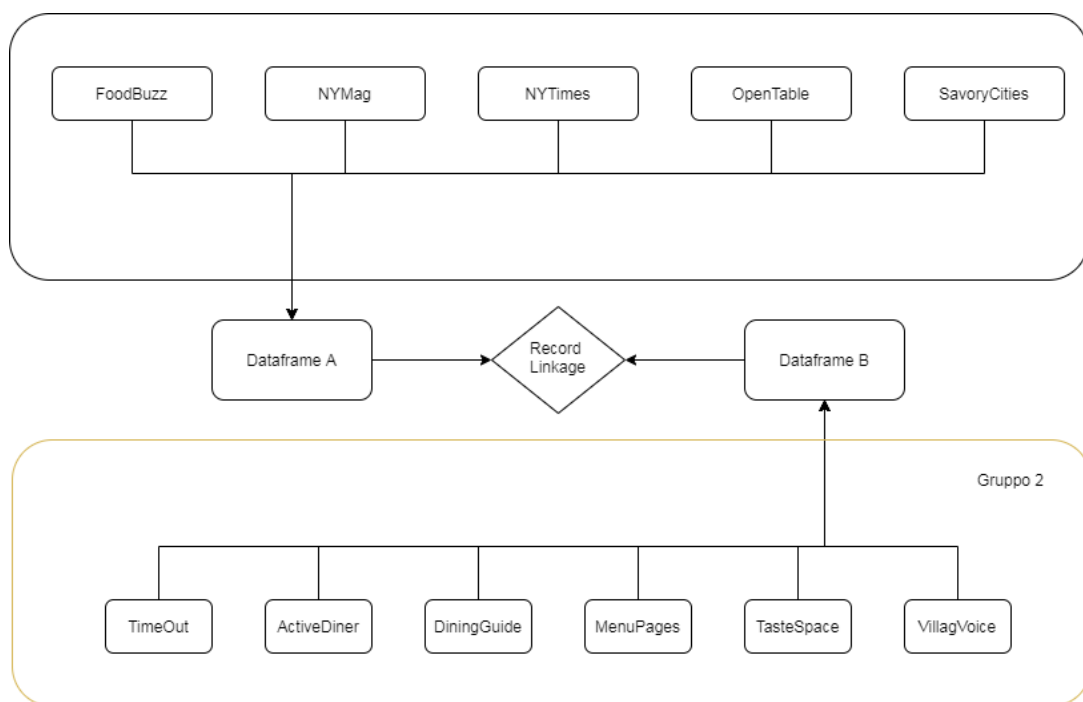


Fig. 7: schematizzazione dei raggruppamenti

Analizzando i dataset ottenuti si individuano delle caratteristiche comuni che ne permettono il raggruppamento in due macro gruppi. Nei dataset *FoodBuzz*, *NYMag*, *NYTimes*, *OpenTable*, *SavoryCities* si hanno esclusivamente il nome del ristorante e il quartiere (solo in *FoodBuzz* abbiamo in aggiunta l'indirizzo, che ci permetterà il linkage con l'altro dataset), nei dataset *TimeOut*, *ActiveDiner*, *DiningGuide*, *MenuPages*, *TasteSpace*, *VillageVoice*, invece, abbiamo nome del ristorante e indirizzo in tutti i file.

Per ogni insieme di dataset, dopo aver uniformato i nomi e l'ordine delle colonne presenti, abbiamo eseguito un append di tutti i dataframe generati importando i file csv. Successivamente sono stati rimossi i duplicati presenti basandoci solo sulle colonne comuni per tutti, ossia *restaurant* e *neighbourhood* nel primo caso e *restaurant* e *addressGoogle* nel secondo.

Successivamente viene applicata la **deduplicazione**, ossia una tecnica usata per eliminare copie duplicate di dati ripetuti. È possibile applicarla sempre attraverso la libreria *Record Linkage Toolkit* in quanto viene vista come un record linkage effettuato tra il database e

se stesso. Come tecnica di *indexing* usiamo *sortedneighbourhood* ed effettuiamo la fase di *comparing* sugli attributi comuni nei rispettivi gruppi di dataset.

In questa fase iniziale per classificare i match ci siamo basati esclusivamente sul punteggio di score ottenuto dalla fase di comparing, in modo da poter includere con sicurezza record che differivano solo di poche lettere (ad esempio i ristoranti "Bubba Gump Shrimp Co." e "Bubba Gump Shrimp Company"), scegliendo come valore di soglia dello score 1.75.

Una volta individuati i match sono stati unificati i dati in modo da creare un dataframe contenente le informazioni uniche utili per ogni match trovato. Per ottenere il dataset finale dal primo dataframe ottenuto come l'append dei csv importati sono state rimosse tutte le righe delle coppie presenti nei match e sono state aggiunti i record unificati contenenti le informazioni complete.

Da questa fase sono risultati due dataset, rispettivamente di 6468 e di 15093 record, invece dei 58614 record totali derivati dall'unione di tutti i 12 file iniziali.

True Matches

I dati di addestramento non sono generalmente disponibili nelle applicazioni di record linkage perché sono set di dati specifici del campione, infatti nel nostro caso non sono forniti dei true matches.

Python Record Linkage Toolkit fornisce un'interfaccia utente per la classificazione manuale delle coppie di record. Il software RecordLinkage Annotator utilizza un file di annotazione strutturato, che si può creare attraverso la funzione `recordlinkage.write_annotation_file()` sia per il linkaggio che per la deduplicazione.

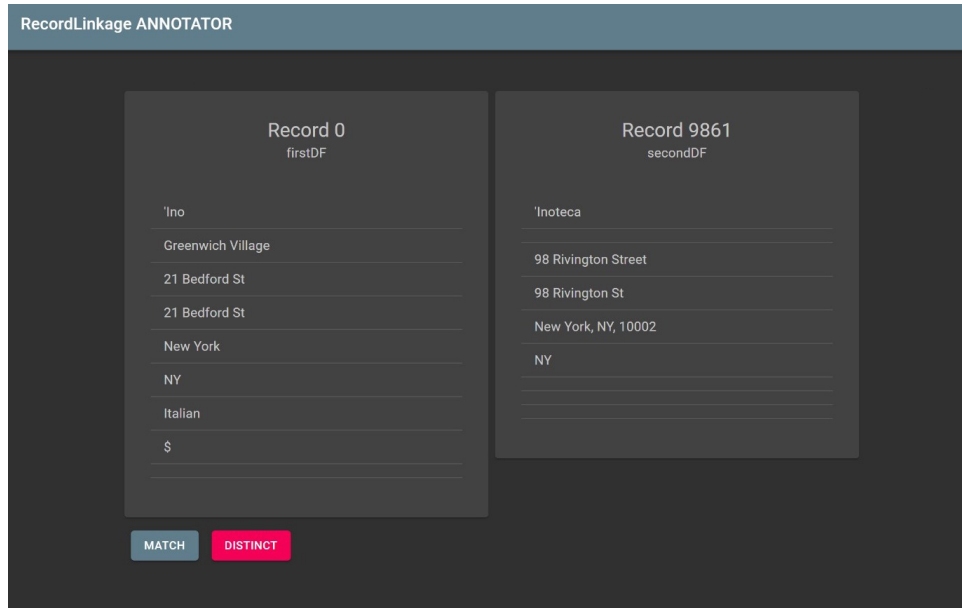


Fig. 8: *Interfaccia web per la classificazione manuale*

Nel nostro caso abbiamo confrontato le prime 100 coppie derivate dalla fase di *comparison*, ottenendo X match. Queste informazioni saranno fondamentali per poter valutare l'efficacia degli algoritmi di classificazione usati.

Risultati:

I dataset di partenza rappresentano un caso reale, siamo stati posti di fronte a dati inconsistenti, codificati in modi diversi, con attributi differenti e senza informazioni sui dati "veri" da poter usare come confronto per valutare l'efficacia del codice sviluppato, per questo possiamo analizzare in modo principalmente empirico i risultati ottenuti.

Grazie all'interfaccia fornita dalla libreria *Record Linkage Toolkit* abbiamo generato un insieme di matches verificati manualmente su un insieme di 100 coppie, grazie ai quali possiamo valutare l'efficacia degli algoritmi di classificazione utilizzati su un sottoinsieme di 100 coppie individuate.

Con la matrice di confusione vogliamo calcolare l'accuratezza statistica della predizione dei dati che esegue la nostra macchina. Questo calcolo sarà effettuato sui modelli di

machine learning che abbiamo scelto.

L'algoritmo ECM ha ottenuto dei piccoli punti percentuali maggiori di accuratezza rispetto al KMeans, vediamo di seguito la sua matrice di confusione ottenuta dalla presisione sui dati di test:

	True Positive	True Negative
Predicted Positive	22	10
Predicted Negative	1	67

Fig. 9: *matrice di confusione con ECM*

L'accuratezza dell'algoritmo ECM è 89%. Dall'accuratezza sufficientemente alta notiamo che i risultati sono buoni, infatti si riesce a ottenere 67 veri negativi, classificando solo 1 come falso positivo. D'altra parte i risultati sono meno soddisfacenti se guardiamo le predizioni positive infatti abbiamo 10 falsi positivi e 22 veri positivi, quindi circa un terzo sono stati classificati non correttamente.

Measure	Value	Derivations
Sensitivity	0.9565	$TPR = TP / (TP + FN)$
Specificity	0.8701	$SPC = TN / (FP + TN)$
Precision	0.6875	$PPV = TP / (TP + FP)$
Negative Predictive Value	0.9853	$NPV = TN / (TN + FN)$
False Positive Rate	0.1299	$FPR = FP / (FP + TN)$
False Discovery Rate	0.3125	$FDR = FP / (FP + TP)$
False Negative Rate	0.0435	$FNR = FN / (FN + TP)$
Accuracy	0.8900	$ACC = (TP + TN) / (P + N)$
F1 Score	0.8000	$F1 = 2TP / (2TP + FP + FN)$

Fig. 10: *Misurazioni con ECM*

	True Positive	True Negative
Predicted Positive	15	17
Predicted Negative	0	68

Fig. 11: *matrice di confusione con KMeans*

In questo caso il livello di accuratezza è inferiore, infatti è del 83%. Vengono predetti correttamente 15 casi su 32 per le predizioni positive e 68 casi su 68 per le predizioni negative. Seppure il risultato per le predizioni negative è perfetto, il classificatore KMean perde in accuracy rispetto a ECM perché sbaglia a predire i casi positivi.

Measure	Value	Derivations
Sensitivity	1.0000	$TPR = TP / (TP + FN)$
Specificity	0.8000	$SPC = TN / (FP + TN)$
Precision	0.4688	$PPV = TP / (TP + FP)$
Negative Predictive Value	1.0000	$NPV = TN / (TN + FN)$
False Positive Rate	0.2000	$FPR = FP / (FP + TN)$
False Discovery Rate	0.5313	$FDR = FP / (FP + TP)$
False Negative Rate	0.0000	$FNR = FN / (FN + TP)$
Accuracy	0.8300	$ACC = (TP + TN) / (P + N)$
F1 Score	0.6383	$F1 = 2TP / (2TP + FP + FN)$

Fig. 12: *Misurazione con KMeans*

Dopo aver individuato i match il dataset finale è dato dai record linkati (un record per coppia individuata) e i record per i quali non è stato individuato un match. Con il primo metodo abbiamo ottenuto X record per il dataset generato dall'algoritmo ECM e X per il dataset generato dall'algoritmo Kmeans.

2.5.2 Metodo 2: Record linkage multiplo

Appurato il funzionamento e l'efficacia degli algoritmi ECM e K-Means è stato studiato un'approccio più dinamico e flessibile, al fine di evitare il processo manuale di categorizzazione della fase 1 e per parametrizzare, dunque rendere facile da impostare, l'ottimizzazione di ogni comparatore.

Nella seguente sezione dunque verrà presentato il funzionamento di tale approccio, correlato di un esempio di applicazione sui dataset analizzati.

Supponiamo di voler linkare N database, ottenendo come output un singolo dataset contenente quanta più informazione possibile dai database originali.

Il processo di linkage multiplo esegue i seguenti passaggi:

1. Scelta di un dataset i tra gli N datasets in input, da cui partirà il processo. La scelta può avvenire in modo randomico, oppure scegliendo il dataset che massimizza una determinata misura di qualità dei dati (es. completezza), per ottimizzare le fasi successive del processo.
2. Dividere i dataset in due insiemi:

$$Analizzati = \emptyset, Da_analizzare = N$$

3. Per ogni dataset $k \in Da_analizzare$, con $k \neq i$:
 - (a) Record linkage del dataset i con il dataset k ;
 - (b) Per ogni match trovato accorpate (eventuale) informazione aggiuntiva del record $\in k$ al record $\in i$;
 - (c) Eliminare da k ogni record di match individuato.
4. Aggiornamento dei due insiemi:

$$Analizzati = Analizzati \cup \{i\}, Da_analizzare = Da_analizzare - \{i\}$$

5. Se $Da_analizzare \neq \emptyset$ scegliere un nuovo dataset $i \in Da_analizzare$, secondo

criteri descritti nel passo 1, e ripetere passo 3.

Alla fine di ogni iterazione nel dataset i sono presenti record rappresentati entità univoche oppure record di cui sono state trovate copie in altri dataset, a cui viene aggiunta (eventuale) nuova informazione.

Una volta processati tutti i datasets non rimane che:

6. Concatenare ogni dataset $\in \textit{Analizzare}$,

producendo in output il dataset cercato.

Applicazione

Per semplificare la gestione del metodo, in particolare la fase di integrazione delle informazioni (3.b), si è scelto di eseguire un'ulteriore fase di preprocessing dei dati. Tale fase prevede:

- Per ogni dataset contenente informazioni circa indirizzo standardizzato via Google API: rimozione record senza indirizzo e rimozione colonne circa altri indirizzi.
- Deduplication con compare su nome ristorante, quartiere ed indirizzo standardizzato (se presenti).
- Standardizzazione dello schema di ogni dataset. Se una colonna non esiste viene aggiunta con valori vuoti. Ciò consentirà uno scambio di informazioni più rapido e semplice tra datasets.

Dunque viene eseguita la funzione **link_reduce**, che implementa record linkage multiplo. Questa prende in input i datasets, il nome del dataset di partenza ed i parametri utilizzati nelle comparazioni (es. thresholds per valutazione match effettivi, windows, algoritmo ML, ecc.).

Tutti i match trovati tramite l'algoritmo sono stati considerati match possibili: i match reali sono identificati calcolando la distanza Jaro-Winkler tra il valore del nome del ristorante dei match possibili.

Nella *Fig. 9* si vedono in azione le prime tre iterazioni, partendo dal dataset *VillageVoice* con algoritmo ML K-Means. Sono riportati il numero di match reali per ogni coppia di database linkati e l'evoluzione del numero totale di record per dataset.

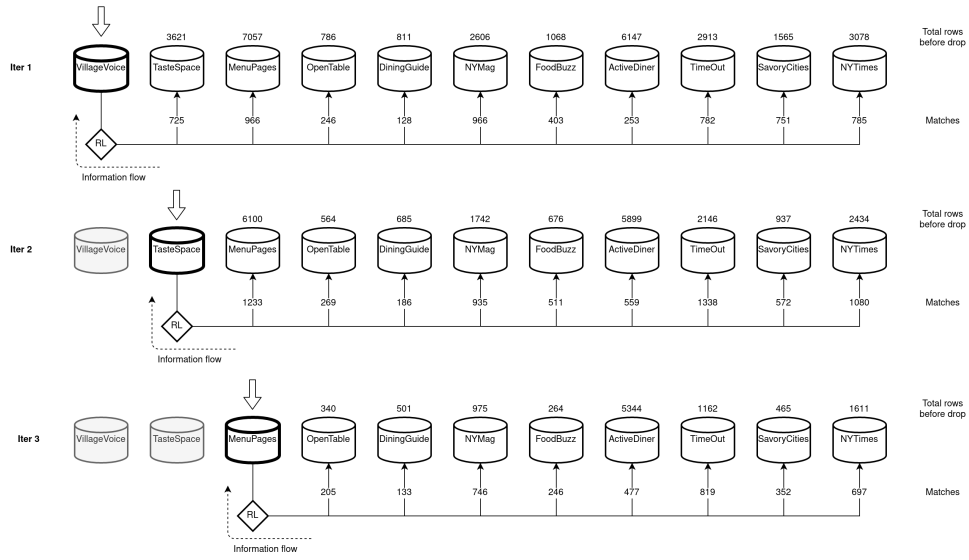


Fig. 13: Prime tre iterazioni del record linkage multiplo sui dataset dei ristoranti. Notare come ad ogni iterazioni il numero di record per dataset si riduce.

Una volta esauriti i dataset da analizzare questi vengono concatenati per produrre l'output finale, a cui viene ulteriormente controllata la presenza di un indirizzo.

Bibliografia

- [1] Carlo Batini et al. “Methodologies for Data Quality Assessment and Improvement”. In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541883. URL: <https://doi.org/10.1145/1541880.1541883>.
- [2] I. P. Fellegi and A. B. Sunter. “A Theory for Record Linkage”. In: *Journal of the American Statistical Association* 64 (1969), pp. 1183–1210.
- [3] Matthew A. Jaro. *UNIMATCH: A Record Linkage System: User’s Manual*. Tech. rep. U.S. Bureau of the Census, Washington, D.C., 1976.
- [4] William E. Winkler. “String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage”. In: *Proceedings of the Section on Survey Research*. Washington, DC, 1990, pp. 354–359.