1 Data quality

Un esempio di scarsa data quality è quella di avere lo stesso dato con valori diversi nello stesso posto, anche semplicemente una pagina web.
I dati sono una rappresentazione della realtà e questo porta al fatto che la realtà viene da noi modellata tramite alcuni dati specifici. Si definiscono quindi:

- **utilità** come precisione della rappresentazione interna del dato rispetto a il compito svolto (e quindi, ad esempio, un'immagine ritoccata potrebbe essere più utile per determinati scopi, si vede meglio anche attraverso la nebbia)
- **fedeltà** su come una determinata interpretazione aderisce al mondo reale

1.1 Concetti Fondamentali

Alcuni dei concetti fondamentali, che vengono appresi e che sono un punto di riferimento, quando si parla di qualità dei dati sono:

- **qualità**: caratteristiche di un artefatto che influiscono sulla sua capacità di soddisfare le esigenze e le aspettative dell'utente, dichiarate o implicite. Un dato è di qualità se è adatto all'uso che se ne vuole fare. Si ha il concetto di **fitness for use**. *La qualità dei dati è negli occhi di chi li usa e non nelle mani di chi li produce.*
- **dimensione**(o caratteristica): una caratteristica specifica che descrive la qualità delle informazioni associato al dato, solitamente non misurabile. Alcune volte è viene misurata, li dimensioni come le sottodimensioni attraverso le *metriche*.
- **sottodimensioni**: sotto-caratteristiche che spesso classificano una certa dimensione.

Le metriche sono una procedura che spesso mi consentono di misurare un certo valore. Rifacendoci alle slide ci riferiamo alla definizione presente nello standard ISO 9126-1 e secondo il framework ISM3. Quello che emerge è che le metriche racchiudono sia:

- una procedura (o metodo) di misurazione, ad esempio un algoritmo che prende l'elemento da misurare e lo associa a una misura (sia esso un valore ordinale o un intervallo)
- che una corretta unità di misura (o scala), cioè il dominio dei valori restituiti dalla procedura di misurazione

Si possono avere metriche diverse per la stessa dimensione.

1.2 Le dimensioni di qualità dei dati

La qualità dei dati è un concetto che può essere espresso attraverso molteplici dimensioni, ad esempio la accuratezza (magari anche solo per errori di typo nei dati), la comprensibilità, la completezza (avendo magari valori a NULL), l'inconsistenza e altro.

Un altro aspetto da tenere in considerazione è che si hanno metriche:

- **oggettive**, modi formali e precisi per misurare le metriche per una dimensione di qualità in termini di valori di un dominio, indipendentemente dalla percezione/valutazione umana
- **soggettive**, modi per misurare le metriche per una dimensione di qualità che dipendono dalla percezione (di solito valutazione esplicita e ordinale) delle persone coinvolte nel processo di misurazione, dipendente dalla percezione/valutazione umana

1.3 Data quality dimensions

Si hanno diverse dimensioni e metriche; vengono di seguiuto trattate (sottosezioni)

1.3.1 Accuratezza

La precisione di un valore v è definita come la vicinanza tra v e un valore v', questo viene considerato come la corretta rappresentazione del fenomeno del mondo reale che v intende rappresentare. Si può applicare alle tuple e alle relazioni.

Si hanno due sottodimensioni di questa qualità:

1. **accuratezza sintattica**, magari controllando se la stringa, inserita in un oggetto o una tupla, è presente in un insieme di valori di riferimento per il dominio trattato (magari un vocabolario, una lista di città etc...). Ci sono dati su cui non è applicabile. Generalmente si fa in base alle stringhe oppure in base ai singoli token di una stringa.
2. **accuratezza semantica**, molto difficile da verificare (magari tramite un'altra sorgente dati per cross validare). Per piccolissimi db si fa a mano ma è generalmente impensabile.

Conoscere lo schema aiuta a valutare la accuratezza dei valori e si hanno diverse metriche. Posso verificare se un dato è vicino a quello che mi aspetto in un certo dominio tramite una funzione di distanza che confronta il valore che ho e il dominio di riferimento. Dopo questo posso pensare a correggere eventualmente il dato. Si cerca il valore più vicino al dominio di riferimento.

Potrei anche fare una stima statistica per capire quale sia il valore corretto. Per il calcolo delle distanze su stringhe uso la solita **distanza di edit**, ci sono altri metodi ma useremo questa per ora. In fase di assessment considero solo valori con una massima distanza di edit stabilita e nella fase di improvement posso stimare la giusta correzione per ottenere la stringa magari corretta.

La distanza sintattica potrebbe però essere poco informativa in merito alla semantica (si pensi a Domenico spesso abbreviato in Mimmo). Possiamo introdurre la **distanza di edit normalizzata** come:

$$1 - \frac{edit(a, b)}{n}$$

con n numero massimo di simboli. Si ottiene un valore tra 0 e 1, con 1 che indica che i valori che sono identici, accuratezza pari a uno implica distanza nulla (misuro il complemento a 1 della distanza). Tali valori tra 0 e 1 sono comodi in quanto spesso le metriche danno risultati che sono in questo range e posso quindi effettuare una somma pesata tra le varie metriche.
Si hanno vari tipi di funzione di edit (ma limitate a stringhe corte):

- Hamming distance
- Levenshtein distance
- Jaro-Winkler

o di distanza di token (per stringhe composte da più stringhe, esempio "Sesto San Giovanni"):

- Jaccard index
- Sorensen-Dice
- N-gram

Si possono avere problemi nel trovare la reference table di un certo dominio.

1.3.2 Completezza

La completezza riguarda ogni tipo di rappresentazione del dato. Definiamo la **completezza** (di un insieme di dati) come la copertura con la quale il fenomeno osservato è rappresentato nell'insieme di dati. Avere ad esempio dei NULL impedisce la completezza.
Impatta valori alfanumerici e numerici e si può applicare alle tuple, agli attributi, agli oggetti e alle relazioni. Tipicamente si esegue una sottrazione in termini di $1-$ in modo da poter rappresentare la massima completezza a 1:

- la completezza di **tupla**, che identifica la presenza di valori NULL in una tupla, è basata sul numero di NULL di una certa riga rispetto al numero di attributi presenti
- la completezza di **attributo**, che presenta valori NULL all'interno delle colonne di una tabella. è basata sul numero di NULL di una certa colonna rappresentante un attributo

- la completezza di **tabella** è basata sul numero di NULL dell'intera tabella
- la completezza di **oggetto** è il numero di valori non nulli in tutti gli oggetti rappresentati nelle tuple

Nelle precedenti definizioni, abbiamo assunto una ipotesi di **mondo chiuso**, ovvero tutto ciò che è rappresentato nella base di dati è vero, tutto il resto è falso. Questa è la tipica ipotesi che si fa nei DBMS. Una ipotesi alternativa è quella di un **mondo aperto**, come nei DBMS NoSQL, dove di tutto ciò che non è rappresentato non si sa nulla. In questo caso introduciamo la **object completness**, che tiene conto del fatto che gli oggetti rappresentabili sono più delle tuple della tabella, e di tale cardinalità serve una stimma indiretta. Quindi nel mondo non relazionale è difficile stimare la completezza, non avendo certezza del numero di oggetti.

1.3.3 Currency

Una delle **proprietà temporali**, di livello di aggiornamento, ovvero di **currency**.
Difficile parlare di proprietà temporali perché dare una definizione di tempo risulta difficile,e in secondo luogo bisogna avere ben presente la semantica della dimensione di qualità che dobbiamo considerare.
Si ha che la **currency** misura con quale rapidità i dati sono aggiornati rispetto al corrispondente fenomeno del mondo reale. 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 ma questa misura è costosa in quando generalmente l'evento non è ben noto.

Un'altra metrica di currency è vederla come come differenza tra tempo di arrivo alla organizzazione e tempo in cui è effettuato l'aggiornamento (cosa misurabile in presenza di log degli arrivi e degli update). Un'altra metrica è la differenza rispetto al metadato ultimo aggiornamento effettuato che, per valori con periodicità di aggiornamento nota, currency calcolabile in maniera approssimata ma poco costosa (ma non si hanno informazioni in merito alle modifiche del dato).

La **tempestività** misura quanto i dati sono aggiornati rispetto a un particolare processo (o ai processi) che li utilizza. Questa tecnica, al contrario della currency, è dipendente dal processo, ed è associabile al momento temporale in cui deve essere disponibile per il processo che utilizza il dato. Posso avere dati obsoleti per il processo di chi li usa ma con alta currency.

1.3.4 Consistenza

Si definisce la **consistenza** in due modi:

1. come consistenza dei dati con i vincoli di integrità o dipendenze funzionali definiti sullo schema (ad esempio i vincoli di integrità nel modello relazionale).
   Si hanno anche dei vincoli di consistenza, detti business rule, che possono riguardare uno o più attributi, relazioni o altro. Possono essere espressi in termini di probabilità. Si usano anche vincoli di integrità in logica e si hanno dei data edit nelle indagini statistiche
2. come consistenza delle diverse rappresentazioni di uno stesso oggetto della realtà presenti nella base di dati

1.3.5 Accessibilità

Invece con **accessibilità** si esprime la capacità di un utente di accedere ai dati a partire dalla propria cultura, stato fisico e psichico e dalla tecnologie disponibilità. Un altro aspetto importante che va sempre considerato è il **tradeoff tra varie qualità**, ad esempio consistenza e completezza nel modello relazionale possono essere non conciliabili quando si voglia rispettare l'integrità referenziale. Nel dominio statistico si ha tradeoff tra tempestività e completezza/accuratezza, che vengono però privilegiate (bisogna quindi studiare quanto è sporco un certo dato). Nel web si preferisce la tempestività rispetto ad accuratezza/completezza.

2 Quality improvement

Avendo quindi già introdotto la fase di **quality assesment** passiamo a quella di **quality improvement**, ovvero al miglioramento dei dati stessi.
L'obiettivo della fase di miglioramento, una volta misurata la qualità dei dati mi accorgo che i dati sono di bassa qualità rispetto alle esigenza, si deve cercare di migliorare i dati. La qualità dei dati è un problema di tipo multidimensionale, potendo decidere di migliorare anche solo alcuni aspetti relativi alla qualità dei dati (come la completezza, la consistenza, l'aggiornamento temporale). Sappiamo inoltre che ci sono dei tradeoff

quindi il miglioramento di alcuni aspetti va a discapito della qualità di altri. Si hanno in generale due strategie in fase di miglioramento:

1. **data-driven**, migliorando il dato stesso in quanto tale. Si punta a migliorare la qualità dei dati modificando direttamente il valore dei dati attraverso il confronto con altri dati ritenuti di buona qualità. Ad esempio, i valori dei dati obsoleti vengono aggiornati aggiornando un database caratterizzato da una currency, quindi da una misura eseguita attraverso una metrica temporale, più alta. Si migliora il dataset stesso.
   Si hanno varie procedure:

   (a) **acquisizione di nuovi dati**, migliora i dati attraverso l'acquisizione di dati di qualità alta che andranno a sostituire i valori che sollevano problemi di qualità. Questa strategia può essere utilizzata per tutte le dimensioni esaminate nella fase di valutazione (assessment)

   (b) **record linkage**, detta anche **identificazione degli oggetti**, che confronta i dataset, che contengono valori sporchi, con una fonte certificata o di qualità superiore, identificando le tuple/record nei due dataset che potrebbero fare riferimento allo stesso oggetto del mondo reale. In seguito si va eseguire una *pulizia* dei dati sporchi in favore dei dati con qualità più alta.

   (c) **affidabilità della fonte**, dove si selezionano le fonti di dati sulla base della qualità dei loro dati

2. **process-driven**, migliorando il processo di acquisizione dei dati (datti errati possono portare ad errori sistematici), ottenendo che il dataset viene alimentato con dati corretti. Si punta quindi a migliorare la qualità ridisegnando i processi che creano o modificano i dati. Ad esempio, un processo può essere riprogettato includendo un'attività che controlla il formato dei dati prima della loro archiviazione. Tale strategia fa riferimento all'ambito detto **Business Process Reengineering (BPR)**, ovvero avendo la possibilità di riprogettare i processi, magari scoprendo che una sorgente era di qualità troppo bassa.
   Si hanno due tecniche:

   (a) **process control**, dove si aggiungono elementi e/o procedure di controllo nel processo di produzione dei dati quando: vengono creati nuovi dati, vengono aggiornati i set di dati o il processo accede a nuovi set di dati. Verificando errori e la qualità dei dati stessi.
   In questo modo, viene applicata una strategia reattiva, che quindi agisce tempestivamente, agli eventi di modifica dei dati, evitando così la degradazione dei dati e la propagazione degli errori.

   (b) **process redesign**, dove si ridisegnano i processi per rimuovere le cause della scarsa qualità e introduce nuove attività che producono dati di qualità più alta. Se la riprogettazione del processo è radicale, questa tecnica viene definita *business process reengineering*.

   Queste tecniche sono puntuali e non sono facilmente integrabili in un sistema di raccolta continua dei dati.

Nel lungo termine le tecniche process-driven sono più efficaci, eliminando il problema alla radice, ma sono estremamente costose nel breve termine. Le tecniche data-driven sono più economiche ma costose nel lungo e quindi sono adatte per un'applicazione one-shot, solitamente sono consigliate per i dati statici.

Si hanno miglioramenti specifici per la dimensione:

- **accuratezza** tramite confronto dei valori con un dominio di riferimento. Questo è il tipo di tecnica che abbiamo considerato per l'accuratezza sintattica nella fase di quality assesment.
- **completezza** tramite completamento di dati incompleti con tecniche specifiche che sfruttano la conoscenza sui dati.
- **consistenza** tramite l'identificazione dei dati corretti sfruttando vincoli di integrità, dipendenze funzionali o dati derivati, in quale modo posso poi andare a correggere i miei dati tramite sorgenti esterni o acquisendo nuovamente i dati.

Sulle slide è segnata come "d", ma ti giuro che non solo non c'è la "c", ma non ci sono nemmeno "a" e "b". Apparso a caso, magari se capisci dove collocarlo dimmi rip. Inoltre poi abbimao la deduplica che è un punto 3, ma esiste solo il punto 1, che questo error cose sia il punto 2 e non il d?????

Come tecnica si usa anche la **error localization and correction** che identifica ed elimina gli errori di qualità dei dati rilevando i valori che non soddisfano un dato insieme di regole, dette **edits**. Queste tecniche sono principalmente studiate nel dominio statistico. Rispetto ai dati elementari, i dati statistici aggregati (Da wiki: Le aggregazioni di dati sono dati di alto livello composti da una moltitudine, o da una loro combinazione, di dati individuali.) (come media, somma, massimo). Questi dati sono meno sensibili alla localizzazione probabilistica e alla correzione dei valori probabilmente errate. Sono state proposte tecniche per la localizzazione e

correzione degli errori per incongruenze, dati incompleti e valori anomali, ovvero valori significativamente diversi da tutti gli altri valori in un set di dati per dati elementari e dati statistici.

Un altra tecnica di quality improvement è la **deduplica** che corrisponde al raggruppamento di record di dataset che si riferiscono alla stessa entità nel mondo reale (raggruppo i doppioni). Praticamente la deduplica è un'operazione di record linkage fatta su un'unica tabella. La tabella risultate è strutturata in modo tale da avere le tuple suddivise in 3 gruppi:

1. coppie/gruppi di tuple che matchano, corrispondenti allo stesso oggetto reale
2. coppie di tuple che non matchano che corrispondono a entità distinte nel mondo reale
3. possibili coppie/gruppi di tuple, per le quali non è stato possibile giungere una conclusione definitiva, e sono necessarie ulteriori indagini (e costi) per assegnarle al gruppo match o mismatch

Un'altra attività classica, propedeutica ad altre attività (è una sorta di fase di preprocessamento), è la fase di **normalizzazione**, trasformando le stringhe, scritte nello standard del dominio di riferimento, effettuando per ogni dimensione, il riconoscendo della stessa e il relativo miglioramento. Si ha quindi il miglioramento di singoli valori o di intere tuple, migliorando accuratezza sintattica, completezza, currency e consistenza, in base a certe metriche.

Ogni valore che non matcha con la tabella di riferimento (spesso disponibili online a seconda del dominio ma non sempre) deve essere corretto. Il miglioramento dell'accuratezza sintattica viene effettuato sostituendo il valore errato con quello a distanza inferiore nella tabella di riferimento, dove la distanza può essere valutata dopo una precedente verifica dei valori e delle loro caratteristiche (si nota come le attività data-driven sono costose dal punto di vista temporale).

Mentre per migliorare l'accuratezza possiamo adottare procedure standard di confronto con le tabelle di riferimento, per la completezza è molto più complesso. Non avendo un elemento da cui partire, avendo a disposizione solo un valore NULL, non sempre si riesce a ricostruire il valore corretto per sostituirlo al NULL. Spesso comunque si ha un contesto per aiutare a rimuovere i NULL. La procedura più intuitiva è eseguire una nuova acquisizione di dati incompleti riempiendo di volta in volta più valori NULL possibili. Poiché di solito si tratta di un'attività molto costosa, possiamo adottare diverse euristiche che di solito dipendono dal contesto.

Per la completezza si usano anche i **dati derivati**, definiti come dati per i quali esiste una formula matematica per la quale questi dati possono essere ottenuti a partire da un altro set di dati già conosciuto.
Bisogna migliorare la consistenza vedendo se la dipendenza funzionale sui dati derivati e altri vincoli di integrità intra-relazionale possono essere sfruttati, oltre che per la completezza completezza, anche per la consistenza.

Dopo tutte queste operazioni si ottiene un db già abbastanza pulito ma si hanno, solitamente, ancora problemi con spazio di miglioramento. Si userà quindi il **record linkage**.

3 Record linkage

Il **record linkage** viene usato anche in altri contesti oltre il data quality con anche altri nomi (come deduplicazione, se si tratta di un solo dataset).
Prima della fase di data integration si ha lo *scheme integration*, con approcci architetturali. Dal punto di vista di integrazione si hanno, si ricorda, gli approcci GAV o LAV, ma in entrambi i casi si hanno elementi rappresentanti lo stesso oggetto reale in db diversi che potrebbero non essere facilmente integrati a causa, magari, dell'assenza di una chiave primaria univoca. Bisogna quindi studiare come collegare i record presenti nei vari db tramite **record linkage** e mettere poi insieme i risultati con un'operazione di **fusion**, cercando di studiare eventuali ridondanze o ambiguità.

Il record linkage consiste quindi nell'identificare le stesse osservazioni in diversi file/db. In generale si parla di **entity resolution task/record linkage** avendo varie varianti:

- **deduplicazione**, che considerando una sola tabella è usata principalmente con il modello ER. Si procede raggruppando record che corrispondono allo stesso oggetto reale normalizzando lo schema e riducendo il numero di record nel dataset. Come variante si hanno i cluster di calcolo
- **record linkage**, dove appunto si matchano da un archivio dati deduplicato a un altro (bipartito). Si ha anche la versione k-partita lavorando con più data store. Generalmente questo metodo proposto nei datastore relazionali, ma più frequentemente applicato a record non strutturati da varie fonti

- **canonicalizzazione**, che generalmente fornisce il record più completo, attribuisce i valori tramite la fusione, costruendo dei **single version of truth**, usando metodi probabilistici. Questo metodo è tipico nel *master data management*, dove si raccolgono tutti i dati per poi integrarli
- **referencing**, detto anche **entity disambiguation**, dove si matchano record sporchi con uno pulito, con una tabella di riferimento deduplicata che è già stata canonizzata. Questo metodo generalmente utilizzato per atomizzare più record sulla stessa chiave primaria e donare informazioni aggiuntive al record

Si hanno vari tool per la entity resolution:

- *NTLK*, un natural language toolkit
- *Dedupe**, per lo structured deduplication
- *Distance*, un'implementazione in C per distance metrics
- *Scikit-Learn*, per machine learning models
- *Fuzzywuzzy*, per fuzzy string matching
- *PyBloom*, per probabilistic set matching

In input al record linkage si ha quindi una serie di tabelle e in output un insieme di tuple che matchano e un insieme di tuple che non matchano, nel caso relazionale. Si hanno anche tuple per cui non si sa dire nulla o per le quali non si ha certezza su match/mismatch. Lo stesso ragionamento si applica anche a modelli non relazionali
Si hanno varie tecniche per la comparazione delle coppie:

- quella **empirica**, basata sulla distanza di simboli, dal punto di vista lessicografico, nei valori delle tuple (Crlo è simile a Carlo)
- quella **probabilistica**, dove presumo di conoscere un campione di frequenze di distanze tra coppie di tuple, note come corrispondenti o non corrispondenti e dove potrei decidere quelli corrispondenti e quelli non corrispondenti nell'universo proiettando tale conoscenza sul campione nell'universo della coppia. Si usano soglie per poter discernere match e mismatch
- quelle **knowledge based**, dove si decide in base a regole per il match delle tuple
- **mista**, sia probabilistica che knowledge based
- le recenti tecniche di **machine learning**

Ovviamente grosse tabelle portano, confrontando tutti gli elementi ogni volta, una alta complessità, dal punto di vista prestazionale inefficienti. Si hanno quindi vari step per il record linkage:(non ho capito se parla solo di tabelle)

- costruzione dello spazio di ricerca iniziale come prodotto cartesiano di tutti i dataset in input.Questa è una fase di preprocessamento e in questa fase si cerca di ottenere un formato unico per tutti i sorgenti, sia dal punto di vista del modello che da quello dei dati (normalizzando i vari valori di uno stesso dominio, ad esempio avendo un solo modo per definire la via, di non usare abbreviazioni)
- si usano tecniche di blocking per ridurre lo spazio di ricerca, ottenendo uno spazio di ricerca ridotto. Si hanno varie tecniche di blocking in letteratura e vari algoritmi. Tra tutti si ha il *sorted neighbour* dove si prende un attributo e cerco di ordinare la tabella sugli insiemi degli attributi in modo tale che i gruppi degli attributi che confronto siano vicini tra loro.

  Ci si muove tra gli elementi tramite sliding window per specificare meglio i controlli, riducendo i confronti (senza la finestra ho n^2 confronti, con n righe, con la finestra di lunghezza m righe ho $m^2 \frac{n}{m}$, quindi $m \cdot n$ confronti, avendo $m \ll n$). Questa fase è facilmente distribuibile, facendo finestre di controllo in parallelo i più nodi
- sullo spazio ridotto si usa un metodo di comparazione, a cui è associato un metodo di decisione, per definire i match, i mismatch e quelli di cui non si ha certezza, che chiamiamo possible match (non sempre si hanno, dipende dalle soglie usate, una soglia di certo non permette di creare i possible match due invece si, nel mezzo tra le due si hanno i possible match). Si hanno metodi probabilistici, metodi non supervisionati come il clustering e anche supervisionati come: svm, logistic, random forest, boosting gradient. Confrontando quindi le varie coppie di dati

Questo è lo schema di base e in ogni fase si ha un processo di quality assesment.
Il primo problema è che i dataset potrebbero essere di formato diverso, relazionale o meno, e quindi, come i data integration, si cerca di trasformare un modello nell'altro. Potrei dover unire in un unico formato anche immagini e mappe, oltre a documenti json o csv.

Approfondiamo quindi l'uso di tecniche **probabilistiche** per la fase di comparazione, avendo un **probabilistic record linkage**, dove comunque le prime fasi restano normali. Quindi si cerca la probabilità di match tra le coppie usando dei vettori di distanza, usando la somma pesata dei vettori:

$$P_{match} = \sum_{score \in A} w \cdot score$$

Avendo:

- w pesi tra 0 e 1, avendo peso maggiore per feature meglio predittibili
- A vettore degli attributo ciascuno con un punteggio $score$

Se il punteggio così pesato supera una soglia ottengo un match.
Si hanno anche meccanismi basate su regole, sebbene la loro formulazione sia difficile, è possibile applicare regole specifiche del dominio rendendo più specifico l'approccio.
La soluzione proposta da **Fellegi Sunter**, quella probabilistica, è ottimale quando gli attributi di confronto erano condizionatamente indipendenti, permettendo comunque di avere un'area di ambiguità date dalla distribuzione dai match/mismatch.

Bisogna quindi per ogni coppia definire una funzione di distanza, imponendo un insieme finito di attributi da usare per valutare tale distanza, potrebbero esserci attributi non rilevanti. Un modo ottimale è scegliere attributi dello stesso significato e con la maggior accuratezza possibile per evitare errori. Si hanno anche attributi discriminatori per i quali i domini contengono un alto numero possibile di valori. Meno attributi si prendono in considerazione più rischio falsi positivi, aumentando i matching, anche se aumento le prestazioni. Scegliere troppi attributi può portare a falsi negativi, avendo tantissimi mismatch. È quindi un discorso che va approfondito di caso in caso.

L'algoritmo di Fellegi Sunter, per la fase decisionale, suggerisce di campionare la frequenza di match e non match, prendendo una coppia che si sa già che matcha e valutando per ogni distanza la frequenza di match/-mismatch andando a calcolare in seguito la probabilità di match in base alla percentuale di match/mismatch dati dalla distanaza di edit (ex: distanza di edit =2, abbiamo 27 match e 12 mismatch, la percentuale è 27/(27+12)=70%). Si fa un ragionamento in stile SVM anche se senza iperpiani ma con un separatore più complesso.

All'aumentare della distanza ovviamente aumentano i mismatch e viceversa. Alla fine del ragionamento posso produrre le soglie (o la soglia) da usare per lo studio dell'intero dataset. Si rischiano comunque falsi positivi e falsi negativi che possono essere ridotti aumentando i possibile match, allargando le soglie. Il record linkage, nel calcolo della distanza, è comunque molto legato al dominio.

Approfondiamo ora le tecniche di comparazione basate sul machine learning. Si hanno approcci supervisionati, SVM, random forest, conditional random fields e altri.
Ci sono anche tecniche non supervisionate, tramite clustering, con l'idea di usare K-means per raggruppare elementi simili.
Matchare testi descrittivi lunghi è difficile ma con tecniche di distanza non euclidea e di embedding, in un contesto comunque di machine learning, si riesce ad ottenere dei risultati.
In generale comunque il record linkage deve essere veloce, anche a discapito di un minimo di qualità.
Bisogna quindi valutare la qualità in termini di: true positive (TP), true negatives (TN), false positive (FP) e false negative (FN); ripotando le notazioni di recall e precisione dei dati:

$$recall = \frac{TP}{TP + FN} \qquad precision = \frac{TP}{TP + FP}$$

Avendo un'elevata conoscenza di dominio posso usare tali conoscenze per la comparazione e la cosa è difficilmente generalizzabile.

3.1 Data fusion

Passiamo quindi alla fusione dei record che rappresentano lo steso oggetto nel mondo reale. Bisogna gestire i conflitti. Si hanno varie tecniche:

- **ignorare il conflitto**, non è una buona soluzione. Queste strategie non consentono di prendere la decisione in base a ciò che è in conflitto con i dati e a volte non vengono rilevati i conflitti di dati.
- **evitare il conflitto**, con strategie basate su istanze e metadati, scegliendo di prendere istanze senza conflitti, prendendo solo valori consistenti senza dire niente degli altri.
- **risolvere il conflitto**, prendendo una sorgente più sicura o facendo una scelta a basata su vari aspetti. La decisione tiene conto o dei metadati o delle singole istanze per valutare la strategia. Si hanno due strategie:
    1. decidere quando scegliere un valore tra tutti i valori già presenti
    2. strategie di mediazione, quando scelgono un valore che non necessariamente esiste tra i valori inconsistenti (ad esempio faccio la media tra due valori per fare non avere conflitti(instance based) oppure prendere il valore più aggioranto (metadata based))

La scelta di fusione deve essere comunicata a chi poi usa i dati altrimenti si rischiano molti problemi.
È il tema della **provenance dei dati**, il tenere traccia delle decisioni prese. Con al data fusion si ottiene potenzialmente qualcosa con pochi NULL.
Alcune note con conclusive.
La **data preparation** è un buon modo per ottenere ottimi risultati, tramite normalizzazione di dati e schemi e *imputation*. Si rendono più facili i riscontri tramite distanze sintattiche.