



UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Dipartimento di Informatica, Sistemistica e  
Comunicazione

Corso di Laurea Magistrale in Informatica

# **STAN: UN TOOL PER L'ANNOTAZIONE SEMANTICA INCREMENTALE DI TABELLE SUL WEB**

**Relatore:**

*Dott. Matteo PALMONARI*

**Co-relatore:**

*Dott. Riccardo PORRINI*

**Relazione della prova finale di:**

Davide Brando PREDA

Matricola 735850

**ANNO ACCADEMICO 2014-2015**



# Indice

<b>1</b>	<b>Introduzione</b>	<b>11</b>
<b>2</b>	<b>Motivazioni e Casi di studio</b>	<b>17</b>
2.1	Casi di studio . . . . .	18
2.1.1	School closing, Chicago . . . . .	18
2.1.2	Ecommerce, 7Pixel . . . . .	21
2.2	Criticità . . . . .	24
2.2.1	Integrazione di tabelle sul web . . . . .	24
2.2.2	Automatizzazione dell'integrazione . . . . .	25
2.2.3	Strumento per non tecnici . . . . .	26
2.3	Obiettivi . . . . .	27
<b>3</b>	<b>Modelli</b>	<b>29</b>
3.1	Annotare una tabella . . . . .	29
3.2	Modello Star . . . . .	32
3.3	Modello Composed Star . . . . .	33
3.4	Modello Snow-Flake . . . . .	34
3.5	Modello Simple Snow-Flake . . . . .	35
3.6	Modello Hidden-subject . . . . .	36
<b>4</b>	<b>Stato dell'arte</b>	<b>39</b>
4.1	Data Integration . . . . .	39
4.2	Web Data Integration . . . . .	40
4.3	Table Integration . . . . .	42
4.4	Tools . . . . .	42
4.4.1	Karma . . . . .	42
4.4.2	Open Refine . . . . .	45
4.4.3	Confronto delle funzionalità . . . . .	48
4.5	Integrazione sfruttando una knowledge-base . . . . .	49
4.6	Integrazione sfruttando una ontologia di dominio . . . . .	50
4.7	Integrazione senza sfruttare risorse esterne . . . . .	52
4.8	Approcci . . . . .	52
4.8.1	Karma . . . . .	52

## INDICE

---

4.8.2	Table Miner . . . . .	53
4.8.3	Semantic message passing . . . . .	54
4.8.4	ONDINE . . . . .	55
4.8.5	QEWY . . . . .	56
4.8.6	Giva . . . . .	58
4.8.7	Framework ibrido macchina-crowdsourcing . . . . .	58
4.8.8	Confronto . . . . .	60
4.9	Analisi conclusive . . . . .	60
<b>5</b>	<b>STAN</b>	<b>63</b>
5.1	Architettura . . . . .	63
5.2	Processo di sviluppo . . . . .	65
5.3	Funzionalità . . . . .	67
5.3.1	Caricamento della tabella . . . . .	67
5.3.2	Workspace privato . . . . .	68
5.3.3	Visualizzazione della tabella . . . . .	68
5.3.4	Annotazione del soggetto . . . . .	69
5.3.5	Annotazione di una colonna come proprietà . . . . .	70
5.3.6	Suggerimenti di annotazione . . . . .	72
5.3.7	Salvataggio dello stato corrente . . . . .	74
5.3.8	Export dei mapping . . . . .	74
5.3.9	Export delle triple . . . . .	75
5.3.10	Definizione del namespace . . . . .	76
5.3.11	Servizio di API . . . . .	76
<b>6</b>	<b>Sperimentazione</b>	<b>79</b>
6.1	Qualità dei suggerimenti nel contesto dell'eCommerce . . . . .	81
6.1.1	Confronto STAN vs Karma . . . . .	82
6.1.2	Differenza tra colonne numeriche e testuali . . . . .	82
6.1.3	Dettaglio sulle singole colonne . . . . .	83
6.1.4	Qualità in relazione alla dimensione dell'indice . . . . .	84
6.1.5	Qualità in relazione al numero di valori . . . . .	88
6.2	Esperimenti di carico . . . . .	89
6.2.1	Tempi nel contesto dell'eCommerce . . . . .	89
6.2.2	Tempi nel contesto School Closing . . . . .	91
6.3	Considerazioni . . . . .	91
6.4	Vantaggi dell'approccio incrementale . . . . .	92
<b>7</b>	<b>Sviluppi futuri</b>	<b>95</b>
7.1	Utilizzo in 7Pixel . . . . .	95
7.2	Progetto Open Source . . . . .	96
7.3	Endpoint SPARQL . . . . .	96
7.4	Approfondimento su ABSTAT . . . . .	97
7.5	Supporto di altre knowledge base . . . . .	97

7.6	Supporto di altri formati . . . . .	97
7.7	Filtraggio object type . . . . .	98
7.8	Workspace ampliata . . . . .	98
7.9	Selezione del campione più rappresentativo . . . . .	98
<b>8</b>	<b>Conclusioni</b>	<b>101</b>

## INDICE

---

# Elenco delle figure

1.1	Un esempio di tabella estratta da Wikipedia relativa ai presidenti del consiglio italiani. . . . .	12
1.2	Un riquadro informativo in Google contenente informazioni strutturate estratte dal web. . . . .	14
2.1	Esempio di una tabella sul web la cui semantica va ricostruita affinché sia interpretabile da una macchina. . . . .	18
2.2	Esempio di integrazione tra due tabelle del dataset del caso di studio di Chicago . . . . .	21
2.3	Il processo di mapping in 7Pixel. . . . .	23
2.4	Il processo di annotazione di STAN. . . . .	25
3.1	Schematizzazione di una tabella T. . . . .	29
3.2	Esempio di modello Star. . . . .	33
3.3	Esempio di modello Composed Star. . . . .	34
3.4	Esempio di modello Snow-Flake. . . . .	35
3.5	Esempio di modello Simple Snow-Flake. . . . .	36
3.6	Esempio di modello Hidden-subject. . . . .	37
4.1	Schematizzazione degli approcci fisico e virtuale di Data Integration. . . . .	39
4.2	L'annotazione di dati sul web sfruttando un'ontologia come schema globale . . . . .	41
4.3	Il processo di generazione dei mapping tramite il tool Karma (fonte: [26]) . . . . .	43
4.4	L'interfaccia del tool Karma . . . . .	44
4.5	L'interfaccia del tool Open Refine. . . . .	47
4.6	La rappresentazione dell'OTR con le due componenti: concettuale e terminologica (fonte: [7]) . . . . .	56
4.7	Un frammento di QuTree. L'ontologia delle quantità e delle unità di misura. (fonte: [38]) . . . . .	57
5.1	Lo schema dell'architettura di STAN. . . . .	63
5.2	L'interfaccia di STAN durante l'import di una tabella. . . . .	67

## ELENCO DELLE FIGURE

---

5.3	La visualizzazione di una tabella dopo aver effettuato l'importazione in STAN. . . . .	69
5.4	L'annotazione di una colonna come soggetto della tabella in STAN. . . . .	70
5.5	L'interfaccia di STAN durante l'annotazione di una colonna come proprietà e i suggerimenti forniti dall'autocompletamento. . . . .	71
5.6	I suggerimenti forniti da STAN durante l'annotazione di una colonna. . . . .	72
5.7	Schematizzazione del problema dell'annotazione per l'algoritmo di STAN. . . . .	74
6.1	MRR calcolato sulle singole colonne per STAN e Karma. . . . .	84
6.2	Recall at 5 calcolata sulle singole colonne per STAN e Karma. . . . .	84
6.3	L'andamento del MRR al crescere della dimensione dell'indice per STAN. . . . .	85
6.4	L'andamento della Recall at 5 al crescere della dimensione dell'indice per STAN. . . . .	85
6.5	L'andamento del MRR al crescere della dimensione dell'indice per Karma. . . . .	86
6.6	L'andamento della Recall at 5 al crescere della dimensione dell'indice per Karma. . . . .	86
6.7	Il confronto tra STAN e Karma relativo al MRR al crescere della dimensione dell'indice. . . . .	87
6.8	Il confronto tra STAN e Karma relativo alla Recall al crescere della dimensione dell'indice. . . . .	87
6.9	L'andamento del MRR al crescere del numero di celle per STAN. . . . .	88
6.10	L'andamento della Recall at 5 al crescere del numero di celle per STAN. . . . .	88



# Elenco delle tabelle

2.1	Esempio di uno dei listini commerciali di 7Pixel. . . . .	24
3.1	Estratto della tabella relativa ai crimini commessi a Chicago. . . . .	33
3.2	Estratto della tabella relativa ai crimini commessi a Chicago con relative coordinate geografiche. . . . .	34
3.3	Estratto della tabella relativa alle scuole di Chicago. . . . .	35
3.4	Estratto di uno dei listini commerciali di 7Pixel in cui per ciascuna offerta vengono specificati la marca, i tempi di spedizione e il peso. . . . .	36
3.5	Estratto di uno dei listini commerciali di 7Pixel in cui il soggetto è implicito. . . . .	37
4.1	Tabella di confronto tra le funzionalità di Karma e Open Refine. . . . .	48
4.2	Tabella riassuntiva degli approcci analizzati. . . . .	60
5.1	Estratto di una delle tabelle relativa al caso di studio di Chicago. . . . .	70
6.1	Confronto di qualità tra STAN e Karma. . . . .	82
6.2	Confronto di qualità tra STAN e Karma nel dettaglio tra colonne numeriche e testuali. . . . .	83
6.3	I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto eCommerce in rapporto al numero di celle. . . . .	90
6.4	I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto eCommerce in rapporto alla dimensione dell'indice. . . . .	90
6.5	I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto School Closing in rapporto al numero di celle. . . . .	91
6.6	La percentuale di colonne annotate con una ontologia locale per ciascuna tabella del caso di studio delle School Closing. . . . .	93

## ELENCO DELLE TABELLE

---

# Capitolo 1

## Introduzione

Le tabelle sono un veicolo tramite il quale è possibile esprimere importanti informazioni che difficilmente sarebbe possibile esprimere in un testo libero. Le righe e le colonne di una tabella, infatti, conferiscono ai contenuti di essa una struttura chiara e precisa. Per un essere umano comprendere la semantica di una tabella può essere intuitivo anche proprio grazie alla sua struttura. Tuttavia lo stesso non si può dire di una macchina che si trova a dover interpretare una tabella presente sul web.

Le tabelle sono una delle poche pratiche di strutturazione dei dati sul web e probabilmente una delle più utilizzate. Infatti si conta che nei documenti web indicizzati da Google sono presenti oltre 154 milioni di tabelle contenenti dati relazionali [8]. In questa tesi ci si concentra proprio sulla semantica dei dati contenuti nelle tabelle sul web, un problema che si inserisce in un contesto più generale che è l'interpretazione dei dati presenti nelle pagine web. La maggior parte delle informazioni contenute sul web infatti viene processata dai motori di ricerca sotto forma di semplici stringhe di testo non strutturate. In fase di ricerca dunque il discriminante per stabilire se un documento deve essere considerato rilevante o meno si basa fondamentalmente su metriche di frequenza e similarità dei termini cercati rispetto a quelli contenuti nei documenti. In questo modo però la semantica intrinseca nei documenti viene persa e questo impedisce ad una macchina di cogliere il significato dei contenuti di una pagina web. Ad esempio considerando la tabella 1.1 estratta da una pagina web una macchina non è in grado di rispondere ad una domanda come "Quali sono stati i presidenti del consiglio italiani appartenenti al partito democratico?".

Il Semantic Web affronta questa classe di problemi definendo degli standard per rendere i contenuti presenti sul web maggiormente strutturati. Il web infatti si è sempre basato sul linguaggio HTML che consente di strutturare una pagina web dal punto di vista grafico in modo che sia visualizzabile all'interno di un browser. L'HTML puro tuttavia non consente di strutturare i contenuti in modo che siano comprensibili da una macchina. Negli ultimi anni tuttavia si è diffuso l'utilizzo di

## CAPITOLO 1. INTRODUZIONE

annotazioni semantiche che, correttamente abbinate al linguaggio HTML, consentono di conferire una semantica non ambigua e condivisa a precisi elementi della pagina. Sfruttando dunque dei linguaggi di annotazione appositi come ad esempio RDFa [1] è possibile arricchire i contenuti di una pagina con utili metadati. Queste informazioni aggiuntive sono poi estratte dai motori di ricerca e dai web crawlers per fornire agli utenti un'esperienza di ricerca più ricca ed esaustiva. Ad esempio Google sfrutta tali annotazioni per costruire automaticamente dei riquadri informativi a fianco dei consueti risultati di ricerca (figura 1.2).

N.	Ritratto	Nome (Nascita-morte)	Mandato		Partito	Governo e composizione		Leg.	Presidente della Repubblica
			Inizio	Fine					
		<b>Romano Prodi</b> (1939- )	17 maggio 2006	8 maggio 2008	L'Ulivo; Partito Democratico	Prodi II	L'Unione DS-DL/PD-PRC-RnP (SDI-RI) -PdCI-HdV-FdV UDEUR-SI-DCU-AL-SD-LD-MRE	XV (2006)	 (2006-2015) <sup>[4]</sup>
		<b>Silvio Berlusconi</b> (1936- )	8 maggio 2008	16 novembre 2011	Il Popolo della Libertà	Berlusconi IV	Centro-destra PdL-LN-MpA-CN-PT-FdS-DC	XVI (2008)	
		<b>Mario Monti</b> (1943- )	16 novembre 2011	28 aprile 2013	Indipendente	Monti	Governo tecnico con l'appoggio esterno di: PdL-PD-UdC-FLI-ApI		
		<b>Enrico Letta</b> (1966- )	28 aprile 2013	22 febbraio 2014	Partito Democratico	Letta	Grande coalizione PD-PdL/NCD-SC-UdC-PI-RI		 (2015-)
		<b>Matteo Renzi</b> (1975- )	22 febbraio 2014	<i>in carica</i>	Partito Democratico	Renzi	PD-NCD-SC-UdC-Dem. Solidale-PSI	XVII (2013)	

**Figura 1.1:** Un esempio di tabella estratta da Wikipedia relativa ai presidenti del consiglio italiani.

Annotare semanticamente un elemento significa arricchirlo in modo da assegnargli un significato non ambiguo e condiviso. In particolare l'annotazione mira ad assegnare a precisi elementi la semantica di un concetto o una proprietà contenuti in una ontologia. Un'ontologia è una rappresentazione formale, condivisa ed esplicita di una concettualizzazione di un dominio di interesse ed è interpretabile da una macchina. Dunque sfruttando le annotazioni e la conoscenza racchiusa nelle ontologie una macchina è in grado di interpretare i contenuti di una pagina web o i dati di una tabella. Una volta strutturati i contenuti diventa possibile integrarli tra loro unendo pezzi di informazioni in relazione tra di loro. Conseguentemente all'integrazione dei dati è possibile fornire una risposta ad interrogazioni complesse che implicano la comprensione di dati provenienti da fonti eterogenee. Ad esempio, date due tabelle estratte dal web relative rispettivamente ai dati statistici sui terremoti avvenuti negli ultimi cento anni in Italia e i dati edilizi relativi al numero

---

di edifici costruiti nelle città italiane, quello che si vorrebbe essere in grado di fare è trovare una correlazione tra questi eventi utilizzando le informazioni presenti in entrambe le tabelle. Per poter portare a termine questo compito è necessario comprendere la struttura e la semantica delle due tabelle, annotarle ed infine integrarle.

Quello dell'annotazione di fatto è un problema noto nell'area di ricerca della Data Integration. Nella Data Integration classica, infatti, una delle tecniche utilizzate per l'integrazione prevede la creazione di un dataset integrato virtuale mediante la definizione di mapping tra gli schemi locali dei dataset da integrare e uno schema globale. Di fatto quindi l'annotazione può essere vista proprio come la definizione di un mapping verso uno schema globale che è rappresentato da una ontologia.

L'obiettivo di questa tesi dunque è quello di concentrarsi sui dati in formato tabellare ed implementare uno strumento che consenta di effettuare annotazioni semantiche su di essi. Il risultato di questa tesi, infatti, è STAN una applicazione web, accessibile all'indirizzo <http://stan.disco.unimib.it/>, che consente tramite interfaccia grafica di importare una tabella, annotarla con la semantica di una ontologia e infine esportare il risultato dei mapping definiti.

L'implementazione del tool è stata guidata da due casi di studio reali attraverso i quali è stato possibile studiare e comprendere le problematiche relative all'integrazione dei dati. I due casi di studio fanno riferimento a due domini diversi e hanno fatto emergere esigenze e problematiche diverse. Il primo caso di studio è legato a temi socio-economici e tratta della decisione della città di Chicago di chiudere in massa 47 scuole della città per ridurre i costi di gestione. In questo caso di studio esistono una molteplicità di sorgenti di dati in formato tabellare (e.g. crimini, rendimento delle scuole, etc) che dovrebbero essere integrate da esperti di dominio e poi analizzate per rispondere a domande complesse. Il secondo invece riguarda il mondo dell'eCommerce, e in particolare l'azienda 7Pixel, un'azienda italiana leader nel settore della comparazione di prezzi online e proprietaria dei siti di TrovaPrezzi.it<sup>1</sup> e Kirivo.it<sup>2</sup>. Nello specifico il caso di studio tratta il problema di gestire e integrare in modo dinamico offerte commerciali provenienti da fonti eterogenee che nella maggior parte dei casi arrivano all'azienda in formato tabellare. L'analisi di questi due casi di studio ha permesso di dedurre alcuni dei problemi e delle sfide principali che il tool implementato si pone di affrontare. In particolare i temi principali emersi sono la difficoltà nell'integrazione di dati eterogenei e nella loro interpretazione semantica, la necessità di automatizzazione del processo di integrazione nel caso di grandi volumi di dati e l'esigenza di rendere il processo eseguibile anche da persone che non possiedono competenze tecniche.

STAN non è l'unico strumento di annotazione tabellare esistente. Infatti dall'analisi dello stato dell'arte è emerso che esistono già alcuni tool che affrontano questo

---

<sup>1</sup><http://www.trovaprezzi.it>

<sup>2</sup><http://www.kirivo.it>

## CAPITOLO 1. INTRODUZIONE

problema e anche diversi approcci per l'annotazione automatica dei dati di una singola colonna. Data una colonna e una ontologia questi approcci si prefiggono di annotare automaticamente la colonna con un concetto o una proprietà dell'ontologia. Analizzando la colonna *Partito* della tabella 1.1 tali algoritmi esaminano i valori contenuti in essa per annotarla ad esempio con la proprietà *dbp:party* di DBPedia. Nel corso della tesi dunque si condurranno due diversi confronti in rapporto allo stato dell'arte: uno legato alle funzionalità di STAN e uno legato alle caratteristiche e all'accuratezza dell'algoritmo di annotazione adottato in STAN. L'algoritmo di annotazione che viene utilizzato in STAN non è oggetto della tesi pertanto non sarà descritto in modo approfondito tuttavia propone un approccio alternativo e quindi la sua accuratezza sarà sperimentata in relazione ad un altro algoritmo da stato dell'arte.

**Harrison Ford - Wikipedia**  
[https://it.wikipedia.org/wiki/Harrison\\_Ford](https://it.wikipedia.org/wiki/Harrison_Ford)  
Harrison Ford (Chicago, 13 luglio 1942) è un attore statunitense. Tra la fine degli anni settanta e l'inizio dei novanta del XX secolo, ha partecipato ad alcuni dei ...  
[Biografia](#) - [Filmografia](#) - [Riconoscimenti](#) - [Doppiatori italiani](#)

**Harrison Ford | MYmovies**  
[www.mymovies.it/biografia/?a=262](http://www.mymovies.it/biografia/?a=262)  
Eroe delle saghe più appassionanti del cinema americano, Harrison Ford è stato protagonista di numerosi film di successo, firmati dai più acclamati registi ...

**Harrison Ford - IMDb**  
[www.imdb.com/name/nm0000148/](http://www.imdb.com/name/nm0000148/) Traduci questa pagina  
Harrison Ford was born on July 13, 1942 in Chicago, Illinois, to Dorothy (Nidelman), a radio actress, and Christopher Ford (born John William Ford), an actor ...

**Harrison Ford - Trovacinema - La Repubblica**  
[trovacinema.repubblica.it/attori-registi/harrison-ford/177002](http://trovacinema.repubblica.it/attori-registi/harrison-ford/177002)  
Harrison Ford è nato a Chicago, Illinois, il 13 luglio 1942. Figlio di un irlandese e di un'ebrea russa, Harrison è cresciuto nell'area suburbana di Des Plaines, ...

**Harrison Ford biografia - Comingsoon.it**  
[www.comingsoon.it/personaggi/harrison-ford/40439/biografia/](http://www.comingsoon.it/personaggi/harrison-ford/40439/biografia/)  
Info su Harrison Ford biografia filmografia discografia video foto citazioni curiosità frasi celebri news carriera.

**Harrison Ford | Film | The Guardian**  
[www.theguardian.com/film/harrisonford](http://www.theguardian.com/film/harrisonford) Traduci questa pagina  
Star Wars force awakens with standing ovation for Harrison Ford at Comic-Con ...  
Harrison Ford is welcomed by screaming fans at Comic-Con in his first public ...

**Comic - Con 2015, quante sorprese: arrivano Harrison Ford ...**  
[www.tgcom24.mediaset.it/.../comic-con-2015-quante-sorprese-arrivano-...](http://www.tgcom24.mediaset.it/.../comic-con-2015-quante-sorprese-arrivano-...)  
11 lug 2015 - 10:20 - Harrison Ford è riapparso in pubblico per la prima volta dopo lo schianto aereo in cui ha rischiato la vita lo scorso marzo. Venerdì 10 ...




**Harrison Ford**  
Attore

Harrison Ford è un attore statunitense. Tra la fine degli anni settanta e l'inizio dei novanta del XX secolo, ha partecipato ad alcuni dei film statunitensi di maggiore successo di quel periodo. [Wikipedia](#)

**Data di nascita:** 13 luglio 1942 (età 73), Chicago, Illinois, Stati Uniti  
**Altezza:** 1,85 m  
**Coniuge:** Calista Flockhart (s. 2010), Melissa Mathison (s. 1983–2004), Mary Marquardt (s. 1964–1979)  
**Figli:** Liam Flockhart, Ben Ford, Georgia Ford, Malcolm Ford, Willard Ford  
**Film in uscita:** *Star Wars: Episodio VII*

**Film**  
Visualizza altri 45 elementi

 Star Wars: Episodio VII 2015	 Star Wars: Episodio IV - A New... 1977	 Blade Runner 1982	 I predatori dell'arca perduta 1981	 L'impero colpisce ancora 1980
--	--	---	--	---

**Figura 1.2:** Un riquadro informativo in Google contenente informazioni strutturate estratte dal web.

I contributi principali di questa tesi sono:

1. La formalizzazione del problema e in particolare la formalizzazione dei modelli possibili di annotazione di una tabella.
2. L'implementazione di un tool a supporto dell'annotazione di una tabella.
3. Una sperimentazione che mette a confronto l'algoritmo di annotazione usato nel tool e uno esistente da stato dell'arte.

---

A partire dall'analisi dello stato dell'arte e provando ad annotare le tabelle reali dei casi di studio è emersa l'esistenza di alcuni pattern ricorrenti nell'annotazione di un tabella. Una definizione rigorosa di questi pattern non è presente nello stato dell'arte, per questo uno degli obiettivi della tesi è la formalizzazione di una serie di modelli possibili di annotazione. I modelli quindi sono stati utilizzati con un duplice scopo: per definire in maniera non ambigua i tipi di grafo RDF che vengono prodotti mediante STAN e per condurre un confronto tra i diversi tool di annotazione disponibili.

STAN presenta alcune caratteristiche che lo differenziano dagli altri tool esistenti da stato dell'arte. Innanzitutto STAN, a differenza degli altri tool, consente di definire una propria ontologia in maniera incrementale durante il processo di annotazione e questo permette agli utenti di utilizzare una semantica propria tramite la definizione di nuove proprietà e delimitandone la semantica. Un'altra differenza tra STAN e gli altri tool è l'architettura puramente pensata per il web. STAN infatti è una applicazione web utilizzabile online che permette a ciascun utente di avere il suo spazio di lavoro personale. Gli altri tool invece sono pensati per essere scaricati ed utilizzati da un singolo utente sulla propria macchina. STAN inoltre è l'unico tool che sfrutta un algoritmo di annotazione automatica *instance based* basato sul confronto di un campione dei valori della colonna con i valori di migliaia di proprietà di una knowledge base. STAN inoltre a differenza degli altri tool è uno strumento modulare che può essere esteso con diversi algoritmi di annotazione. L'unico requisito è che i nuovi algoritmi rispettino un certo formato per l'invocazione e la risposta. Infine STAN è l'unico tool che, sfruttando il proprio algoritmo di annotazione, espone un servizio di API pubbliche le quali possono essere interrogate da applicazioni di terze parti per effettuare l'annotazione automatica di gruppi di valori. Il servizio di API è stato pensato appositamente ragionando sul caso di studio di 7Pixel. Uno degli obiettivi della tesi, infatti, è quello di rendere STAN un provider di annotazioni all'interno dei sistemi di 7Pixel e le API sono il meccanismo più semplice per permettere l'integrazione con i sistemi legacy propri dell'azienda.

L'obiettivo della sperimentazione infine è quello di valutare l'algoritmo di annotazione in termini di qualità e di performance in relazione allo stato dell'arte. La sperimentazione è infatti stata condotta per studiare l'algoritmo sotto due diversi punti di vista: per valutarne la qualità dei suggerimenti nel caso specifico dell'e-Commerce e per valutarne le performance in termini di tempo. In particolare, dato che l'obiettivo è quello di integrarsi con i sistemi di 7Pixel, la sperimentazione è stata condotta su un dataset di listini commerciali di proprietà dell'azienda utilizzando come verità le annotazioni effettuate in passato. La sperimentazione, inoltre, è stata condotta confrontandosi con un altro algoritmo esistente da stato dell'arte in modo da valutare i punti di forza e debolezza dei due algoritmi.

## **CAPITOLO 1. INTRODUZIONE**

---

La tesi è organizzata come segue. Nel capitolo 2 verranno introdotti i casi di studio analizzati e le criticità risultanti da essi. Nel capitolo 3 si formalizzano i modelli ricorrenti di annotazione individuati a partire dall'analisi dei casi di studio. Nel capitolo 4 si contestualizza il problema all'interno dello stato dell'arte effettuando una analisi approfondita degli strumenti e degli approcci esistenti. Nel capitolo 5 si descrive STAN, la sua architettura e l'analisi funzionale. Nel capitolo 6 si analizzano i risultati della sperimentazione condotta sull'algoritmo di annotazione di STAN nel dominio dell'eCommerce. Infine nel capitolo 7 si descrivono gli sviluppi futuri previsti per il tool.



## Capitolo 2

# Motivazioni e Casi di studio

Il web è una collezione di documenti ed informazioni immensa. Si conta infatti che ad oggi il numero di pagine indicizzate dai motori di ricerca siano almeno 47 miliardi<sup>1</sup>. Tuttavia i dati contenuti nei documenti presenti sul web sono pensati prevalentemente per essere compresi dagli esseri umani. Per questo motivo si presta molta attenzione all’impaginazione e allo stile piuttosto che alla strutturazione di essi. Questo rende spesso l’estrazione e l’integrazione di tali dati molto difficoltosa. I principali ostacoli da affrontare, infatti, sono l’enorme eterogeneità di contenuti e formati dei dati oltre alla quasi totale assenza di semantica a descrizione di essi. Ad esempio la tabella in figura 2.1 fa riferimento alla produzione di caffè in alcuni paesi nel corso dell’anno 2006. Tuttavia considerando solamente la tabella non è possibile comprendere né che si stia parlando della produzione di caffè né che siano dati riferiti all’anno 2006. Infatti si riesce a comprendere la corretta interpretazione dei dati solo leggendo la descrizione di contesto che si trova al di fuori della colonna.

Di conseguenza per integrare i dati presenti nelle pagine web, ed in particolare nelle tabelle, uno dei problemi principali è ricostruire la semantica dei loro schemi. In particolare cercando di assegnare ai dati la semantica data da precise ontologie e vocabolari. Una volta esplicitata la semantica dei dati risulta più semplice effettuare l’integrazione. L’obiettivo della tesi è quello di implementare un tool che semplifichi la comprensione e la ricostruzione di tale semantica. Per guidare la modellazione e lo sviluppo di questo tool sono stati presi in considerazione due casi di studio in modo da concentrarsi su casi e problemi concreti. L’integrazione dei dati si applica in numerosi domini, nel caso di questa tesi in particolare ci si è concentrati su due domini: uno di contesto socio-economico e uno commerciale. Dall’analisi di questi due casi di studio è possibile dedurre alcuni dei problemi e delle sfide principali da affrontare nell’integrazione dei dati. Nelle prossime sezioni saranno descritti dunque i due casi di studio e a seguire verranno affrontate più nel dettaglio le sfide da affrontare nello sviluppo di STAN.

---

<sup>1</sup><http://www.worldwidewebsize.com/>

The following table lists the total coffee production of each [coffee exporting country](#) in the year 2006<sup>[1]</sup>.

Country	60 kilogram bags	Kilograms	Pounds
Brazil	42,512,000	2,550,720,000	5,611,584,000
Vietnam	15,000,000	900,000,000	1,980,000,000
Colombia	11,600,000	696,000,000	1,531,200,000
Indonesia	6,850,000	411,000,000	904,200,000
Ethiopia	5,500,000	330,000,000	726,000,000
India	5,005,000	300,300,000	660,660,000
Mexico	4,500,000	270,000,000	594,000,000
Guatemala	4,000,000	240,000,000	528,000,000
Peru	3,500,000	210,000,000	462,000,000
Honduras	2,700,000	162,000,000	356,400,000
Uganda	2,500,000	150,000,000	330,000,000
Ivory Coast	2,350,000	141,000,000	310,200,000
Costa Rica	1,808,000	108,480,000	238,656,000

**Figura 2.1:** Esempio di una tabella sul web la cui semantica va ricostruita affinché sia interpretabile da una macchina.

## 2.1 Casi di studio

### 2.1.1 School closing, Chicago

#### Il problema

Il 22 Maggio 2013 la Chicago Board of Education ha votato per la chiusura di 47 scuole elementari pubbliche sottoutilizzate nella città di Chicago [13], la più grande chiusura di massa di scuole del paese. La decisione è stata presa per fronteggiare il miliardo di dollari di debito che era stato ormai accumulato. Da dati ufficiali gli studenti iscritti nelle scuole pubbliche di Chicago erano 403,000 a fronte di 511,000 posti disponibili nelle scuole. Per ridurre dunque gli sprechi è stato deciso di chiudere quelle scuole a rischio che contavano un numero troppo basso di iscritti. Le 47 scuole chiuse infatti contavano tutte un numero di studenti non superiore a 600 e con un numero di iscritti inferiore al 70% del numero di posti massimo.

La decisione è stata accolta con grande riluttanza dall'opinione pubblica dato che la chiusura ha interessato oltre 12,000 studenti. In particolare le famiglie e l'associazione degli insegnanti di Chicago hanno espresso la loro preoccupazione circa la sicurezza degli studenti dato che essi avrebbero dovuto muoversi più lontano da casa per raggiungere le nuove scuole dovendo magari passare per aree della città poco sicure. Dunque per evitare che gli studenti deallocati finissero in scarsi ambienti educativi e che soffrissero da un punto di vista sia emotivo che accademico,

la Chicago Board of Education ha assegnato ciascuno studente ad una *"welcoming school"*. Ciascuno studente infatti è stato riallocato in una scuola con un livello di istruzione pari o migliore della precedente. Inoltre sono stati fatti investimenti da parte dei distretti sia in programmi aggiuntivi nelle *"welcoming school"* che nei servizi di trasporto e collegamento verso di esse. I criteri utilizzati nella scelta della riallocazione degli studenti sono stati:

- Un livello di istruzione pari o superiore alla scuola precedente basandosi sui dati di performance relativi all'anno scolastico 2012-13.
- Una distanza non maggiore di un miglio dalla scuola precedente.
- La disponibilità di posti per gli studenti nella nuova scuola.

La scelta della riallocazione non è stata imposta alle famiglie che infatti hanno potuto scegliere di iscrivere i propri figli nella scuola suggerita oppure in una qualsiasi altra. In molti casi le famiglie hanno considerato criteri differenti nella scelta della nuova scuola come ad esempio: la vicinanza da casa, la presenza di buoni insegnanti, la presenza di amici o familiari, l'esistenza di servizi di trasporto adeguati e altri fattori legati alla sicurezza come il tasso di criminalità. Alla fine, durante il successivo anno scolastico, il 93% degli studenti provenienti da una delle scuole chiuse si è iscritto ad una scuola con un livello di istruzione maggiore della precedente. Tuttavia un quarto di essi ha scelto di iscriversi in una scuola con livello di istruzione inferiore rispetto alla *"welcoming school"* a cui era stato assegnato.

Analizzando questo scenario si capisce facilmente quanto sia complicato prendere una decisione di questo tipo e quanti fattori la influenzino, sia dal punto di vista dei genitori che dello stato. La Chicago Board of Education sicuramente ha condotto delle analisi per stabilire gli accoppiamenti tra le scuole da chiudere e le *"welcoming school"*, tuttavia sono stati trascurati alcuni fattori ritenuti invece rilevanti dalle famiglie. Una analisi di questo genere, tuttavia, non è affatto semplice da condurre considerando l'eterogeneità dei formati e delle fonti dei dati a disposizione: dati anagrafici sugli studenti, dati accademici relativi alle scuole, dati del censimento relativi alla popolazione e alla povertà dei quartieri, dati relativi ai trasporti pubblici e infine dati della polizia relativi ai crimini.

Per produrre una integrazione dei dati di qualità è necessario un intervento umano che tuttavia richiede di possedere alcune competenze tecniche. In un caso come questo dunque sarebbe potuto essere utile uno strumento come STAN in grado di conferire una semantica ai dati tramite interfaccia grafica al fine di integrarli e analizzarli nel loro insieme. In generale comunque l'integrazione di dati in casi del genere risulta fondamentale per poter condurre un'analisi approfondita che si basi su dati reali. Nel caso specifico integrare i dati a disposizione anche successivamente alla effettiva riallocazione degli studenti può aiutare a rispondere ad alcune domande come: perchè le famiglie hanno deciso di mandare o meno i propri figli

nelle *"welcoming school"*? Quali sono le caratteristiche degli studenti interessati da queste chiusure? Gli studenti allocati in scuole con un livello di istruzione superiore conseguiranno davvero dei risultati accademici migliori?

### Il dataset

Nel contesto di questa tesi sono stati recuperati e analizzati diversi dati in formato Open Data messi a disposizione dal governo di Chicago, in particolare:

- *Scuole*. Dataset sulle scuole di Chicago con alcune informazioni base come il nome, il tipo di scuola, le classi, l'indirizzo, il quartiere e le coordinate geografiche.
- *Qualità delle scuole*. Dataset con tabelle innestate relative ai diversi fattori analizzati per valutare la qualità del servizio accademico fornito dalle scuole elementari e superiori di Chicago. In particolare per ciascuna scuola viene specificato un punteggio ed un livello che sono il risultato aggregato di diversi indici di qualità come l'andamento medio degli studenti in alcune materie rispetto alla media nazionale oppure la media di studenti che frequentano le lezioni.
- *Censimento*. Dati sul numeri di abitanti di Chicago per ogni quartiere della città.
- *Povertà*. Dataset relativo al benessere economico di ciascun quartiere della città di Chicago. In particolare per ciascun quartiere si specificano informazioni come la percentuale di famiglie sotto la soglia di povertà, la percentuale di disoccupazione, la percentuale di persone senza un diploma e la media del reddito annuale pro capite.
- *Crimini*. Dataset della polizia di Chicago relativo ai crimini commessi dal 2010 ad oggi. Ciascun crimine è archiviato con un numero di caso, una data, l'indirizzo in cui è stato commesso, il quartiere, le coordinate, una descrizione e eventualmente l'arresto del colpevole.
- *Confini geospaziali*. Dati geospaziali in formato shapefile relativi sia ai confini dei quartieri di Chicago che ai distretti della polizia.

L'utilizzo di dati reali relativi a questo caso di studio ha consentito una più profonda comprensione di cosa significa trattare dati eterogenei e quanto possa essere utile in vari domini l'integrazione di essi. Nel caso di Chicago, ad esempio, integrando i dati relativi alla qualità delle scuole e quelli geospaziali relativi alla distanza dalle vecchie scuole è stato possibile considerare alcuni fattori chiave nella scelta delle nuove scuole. In futuro se questa situazione si dovesse riproporre, a Chicago o in un'altra città, si potrebbe invece pensare di sfruttare un insieme più grande di dataset in cui ad esempio sono contenute informazioni relative ai

fattori presi in considerazione dalle famiglie di Chicago. Riuscire a fornire una visione di insieme chiara e coerente su un insieme di dati, inoltre, è il primo passo da compiere per chi si occupa di Data Journalism [21], un approccio a metà strada tra la ricerca e l'inchiesta giornalistica che fa un uso intensivo di dati e informazioni per analizzare, raccontare e visualizzare un fenomeno o una notizia, spesso producendo delle aggregazioni di elementi diversi. Nel caso della chiusura delle scuole di Chicago ad esempio si sarebbero potuti integrare i dati relativi alle scuole di Chicago e quelli relativi ai crimini commessi (figura 2.2). In questo modo sarebbe stato possibile tranquillizzare le famiglie riguardo alla sicurezza dei quartieri delle *"welcoming school"* oppure metterle in guardia da eventuali quartieri poco sicuri.

Case Number	Location
HH215971	Lincoln Park
HK256802	Forest Glen

School	Community Area
Prescott elem. school	Lincoln Park
Northside high school	North Park

**Figura 2.2:** Esempio di integrazione tra due tabelle del dataset del caso di studio di Chicago

### 2.1.2 Ecommerce, 7Pixel

#### Il problema

L'importanza della tecnologia Web come mezzo per le transazioni commerciali è cresciuta in modo significativo [15]. Lo sviluppo di nuove tecnologie in questo ambito ha portato a drastici cambiamenti nelle relazioni con i clienti e nei modelli di business. I classici rapporti commerciali venditore-acquirente vengono sostituiti da relazioni multi-a-molti tra clienti e fornitori. Tuttavia, questa nuova flessibilità nel commercio elettronico genera anche importanti sfide. Il problema principale è l'eterogeneità delle descrizioni informative utilizzate dai fornitori e dai clienti. Le descrizioni dei prodotti, i formati dei cataloghi, e i documenti aziendali sono spesso non strutturati e non standardizzati. Nella maggior parte dei casi i venditori hanno il loro modo di descrivere prodotti e offerte, mentre gli acquirenti ne hanno un altro. Strutturare e standardizzare le descrizioni delle offerte è un compito importante nel mondo dell'e-commerce, per assicurarsi poi che i diversi attori coinvolti possano effettivamente comunicare tra loro.

7Pixel<sup>2</sup> è un'azienda italiana leader nella comparazione prezzi e nello shopping online. In particolare 7Pixel è proprietaria dei siti TrovaPrezzi.it<sup>3</sup>, ShoppyDoo.it<sup>4</sup>

<sup>2</sup><http://www.7pixel.it/>

<sup>3</sup><http://www.trovaprezzi.it/>

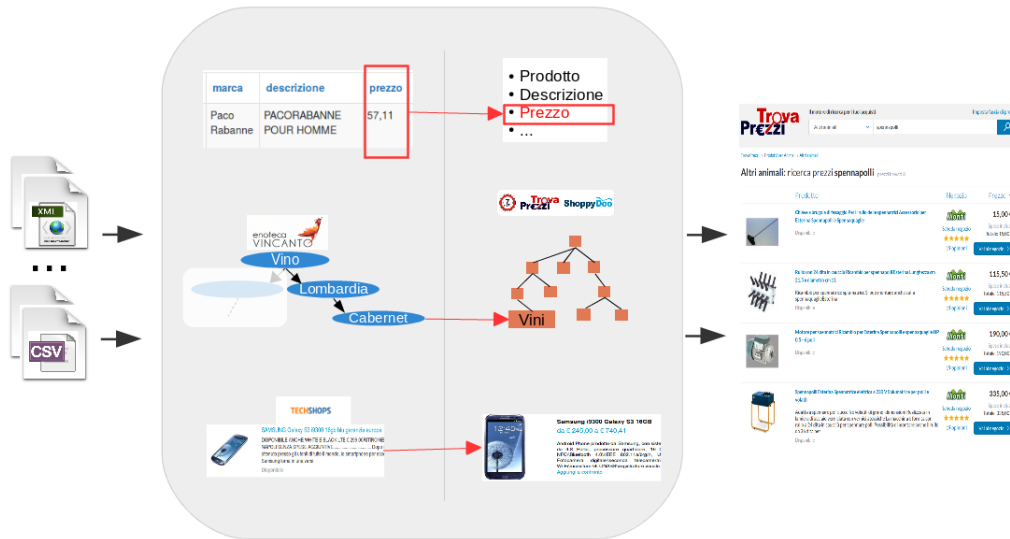
<sup>4</sup><http://www.shoppydoo.it/>

e Kirivo.it<sup>5</sup>. Come nella maggior parte delle aziende di ecommerce 7Pixel si trova ad affrontare tutti i problemi di eterogeneità citati in precedenza. Infatti, i commercianti che desiderano apparire sui siti di 7Pixel inviano migliaia di listini di offerte commerciali eterogenei tra di loro. Ciascun listino, infatti, possiede la semantica che il commerciante gli ha assegnato. Tuttavia 7Pixel per interpretare e integrare le offerte commerciali sfrutta una semantica propria raffinata negli anni. Per questo è necessario un processo di mappatura tramite il quale viene riconciliata la semantica dei listini con la semantica di 7Pixel (figura 2.3). Questo processo ad oggi viene eseguito manualmente da degli esperti di dominio che si occupano dei contenuti da pubblicare sui siti di 7Pixel. Tale lavoro è supportato da alcuni software sviluppati internamente all'azienda ma il compito di mappatura è lasciato esclusivamente al ragionamento umano. Durante la fase di mappatura, in particolare, sono tre gli elementi principali che vengono mappati:

- **Schema:** i listini che arrivano a 7Pixel sono, o possono essere visti, in formato tabellare (CSV e XML). Preliminarmente a qualsiasi altra attività di mappatura quindi deve essere riconosciuto lo schema di tali listini. Ciascun campo del listino infatti viene annotato dagli esperti di 7Pixel con un valore identificativo che ne determina la semantica (e.g. Prodotto, Prezzo, Categoria, ..). Capita che non tutti i campi del listino siano importanti per 7Pixel per cui gli esperti hanno anche la possibilità di ignorare alcuni campi.
- **Categorie:** i commercianti inseriscono ciascuna delle proprie offerte in una categoria merceologica. In questo modo quando le offerte vengono pubblicate sui siti di 7Pixel sono visualizzabili in una precisa categoria. Tuttavia ciascun commerciante definisce le categorie a modo suo e 7Pixel a sua volta possiede un proprio albero delle categorie. Per questo motivo è necessario che gli esperti di dominio individuino delle corrispondenze tra le categorie utilizzate dai commercianti e quelle di 7Pixel. Spesso, inoltre, non esistono corrispondenze dirette. Infatti capita che le categorie dei commercianti siano generiche rispetto a quelle di 7Pixel oppure viceversa.
- **Prodotti:** per poter offrire ai propri utenti dei contenuti maggiormente informativi, 7Pixel raggruppa le offerte che fanno riferimento ad uno stesso prodotto. Questo consente agli utenti di trovare l'offerta più vantaggiosa per un determinato prodotto. Per rendere questo possibile, tuttavia, è necessario riconoscere tra le offerte dei commercianti quelle che fanno riferimento ai prodotti censiti sui siti di 7Pixel. Dei tre tipi di mapping questo è quello più automatizzato. Infatti, esiste un software che tramite un approccio supervisionato è in grado di riconoscere il prodotto di riferimento per un'offerta basandosi su titolo, descrizione e categoria dell'offerta.

---

<sup>5</sup><http://www.kirivo.it/>



**Figura 2.3:** Il processo di mapping in 7Pixel.

Uno degli obiettivi di 7Pixel per i prossimi anni è quello di cercare di rendere queste fasi di mapping il più automatizzate possibili. Quanto meno il desiderio è quello di far svolgere il grosso del lavoro dalle macchine e lasciare agli esperti di dominio solo un compito di supervisione. Lo strumento che è stato sviluppato nel corso di questo lavoro in quest'ottica è utile per rendere il mapping dello schema automatico. Infatti, STAN è stato pensato per potersi integrare con i sistemi attualmente utilizzati in 7Pixel per effettuare la mappatura. Dato un nuovo listino di un commerciante, infatti, STAN è in grado di proporre una mappatura completa dello schema. Tramite il servizio di API che STAN offre è possibile popolare in modo automatico i campi relativi alla mappatura del listino. In questo modo si semplifica notevolmente il lavoro degli esperti di dominio ai quali non resta che supervisionare tale mappatura ed eventualmente modificarla.

## Il dataset

I commercianti che desiderano comparire sui siti di comparazione prezzi di 7Pixel inviano le loro offerte sotto forma di listino. Ciascun listino può contenere un numero variabile di offerte che va dalle decine alle migliaia. Ogni offerta è caratterizzata mediamente da 12 campi che la dettagliano (e.g. Prezzo, Descrizione, Marca, ..). Al momento, sui siti di TrovaPrezzi.it e ShoppyDoo.it sono pubblicate oltre 12 milioni di offerte di oltre 2500 commercianti diversi. Considerando questa mole di dati è facile rendersi conto di quanto sia onerosa la fase di mappatura dei listini.

Per le finalità di questa tesi sono stati utilizzati i listini pubblicati su TrovaPrezzi.it ad una certa data. Il dataset così definito è formato da oltre 2300 listini di commercianti diversi. Per condurre la sperimentazione, inoltre, sono stati utiliz-

zati come gold standard i mapping già definiti manualmente tra i campi dei listini e l'ontologia specifica di dominio utilizzata in 7Pixel.

SKU	Marca	Prodotto	Prezzo	Categoria	Disponibilità
90183	Rolex	Rolex Date	2300	Orologeria	7
67127	Omega	Omega Seamaster	1700	Orologeria	10
09972	Zenith	El Primero	3000	Orologeria	5
09972	Pandora	Anello Eternity	349	Gioielleria	20
09972	Swarovski	Christie Oval	79	Gioielleria	20

**Tabella 2.1:** Esempio di uno dei listini commerciali di 7Pixel.

## 2.2 Criticità

Dall'analisi dei casi di studio emergono alcune criticità e problemi da affrontare. La necessità principale è quella di integrare dati eterogenei per creare una nuova risorsa informativa aggregata. Questa sfida inoltre ne implica altre come la corretta interpretazione della semantica dei dati. In particolare le tematiche che emergono prevalentemente dall'analisi dei casi di studio sono:

- Le difficoltà nell'integrazione dei dati tabellari contenuti nei documenti sul web. La classe di problemi che riguarda l'eterogeneità dei dati e l'ambiguità della loro interpretazione semantica.
- La necessità di automatizzare il processo di integrazione nel caso di grandi volumi di dati.
- L'esigenza di rendere il processo eseguibile anche da persone che non hanno competenze tecniche.

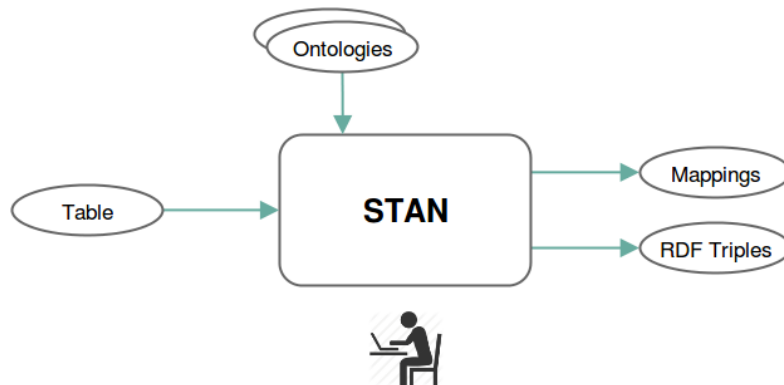
Nelle prossime sezioni vengono approfondite tali tematiche specificando in particolare come STAN, il tool sviluppato durante questa tesi, li affronta e cerca di risolverli.

### 2.2.1 Integrazione di tabelle sul web

L'obiettivo di questa tesi è quello di creare uno strumento che fornisca supporto ad un utente che voglia ricostruire la semantica di una tabella. In questo modo, date due o più tabelle annotate con una semantica coerente risulta più semplice integrarle per estrarre nuova informazione risultante dalla combinazione di tali dati. Una delle tecniche utilizzate nella Data Integration sfrutta ontologie e vocabolari per annotare i dati al fine di integrarli [11]. Questo approccio consente sia di riutilizzare ontologie condivise che di definirne di proprie. Annotare diverse tabelle con una propria ontologia consente l'integrazione di esse. Tuttavia è proprio l'utilizzo



di ontologie esistenti e condivise che evidenzia i vantaggi offerti dall'annotazione semantica dei dati. Sul web infatti esiste già una grande quantità di dati disponibili e annotati con ontologie condivise: la Linked Open Data Cloud (LOD) [39], una collezione di dati che conta al 2014<sup>6</sup> una gran mole di informazioni provenienti da oltre 1000 dataset di diversi domini (e.g. Scienza, Geografia, Musica, ...). Utilizzare ontologie condivise consente quindi di integrare i propri dati con quelli della LOD dato che la semantica che li descrive è la stessa. Inoltre, l'annotazione mediante ontologie condivise consente anche la pubblicazione dei dati nella LOD in modo da contribuire alla codifica di nuova conoscenza.



**Figura 2.4:** Il processo di annotazione di STAN.

Il tool sviluppato, STAN, consente dunque agli utenti di annotare in modo interattivo una tabella con i concetti e le proprietà di una o più ontologie. In particolare l'utente ha la possibilità sia di definire una propria ontologia che di utilizzarne una esistente. L'output del processo si traduce nella strutturazione della conoscenza contenuta nella tabella in input. In particolare gli output del sistema sono due. I mapping tra la tabella e le ontologie, ossia le annotazioni semantiche prodotte dal tool, e i dati strutturati in formato RDF da pubblicare nella LOD o in un qualsiasi triple store.

### 2.2.2 Automatizzazione dell'integrazione

Quando si hanno grandi volumi di dati da integrare l'annotazione manuale al fine di integrare i dati può rivelarsi troppo onerosa. Per questo motivo uno degli obiettivi della tesi è quello di cercare di rendere la procedura di annotazione automatica in modo da semplificare questo compito alle persone rendendolo eseguibile da una macchina.

Per raggiungere questo obiettivo STAN incorpora al suo interno un algoritmo in grado di fornire una lista ordinata di annotazioni possibili per ciascuna colonna della tabella. Inoltre, per rendere il processo integrabile in altri sistemi STAN offre

<sup>6</sup><http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

anche una interfaccia interrogabile tramite API HTTP [18]. Sfruttare un simile algoritmo può essere utile in diversi domini e applicazioni. Ad esempio, come analizzato nel caso di 7Pixel, nel mondo dell'eCommerce spesso esiste la necessità di integrare offerte di commercianti diversi che, nella maggior parte dei casi, sono tutte caratterizzate da una semantica propria. Per poterle integrare e visualizzare online dunque risulta necessario riconciliare la semantica delle singole offerte con una semantica specifica del sito di eCommerce. Nel caso di 7Pixel, ad oggi, la mappatura è effettuata a mano. Tuttavia l'utilizzo di STAN come provider di possibili annotazioni può rendere questo procedimento automatico oppure semi-automatico con la supervisione di un esperto di dominio.

### 2.2.3 Strumento per non tecnici

L'approccio classico di annotazione dei dati prevede la definizione di precise regole di mapping tra i dati e una o più ontologie. Tali regole tuttavia sono definite in modo da poter essere processate da una macchina, e per questo motivo devono rispettare la sintassi di un preciso linguaggio. Esistono diversi linguaggi che consentono la definizione di tali regole (e.g. RML [14], R2RML [12], D2RQ<sup>7</sup>, ..), ciascuno con la propria sintassi e formalismi. Per questo motivo per affrontare la fase di mappatura è necessario conoscere il linguaggio di mapping e possedere un minimo di competenze tecniche. Nella maggior parte dei casi questo significa che per analizzare dati provenienti da fonti diverse è necessario l'intervento di un tecnico che si occupi dell'integrazione preliminare dei dati. Questo significa, ad esempio, che un data journalist intenzionato a scrivere un articolo basato su dati di diverse fonti deve necessariamente essere affiancato da un tecnico e non può essere indipendente nella sua indagine.

Per questo motivo si è deciso di sviluppare un tool per consentire anche a persone che non possiedono competenze tecniche di effettuare la mappatura dei dati. Infatti con STAN è possibile svolgere l'intero processo senza conoscere i tecnicismi del linguaggio di mapping che vengono nascosti all'utente dall'interfaccia del tool. STAN propone un'interfaccia intuitiva per l'annotazione di ciascuna colonna. Infatti è sufficiente selezionare una proprietà e il tipo dei valori raggiunti da quella proprietà. Per facilitare ulteriormente l'utente, inoltre, sono disponibili un sistema di autocompletamento che aiuta in fase di compilazione e una lista di annotazioni suggerite generata automaticamente dal tool.

---

<sup>7</sup><http://d2rq.org/d2rq-language>

## **2.3 Obiettivi**

Partendo dai casi di studio reali di 7Pixel e di Chicago, l'obiettivo di questa tesi è comprendere e analizzare il processo di annotazione semantica di una tabella e sviluppare un tool che lo supporti. Il tool sviluppato deve consentire il riutilizzo di ontologie e vocabolari condivisi ma anche dare la possibilità di definire una propria ontologia in modo incrementale parallelamente all'annotazione. Il tool, inoltre deve consentire anche a persone che non hanno una formazione tecnica di portare a termine l'annotazione. Per questo motivo è necessario che il tool nasconda agli utenti il linguaggio di mapping e allo stesso tempo li supporti in fase di annotazione con un'interfaccia intuitiva e suggerimenti.

Posti questi obiettivi i contributi portati da questa tesi sono:

- La concettualizzazione del problema ed, in particolare, la definizione dei modelli possibili di annotazione.
- Lo sviluppo di un tool per l'annotazione incrementale di tabelle sfruttando una o più ontologie.
- Una sperimentazione sulla qualità dei suggerimenti forniti da due diversi algoritmi di annotazione automatica nel caso reale dell'annotazione di listini commerciali.

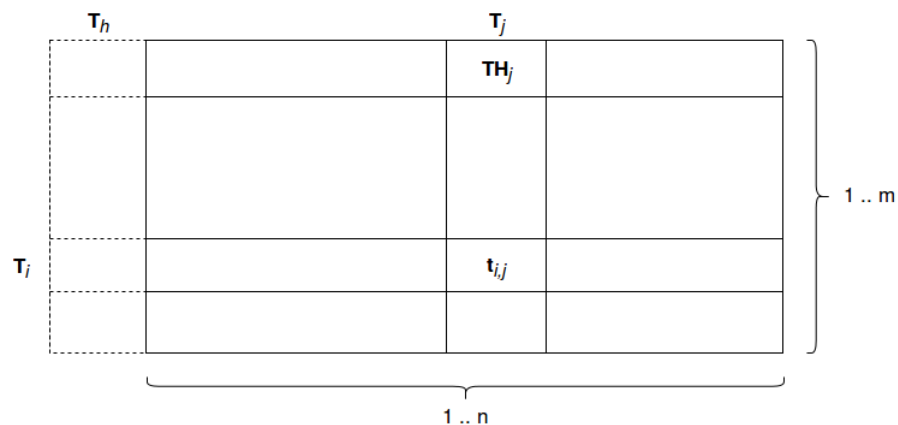


## Capitolo 3

# Modelli

### 3.1 Annotare una tabella

Dall'analisi dei casi di studio e dello stato dell'arte è emerso che le strutture con cui vengono presentati i dati delle tabelle spesso presentano dei pattern comuni e ricorrenti. In particolare è emerso che spesso una tabella è caratterizzata da un insieme di oggetti principali raggruppati in una colonna soggetto. Le rimanenti colonne, invece, descrivono le proprietà degli oggetti contenuti nella colonna soggetto oppure descrivono le proprietà di altre colonne della tabella. La colonna soggetto, in particolare, può essere sia esplicita che implicita nella tabella. Infatti, come schematizzato in figura 3.1, può capitare che la colonna soggetto non sia effettivamente compresa tra i dati della tabella ma che esista implicitamente. Le tabelle contenute nei listini commerciali di 7Pixel, ad esempio, sono caratterizzate da colonne che descrivono un insieme di offerte. Tuttavia all'interno delle tabelle stesse non esistono oggetti di tipo offerta poiché sono impliciti.



**Figura 3.1:** Schematizzazione di una tabella T.

L'annotazione di una tabella dunque prevede l'assegnazione di una semantica non ambigua e condivisa alle colonne utilizzando un'ontologia di riferimento. Data una colonna soggetto, l'annotazione prevede per ciascuna colonna la specifica di una proprietà che collega gli oggetti di un'altra colonna con quelli della colonna in questione, e la specifica di un tipo per i valori della colonna. A seconda del tipo specificato per una colonna i valori in essa assumono una diversa connotazione, in particolare i valori possono essere considerati come:

- *Literal*: valori come numeri, stringhe e date.
- *Uri*: istanze di una classe dell'ontologia.

### Definizione di annotazione

Date:

- Una tabella relazionale  $T$ , avente  $i$  righe e  $j$  colonne.  $T_i$  denota la  $i$ -esima riga della tabella,  $T_j$  denota la  $j$ -esima colonna,  $T_s$  denota la colonna soggetto,  $TH_j$  denota l'header della colonna  $j$  che può esistere o essere stringa vuota, e  $t_{i,j}$  è una cella alla riga  $T_i$  e colonna  $T_j$ .
- Una ontologia  $O$  definita come un insieme di assiomi sui termini definiti da  $P$  e  $X$  formulati in un linguaggio formale  $L$ .  $P = P_o \cup P_d$  è l'insieme di tutte le proprietà dell'ontologia con  $P_o$  insieme di tutte le *ObjectProperty* e  $P_d$  insieme di tutte le *DatatypeProperty*.  $X = C \cup D$  è l'insieme di tutti i tipi dell'ontologia con  $C$  insieme di tutte le classi e  $D$  insieme di tutti i datatype.

Annotare una tabella  $T$  nei termini dell'ontologia  $O$  significa:

- Definire una sola colonna soggetto  $T_s$  associata ad una classe  $C_s$  che rappresenta il tipo dei valori in  $T_s$ .
- Per ogni altra colonna  $T_j$  che si desidera annotare, definire una proprietà  $p_j \in P$ , una colonna  $T_k$  con  $k \neq j$  tale che  $T_k$  è il dominio di  $p_j$ , e un tipo  $x_j \in X$  definito come il tipo dei valori in  $T_j$ .

---

\*Si osserva che una proprietà  $p$  può essere usata per una o più colonne.

Tramite l'annotazione, dunque, non solo è possibile definire la semantica di una singola colonna ma anche specificare le relazioni tra le diverse colonne. Esistono diverse strategie per annotare i dati di una tabella che identificano diversi modelli di annotazione, tuttavia una definizione rigorosa di esse non è presente nello stato dell'arte. Per questo motivo nel seguito di questo capitolo si cerca di identificare e formalizzare i modelli di annotazione possibili. La definizione dei modelli nel seguito della tesi ha inoltre un duplice scopo: condurre un confronto tra i diversi

tool di annotazione disponibili e definire in maniera non ambigua i tipi di grafo RDF che è possibile derivare dall'annotazione di una tabella. Ciascun modello di annotazione produce, infatti, dalla tabella T un grafo G definito come segue.

#### Definizione del grafo risultante dall'annotazione

L'annotazione specifica tutte le informazioni necessarie a definire dei mapping in un linguaggio dato che permettono di estrarre un grafo RDF dalla tabella T sulla base delle annotazioni specificate. In particolare G è composto dall'unione di tanti alberi  $G_i$  quante sono le righe della tabella T. Ciascun albero è definito come  $G_i = \langle V, E, type, prop \rangle$  dove:

- V è l'insieme dei nodi dell'albero, e in particolare  $V \subseteq T_i$
- E è l'insieme degli archi dell'albero, e in particolare ciascun arco  $e$  è definito come  $e = (u, v)$  con  $u, v \in V$
- $type$  è una funzione totale definita come  $type: V \rightarrow X$
- $prop$  è una funzione totale definita come  $prop: E \rightarrow P$
- $v_s$  è il nodo radice di  $G_i$  ed è l'i-esimo valore estratto dalla colonna soggetto  $T_s$
- $type(v_s) \in C$ , mentre  $type(v_j) \in X$
- Per ogni  $G_i$ ,  $type(v_s)$  è sempre lo stesso
- Per ogni arco  $e \in E$ ,  $type(u) \in C$  e  $type(v) \in X$

Nel corso di questo capitolo verranno presentati i modelli che sono stati individuati nell'annotazione delle tabelle. In particolare ciascun modello sarà descritto nei termini dei vincoli che impone nella creazione del generico grafo G. I modelli identificati sono i seguenti:

- *Star*
- *Composed Star*
- *Snow-Flake*
- *Simple Snow-Flake*
- *Hidden-subject*

Infine, gli output del processo di annotazione che è possibile produrre tramite STAN sono tre:

- $M$ , un insieme di mapping eseguibili definiti in funzione delle annotazioni e specificati con un linguaggio formale  $L$ .
- $\Sigma$ , un insieme di triple risultante dall'applicazione dei mapping alle righe della tabella  $T$ .
- $\bar{O}$ , una ontologia locale definita come un insieme di assiomi  $\mathcal{A}$  su  $P$  e  $X$  tali che, per ogni annotazione di una colonna con una proprietà  $p_j$  con dominio  $c$  e valori di tipo  $x$ , vale la formula  $\forall k (c(k) \rightarrow \exists h (p_j(k,h)) \wedge x(h))$ . La formula in questione, rappresentata in First Order Logic per semplicità, equivale a un assioma di OWL2 [24] che ha la forma  $\exists c \sqsubseteq p_j . x$ .

### 3.2 Modello Star

Il modello *Star* si applica alle tabelle in cui una particolare colonna viene vista come soggetto e le altre descrivono gli oggetti contenuti in essa. Questa struttura implica che le uniche relazioni esistenti si dipartono dalla colonna soggetto andando di fatto a formare una struttura a stella. Infatti, il soggetto della tabella è unico e non esistono relazioni che connettono le rimanenti colonne. Inoltre, le proprietà definite in questo particolare modello sono tutte delle *Datatype Property*. Dunque l'albero costruito per ciascuna riga della tabella derivante da questo tipo di mappatura risulta caratterizzato da un unico nodo centrale, il soggetto, e i restanti nodi foglia di tipo *literal*.

Data la definizione di  $G_i$  il modello *Star* possiede le seguenti caratteristiche:

- La profondità massima di  $G_i$  è 1.
- $prop$  è definita come  $prop: E \rightarrow P_d$ .
- Per ogni  $v \in V$ , escluso  $v_s$ ,  $type(v) \in D$ .

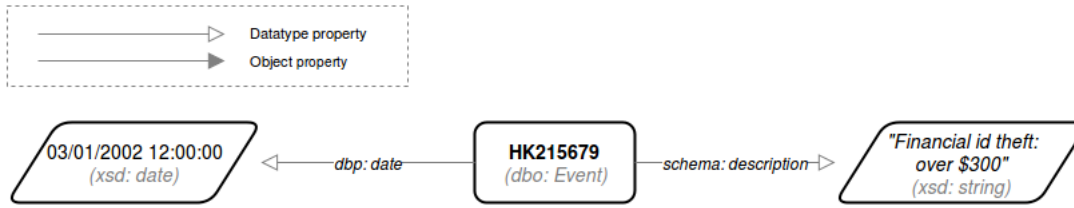
Considerando ad esempio è la tabella 3.1 in cui la colonna *Case Number* rappresenta il soggetto dell'intera tabella. Sfruttando il modello *Star* e le ontologie di Schema.org e DBPedia, è possibile costruire un albero come quello modellato in figura 3.2.



### 3.3. MODELLO COMPOSED STAR

Case Number	Date	Description
HH215971	03/01/2002 13:01:00	Armed: knife/cutting instrument
HH216038	03/01/2002 00:15:00	Poss: cannabis 30gms or less
HK215679	03/01/2002 12:00:00	Financial id theft: over \$300

**Tabella 3.1:** Estratto della tabella relativa ai crimini commessi a Chicago.



**Figura 3.2:** Esempio di modello Star.

### 3.3 Modello Composed Star

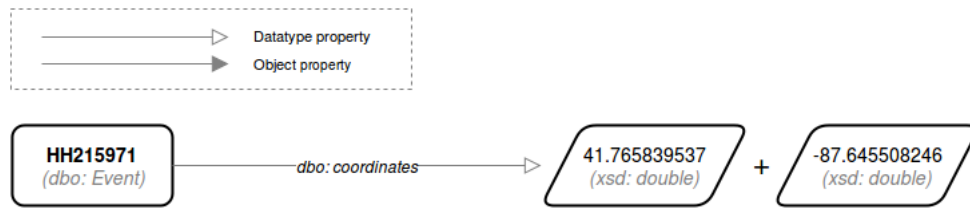
Il modello *Composed Star* è un caso particolare del modello *Star* nel quale si desidera comporre più colonne per crearne una sola. A volte, infatti, si ha la necessità di definire un concetto a partire dall'aggregazione di più colonne. In questo modello, infatti, si definiscono proprietà il cui oggetto è formato dalla composizione di due o più colonne. Per esempio è possibile che nella tabella 3.2 si vogliano considerare congiuntamente le colonne *Latitude* e *Longitude*. Il modello *Composed Star*, come il modello *Star*, si basa solo su *DatatypeProperty* per cui per comporre i valori di più colonne si può sfruttare semplicemente la concatenazione tra stringhe (figura 3.3).

Data la definizione di  $G_i$  il modello *Composed Star* possiede le seguenti caratteristiche:

- La profondità massima di  $G_i$  è 1.
- $prop$  è definita come  $prop: E \rightarrow P_d$ .
- Per ogni  $v \in V$ , escluso  $v_s$ ,  $type(v) \in D$ .
- Ogni nodo  $v \in V$ , escluso  $v_s$ , può essere definito come la composizione di due o più valori della riga  $T_i$ .

Case Number	Latitude	Longitude
HH215971	41.765839537	-87.645508246
HH216038	41.906690424	-87.640736236
HK215679	42.006627736	-87.675994307

**Tabella 3.2:** Estratto della tabella relativa ai crimini commessi a Chicago con relative coordinate geografiche.



**Figura 3.3:** Esempio di modello Composed Star.

### 3.4 Modello Snow-Flake

A differenza dei modelli di tipo *Star* un modello *Snow-Flake* esprime in modo più articolato la semantica di una tabella. Se nel caso di un modello *Star* è possibile definire solo la colonna soggetto come colonna da cui si dipartono delle proprietà nei modelli *Snow-Flake* se ne possono definire diverse. La differenza principale con i modelli *Star* è l'utilizzo combinato di *DatatypeProperty* e *ObjectProperty*. Sfruttando le *ObjectProperty*, infatti, è possibile considerare gli elementi di una colonna come entità e non più solo come valori letterali. Ciascuna valore della tabella trattato come un'entità viene trasformato in uno URI. Inoltre, viene automaticamente arricchito con due proprietà, *rdf:type* e *rdfs:label*, che ne descrivono il tipo e la label. L'albero prodotto tramite questo tipo di modello dunque può contenere un numero variabile di nodi intermedi caratterizzati a loro volta da proprie proprietà. Un esempio è la mappatura in figura 3.4 della tabella 3.3.

Data la definizione di  $G_i$  il modello *Snow-Flake* possiede le seguenti caratteristiche:

- La profondità massima di  $G_i$  non è limitata superiormente.
- Per ogni  $v \in V$  tale che  $type(v) \in C$ ,  $v$  è descritto ulteriormente dalla proprietà *rdfs:label* che lo associa alla relativa cella  $t_{i,j}$  della tabella.

### 3.5. MODELLO SIMPLE SNOW-FLAKE

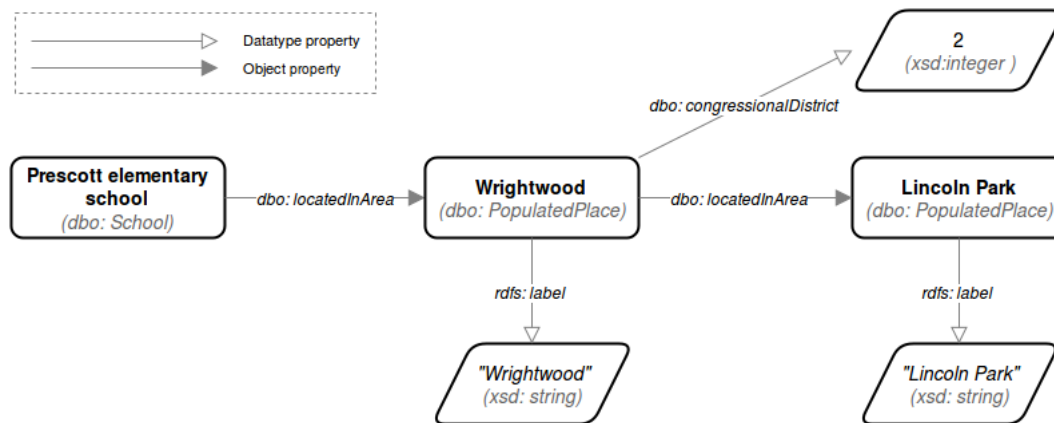


Figura 3.4: Esempio di modello Snow-Flake.

School Name	Community Area	Neighborhood	Congr. District
Prescott elementary school	Lincoln Park	Wrightwood	2
Edgebrook elementary school	Forest Glen	Edgebrook	1
Northside Prep high school	North Park	River's Edge	5

Tabella 3.3: Estratto della tabella relativa alle scuole di Chicago.

### 3.5 Modello Simple Snow-Flake

Il modello *Simple Snow-Flake* è un caso semplificato del modello *Snow-Flake*. In questo modello, infatti, la profondità dell'albero risultante dalla mappatura è limitato a 2 mentre nel modello *Snow-Flake* non ci sono limiti. Come nel modello *Snow-Flake* è possibile utilizzare le *ObjectProperty* per annotare gli elementi di una colonna come entità. Tuttavia, le colonne annotate da una *Object Property* non possiedono altre proprietà uscenti a parte le due proprietà uscenti di default *rdfs:label* e *rdf:type*.

Data la definizione di  $G_i$  il modello *Simple Snow-Flake* possiede le seguenti caratteristiche:

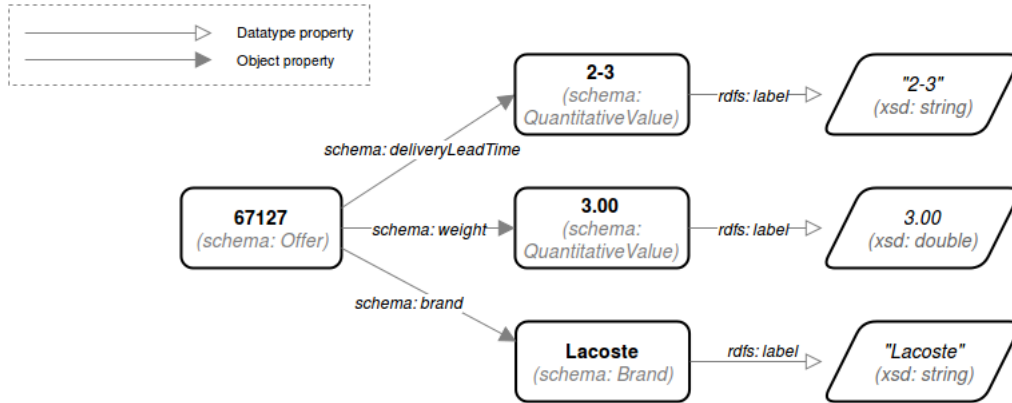
- La profondità massima di  $G_i$  è 2.

- Per ogni  $v \in V$  tale che  $type(v) \in C$ ,  $v$  è descritto ulteriormente dalla proprietà  $rdfs:label$  che lo associa alla relativa cella  $t_{i,j}$  della tabella.

In figura 3.5, ad esempio, si mappa la colonna *Brand* della tabella 3.4 con la proprietà *brand* di Schema.org che è una *ObjectProperty*. Gli elementi della colonna di conseguenza saranno tutti di tipo *schema:Brand* e ciascuno di essi sarà trasformato in uno URI con associata una *label*.

Offer ID	Brand	Shipping Time	Weight
90183	Paco Rabanne	2-3	2.50
67127	Lacoste	2-3	3.00
09972	Armani	2-3	3.00

**Tabella 3.4:** Estratto di uno dei listini commerciali di 7Pixel in cui per ciascuna offerta vengono specificati la marca, i tempi di spedizione e il peso.



**Figura 3.5:** Esempio di modello Simple Snow-Flake.

### 3.6 Modello Hidden-subject

Il modello *Hidden-subject* è un caso particolare che estende ciascuno dei modelli presentati fino ad ora. In particolare questo tipo di modello è pensato per quelle tabelle in cui non compare esplicitamente una colonna soggetto ma le colonne della tabella fanno riferimento a degli oggetti principali impliciti nella descrizione della tabella. In questi casi dunque si considera un'estensione della tabella. In particolare oltre alle colonne esplicite della tabella se ne considera una aggiuntiva  $T_h$  che non compare nella tabella e che sarà considerata come la colonna soggetto della tabella. Come per gli altri modelli i valori contenuti nella colonna soggetto sono trasformati in entità a cui è associato un tipo in  $C$ . Le restanti colonne invece possono essere modellate come uno dei modelli descritti in precedenza.

### 3.6. MODELLO HIDDEN-SUBJECT

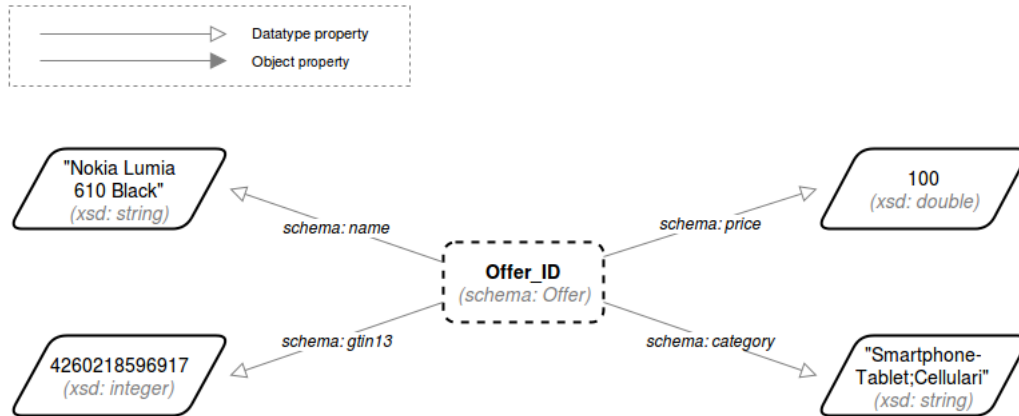
Data la definizione di  $G_i$  il modello *Hidden-subject* possiede le seguenti caratteristiche:

- $v_h$  è il nodo soggetto di  $G_i$ .
- $v_h \in T_h$  con  $T_h \not\subseteq T$ .
- $type(v_h) \in C$ .

Un esempio di tabella su cui applicare il modello *Hidden-subject* è la tabella 3.5 nella quale si descrivono gli attributi di offerte di prodotti commerciali. In questo caso ciascuna colonna caratterizza i campi di un'offerta sebbene nella tabella non sia presente alcun riferimento al concetto di offerta. Una possibile annotazione di questa tabella, sfruttando l'ontologia di Schema.org è quella modellata in figura 3.6. In questo caso quando vengono generate le triple risultanti dall'annotazione è necessario che per ciascuna riga venga creata un'entità univoca che possa essere utilizzata come soggetto delle altre colonne.

Product	EAN code	Price	Category
Nokia Lumia 610 Black	4260218596917	100	Smartphone-Tablet;Cellulari
Navigatore Tom Tom	5060213363939	139.9	Navigatore GPS;
Digital camera Nilox	4260155913297	46.8	Fotocamere Digitali;

**Tabella 3.5:** Estratto di uno dei listini commerciali di 7Pixel in cui il soggetto è implicito.



**Figura 3.6:** Esempio di modello Hidden-subject.



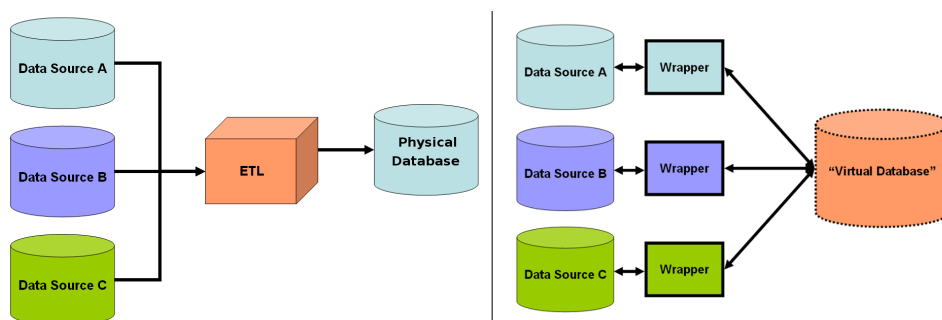
## Capitolo 4

# Stato dell'arte

### 4.1 Data Integration

Il problema della combinazione di dati provenienti da diverse sorgenti è il focus dell'area di ricerca della Data Integration [16]. Questo problema, infatti, è importante in molte applicazioni in cui si necessita di una vista unificata su un insieme di dati. In questo modo in fase di ricerca diventa possibile effettuare una query nei termini di un dataset integrato e non singolarmente sulle sorgenti dei dati. I principali paradigmi di integrazione di dati esistenti in letteratura sono due:

- *Fisico*. Si crea fisicamente una nuova base di dati tramite un processo di ETL (estrazione, trasformazione e caricamento) importando i dati da due o più sorgenti. Alla fine di questo processo la base di dati risultante può avere una struttura completamente diversa da quelle di partenza.
- *Virtuale*. Gli schemi locali delle sorgenti vengono mappati ad uno schema globale. In questo modo i dati continuano a risiedere nelle sorgenti e si crea un nuovo dataset virtuale. L'accesso ai dati in fase di query è poi mediato da un agente software che si preoccupa di rendere trasparente la frammentazione dei dati sorgenti e di tradurre le query dallo schema globale a quelli locali.



**Figura 4.1:** Schematizzazione degli approcci fisico e virtuale di Data Integration.

L'approccio fisico è utilizzato specialmente nei sistemi di data warehouse [16]. Tuttavia si rivela poco efficace per dataset che vengono frequentemente aggiornati. In questi casi infatti sarebbe richiesta la continua esecuzione di processi di ETL per sincronizzare i dati. Tramite l'approccio virtuale, invece, si può fornire l'accesso in tempo reale ai dati passando attraverso un mediatore. Il mediatore si occupa di tradurre le query in query specifiche sugli schemi delle sorgenti di dati basandosi sui mapping definiti tra essi e uno schema globale. Tali mapping possono essere definiti in due modi [30]: come mapping dalle entità dello schema globale verso quelle degli schemi locali (approccio *Global As View* (GAV)) o viceversa (approccio *Local As View* (LAV)).

Formalmente si può quindi definire un sistema di Data Integration come una tripla  $\langle G, S, M \rangle$  dove  $G$  è lo schema globale,  $S$  è l'insieme delle sorgenti di dati e i relativi schemi locali, e  $M$  è l'insieme di mapping che legano le sorgenti di dati allo schema globale. La definizione di tali mapping, tuttavia presenta alcune complessità legate all'esistenza di eterogeneità semantica tra i dati. Le basi di dati infatti sono create per scopi differenti e da persone diverse. Per questo motivo è facile che gli schemi definiti abbiano stili e semantiche differenti. Nel definire quindi questi mapping bisogna prevedere un effort cognitivo considerevole da parte di un essere umano e un effort ancora maggiore se si vuole affidare questo compito ad una macchina. Principalmente gli approcci per definire un mapping si basano su tecniche di matching tra i valori degli schemi (*Name based*) oppure sulla similarità tra le istanze (*Instance based*).

## 4.2 Web Data Integration

Il web rappresenta un caso specifico e più recente in cui si applicano i principi generali appena descritti di Data Integration [31]. Sicuramente il web fornisce una fonte di dati immensa dalla quale possono essere estratte informazioni di grande valore. Tuttavia l'estrazione e l'integrazione di tali informazioni risulta spesso molto complicato per i seguenti motivi:

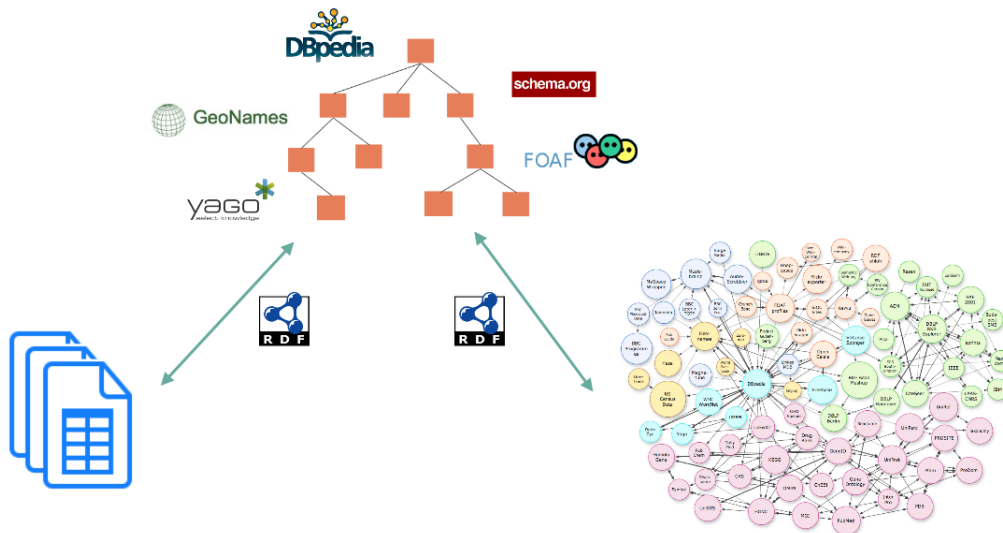
- Enorme eterogeneità di contenuti. Sul web infatti si tratta di qualsiasi argomento con informazioni contrastanti e con diversi livelli di qualità.
- Diversità di formati (tabelle, liste, paragrafi, ...).
- Difficoltà nell'estrazione dell'informazione.
- Assenza di semantica che deve essere ricostruita a partire dal contesto.

I dati presenti sul web sono pensati prevalentemente per essere compresi da esseri umani. Per questo si presta molta attenzione all'impaginazione e allo stile piuttosto che alla strutturazione stessa dei dati. Considerando, ad esempio, le tabelle presenti sul web è facile imbattersi in tabelle ricche di informazioni utili ma in rari casi esse specificano uno schema o la semantica dei dati (esempio in figura



2.1). Inoltre, la maggior parte dei dati presenti sul web è nascosto dietro i form di compilazione presenti nelle pagine web (*Deep web* [22]). Questi dati non sono accessibili, ad esempio, ai comuni motori di ricerca perchè non esistono link a tali pagine.

Per far fronte a tali problemi si è ormai affermato il concetto di Semantic web [39] che fin dalla sua prima teorizzazione nel 2001 [6] si pone l'obiettivo di rendere i dati presenti sul web interpretabili dalle macchine. L'idea base è quella di annotare il contenuto delle pagine web con informazioni semantiche. Ad esempio per specificare che una pagina web parla di un attore annotandone alcuni dati anagrafici e la sua filmografia. Tali annotazioni sono effettuate principalmente sfruttando il linguaggio RDF [40], un linguaggio basato sulla definizione di triple <oggetto, predicato, oggetto>. RDF è un linguaggio pensato per il web, infatti, ciascuna risorsa è definita come uno URI e quindi accessibile sul web, inoltre, RDF consente di definire delle relazioni tra i concetti e questo si sposa alla perfezione con il paradigma dell'ipertesto tipico del mondo del web.



**Figura 4.2:** L'annotazione di dati sul web sfruttando un'ontologia come schema globale

L'annotazione dei dati di fatto corrisponde alla definizione di un mapping verso uno schema globale nella Data Integration classica. Infatti, nella maggior parte dei casi l'annotazione viene eseguita basandosi su ontologie diffuse e condivise (e.g. Schema.org<sup>1</sup>, DBpedia<sup>2</sup>, GoodRelations<sup>3</sup>, ...) che formano di fatto uno schema globale. Sul web esiste già una grande quantità di dati pubblici disponibili in

<sup>1</sup><http://schema.org/>

<sup>2</sup><http://wiki.dbpedia.org/>

<sup>3</sup><http://www.heppnetz.de/projects/goodrelations/>

formato RDF: la Linked Open Data Cloud (LOD) [39] che conta al 2014<sup>4</sup> più di 31 miliardi di triple provenienti da oltre 1000 dataset di diversi domini (e.g. Scienza, Geografia, Musica, ...). Una volta riconciliati i dati presenti sul web ad uno schema globale risulta quindi possibile integrarli con quelli presenti nella LOD (figura 4.2). In particolare è possibile sia arricchire i propri dati con quelli presenti nella LOD che pubblicare i propri dati nella LOD dopo averli modellati in formato RDF.

### 4.3 Table Integration

Come già accennato in precedenza le tabelle sono una delle strutture maggiormente utilizzate sul web per presentare i dati, tuttavia raramente sono accompagnate da una descrizione semantica. Per questo negli ultimi anni la comunità scientifica si è concentrata sul task di annotazione dei dati in formato tabellare presenti sul web. Durante questa tesi dunque è stato realizzato STAN un tool che permette di annotare in modo semi automatico una tabella con i concetti e le proprietà di una o più ontologie. Nello sviluppo del tool ci si è confrontati con alcuni dei tool esistenti che si dedicano allo stesso obiettivo: Karma e Open Refine, che sono analizzati nel dettaglio nella sezione 4.4. Inoltre, all'interno del tool è stato integrato un algoritmo per l'annotazione automatica dei valori di una colonna che utilizza un approccio *instance based* basato sul confronto di un campione dei valori della colonna con i valori di migliaia di proprietà di una knowledge base. Anche in questo caso ci si confronta con gli approcci esistenti in letteratura, in particolare considerando anche quegli algoritmi che non sono integrati in un tool. Gli algoritmi sono analizzati singolarmente nella sezione 4.8. Nelle sezioni 4.5, 4.6 e 4.7, invece, sono descritte in modo generale le tipologie a cui appartengono tali algoritmi che da letteratura sono principalmente tre:

- Approcci basati sull'utilizzo di una base di conoscenza.
- Approcci basati sull'utilizzo di una ontologia di dominio.
- Approcci che non si basano su risorse esterne.

### 4.4 Tools

#### 4.4.1 Karma

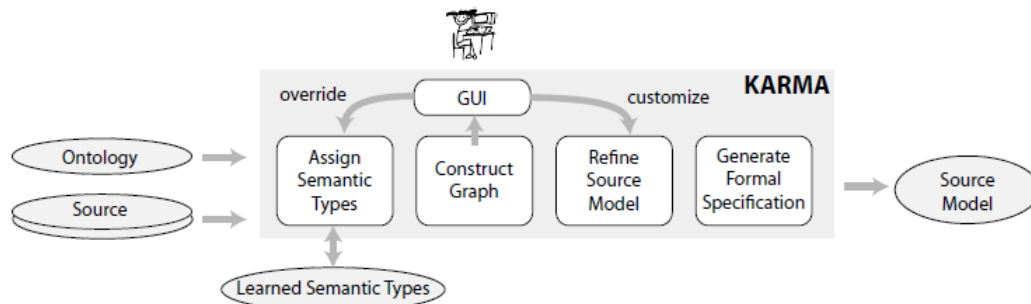
Karma [26] [37] è un tool per effettuare la trasformazione di dati in forma tabellare in dati in formato RDF che si basa sulla presenza di una o più ontologie. In particolare l'obiettivo di Karma è quello di rendere questo procedimento semi automatizzato e comprensibile anche a persone con competenze non tecniche. Il tool infatti genera dei possibili mapping tra la tabella e l'ontologia che poi vengono

---

<sup>4</sup><http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

supervisionati dall'utente tramite l'utilizzo di una interfaccia grafica. L'utente è in grado di indicare i mapping corretti al sistema, e il sistema basandosi su tali correzioni è in grado di imparare a definire nuovi mapping.

Gli input del sistema sono: una o più ontologie, una collezione di dati e un database di tipi semantici. I primi due vengono importati dagli utenti i tipi semantici, invece, sono quelli che il sistema ha imparato a riconoscere in base ai precedenti utilizzi. L'output del sistema infine è un file di mapping che specifica le corrispondenze tra i dati tabellari e l'ontologia. Un output secondario, invece, è il database dei tipi semantici aggiornato e raffinato durante il processo per incorporare i nuovi tipi semantici appresi.



**Figura 4.3:** Il processo di generazione dei mapping tramite il tool Karma (fonte: [26])

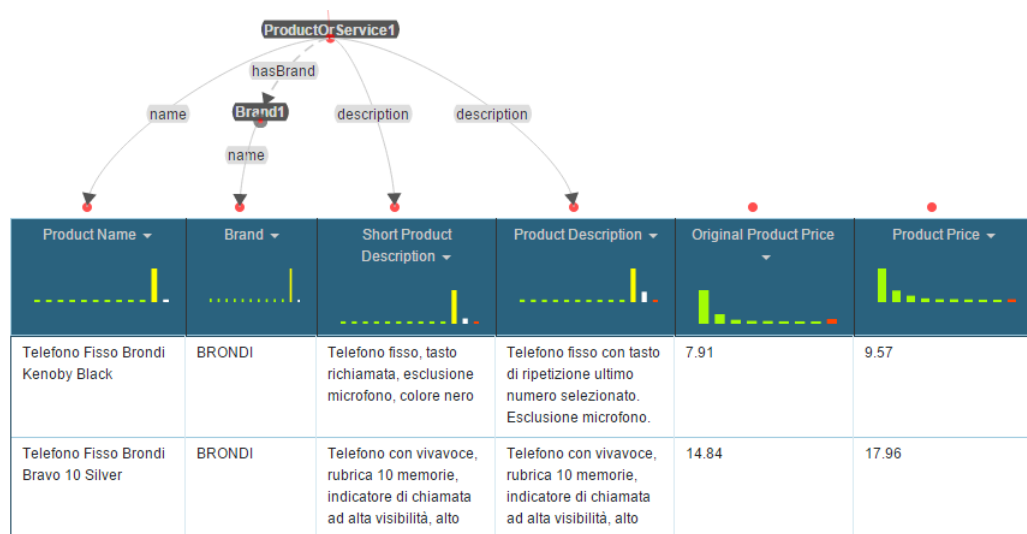
Karma è una applicazione web scritta in Java utilizzabile tramite browser e compatibile con i principali sistemi operativi. Karma è un tool Open-Source con licenza Apache 2<sup>5</sup> per cui è liberamente scaricabile ed utilizzabile. Una volta scaricata è possibile eseguire l'applicazione in locale facendo girare sia la parte server che quella client, oppure è possibile installare Karma su un server e usare l'applicazione da altre macchine. Le principali funzionalità di Karma sono:

- L'importazione di dati in diversi formati: tabelle di database, CSV, JSON, XML, XLS.
- Il caricamento da file di ontologie OWL a scelta dell'utente.
- Alcune operazioni di elaborazione della tabella come la creazione di nuove colonne o righe e la possibilità di splittare una colonna in più colonne.
- La visualizzazione grafica della varianza dei valori all'interno di ciascuna colonna.
- L'annotazione delle colonne supportata da un sistema di auto completamento.
- Il suggerimento nell'annotazione delle colonne basato sui mapping precedentemente definiti.

<sup>5</sup><http://www.apache.org/licenses/LICENSE-2.0>

## CAPITOLO 4. STATO DELL'ARTE

- La visualizzazione strutturata sotto forma di albero dei mapping effettuati tra la tabella e l'ontologia.
- L'export in formato R2RML [12] del mapping tra la tabella e l'ontologia.
- L'export in formato RDF o JSON-LD del dataset dati i mapping.
- La possibilità di effettuare query SPARQL e maneggiare le triple create sfruttando il framework OpenRDF Workbench<sup>6</sup> installato assieme a Karma.



**Figura 4.4:** L'interfaccia del tool Karma

Nello specifico, l'annotazione delle colonne consente di definire relazioni di tipo *Star*, *Simple Snow-Flake* e *Snow-Flake*. In particolare ogni colonna può essere annotata come URI o come property di una classe. Nel primo caso il sistema richiede semplicemente di specificare la classe da assegnare come tipo alla colonna selezionata come URI. Nel secondo caso si deve selezionare una *DatatypeProperty*, tra quelle presenti nell'ontologia importata, e la classe che farà da soggetto di quella property. Una volta selezionata la property il sistema filtra in automatico le classi in modo da riflettere quella che è la struttura dell'ontologia. Vengono infatti mostrate quelle classi che nell'ontologia compaiono come soggetto della property selezionata. Nonostante ciò all'utente è consentito violare la struttura dell'ontologia selezionando una qualsiasi delle altre classi. Questo per permettere una maggior flessibilità in fase di mappatura dei dati.

<sup>6</sup><http://rdf4j.org/>

#### 4.4.2 Open Refine

Open Refine<sup>7</sup> (in precedenza Google Refine) è uno strumento per lavorare con dati grezzi in formato tabellare. Il tool consente agli utenti di effettuare diverse operazioni sui dati tramite una interfaccia grafica. Ad esempio il tool consente di trasformare i dati in diversi formati, estenderli con informazioni presenti sul web e integrarli con le informazioni contenute in basi di conoscenza online.

Il tool è Open Source e liberamente scaricabile. Open Refine è una applicazione web, ma differentemente dalla maggior parte delle applicazioni web, è pensata per essere eseguita sulla propria macchina e utilizzata da un solo utente. Questo per garantire la possibilità di maneggiare dati sensibili sul proprio computer senza doverli caricare su internet. La parte server dell'applicazione mantiene lo stato dei dati, mentre la parte client è dedicata all'interfaccia utente. La comunicazione tra client e server avviene tramite chiamate ajax per effettuare modifiche ai dati e per recuperare lo stato corrente del server. Le principali funzionalità di Open Refine sono:

- L'importazione di dati in diversi formati: CSV, JSON, XML, XLS, Google Spreadsheet, Google fusion tables.
- L'individuazione di elementi simili all'interno delle colonne per poterli normalizzare.
- La modifica di più celle applicando una regola di trasformazione specificata con un linguaggio apposta messo a disposizione dal tool.
- La creazione di nuove colonne a partire da colonne esistenti.
- La visualizzazione della distribuzione dei valori numerici presenti in una colonna.
- La riconciliazione semi-automatica delle celle di una colonna con le entità contenute in una base di conoscenza tramite l'interrogazione via API di servizi esterni.
- L'aggiunta automatica di colonne contenenti dati aggiuntivi prelevati da una base di conoscenza. Una volta riconciliati le celle di una colonna a entità di una base di conoscenza è possibile estenderli con informazioni aggiuntive sotto forma di nuove colonne. Ad esempio, data una colonna contenente entità di tipo *Film* è possibile aggiungere e riempire in automatico la colonna *Directed by*.
- L'allineamento dello schema della tabella con l'ontologia di Freebase definendo le proprietà esistenti tra le colonne.

---

<sup>7</sup><http://openrefine.org/>

A differenza di Karma (sezione 4.4.1), Open Refine non integra un algoritmo di annotazione delle colonne. Infatti l'allineamento dello schema della tabella ad una ontologia viene effettuato a mano tramite interfaccia. In questa fase è possibile definire modelli di annotazione di tipo *Star*, *Snow-Flake* e *Simple Snow-Flake*. Open Refine permette, infatti, di annotare le relazioni che collegano una colonna *subject* alle altre. In particolare è possibile selezionare sia le *DatatypeProperty* che le *ObjectProperty* di una ontologia con l'ausilio di un sistema di autocompletamento. Nativamente, Open Refine, supporta soltanto Freebase come base di conoscenza per definire la semantica della tabella. Tuttavia esistono estensioni che permettono di utilizzare anche i concetti e le relazioni di altre ontologie. Un esempio è *RDF Refine*<sup>8</sup> che implementa l'ontologia di DBPedia oltre ad aggiungere la funzionalità di export del dataset in formato RDF.

La funzionalità di riconciliazione semi-automatica dei dati di una colonna, invece, tenta di matchare singolarmente tutti gli elementi di una colonna con le entità di una base di conoscenza. Ad esempio, nel caso venga utilizzato Freebase, identificata una colonna contenente dei film la stringa "Ocean's Eleven" verrà associata all'entità [http://www.freebase.com/view/en/oceans\\_eleven](http://www.freebase.com/view/en/oceans_eleven) di Freebase. Quello di Freebase è il servizio di riconciliazione di default messo a disposizione da Open Refine ma l'utente ha la possibilità di scegliere uno tra i servizi di riconciliazione esterni esistenti associati a basi di conoscenza che si occupano di riconciliare un insieme di valori testuali con le entità della base di conoscenza a cui fanno riferimento. I servizi di riconciliazione citati nella documentazione sono:

- DBpedia
- Reconcile-csv
- Nomenklatura
- SPARQL endpoints
- VIVO Scientific Collaboration Platform
- FundRef
- JournalTOCs
- VIAF
- FAST (Faceted Application of Subject Terminology)
- Library of Congress Subject Headings
- OpenCorporates
- Ordnance Survey

---

<sup>8</sup><http://refine.deri.ie/>

## 4.4. TOOLS

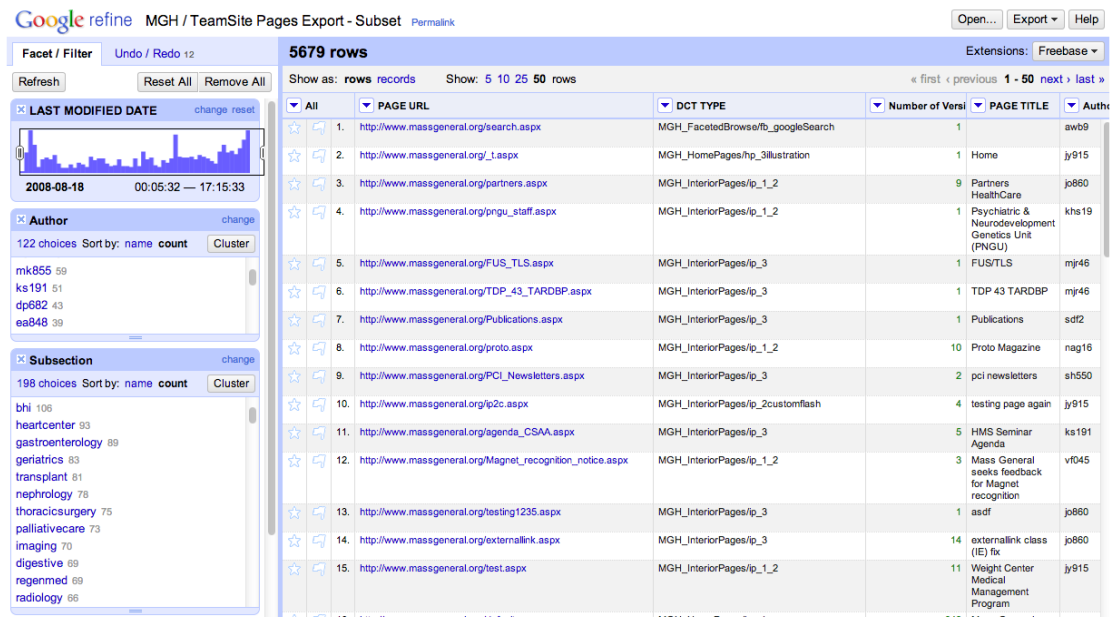


Figura 4.5: L'interfaccia del tool Open Refine.

#### 4.4.3 Confronto delle funzionalità

Funzionalità	Karma	Open Refine
Import tabelle	CSV, XML, JSON, XLS	CSV, XML, JSON, XLS, Google Spreadsheet, Google fusion tables
Import ontologie	✓	✗
Operazioni di modifica sulla tabella	✓	✓
Clustering sui valori di una colonna	✗	✓
Visualizzazione varianza valori colonna	✓	✓
Autocomplete in fase di annotazione	✓	✓
Suggerimento tipo della colonna	✓	✗
Riconciliazione automatica delle entità	✗	✓
Aggiunta automatica di nuove colonne	✗	✓
Export mapping	✓	✗
Export triple RDF	✓	✓
SPARQL endpoint	✓	✗

**Tabella 4.1:** Tabella di confronto tra le funzionalità di Karma e Open Refine.



## 4.5 Integrazione sfruttando una knowledge-base

Gli approcci appartenenti a questa tipologia di integrazione fanno affidamento su di una base di conoscenza esterna per effettuare la riconciliazione semantica dei dati delle tabelle da integrare. La base di conoscenza, in particolare, viene sfruttata per annotare le colonne delle tabelle con le proprietà ed i concetti presenti in essa. In questo modo è possibile successivamente individuare le relazioni esistenti tra le diverse colonne. Le basi di conoscenza più famose e quelle utilizzate negli approcci analizzati sono:

- DBpedia<sup>9</sup>, un progetto Open-Source per l'estrazione di informazioni strutturate da Wikipedia<sup>10</sup> e il riutilizzo di esse sul web.
- Yago<sup>11</sup>, una base di conoscenza derivata manualmente da Wikipedia, WordNet<sup>12</sup> e GeoNames<sup>13</sup>.
- Freebase<sup>14</sup>, una collezione di dati online composta principalmente grazie al contributo dei membri della sua comunità e acquisita nel 2010 da Google.
- Wikitology<sup>15</sup>, una base di conoscenza ibrida ottenuta combinando informazioni strutturate e non da Wikipedia, DBpedia e Yago.

Astraendo dagli approcci analizzati per questa tipologia [33] [43] si identificano cinque fasi principali nel processo di integrazione delle tabelle in input:

- Pre-processing delle tabelle durante il quale vengono effettuate ad esempio operazioni di campionamento, riconoscimento dei valori testuali/numerici, individuazione di acronimi.
- Generazione di  $n$  tipi candidati per l'annotazione di ciascuna colonna.
- Ordinamento e selezione dei  $k$  (con  $k < n$ ) tipi più rappresentativi per ciascuna colonna.
- Selezione delle relazioni esistenti tra le colonne.
- Export dei dati prodotti in formato RDF.

La fase fondamentale del processo su cui si differenziano gli approcci analizzati è sicuramente la generazione dei tipi candidati per l'annotazione delle colonne. In generale quello che si cerca di fare è di definire una funzione di similarità  $f=<V$ ,

---

<sup>9</sup><http://wiki.dbpedia.org/>

<sup>10</sup><https://en.wikipedia.org/>

<sup>11</sup>[www.mpi-inf.mpg.de/yago/](http://www.mpi-inf.mpg.de/yago/)

<sup>12</sup><http://wordnet.princeton.edu/>

<sup>13</sup><http://www.geonames.org/>

<sup>14</sup><https://www.freebase.com/>

<sup>15</sup><http://ebiquity.umbc.edu/project/html/id/83/Wikitology>

$E$  tra le celle della colonna ( $V$ ) e le entità presenti nella base di conoscenza ( $E$ ). In questo modo diventa possibile individuare per ciascuna cella della colonna  $v_i \in V$  un gruppo di entità della base di conoscenza  $E_i \subset E$  che potenzialmente la rappresentino. A partire poi dall'insieme di  $E' = E_1 \cap E_2 \dots \cap E_n$  e dal numero di occorrenze di ciascuna entità è possibile cercare di assumere quale tipo le rappresenta meglio. Questo viene fatto considerando le classi che rappresentano le entità contenute in  $E'$ . Tra esse vengono selezionate le prime  $k$  per essere utilizzate come tipo della colonna.

Ciascun approccio presenta delle peculiarità durante questa fase. In [33], ad esempio, ciascuna tabella viene interpretata come una rete di Markov e si sfruttano i *Probabilistic Graphical Models* [27] in fase di inferenza per individuare i tipi candidati più rappresentativi. Questo approccio, tuttavia, si rivela poco efficace per quelle colonne che contengono valori definiti *literal constants* ossia misure, numeri, ecc.. Questo ha portato infatti ad escludere tali colonne dal processo di annotazione.

Il processo proposto in [43], invece, prevede di non considerare tutte le celle della colonna in modo esaustivo ma solamente un sottoinsieme di esse. La dimensione del sottoinsieme è variabile, infatti, vengono selezionati i primi  $k$  valori che soddisfano una certa soglia di affidabilità nell'assegnazione di un tipo alla colonna. Questo, tra l'altro, consente di rendere il meccanismo più efficiente in termini di tempo.

Spesso, infine, la funzione di similarità  $f$  utilizza come input alcune informazioni di contesto aggiuntive oltre ai soli valori della colonna. Questo deriva dall'intuizione che nell'intorno della tabella, specialmente per tabelle sul web, possono essere presenti delle utili informazioni riguardanti la semantica ed i contenuti di essa. Ad esempio in [33] vengono utilizzate come informazioni di contesto l'header della colonna e i restanti valori contenuti nella stessa riga della cella in analisi. In [43], invece, si analizzano le informazioni presenti nella pagina web dalla quale è stata estratta la tabella. Ad esempio la didascalia, il titolo della pagina, i paragrafi vicini alla tabella e alcune eventuali annotazioni.

## 4.6 Integrazione sfruttando una ontologia di dominio

Gli approcci di questa tipologia sfruttano risorse ontologiche specifiche di un determinato dominio per poter integrare i dati di diverse tabelle. Questo comporta che gli approcci di questo tipo non si prefiggono l'obiettivo di annotare tutte le tabelle presenti sul web, ma in particolare quelle che sono estratte da documenti identificati come inerenti ad uno specifico dominio.

L'approccio classico nell'integrazione di dati adottato nei sistemi di gestione di database relazionali prevede l'utilizzo di *global as view* (GAV) e *local as view* (LAV) [30]. In questo modo è possibile definire dei mapping che traducono in modo automatico le query effettuate sullo schema globale in query sugli schemi locali. Nel Semantic Web [39] invece quello che si cerca di fare è definire un mapping

#### 4.6. INTEGRAZIONE SFRUTTANDO UNA ONTOLOGIA DI DOMINIO

tra gli schemi delle tabelle e i concetti e le relazioni di una ontologia. In questo modo è possibile descrivere più tabelle in termini di una stessa ontologia per poi integrarne i dati. Per definire uno schema di qualità per un dataset e rendere i dati di esso interoperabili con altri dati pubblicati sul web i passi consigliati da seguire sono [2]:

- Comprendere a fondo i tipi delle entità descritte nel dataset e le loro relazioni.
- Scegliere se sia più opportuno creare delle nuove classi o proprietà piuttosto che riutilizzare ontologie già esistenti.
- Nel caso si decida di creare nuove classi/proprietà allinearle a ontologie già esistenti.

Inoltre esistono due strategie principali che possono essere adottate nella pubblicazione di un nuovo dataset riguardanti il riutilizzo di ontologie esistenti [2]:

- *Maximum reuse strategy*: applicando questa strategia si cerca di includere il maggior numero possibile di predicati provenienti da ontologie esistenti e popolari. Questa strategia considera affidabili la maggior parte delle ontologie e vocabolari esistenti. L'obiettivo è quello di garantire la massima interoperabilità per la futura integrazione con altri dataset.
- *Minimum reuse strategy*: al contrario l'applicazione di questa strategia comporta una minima presenza di termini e predicati già esistenti nella definizione dello schema. L'obiettivo di questa strategia è quello di aderire rigorosamente alla specifica nomenclatura di un dominio.

L'obiettivo degli approcci che sfruttano una ontologia di dominio per l'integrazione di dati tabellari è principalmente quello di produrre un mapping tra i dati della tabella e l'ontologia. Una volta generato il mapping diventa quindi possibile trasformare la tabella nei termini dell'ontologia utilizzando un formato che faciliti l'integrazione ed il riutilizzo: nella maggior parte dei casi RDF. Esistono diversi tool e linguaggi che consentono di effettuare manualmente tale mappatura come ad esempio D2R<sup>16</sup>, R2RML [12] e R2R<sup>17</sup>. Il contributo degli approcci analizzati, tuttavia, consiste nel cercare di rendere questo procedimento automatico o semi-automatico.

Uno dei temi più importanti emersi dall'analisi di questi approcci è la distinzione tra l'annotazione di dati testuali e numerici. Infatti secondo un'analisi del 2014 [38] circa il 40% delle tabelle presenti sul web contiene dati numerici. L'annotazione dei dati numerici è quindi un tema molto importante. Tuttavia presenta delle difficoltà intrinseche legate all'ambiguità e talvolta all'assenza delle unità di misura, elemento principale utilizzato per riconoscere il tipo di un dato numerico. Questo ha portato spesso in letteratura a trattare il problema con approcci appositi che

---

<sup>16</sup><http://d2rq.org/d2r-server>

<sup>17</sup><http://wifo5-03.informatik.uni-mannheim.de/bizer/r2r/>

differiscono da quelli studiati per i dati testuali [37] [7] [38].

In Karma [26] [37], ad esempio, le colonne di una tabella sono distinte in base al loro contenuto. A seconda che esse siano giudicate come testuali o numeriche il tool seleziona l'algoritmo migliore per effettuare l'annotazione. In particolare per annotare le colonne testuali Karma sfrutta la *Cosine similarity* e la funzione di peso TF-IDF [32] mentre per le colonne numeriche si analizza la distribuzione dei valori numerici applicando il test di verifica statistico di Kolmogorov-Smirnov [29]. Altri approcci, invece, come ONDINE [7] e QEWT [38] si basano sulla costruzione di una risorsa ontologica propria con cui effettuare in seguito l'annotazione dei dati. In entrambi i casi si presta particolare attenzione ai valori numerici. In ONDINE vengono confrontate le unità di misura identificate nelle celle di una colonna con i concetti presenti nella risorsa ontologica. Infatti tale ontologia contiene al suo interno la definizione di alcuni concetti relativi alle unità di misura.

In QEWT, invece, si mira a realizzare un sistema che si focalizza esclusivamente sull'annotazione di dati numerici. Per questo la risorsa ontologica creata contiene solamente quantità (e.g. *Length*, *Area*) e unità di misura (e.g. *km/h*, *kg*, *\$*).

### 4.7 Integrazione senza sfruttare risorse esterne

A differenza degli approcci visti fino ad ora, quelli appartenenti a questa tipologia non si basano su nessuna risorsa esterna per effettuare la riconciliazione dei dati tabellari. Infatti è possibile ricondurre questi approcci al task dello *Schema Matching* [36]. Lo *Schema Matching* si occupa di risolvere il problema di identificare le corrispondenze semantiche tra due tabelle. Questo task nasce in origine per le tabelle relazionali dei sistemi di archiviazione dati e si è sviluppato in seguito nell'ambito delle tabelle presenti sul web. Tuttavia, questo task applicato alle tabelle presenti sul web risulta essere complicato per due motivi principali:

- Incompletezza dei dati. Spesso infatti i dati presenti nelle tabelle sul web contengono solo un numero limitato di informazioni. Per questo può risultare difficile riconoscere che due colonne modellano gli stessi concetti.
- Assenza di metadati. Informazioni come gli header delle colonne o il tipo di dato che esse contengono non sempre sono presenti nelle tabelle presenti sul web.

## 4.8 Approcci

### 4.8.1 Karma

Karma [26] [37] è un tool per effettuare l'annotazione di dati in forma tabellare che si basa sulla presenza di una o più ontologie. In una prima versione del tool [26] la fase di annotazione delle colonne veniva effettuata sfruttando i *conditional random*

*field* (CRF) [28]. Tramite i CRF il tool era in grado di imparare una funzione di labelling che, prendendo in input il nome della colonna e tutti i valori di essa, restituisse il nodo dell'ontologia più rappresentativo per quella colonna.

In un lavoro successivo [37], tuttavia, è stato sperimentato un nuovo approccio di labelling che si è dimostrato essere più efficace. In questo caso la strategia di annotazione differisce a seconda della tipologia di dati presenti nella colonna. In particolare si distinguono tre tipologie di colonne:

- *Testuale*: se la frazione di dati puramente numerici in una colonna è minore del 60%.
- *Numerica*: se la frazione di dati puramente numerici in una colonna è maggiore del 80%.
- *Ibrida*: se la frazione di dati puramente numerici in una colonna è compresa tra il 60% e l'80%.

L'approccio si basa nel trattare i valori di ciascuna colonna precedentemente annotata come un unico *documento* associato ad una label. In fase di annotazione, invece i valori della nuova colonna da annotare sono utilizzati come *query*. A seconda della tipologia di colonna da annotare in fase di query esistono due diverse strategie di annotazione: testuale o numerica. Il discriminante nella scelta tra le due strategie è ancora la frazione di dati puramente numerici. Se tale frazione per i valori della colonna da annotare supera il 70% allora si utilizzerà la strategia numerica, altrimenti quella testuale.

La strategia testuale si basa sul rappresentare ciascun documento come un vettore di termini. A ciascun termine viene assegnato un peso definito tramite la funzione TF-IDF [32] che tiene conto della frequenza del termine nel documento e nella collezione. In questo modo in fase di query è possibile calcolare la *Cosine Similarity* tra il vettore query e i documenti in modo da recuperare i k documenti più significativi e le relative label.

Invece, la strategia numerica si basa sull'analisi della distribuzione dei valori numerici associati ad una label. Quanto più la distribuzione dei valori utilizzati come query è simile alla distribuzione dei valori di uno dei documenti e tanto più è probabile che le due distribuzioni siano associate alla stessa label. In particolare la similarità tra distribuzioni è calcolata tramite il test di verifica statistico di Kolmogorov-Smirnov [29] che verifica la forma delle distribuzioni campionarie e che non è condizionato dalla dimensione del campione.

#### 4.8.2 Table Miner

Table Miner [43] è un sistema dedicato al labelling delle colonne basato sulla base di conoscenza di Freebase. Questo approccio presenta alcune peculiarità. La prima è che per determinare il tipo da assegnare ad una colonna non si tiene conto di tutte le celle appartenenti a tale colonna, ma solo di un suo sottoinsieme. La

seconda è che nell'individuare il tipo da assegnare alla colonna vengono valutate anche altre informazioni presenti nella pagina web dalla quale è stata estratta la tabella. Ad esempio la didascalia, il titolo della pagina, i paragrafi vicini alla tabella e alcune eventuali annotazioni.

Il procedimento utilizzato dal sistema Table Miner per automatizzare l'assegnazione di un tipo alle colonne di una tabella prevede un approccio basato su due fasi. La prima fase, denominata *forward learning*, prevede a sua volta più passaggi. Per prima cosa viene effettuata la ricerca di entità candidate nella base di conoscenza a partire da ogni singola cella della colonna in analisi. La peculiarità di questo approccio in questa fase sta nel non considerare tutte le celle della colonna in modo esaustivo ma solamente un sottoinsieme di esse che consenta di raggiungere una certa soglia di affidabilità. Questo, inoltre, rende il meccanismo più efficiente in termini di tempo. Dopo aver selezionato le entità candidate si procede con la disambiguazione di esse basandosi su un valore di confidenza che considera due fattori: la similarità tra il nome dell'entità e quello della cella, e la similarità tra le entità e le informazioni di contesto estratte dalla pagina web dove è presente la tabella. Successivamente alla fase di disambiguazione si procede con l'ordinamento delle entità candidate. Questi passi vengono ripetuti per ciascuna cella della colonna fino a quando non si converge ad un risultato che supera un certo livello di confidenza. Una volta raggiunta tale soglia questa fase termina anche se non sono state prese in considerazione tutte le celle della colonna. Date le entità candidate, a questo punto, è possibile inferire quale possa essere il tipo più adeguato da assegnare alla colonna.

Durante la seconda fase del procedimento, detta *backward update*, si cerca invece di interpretare le celle rimanenti basandosi sul risultato della fase precedente. Se durante questa seconda fase risulta che il tipo assegnato non è valido per le nuove celle analizzate si aggiorna tale valore candidato e si riesegue questo procedimento.

### 4.8.3 Semantic message passing

Semantic message passing [33] è un algoritmo per inferire la semantica di una tabella sfruttando una base di conoscenza. In particolare questo algoritmo basa la propria inferenza sulle basi di conoscenza di DBpedia, Yago e Wikitology.

Il problema viene suddiviso in quattro fasi. La prima fase prevede un pre-processing delle tabelle durante il quale vengono effettuate operazioni di campionamento delle righe e individuazione di acronimi e valori letterali. In un secondo momento si procede con la ricerca e l'individuazione di un set di valori candidati per gli header delle colonne, i valori delle celle e le relazioni tra le colonne. La fase successiva prevede, invece, l'ordinamento di questi candidati e la conseguente selezione dei valori in modo che risultino essere i più opportuni da assegnare. L'ultimo passo, infine, prevede di utilizzare le relazioni scoperte per generare automaticamente della nuova conoscenza sotto forma di triple RDF in modo che questa possa poi essere archiviata in qualche base di conoscenza.

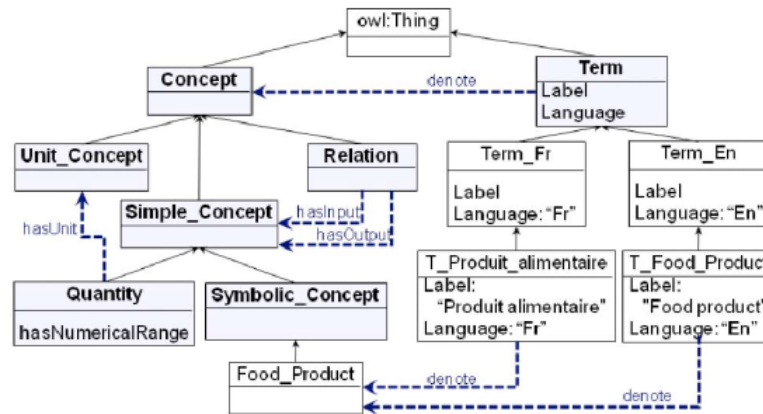
La seconda e la terza fase sono ovviamente quelle centrali e meritano un po' più di attenzione. La fase di individuazione dei candidati, definita *query and rank*, per prima cosa si occupa di trovare delle entità nella base di conoscenza di Wikitology che possano coincidere con i valori contenuti nelle celle di una colonna. Nel portare a termine questo compito oltre al valore stesso di una cella vengono utilizzate come informazioni di contesto l'header della colonna e i restanti valori contenuti nella stessa riga. Da questa analisi vengono escluse quelle colonne che contengono valori definiti *literal constants* ossia misure, numeri, ecc.. perchè considerati difficili da matchare con delle entità della base di conoscenza. Una volta individuate le entità si utilizzano le classi di tali entità come candidati per definire la label da associare a ciascuna colonna. Infine, i candidati per le relazioni tra due colonne sono ricavati cercando sulla base di conoscenza tutte le proprietà che collegano tutte le entità della prima colonna con le entità della seconda colonna. Tutti i candidati individuati fino a questo momento sono poi utilizzati nella successiva fase di inferenza. L'inferenza sfrutta i *Probabilistic Graphical Models* [27] per individuare i valori più appropriati e, in particolare, ciascuna tabella viene vista come una rete di Markov.

#### 4.8.4 ONDINE

Un altro approccio è quello utilizzato in ONDINE [7], un sistema che consente il caricamento di dati in formato tabellare presenti sul web e l'integrazione di essi con dati preesistenti in sistemi di data warehouse. La peculiarità di questo approccio nell'affrontare questo problema risiede nell'utilizzo di una risorsa sia ontologica che terminologica (OTR) (figura 4.6) composta principalmente di due parti:

- Una generica, composta da un insieme di concetti generici utilizzati in fase di Data Integration
- Una specifica, composta da una terminologia ed un insieme di concetti specifici di un preciso dominio

L'OTR si compone di due distinte componenti: una concettuale ed una terminologica. La componente concettuale è una ontologia di concetti, alcuni generici e altri specifici del dominio. La particolarità di tale ontologia è la presenza di concetti relativi alle unità di misura che servono per riconciliare le colonne delle tabelle che contengono dati numerici. La componente terminologica, invece, contiene tutti i termini del dominio. Dove un termine è una sequenza di una o più parole in uno specifico linguaggio a cui si associa una label. I termini sono necessari per denotare i concetti dell'ontologia e, in particolare, un termine può denotare uno o più concetti.



**Figura 4.6:** La rappresentazione dell'OTR con le due componenti: concettuale e terminologica (fonte: [7])

L'annotazione semantica dei dati tabellari segue principalmente le seguenti fasi:

- Distinzione tra le colonne contenenti valori numerici e non.
- Identificazione del concetto da assegnare alle colonne contenenti valori testuali. In questa fase si computa il punteggio di un concetto dell'ontologia per una colonna basandosi sulle similarità tra le celle della colonna e i termini definiti nell'OTR.
- Identificazione del concetto da assegnare alle colonne contenenti valori numerici. In questa fase si confrontano le unità di misura identificate nelle varie celle della colonna con i concetti del dominio per trovare la più verosimile. Inoltre, si analizzano i valori numerici per verificare che l'intervallo numerico in cui si trovano sia compatibile con il concetto a cui si vuole mappare la colonna.
- Identificazione delle relazioni esistenti tra le colonne della tabella.

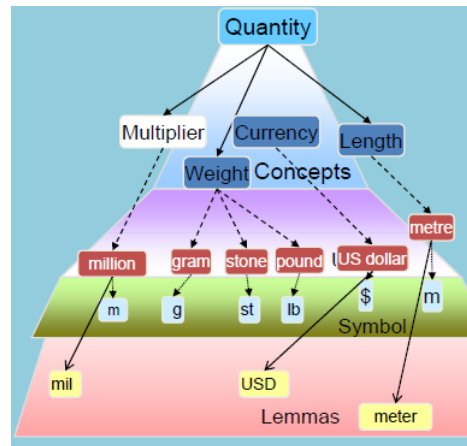
#### 4.8.5 QEWT

L'approccio adottato in QEWT [38] mira a realizzare un sistema per l'interrogazione di dati esclusivamente numerici provenienti da tabelle estratte dal web. Secondo una loro analisi, infatti, circa il 40% delle tabelle presenti sul web contiene dati numerici. Per questo hanno deciso di concentrarsi sull'estrazione e l'annotazione di dati numerici.

I problemi principali relativi all'annotazione di dati numerici sono legati all'ambiguità e talvolta all'assenza delle unità di misura, elemento principale per riconoscere il tipo di un dato numerico. Per questo è stata costruita una ontologia delle quantità e delle unità di misura definita QuTree (figura 4.7). A partire da



Wikipedia sono stati definiti 44 tipi quantitativi (e.g. *Length*, *Area*, *Speed*), e 750 unità di misura (e.g. *km/h*, *kg*, *\$*). Ogni quantità è associata ad una unità di misura canonica e altre unità di misura sono dedotte applicando un fattore di conversione. Ogni unità di misura, inoltre è associata ad una label che ne rappresenta il nome completo. Inoltre nell'ontologia QuTree è presente anche il concetto di moltiplicatore per denotare quelle quantità che sono senza unità di misura ma che sono espresse come istanze ad esempio di centinaia, migliaia, milioni.



**Figura 4.7:** Un frammento di QuTree. L'ontologia delle quantità e delle unità di misura. (fonte: [38])

Data una colonna contenente dati numerici con un header testuale  $H$ , l'obiettivo dell'approccio è quello di annotare la colonna con uno dei tipi presenti in QuTree. Per fare questo si cerca di estrarre da  $H$  le unità di misura, se esistenti, matcharle con le entità presenti in QuTree e infine associarle ai numeri presenti nella colonna. Sono gestiti quattro tipi di unità di misura:

- unità atomiche (e.g. *year*, *meter*, *percentage*)
- unità con moltiplicatori (e.g. *million pounds*)
- unità composte (e.g. *joule/kilogram*)
- liste di unità (e.g. *metre—foot*)

Per annotare le colonne l'approccio prevede l'utilizzo di una grammatica context-free (CFG)<sup>18</sup> per riconoscere le varie componenti dell'header in modo da riuscire ad isolare le unità di misura. Tramite la CFG, infatti, è stato possibile costruire un albero delle componenti lessicali dell'header da cui poi vengono estratte le unità di misura. Una volta estratte le unità di misura è possibile calcolare la similarità tra esse e le entità dell'ontologia QuTree in modo da identificare il tipo da assegnare alla colonna.

<sup>18</sup>[https://www.cs.rochester.edu/~nelson/courses/csc\\_173/grammars/cfg.html](https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html)

### 4.8.6 Giva

Un caso particolare di approccio basato su ontologie di dominio è GIVA [9] [10], un framework per l'integrazione, la visualizzazione e l'analisi di dati geospaziali e temporali. Il problema che viene affrontato in questo caso nasce dal fatto che sul web spesso sono disponibili tabelle ricche di informazioni geospaziali provenienti da diversi domini: urbanistica, trasporti, salute pubblica. Tuttavia spesso queste tabelle sono descritte nei termini di un singolo dominio di interesse legato alla fonte di provenienza. Per molti esperti di dominio, invece, è utile aggregare tali dati per scoprire interessanti correlazioni. I dati geospaziali e temporali presentano diverse tipologie di eterogeneità. Tali eterogeneità, per lo più, fanno riferimento alle dimensioni in cui solitamente sono espresse tali tabelle: lo spazio e il tempo. Le ambiguità spaziali sono legate alla diversa granularità delle informazioni (e.g. stati, città, quartieri). I dati temporali, invece, possono essere espressi in diversi formati e a diversi livelli di precisione. Un dato temporale, ad esempio, può essere preciso (e.g. 22/03/1991), generico (e.g. Settembre) oppure può essere anche un intervallo temporale. Anche in questo caso si utilizzano approcci distinti per annotare i valori testuali e quelli numerici. I valori testuali sono riconciliati ai dati contenuti in GeoNames<sup>19</sup> applicando delle tecniche di *Named Entity Recognition* (NER) [34]. Per riconoscere ed annotare i dati numerici, invece, sono stati utilizzati dei template predefiniti.

### 4.8.7 Framework ibrido macchina-crowdsourcing

L'approccio proposto in [17] si pone a cavallo tra gli approcci basati su una base di conoscenza e quelli che non sfruttano alcuna risorsa esterna. L'idea, infatti, si basa sull'intuizione di effettuare l'integrazione di più tabelle sfruttando sia le potenzialità di calcolo delle macchine che la capacità di ragionamento degli esseri umani. Il sistema sviluppato si pone come un ibrido tra algoritmi automatici e *Wisdom of crowds* [41]. Intuitivamente agli algoritmi viene assegnata la parte "facile" del lavoro, ossia accoppiare colonne e concetti, mentre al crowdsourcing viene assegnato il compito di riconoscere i concetti solo per quelle colonne che per gli algoritmi sono ritenute troppo "difficili". L'approccio base applicato dagli algoritmi automatici è molto simile a quelli già presentati nella sezione 4.5 utilizzando Freebase come base di conoscenza. Tuttavia l'interpretazione delle colonne che il sistema reputa più difficili è affidata al crowdsourcing. Ossia l'assegnazione di un tipo a tali colonne è assegnata a persone che si offrono di eseguire manualmente questo compito ricevendo in cambio un compenso economico. I risultati provenienti dal crowdsourcing, inoltre, sono poi utilizzati dagli algoritmi per le successive annotazioni. Per stabilire quali debbano essere le colonne da annotare manualmente viene utilizzata una funzione di utilità che tiene conto di due fattori principali:

---

<sup>19</sup><http://www.geonames.org/export/codes.html>

- La difficoltà degli algoritmi nell'assegnazione di un tipo  $T$  alla colonna  $A$ . Tale difficoltà è modellata come la quantità di entropia nella distribuzione di probabilità che esiste tra i possibili tipi  $t_1, \dots, t_n$  che possono essere assegnati ad  $A$ . Più le probabilità dei concetti  $t_1, \dots, t_n$  sono simili e più è difficile per il sistema assegnare il corretto concetto alla colonna.
- L'influenza di una colonna nel riconoscere i tipi di altre colonne. Spesso conoscere il tipo a cui è associata una colonna aiuta ad inferire i tipi da associare ad altre colonne. Per questo il sistema cerca di massimizzare il numero di colonne da far valutare in crowdsourcing basandosi anche su questo fattore.

Una volta identificate le colonne difficili vengono pubblicate su Amazon Mechanical Turk (AMT)<sup>20</sup>. Per ognuna delle colonne sono fornite alcune opzioni sotto forma di tipi di Freebase tra cui gli utenti possono scegliere. Ogni colonna deve essere valutata da almeno tre utenti e il tipo che ottiene più voti viene scelto come candidato per tale colonna. Infine vengono computati i risultati prodotti dalla macchina e quelli risultanti dal crowdsourcing in modo da ottenere l'annotazione finale delle colonne con i tipi della base di conoscenza.

---

<sup>20</sup><https://www.mturk.com/>

#### 4.8.8 Confronto

Approccio	Tipologia	Risorse	Colonne testuali	Colonne numeriche
Karma [26] [37]	Ontologia	Ontologie a scelta	TF-IDF	Kolmogorov-Smirnov
Table Miner [43]	Base di conoscenza	Freebase	Frequency similarity	-
Semantic message passing [33]	Base di conoscenza	DBpedia, Yago, Wikitology	Probabilistic graphical models	-
ONDINE [7]	Ontologia	OTR	Terms similarity	Units similarity
QEWI [38]	Ontologia	QuTree	-	Context-free grammar
GIVA [9] [10]	Ontologia	GeoNames, Wikipedia	Named Entity Recognition	Template predefiniti
Framework ibrido macchina-crowdsourcing [17]	Schema matching	Freebase	Frequency similarity	-

**Tabella 4.2:** Tabella riassuntiva degli approcci analizzati.

### 4.9 Analisi conclusive

In questo contesto dunque si inserisce STAN, il tool sviluppato nel corso di questa tesi. STAN si pone come una alternativa ai tool già esistenti proponendo un approccio moderno ed user-friendly che renda il processo di mapping ancora più semplice ed intuitivo. Le principali caratteristiche che differenziano STAN dai tool di annotazione analizzati sono:

- La possibilità di utilizzare una ontologia propria definendola in modo incrementale man mano che si definisce la mappatura della tabella. Nei tool analizzati, infatti, Karma permette l'importazione di una ontologia propria solo se essa è stata definita e formalizzata in precedenza, mentre Open Refine non lo consente in nessun modo.

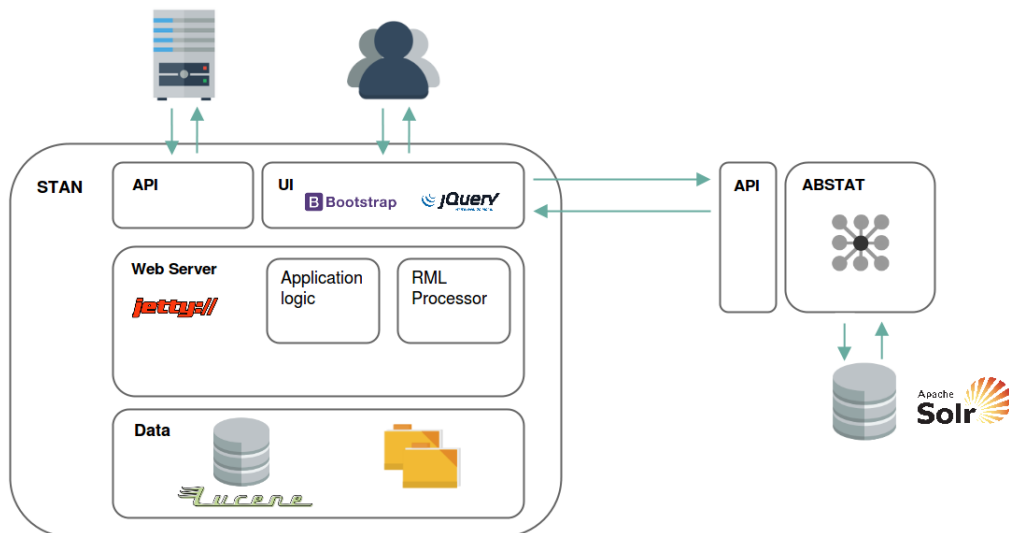
- L'utilizzo di un algoritmo di annotazione che sfrutta una knowledge base a differenza di Karma che utilizza un algoritmo basato su una ontologia di dominio.
- La possibilità di fornire fin dal primo utilizzo suggerimenti in fase di annotazione. Karma invece per sua natura non è in grado di farlo inizialmente dato che i suggerimenti si basano esclusivamente sulle annotazioni effettuate in precedenza.
- L'architettura puramente pensata per il web. Karma ed Open Refine, infatti, sono sviluppate come applicazioni web e sono eseguite all'interno di un browser ma sono pensate per essere utilizzate in locale. Questo implica che le due applicazioni non prevedono alcuna logica di creazione di uno spazio di lavoro privato. STAN, invece, non richiede alcuna installazione né configurazione e consente a ciascun utente di avere il proprio workspace privato online.
- L'esposizione di API pubbliche che rendono STAN un provider di annotazioni di gruppi di valori per applicazioni di terze parti.
- La modularità grazie alla quale è possibile estendere STAN con diversi algoritmi di annotazione. A differenza degli altri tool infatti in STAN è possibile implementare più di un algoritmo di annotazione. L'unico requisito è che i nuovi algoritmi rispettino il formato utilizzato nelle API.



# Capitolo 5

## STAN

### 5.1 Architettura



**Figura 5.1:** Lo schema dell'architettura di STAN.

STAN è una applicazione web in esecuzione su un server remoto e accessibile tramite browser sfruttando il protocollo HTTP. Come nella maggior parte delle applicazioni web STAN si struttura secondo una architettura multi-tier [19], definendosi principalmente in tre layer:

- *Layer di interfaccia*
- *Layer applicativo*
- *Layer dei dati*

Il layer di interfaccia è il punto di accesso tramite il quale è possibile interagire con il sistema STAN. In particolare questo layer espone due interfacce, entrambe basate sul protocollo HTTP, che permettono di comunicare con l'applicazione: una *user-interface* (UI) e un insieme di *application programming interface* (API). La UI permette agli utenti di visualizzare nel proprio browser l'interfaccia grafica del sistema. L'interfaccia in particolare è stata costruita sfruttando due framework noti: Bootstrap<sup>1</sup> e jQuery<sup>2</sup>. Bootstrap è stato utilizzato per il templating grafico degli elementi che costituiscono le pagine dell'applicazione web. jQuery, invece, è stato sfruttato per implementare alcune funzionalità interattive e dinamiche in risposta a precise azioni dell'utente. Se la UI è l'interfaccia tramite la quale comunicano gli utenti, le API sono l'interfaccia tramite la quale comunicano applicazioni e servizi di terze parti. STAN, infatti, espone pubblicamente la funzionalità di annotazione in modo che essa possa essere integrata in sistemi esterni. Quando tali API sono invocate sfruttando il protocollo HTTP, STAN risponde in formato JSON con una lista di possibili annotazioni elaborate sulla base della richiesta effettuata.

Il layer applicativo è di fatto quello che implementa il web server e quello in cui è contenuta tutta la logica applicativa. L'applicazione si basa sul linguaggio Java, implementando in particolare un web server Jetty<sup>3</sup>. Il layer applicativo si occupa di ricevere, elaborare e soddisfare le richieste del client, nel compiere queste operazioni dialoga con il layer di interfaccia tramite il protocollo di comunicazione asincrona AJAX. Il layer applicativo, inoltre, incorpora un engine di generazione di triple (RML Processor [14]) utilizzato per effettuare l'export dei dati contenuti nella tabella sotto forma di triple RDF.

Il layer dei dati, infine, contiene i dati su cui poggia l'applicazione. In particolare l'applicazione gestisce un indice Lucene<sup>4</sup> per ciascuna knowledge-base utilizzata dall'algoritmo di annotazione. Inoltre l'applicazione sfrutta il filesystem della macchina su cui è deployata per archiviare e gestire i file utili a tenere traccia della workspace personale di ciascun utente. In particolare vengono archiviate le tabelle caricate dagli utenti e lo stato della mappatura corrente. In questo modo diventa possibile per gli utenti interrompere la mappatura e riprenderla in seguito dal punto in cui si erano interrotti.

STAN, inoltre, comunica con una applicazione web esterna, ABSTAT [35], che tra le altre cose fornisce una funzionalità di autocompletamento sulle property e sui concetti di una knowledge base. ABSTAT è un'applicazione di *Data Summarization* [23] sviluppata internamente al Dipartimento di Informatica Sistemistica e Comunicazione (DISCO) dell'università Bicocca<sup>5</sup>. L'obiettivo del framework ABSTAT è quello di fornire ai suoi utilizzatori una migliore comprensione di dataset complessi e voluminosi. Per fare ciò ABSTAT estrae dei *summaries* sotto forma di linked data a partire da un modello astratto basato sull'ontologia del dataset.

---

<sup>1</sup><http://getbootstrap.com/>

<sup>2</sup><https://jquery.com/>

<sup>3</sup><http://www.eclipse.org/jetty/>

<sup>4</sup><http://lucene.apache.org/>

<sup>5</sup><http://www.disco.unimib.it/>



I *summaries* sono poi esportati in formato RDF e resi consultabili attraverso un endpoint SPARQL e un'interfaccia web. Inoltre ABSTAT espone delle API di autocompletamento sulle proprietà e sui concetti contenuti nei *summaries* utilizzando Solr<sup>6</sup> come framework di ricerca. Il layer di interfaccia di STAN sfrutta tali API per fornire ai propri utenti un servizio di autocompletamento a supporto delle fasi di annotazione. STAN infatti comunica ad ABSTAT le lettere digitate dall'utente sfruttando il protocollo AJAX e ottiene in risposta una lista JSON di proprietà o concetti da suggerire all'utente.

Infine per quanto riguarda la messa in produzione, STAN è stato deployato su macchine linux in due ambienti differenti: su un server privato di 7Pixel e su un server pubblico della rete dell'università Bicocca. La versione deployata nei sistemi di 7Pixel non è accessibile al di fuori dell'azienda per motivi legati alla privacy dei dati. L'indice utilizzato dall'applicazione in questo caso è infatti costruito a partire dai listini dei commercianti che compaiono sui siti di eCommerce di 7Pixel e pertanto non può essere esposto pubblicamente. L'applicazione tuttavia dovrebbe essere presto utilizzata dagli altri software di 7Pixel tramite API come provider di annotazioni per i campi dello schema dei listini. La versione deployata sui server dell'università Bicocca, invece, è pubblicamente accessibile all'indirizzo <http://stan.disco.unimib.it/> e consultabile sia tramite UI che API.

## 5.2 Processo di sviluppo

Durante la raccolta dei requisiti e lo sviluppo del software di STAN sono stati seguiti i principi della metodologia di sviluppo Agile [4]. I metodi agili si contrappongono al modello a cascata e ad altri processi di sviluppo software tradizionali [5] proponendo un approccio focalizzato sull'obiettivo di consegnare al cliente, in tempi brevi e frequentemente, software funzionante e di qualità.

Nel caso di questa tesi i clienti dell'applicazione STAN sono stati i relatori della stessa che sono stati a tutti gli effetti parte integrante del team, non come sviluppatori ma appunto come responsabili della qualità finale del prodotto. La pianificazione dello sviluppo è stata fatta suddividendo il lavoro da svolgere in piccoli task e concordandone con i clienti la priorità<sup>7</sup>. Ciascun task contiene tutto ciò che è necessario per rilasciare un piccolo incremento nelle funzionalità del software: analisi dei requisiti, progettazione, implementazione e test. Ciascun task inoltre si è ritenuto concluso solo a seguito del deploy nell'ambiente di produzione e dell'accettazione finale dei clienti. Ad esempio il primo task nello sviluppo di STAN è stato sviluppare una semplicissima applicazione web attiva su un server remoto che se interrogata rispondesse con il messaggio "STAN is alive". In seguito incrementalmente sono state aggiunte tutte le funzionalità.

---

<sup>6</sup><http://lucene.apache.org/solr/>

<sup>7</sup>Lo storico di tali task è reperibile presso <http://semanticannotation.wikidot.com/brando>

Per evitare di perdere il focus sull'obiettivo finale e per gestire eventuali cambi di priorità lo sviluppo del software è stato suddiviso in finestre di tempo limitate chiamate iterazioni con, in genere, una durata di qualche settimana. Anche se il risultato di ogni singola iterazione non ha sufficienti funzionalità da essere considerato completo deve essere disponibile nell'ambiente di produzione e, nel susseguirsi delle iterazioni, deve avvicinarsi sempre di più alle richieste del cliente. Alla fine di ogni iterazione, inoltre, è stato analizzato il punto della situazione e sono stati concordati i successivi task da implementare nell'applicazione.

Una delle pratiche delle metodologie agili che è stata adottata nello sviluppo di STAN è il Test-Driven Development (TDD) [3]. Il TDD è un modello di sviluppo del software che prevede che la stesura dei test automatici avvenga prima di quella del software che deve essere sottoposto a test, e che lo sviluppo del software applicativo sia orientato esclusivamente all'obiettivo di passare i test precedentemente predisposti. Più in dettaglio, il TDD prevede la ripetizione di un breve ciclo di sviluppo in tre fasi. Nella prima fase (detta "fase rossa"), il programmatore scrive un test automatico per la nuova funzione da sviluppare, che deve fallire in quanto la funzione non è stata ancora realizzata. Nella seconda fase (detta "fase verde"), il programmatore sviluppa la quantità minima di codice necessaria per passare il test. Nella terza fase (detta "fase grigia" o di refactoring), il programmatore esegue il refactoring del codice per adeguarlo a determinati standard di qualità. Seguendo questa pratica durante lo sviluppo di STAN sono stati implementati oltre 200 test unitari con una copertura sul codice del 70% grazie ai quali è possibile monitorare costantemente il corretto funzionamento dell'applicazione.

Parte integrante dello sviluppo dell'applicazione ha anche riguardato l'automatizzazione dei passi di deployment. Nella pratica delle metodologie agili, dove il rilascio in produzione avviene frequentemente, è importante che la procedura di deployment sia rapida e automatica. A maggior ragione l'automatizzazione è importante se, come nel caso di STAN, prima del deployment in produzione si effettua un deployment transitorio su una macchina di integrazione per testare il corretto comportamento dell'applicazione su una macchina diversa da quella di sviluppo. La procedura di deployment che è stata pensata, sviluppata e infine automatizzata per l'applicazione di STAN prevede diversi passi in sequenza e in caso di fallimento di uno di essi si interrompe. Innanzitutto sulla macchina di sviluppo viene compilato il codice sorgente e vengono fatti girare i test unitari. A seguire sulla macchina di integrazione si scarica la versione aggiornata del software sfruttando GIT<sup>8</sup> come strumento di versioning e si fanno girare i test unitari più alcuni test specifici di integrazione. Infine la versione compilata dell'applicazione viene inviata al server di produzione sul quale viene installata e per concludere il processo viene fatto girare un insieme specifico di test di produzione che testano

---

<sup>8</sup><https://git-scm.com/>

ad esempio che l'applicazione sia effettivamente attiva.

L'attenzione che è stata posta nel processo di sviluppo dell'applicazione è servita sicuramente a creare un software di qualità e manutenibile nel lungo periodo. Le metodologie agili inoltre sono anche una pratica chiave al centro del processo di sviluppo di 7Pixel. Per questo adottarle è stato anche un requisito per poter più facilmente integrare in futuro STAN nei sistemi dell'azienda.

## 5.3 Funzionalità

### 5.3.1 Caricamento della tabella

Per iniziare l'annotazione di una tabella la prima cosa da fare è effettuare il caricamento della stessa nel sistema STAN. Per importare la tabella in STAN esistono due modalità: l'import da file oppure l'import da URL. L'import da file prevede il caricamento diretto di un file dalla macchina dalla quale si sta utilizzando STAN, al contrario l'import da URL prevede semplicemente l'inserimento di un indirizzo web dal quale è possibile scaricare il file della tabella da importare. Questa seconda tipologia di importazione è utile nel caso si voglia utilizzare dati pubblici come, ad esempio, quelli della regione Lombardia o della città di Chicago che sono accessibili e scaricabili online. In questo caso dato un indirizzo URL da cui accedervi è STAN che si occupa di scaricarli e importarli senza che l'utente debba scaricarli sulla propria macchina.

The screenshot shows the STAN Alpha Semantic Table Annotation Tool interface. At the top, the title 'STAN Alpha' is on the left and 'Semantic Table Annotation Tool' is on the right. Below the title bar, there is a navigation bar with 'Home' and 'Upload' links. Underneath, there are two tabs: 'Upload from file' (selected) and 'Upload from URL'. The main content area is titled 'Upload your CSV table'. It features a text input field containing 'CPS\_Schools.csv' with 'Change' and 'Remove' buttons to its right. Below this, a label reads 'Specify a separator character, eventually a text delimiter and if the table has an header:'. There are two input fields: 'Separator' with a comma (',') and 'Text Delimiter' with a double quote ('"'). A checkbox labeled 'Has header' is checked. At the bottom left of the form is an 'Upload' button.

**Figura 5.2:** L'interfaccia di STAN durante l'import di una tabella.

Al momento STAN supporta soltanto il formato CSV dei dati pertanto i file caricati, sia direttamente che tramite URL, devono rispettare tale formato per poter essere visualizzati correttamente. In fase di importazione il sistema chiede inoltre all'utente di specificare alcuni parametri per consentire una corretta visualizzazione della tabella:

- Un carattere separatore per identificare le colonne della tabella.
- Un carattere delimitatore del testo per gestire correttamente i casi in cui una cella contenga il carattere separatore.
- La presenza o meno dell'header delle colonne nella tabella caricata.

### 5.3.2 Workspace privato

Può capitare che durante il lavoro di annotazione ci si debba interrompere o che in un secondo momento ci si renda conto di voler raffinare le annotazioni effettuate. In questi casi è un peccato dover ricominciare il lavoro dall'inizio. Per questo motivo STAN offre a ciascun utente la possibilità di recuperare il lavoro svolto sull'ultima tabella caricata. In questo modo di fatto ciascun utente di STAN ha la possibilità di crearsi il proprio workspace personale e privato. La creazione del workspace avviene in modo automatico e trasparente all'utente che non ha neanche bisogno di registrarsi o effettuare un login. Il sistema infatti sfrutta i cookie per tenere traccia degli utenti e delle tabelle che hanno caricato. Un cookie di fatto è un piccolo pezzo di informazione che viene memorizzato nel browser dell'utente in modo che durante gli accessi successivi sia possibile identificarlo. In particolare STAN memorizza due cookie per ciascun utente: un cookie *user* per memorizzare il nome dell'utente assegnato casualmente da STAN, e un cookie *table* per memorizzare l'ultima tabella su cui ha lavorato quell'utente.

### 5.3.3 Visualizzazione della tabella

Una volta effettuato il caricamento di una tabella sarà possibile visualizzarla all'interno di STAN. Per renderizzare graficamente la tabella è stato utilizzato Bootstrap Table<sup>9</sup>, un framework disponibile per il templating di tabelle e che offre alcune funzionalità aggiuntive alla semplice visualizzazione. Ad esempio, infatti, nel caso di una tabella con molte righe la visualizzazione della tabella può essere suddivisa in pagine e l'utente può selezionare il numero massimo di righe da visualizzare in ogni pagina. La paginazione delle righe della tabella consente a STAN di non dover caricare in memoria l'intera tabella ma soltanto un sottoinsieme di essa. Questo permette all'applicazione di essere efficace e responsiva anche in caso di tabelle molto voluminose. Inoltre, il caricamento delle righe della tabella viene effettuato in modo asincrono rispetto al resto della pagina. Questo significa che se

---

<sup>9</sup><http://wenzhixin.net.cn/p/bootstrap-table/docs/index.html>

anche ci fosse un ritardo nel caricamento delle righe della tabella l'utente sarebbe in grado di navigare ed utilizzare i restanti contenuti della pagina.

## STAN Alpha

Semantic Table Annotation Tool

Home / CPS\_Schools.csv

Annotation Ontologies Namespace Save Export

SchoolID	SchoolName	FullName	SchoolName2	ISBE Name	Street Number	Street Direction	Street Name	City	State	ZIP
400010	Ace Technical Chtr HS	Architecture, Construction, and Engineering(ACE)Technical Charter School	Ace Technical Chtr HS	Ace Technical Charter High School	5410	S	State St	Chicago	IL	60609
609772	Addams	Jane Addams Elementary School	Addams	Addams Elem School	10810	S	Avenue H	Chicago	IL	60617
609773	Agassiz	Louis A Agassiz Elementary School	Agassiz	Agassiz Elem School	2851	N	Seminary Ave	Chicago	IL	60657
610513	Air Force HS	Air Force Academy High School	Air Force HS	Air Force Acad High School	3630	S	Wells St	Chicago	IL	60609
610212	Albany Park	Albany Park Multicultural Academy	Albany Park	Albany Park Multicultural Elem	4929	N	Sawyer Ave	Chicago	IL	60625
609774	Alcott ES	Louisa May Alcott Elementary School	Alcott ES	Alcott Elem School	2625	N	Orchard St	Chicago	IL	60614
610524	Alcott HS	Alcott High School for the	Alcott HS	Alcott Humanities	2957	N	Hovne Ave	Chicago	IL	60618

Showing 1 to 100 of 672 rows 100 records per page

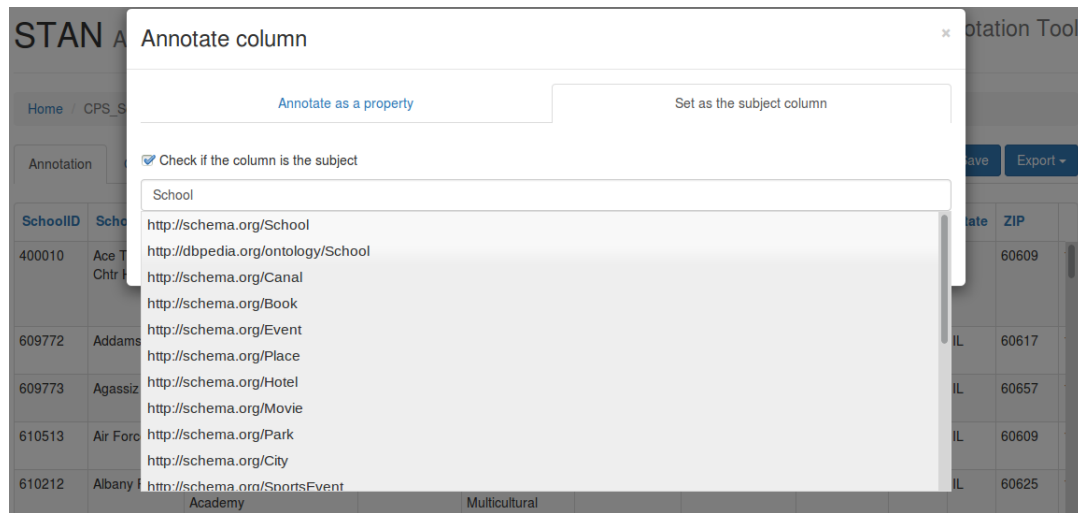
« 1 2 3 4 5 »

**Figura 5.3:** La visualizzazione di una tabella dopo aver effettuato l'importazione in STAN.

#### 5.3.4 Annotazione del soggetto

Dei modelli definiti nella sezione 3 STAN è in grado di supportare il modello *Star* e il modello *Simple Snow-Flake*. Entrambi i modelli richiedono l'identificazione di una colonna come soggetto della tabella. Selezionando una delle colonne infatti all'utente viene mostrato un pannello tramite il quale può scegliere se annotare la colonna come soggetto oppure come proprietà da associare ad un soggetto precedentemente definito. Se si seleziona l'annotazione del soggetto all'utente viene chiesto di specificare il tipo di una ontologia da assegnare alla colonna. Questo significa che la colonna viene arricchita di una semantica tale per cui tutti gli elementi ad essa appartenenti corrispondono ad entità del tipo selezionato.

In particolare l'annotazione può essere eseguita definendo manualmente un tipo oppure riutilizzandone uno esistente. Se l'utente decide di riutilizzare un tipo già esistente STAN lo supporta attraverso un sistema di autocompletamento che suggerisce i tipi estratti dall'ontologia di DBPedia. L'autocompletamento è un servizio sviluppato nel contesto di questa tesi all'interno del framework di ABSTAT ed esposto pubblicamente via API. Al momento l'autocompletamento sui tipi è stato implementato solo in riferimento all'ontologia di DBPedia ma l'idea futura prevede di supportare anche ulteriori ontologie.



**Figura 5.4:** L’annotazione di una colonna come soggetto della tabella in STAN.

### 5.3.5 Annotazione di una colonna come proprietà

L’annotazione di una colonna come proprietà associa la colonna annotata a quella precedentemente definita come soggetto. Per compiere l’annotazione STAN di fatto chiede di definire tramite interfaccia una tripla <soggetto, predicato, oggetto>. Il soggetto è un campo precompilato in cui viene visualizzato il tipo assegnato alla colonna definita come soggetto. La proprietà può essere scelta con l’ausilio di un sistema di autocompletamento che suggerisce le proprietà di DBPedia oppure può essere definita a mano se si sceglie di utilizzare un’ontologia propria. L’oggetto, infine, definisce il tipo da assegnare ai valori della colonna e può essere sia il concetto di una ontologia che un datatype. Tale annotazione assegna alla colonna una semantica per cui tutti i valori di essa sono messi in relazione alla colonna soggetto dalla proprietà definita e inoltre ad essi viene assegnato il tipo specificato come oggetto della proprietà.

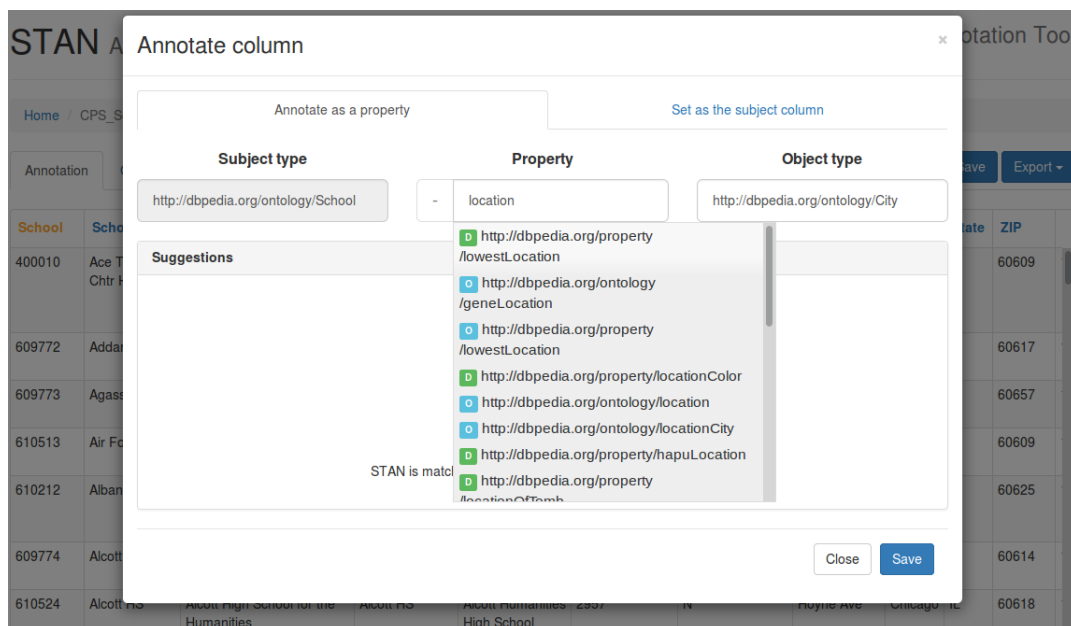
School ID	School Name	City
0001	Prescott elementary school	Chicago
0002	Edgebrook elementary school	Chicago
0003	Northside Prep high school	Chicago

**Tabella 5.1:** Estratto di una delle tabelle relativa al caso di studio di Chicago.

Analizzando ad esempio la tabella 5.1, che rappresenta un estratto della tabella delle scuole del caso di studio di Chicago, è possibile esemplificare alcune possibili annotazioni. In questo caso la colonna *School ID* rappresenta un identificativo di ciascuna scuola quindi potrebbe essere annotata come soggetto della tabella e un

tipo adatto da associare a tale colonna potrebbe essere *dbo:School*. La colonna *School Name*, invece, potrebbe essere annotata con la proprietà *foaf:name* descrivendo così il nome della scuola. Per la colonna *City* infine una buona annotazione potrebbe essere *dbo:location* specificando come object-type *dbo:City*, in questo modo ciascuna scuola verrebbe associata alla città in cui si trova.

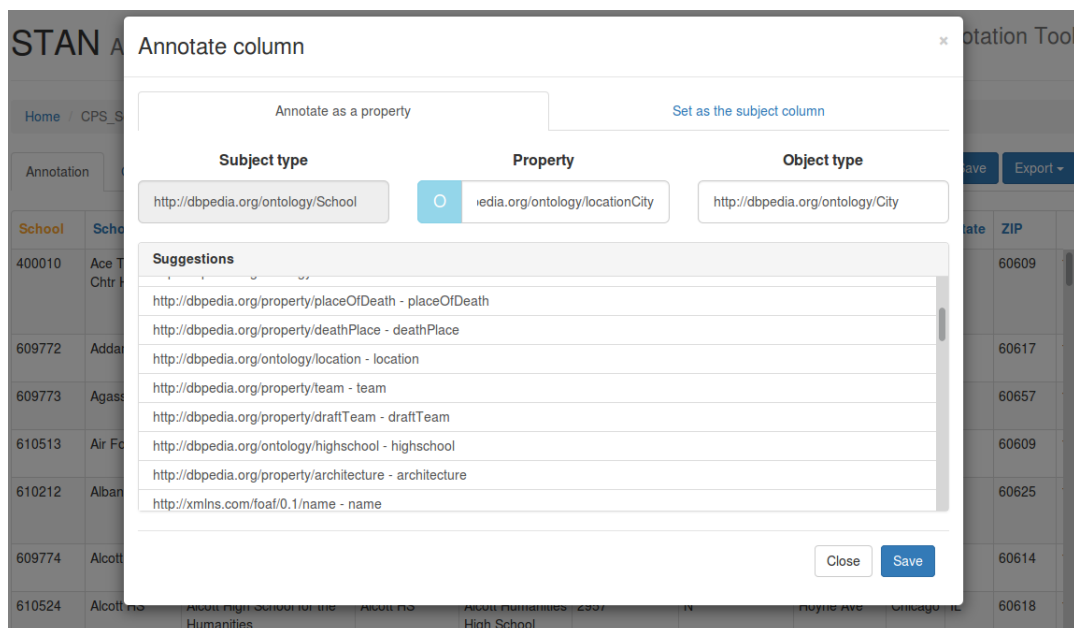
Al momento si è deciso di lasciare agli utenti la libertà di specificare un qualsiasi object-type a prescindere dalla proprietà selezionata, sia che essa sia una *ObjectProperty* che una *DatatypeProperty*. Se da un lato è vero che questo potenzialmente permette agli utenti di definire dei mapping di cattiva qualità da un altro garantisce una maggiore flessibilità nell'annotazione. Infatti bisogna considerare anche che non in tutte le basi di conoscenza si tiene conto di tale classificazione delle proprietà e che essa è una specifica propria dell'ontologia OWL [24]. L'utente è comunque incentivato ad utilizzare le proprietà con la loro accezione più corretta. Infatti i suggerimenti forniti dal sistema di autocompletamento mostrano all'utente un badge grafico che evidenzia il tipo delle proprietà rendendo così l'utente consapevole dell'utilizzo più corretto che andrebbe fatto di tale proprietà. STAN dunque interpreta come utilizzare la proprietà in base al tipo dell'oggetto selezionato e questo influenza di conseguenza la successiva generazione delle triple RDF. Infatti se l'utente seleziona come object-type un concetto allora i valori della colonna annotata saranno considerati come delle entità altrimenti saranno considerati dei valori letterali.



**Figura 5.5:** L'interfaccia di STAN durante l'annotazione di una colonna come proprietà e i suggerimenti forniti dall'autocompletamento.

### 5.3.6 Suggerimenti di annotazione

Per aiutare l'utente in fase di annotazione STAN, oltre a fornire il sistema di auto-completamento, suggerisce una lista di proprietà come possibili annotazioni di una colonna. I suggerimenti sono forniti grazie all'inclusione in STAN di un algoritmo di annotazione automatica sviluppato all'interno del dipartimento di Informatica Sistemistica e Comunicazione dell'università Bicocca. L'algoritmo non è ancora stato presentato alla comunità scientifica ma si inserisce nel contesto degli approcci basati su una knowledge base (sezione 4.5). L'algoritmo sfrutta un approccio *instance based* indipendente dal dominio utilizzato. In particolare l'algoritmo utilizza i valori di una colonna e alcune informazioni di contesto per matchare all'interno della knowledge base la proprietà semanticamente più compatibile per l'annotazione.



**Figura 5.6:** I suggerimenti forniti da STAN durante l'annotazione di una colonna.

Una volta caricate in STAN le tabelle estratte dal web tuttavia perdono le loro informazioni di contesto. Per questo motivo si è pensato ad una strategia alternativa per sfruttare al meglio il modello sui cui si basa l'algoritmo. In particolare si assume che il tipo assegnato al soggetto possa rappresentare una valida informazione di contesto dato che nei modelli *Star* e *Simple Snow-Flake* implementati in STAN tutte le colonne sono in qualche modo in relazione alla colonna soggetto. Nel caso invece dei listini commerciali di 7Pixel in cui non esiste una colonna soggetto si è utilizzato come contesto l'header delle colonne nei casi in cui era disponibile. L'algoritmo nella sua accezione originale utilizza la base di conoscenza di DBPedia ma può modellarsi per essere utilizzato a partire da qualsiasi base di conoscenza.



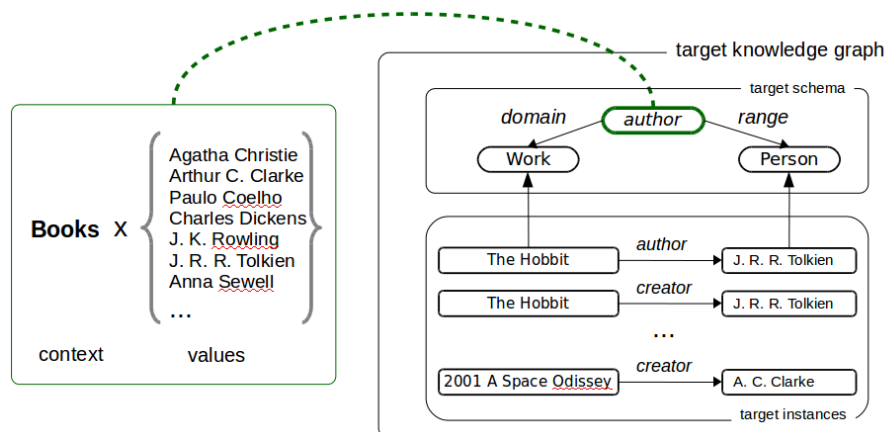
Infatti DBPedia è un'ottima fonte di annotazioni dato che possiede un'ontologia in grado di coprire numerosi domini tuttavia esistono casi, come quello di 7Pixel, in cui è possibile applicare solo un'ontologia di dominio specifica. Per applicare dunque l'algoritmo al caso specifico di 7Pixel è stata costruita una knowledge base a partire dai listini dei prodotti commerciali che sfrutta come ontologia solo quei concetti rilevanti nel contesto di 7Pixel.

L'algoritmo non è oggetto della tesi per cui si descriveranno solo a grandi linee i concetti principali che lo caratterizzano. L'approccio si basa su tre componenti principali: un contesto  $c$ , l'insieme dei valori di una colonna  $V$  e una *knowledge base*. Il contesto rappresenta una classe di entità del mondo reale ed è espresso tramite il tipo assegnato alla colonna soggetto della tabella. Ad esempio, per la tabella 5.1 il contesto è *School*, da associare per esempio all'insieme di valori  $V = \{\text{Prescott elementary school, Edgebrook elementary school, ...}\}$ . Una *knowledge base*, invece, è definita come un insieme di proprietà  $p$ , con  $p \subseteq D \times R$ , aventi un dominio  $D$  ed un range  $R$  che mettono in relazione diverse entità. Ricostruire la semantica di una colonna dunque significa annotare l'insieme dei suoi valori  $V$  con una proprietà  $p$  o un insieme di proprietà  $P = \{p_1, ..., p_n\}$  della knowledge base in modo che la loro semantica sia compatibile. L'intuizione dell'algoritmo per identificare la compatibilità semantica si basa su tre diversi fattori:

- *Copertura*. Dato un contesto  $c$ , i valori di una colonna  $V$  e una proprietà  $p$  il cui dominio matcha lessicalmente con  $c$ . Maggiore è il numero di valori in  $V$  che compaiono come elementi del range di  $p$  e maggiore è la compatibilità semantica tra  $V$  e  $p$ .
- *Frequenza*. Dato un contesto  $c$ , i valori di una colonna  $V$  e una proprietà  $p$  il cui dominio matcha lessicalmente con  $c$ . Quante più volte ciascun valore in  $V$  compare tra gli elementi del range di  $p$  e maggiore è la compatibilità semantica tra  $V$  e  $p$ .
- *Specificità*. Dato un contesto  $c$ , i valori di una colonna  $V$  e una proprietà  $p$  il cui dominio matcha lessicalmente con  $c$ . Più la proprietà  $p$  è specifica in relazione a  $c$  e ai valori di  $V$  e maggiore è la compatibilità semantica tra  $V$  e  $p$ .

L'algoritmo in alcune circostanze impiega un tempo non indifferente per generare i suggerimenti. Per questo motivo si è deciso di rendere la generazione dei suggerimenti asincrona rispetto al caricamento della pagina. La lista dei suggerimenti infatti viene caricata via JavaScript per cui l'utente è libero di proseguire con l'annotazione anche quando l'algoritmo sta ancora generando i suggerimenti. Inoltre, come trattato in modo più approfondito nella sperimentazione, i tempi dell'algoritmo dipendono dal numero di valori della colonna utilizzati per effettuare il matching. Pertanto per migliorare le performance dell'algoritmo non vengono utilizzati tutti i valori di una colonna ma soltanto un sottoinsieme di essi scelto in

modo casuale. In particolare, la selezione dei valori da utilizzare come campione della colonna avviene sfruttando l'implementazione in STAN di un algoritmo di *Reservoir Sampling* [42]. Il *Reservoir Sampling* è un algoritmo per selezionare in tempo lineare un campione casuale di  $k$  elementi da una lista di dimensione  $n$ , dove  $n$  è solitamente un numero molto grande.



**Figura 5.7:** Schematizzazione del problema dell'annotazione per l'algoritmo di STAN.

### 5.3.7 Salvataggio dello stato corrente

Per fare in modo che un utente possa recuperare la propria sessione di lavoro, oltre alla gestione del workspace personale, è necessario tenere traccia delle annotazioni effettuate. Per questo STAN consente in qualsiasi momento di salvare lo stato corrente dell'annotazione semplicemente cliccando sul pulsante di salvataggio. In questo modo si può interrompere il lavoro per poi riprendere dal punto esatto in cui ci si è interrotti. Per tracciare il lavoro di un utente STAN utilizza un file di supporto in cui vengono salvate tutte le informazioni che l'utente ha immesso nel sistema: le annotazioni delle proprietà, gli object-type, le colonne definite come soggetto e il tipo ad essa associato.

### 5.3.8 Export dei mapping

Al termine dell'annotazione l'utente ha due possibilità per usufruire dell'output dell'applicazione: esportare i mapping definiti oppure esportare direttamente i dati della tabella convertiti in triple RDF. Nel caso si scelga di esportare i mapping l'applicazione consente di scaricare un file in formato RML [14] contenente tutti i mapping definiti annotando la tabella. RML è un linguaggio per esprimere regole di mapping che consente di serializzare strutture dati eterogenee secondo il modello RDF dei dati. Il linguaggio RML non è uno standard del W3C tuttavia è candidato per esserlo e si basa sul linguaggio R2RML [12] che invece è uno standard. In particolare RML estende le funzionalità di R2RML consentendo di applicare i

mapping non solo a tabelle relazionali di database ma anche ad altre fonti di dati come file CSV, XML o JSON. STAN al momento supporta solo il formato CSV tuttavia in ottica futura è previsto il supporto anche di altri formati. Per cui proprio la possibilità di utilizzare lo stesso linguaggio di mapping per diversi formati di dati è stato il principale motivo che ha condotto alla scelta di RML.

### 5.3.9 Export delle triple

Come accennato nella sezione 5.1 STAN racchiude al suo interno un engine di generazione di triple che viene sfruttato quando l'utente sceglie di esportare l'output dell'annotazione sotto forma di triple RDF. L'engine RML dunque si occupa di trasformare la tabella importata in STAN in triple RDF basandosi sui mapping che sono stati generati dall'utente. Quando le triple sono state generate viene creato un file in formato N-Triples<sup>10</sup> scaricabile tramite STAN. Una volta scaricato questo file per integrare i dati appena generati è sufficiente che l'utente li carichi in un triple store. In questo modo l'utente avrà immediato accesso all'insieme di dati nella loro visione di insieme e potrà ad esempio esplorarli e interrogarli tramite un endpoint SPARQL. Riprendendo l'esempio di mappatura esposto in 5.3.5 l'export delle triple effettuato per quella annotazione sarebbe:

```
<http://ex.com/stan/0001> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/School> .
<http://ex.com/stan/0001> <http://www.w3.org/2000/01/rdf-schema#label>
    "0001" .
<http://ex.com/stan/0001> <http://xmlns.com/foaf/0.1/name>
    "Prescott elementary school" .
<http://ex.com/stan/0001> <http://dbpedia.org/ontology/location>
    <http://ex.com/stan/Chicago> .
<http://ex.com/stan/0002> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/School> .
<http://ex.com/stan/0002> <http://www.w3.org/2000/01/rdf-schema#label>
    "0002" .
<http://ex.com/stan/0002> <http://xmlns.com/foaf/0.1/name>
    "Edgebrook elementary school" .
<http://ex.com/stan/0002> <http://dbpedia.org/ontology/location>
    <http://ex.com/stan/Chicago> .
<http://ex.com/stan/0003> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/School> .
<http://ex.com/stan/0003> <http://www.w3.org/2000/01/rdf-schema#label>
    "0003" .
<http://ex.com/stan/0003> <http://xmlns.com/foaf/0.1/name>
    "Northside Prep high school" .
<http://ex.com/stan/0003> <http://dbpedia.org/ontology/location>
    <http://ex.com/stan/Chicago> .
<http://ex.com/stan/Chicago> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/City> .
<http://ex.com/stan/Chicago> <http://www.w3.org/2000/01/rdf-schema#label>
    "Chicago" .
```

Le triple generate da STAN per l'esempio in tabella 5.1.

<sup>10</sup><http://www.w3.org/TR/n-triples/>

Si può notare che ciascun valore della colonna soggetto è stato trasformato in un'entità di cui se ne specificano il tipo e la label. Ad tali entità inoltre si collegano le rimanenti proprietà definite. Quando si definisce una *ObjectProperty* in particolare anche l'oggetto della proprietà viene trasformato in entità. L'esempio in questo caso è *http://ex.com/stan/Chicago* che viene appunto utilizzato come entità di cui a sua volta si specificano il tipo e la label.

### 5.3.10 Definizione del namespace

Come si può notare i valori della tabella sono trasformati in entità sotto forma di URI per rispettare il paradigma delle cinque stelle degli Open Data [39]. Infatti, l'utente dopo aver scaricato le triple è libero di caricarle su un proprio server rendendo così i dati accessibili come Linked Data.

La trasformazione in URI dei valori è effettuata autonomamente da STAN ma l'utente è libero di settare dall'interfaccia il namespace che verrà poi utilizzato per la creazione dell'indirizzo web. Nell'esempio appena riportato ad esempio il namespace settato era *http://ex.com/stan/*.

### 5.3.11 Servizio di API

STAN non offre solo un servizio tramite interfaccia web per l'annotazione di una tabella ma espone anche un servizio pubblico di API per effettuare l'annotazione di una singola colonna. Dei tool analizzati STAN è l'unico ad esporre un servizio di API pubbliche per l'annotazione. Karma ed Open Refine, infatti, consentono l'annotazione dei dati solo tramite interfaccia web. In questo modo, inoltre, le API di STAN sono l'unica implementazione concreta di un algoritmo di annotazione automatica che viene resa disponibile per l'utilizzo in applicazioni di terze parti. Infatti, di tutti gli algoritmi di annotazione analizzati solo quello di Karma si inserisce nel contesto concreto di un tool Open Source, gli altri sono approcci teorici presentati alla comunità scientifica che però non sono mai stati integrati in un tool.

L'utilizzo di API da parte di servizi esterni può essere utile in diversi casi. Un esempio sicuramente è il caso di studio di 7Pixel. In questo caso esiste già un sistema utilizzato dagli esperti di dominio di 7Pixel che consente di effettuare la mappatura dei listini. Tuttavia l'output di questo strumento produce un file di mapping specifico per il dominio di 7Pixel che differisce da quello prodotto da STAN. Per questo STAN non può essere utilizzato direttamente per definire i mapping in 7Pixel ma può essere utilizzato come provider esterno di annotazioni sfruttando proprio le API che mette a disposizione.

Attualmente STAN fornisce via API due distinti servizi di annotazione: uno che sfrutta lo stesso algoritmo utilizzato per la generazione dei suggerimenti, e un altro che si basa sull'algoritmo utilizzato in Karma. I due servizi sono accessibili pubblicamente e ciascuno risponde a richieste HTTP, sia GET che POST, ad un

preciso indirizzo web. La richiesta HTTP da formulare a STAN necessita soltanto della definizione di un oggetto JSON che specifichi i seguenti attributi:

- *Values*. Un gruppo di valori da annotare. Il numero di valori possibili non è limitato superiormente ma se eccede i 10 l'algoritmo ne seleziona 10 in maniera casuale.
- *Kb*. La knowledge base da utilizzare come fonte delle annotazioni. Al momento esiste una sola opzione che è DBPedia tuttavia in futuro se ne potranno utilizzare anche altre.
- *Context*. Il contesto opzionale da usare nel caso si utilizzi l'algoritmo integrato in STAN.

In risposta ad una richiesta al servizio di API STAN fornisce una lista di possibili annotazioni in formato JSON. In particolare le annotazioni suggerite fanno tutte riferimento a proprietà della knowledge base utilizzata e per ciascuna di esse si specificano la URI, la label e uno score di affidabilità assegnato dall'algoritmo.

```
{
  "results": [
    {
      "score": "1.5069614918474152",
      "label": "name",
      "uri": "http://dbpedia.org/property/name"
    },
    {
      "score": "1.3208489050625816",
      "label": "name",
      "uri": "http://xmlns.com/foaf/0.1/name"
    },
    {
      "score": "1.1398245601937667",
      "label": "caption",
      "uri": "http://dbpedia.org/property/caption"
    },
    {
      "score": "0.8960444778120581",
      "label": "notableWork",
      "uri": "http://dbpedia.org/ontology/notableWork"
    }
  ]
}
```

Esempio di risposta JSON ad una richiesta alle API di STAN.



## Capitolo 6

# Sperimentazione

La sperimentazione condotta ha due obiettivi principali: valutare la qualità dei suggerimenti prodotti dall'algoritmo di STAN confrontandoli con quelli dell'algoritmo di Karma [37], e valutare le performance in termini di tempo dell'algoritmo di STAN in relazione ad alcuni fattori al fine di comprendere quale sia la combinazione più performante. Karma è stato scelto come paragone da stato dell'arte dato che è uno dei pochi tool completi di Table Annotation e che il suo algoritmo è l'unico il cui codice sia accessibile pubblicamente. Grazie alla modularità di STAN è stato possibile far convivere entrambi gli algoritmi nell'applicazione e di conseguenza condurre la sperimentazione.

La sperimentazione sia in termini qualitativi che di performance si basa sull'analisi di alcune metriche in relazione al variare di alcuni fattori. Al fine di comprendere al meglio la terminologia utilizzata nel corso della sperimentazione si darà una definizione intuitiva dei fattori che influenzano la sperimentazione:

- **Algoritmi**

- *Stan*: l'algoritmo *instance based* implementato in STAN.
- *Karma*: l'algoritmo presentato nella sezione 4.8.1 che sfrutta un approccio differente a seconda che si trattino colonne testuali o numeriche.

- **Dataset**

- *7Pixel*: i listini commerciali dell'azienda 7Pixel relativi al sito di TrovaPrezzi.it. Il dataset è composto da 2.373 tabelle per un totale di oltre 28.000 colonne. La grandezza dei listini varia da un minimo di una decina di righe ad un massimo di circa 125.000 righe.
- *School Closing*: le tabelle pubbliche messe a disposizione dalla città di Chicago rilevanti per analizzare lo scenario della chiusura delle scuole. I dati caratterizzano lo scenario con informazioni relative a scuole, censimento, povertà e criminalità. Il dataset si compone di 11 tabelle per un totale di 225 colonne. La grandezza delle tabelle varia da un minimo di 80 righe ad un massimo di 750.000.

### • Ontologia

- *7Pixel*: nei listini commerciali ciascuna riga descrive un’offerta e l’annotazione viene effettuata utilizzando l’ontologia di dominio specifica di 7Pixel che comprende proprietà come *Prezzo*, *Marca*, *Descrizione*, etc. Per ciascuna tabella, inoltre, è disponibile l’insieme di mapping tra le colonne e l’ontologia. Analizzando dunque lo storico di questi mapping definiti dagli esperti di dominio di 7Pixel sono state ricostruite le annotazioni corrette in modo da creare il gold standard per il dataset.
- *DBPedia*: l’ontologia usata per annotare le tabelle del dataset delle School Closing è quella di DBPedia tuttavia la valutazione della qualità delle annotazioni esula dagli obiettivi di questa tesi perciò non si è creato un gold standard per il dataset a partire da essa.

### • Variabili

- *Contesto*: l’algoritmo di STAN per l’annotazione sfrutta, tra l’altro, una informazione di contesto. Tuttavia il modo di recuperare tale informazione è differente a seconda del tipo di dataset utilizzato. Nel dataset delle School Closing le tabelle possono essere annotate sfruttando il modello *Simple Snow-Flake*, quindi come informazione di contesto per l’annotazione si utilizza il tipo della colonna soggetto che per questa sperimentazione è stato assegnato manualmente a ciascuna tabella. Nel dataset di 7Pixel, invece, le tabelle presentano un modello *Hidden-subject Star* per cui il soggetto non è contenuto nella tabella. Per questo motivo in questo caso come contesto per annotare una colonna si è utilizzato l’header stesso della colonna.
- *Tipo di colonne*: le colonne possono essere classificate come testuali o numeriche a seconda del tipo di valori che contengono. Questa distinzione in particolare è rilevante per Karma dato che applica un approccio differente a seconda del tipo della colonna.
- *Numero di celle*: il numero di valori di una colonna utilizzato dagli algoritmi come campione per effettuare l’annotazione. La selezione delle celle da utilizzare di una colonna viene effettuata casualmente sfruttando un algoritmo di *Reservoir Sampling* facendo attenzione a selezionare solo valori unici. Solo nel caso specifico dell’algoritmo Karma in fase di annotazione di una colonna numerica si ignora il vincolo sull’univocità dei valori. Questo perché Karma si basa sulla distribuzione statistica dei valori quindi anche i duplicati contano.
- *Dimensione dell’indice*: entrambi gli algoritmi sfruttano un indice usato come fonte di informazione. Nel caso delle School Closing l’indice è composto dall’intero insieme delle istanze di DBPedia, mentre nel caso di 7Pixel invece l’indice si compone dei listini già annotati. Dato che



### 6.1. QUALITÀ DEI SUGGERIMENTI NEL CONTESTO DELL'ECOMMERCE

i due algoritmi dipendono fortemente dall'indice si è deciso dunque di provare a simulare diverse situazioni variando la porzione del dataset indicizzato per valutare la relativa accuratezza degli algoritmi.

## 6.1 Qualità dei suggerimenti nel contesto dell'eCommerce

Come già detto l'algoritmo non è oggetto della tesi tuttavia per rendere STAN utilizzabile nell'azienda 7Pixel è necessario condurre una analisi sperimentale al fine di comprendere l'effettiva qualità dei suggerimenti prodotti. Inoltre si è deciso di confrontarsi con un altro algoritmo da stato dell'arte, Karma [37], per meglio comprendere i punti di forza e debolezza dei due approcci. Dunque la sperimentazione qualitativa dell'algoritmo è stata condotta solamente sul dataset dei listini commerciali di 7Pixel valutando, in particolare, la precisione dell'algoritmo nel classificare correttamente ciascuna colonna. Una sperimentazione più generale, che copre anche il caso delle School Closing basandosi sulla knowledge base di DBPedia, è stata invece già condotta al di fuori di questa tesi.

Il confronto relativo alla qualità dei suggerimenti forniti dai due algoritmi è stato valutato utilizzando le seguenti metriche:

- Il *Mean Reciprocal Rank* (MRR), un indice statistico per valutare un processo che produce una lista di possibili annotazioni per una colonna, ordinate per probabilità di correttezza. Calcolato come:

$$MRR = \frac{1}{|C|} \sum_{i=1}^C \frac{1}{rank_i} \quad (6.1)$$

Dove  $C$  è un insieme di colonne e  $rank_i$  è la posizione della prima annotazione corretta nella lista ordinata delle annotazioni prodotta per la  $i$ -esima colonna.

- La *Recall*, la percentuale di annotazioni rilevanti trovate in rapporto a tutte quelle rilevanti esistenti. Calcolata come:

$$R = \frac{|\{\text{annotazioni rilevanti}\} \cap \{\text{annotazioni recuperate}\}|}{|\{\text{annotazioni rilevanti}\}|} \quad (6.2)$$

Nel dataset di 7Pixel esiste una sola annotazione rilevante per ogni colonna. Perciò la *Recall* si traduce nel calcolo della probabilità che l'annotazione rilevante si trovi o meno all'interno della lista di annotazioni prodotta dall'algoritmo. Per questo si analizzerà la *Recall* in base a diverse lunghezze della lista di suggerimenti prodotta dall'algoritmo. In particolare a lunghezza 1, 5 e 10.

### 6.1.1 Confronto STAN vs Karma

Come risulta evidente dalla tabella 6.1 applicando i due algoritmi al caso di 7Pixel in media i suggerimenti forniti dall'approccio di Karma sono di migliore qualità sia in termini di MRR che di *Recall*. Per condurre questa sperimentazione i due algoritmi sono stati testati con diverse configurazioni (sia in termini di numero di celle che di dimensione dell'indice) e i risultati presentati in 6.1 fanno riferimento alla configurazione migliore per entrambi gli algoritmi risultante dall'analisi sperimentale<sup>2</sup>. Per Karma la configurazione migliore è composta dall'utilizzo di 200 celle per l'annotazione e un indice molto piccolo, solo il 10% di tutti i listini. Per STAN invece la configurazione migliore prevede sempre l'utilizzo di 200 celle ma un indice voluminoso, l'80% di tutti i listini. Una trattazione più approfondita del comportamento dei due algoritmi al variare di questi due fattori viene trattata in 6.1.4 e 6.1.5.

Metrics	STAN	Karma
MRR	48.33%	66.74%
Recall at 1	38.74%	53.54%
Recall at 5	62.09%	83.27%
Recall at 10	66.33%	90.65%

**Tabella 6.1:** Confronto di qualità tra STAN e Karma.

### 6.1.2 Differenza tra colonne numeriche e testuali

Nonostante l'approccio di Karma offra mediamente suggerimenti di miglior qualità, l'analisi dettagliata dei risultati mostra che STAN ottiene risultati migliori per quanto riguarda le colonne di tipo numerico. Come emerge dalla tabella 6.2, infatti, l'approccio di STAN fornisce suggerimenti di qualità inferiore per quanto riguarda le colonne testuali ma allo stesso tempo ne fornisce di migliori relativamente alle colonne numeriche. Calcolando la media tra i risultati ottenuti per i due tipi di colonne Karma ottiene risultati migliori perché viene calcolata la media pesata e il numero di colonne contenente valori testuali è maggiore rispetto a quelle contenenti valori numerici (circa 21.000 colonne testuali contro 7.500 colonne numeriche).

Per esempio considerando due colonne dei listini  $C_1 = \{9.5, 2.9, 12, 8.9, \dots\}$  e  $C_2 = \{\text{Canon, Rowenta, Nikon, Olympus, } \dots\}$ , per le quali le annotazioni corrette sono relativamente *Prezzo* e *Marca*, si può notare la differenza dei risultati dei due algoritmi. Per  $C_1$  le prime tre annotazioni suggerite da STAN sono in ordine *Prezzo*, *SpeseSped* e *ManufacturerSku* mentre quelle suggerite da Karma sono *Prodotto*, *Marca* e *Prezzo*. Al contrario per  $C_2$  le annotazioni suggerite da

### 6.1. QUALITÀ DEI SUGGERIMENTI NEL CONTESTO DELL'ECOMMERCE

STAN sono in ordine *Descrizione*, *Prodotto* e *Marca*, mentre per Karma sono *Marca*, *Categoria* e *SpeseSped.*

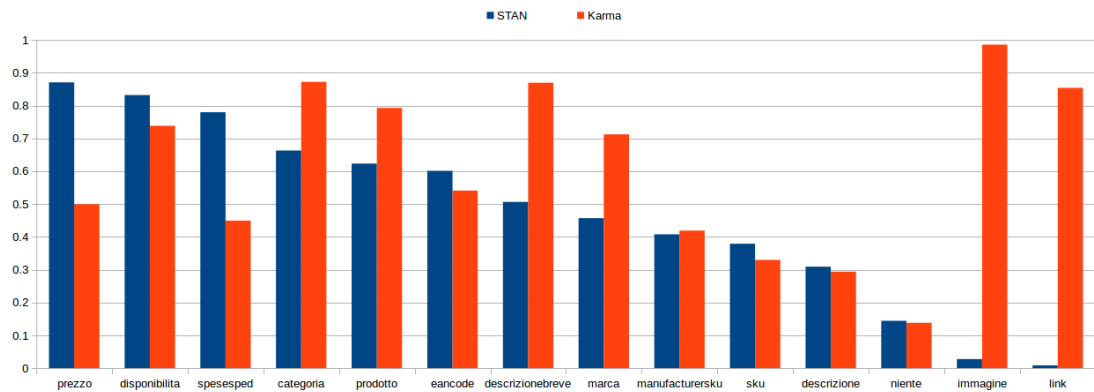
Metrics	STAN		Karma	
	Textual	Numeric	Textual	Numeric
MRR	37.15%	78.64%	75.67%	40.15%
Recall at 1	27.24%	69.92%	65.82%	16.95%
Recall at 5	51.34%	91.23%	87.74%	69.94%
Recall at 10	55.82%	94.80%	92.57%	84.96%

**Tabella 6.2:** Confronto di qualità tra STAN e Karma nel dettaglio tra colonne numeriche e testuali.

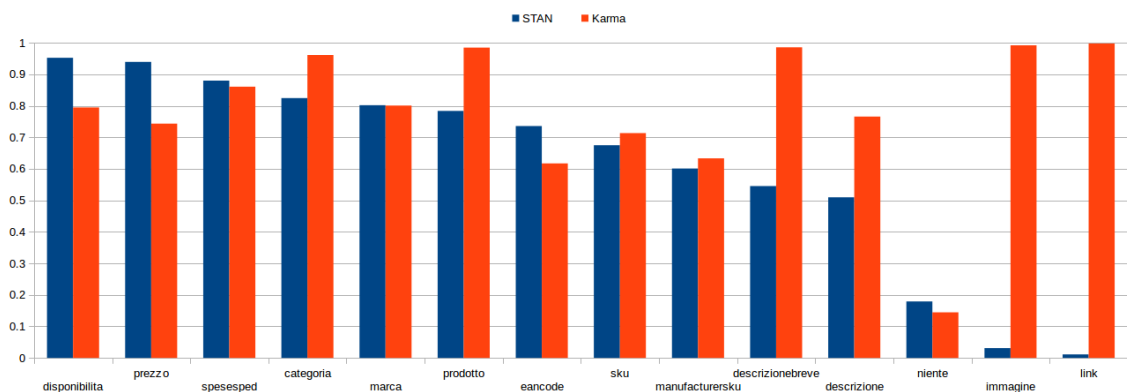
#### 6.1.3 Dettaglio sulle singole colonne

Nei grafici 6.1 e 6.2 viene mostrato il dettaglio relativo a MRR e *Recall* della qualità nei suggerimenti evidenziando gli specifici campi dell'ontologia di 7Pixel. L'ontologia di dominio di 7Pixel è pensata per descrivere una offerta commerciale, infatti come si vede nei grafici i campi ne descrivono il prodotto, la marca, il prezzo e altre informazioni utili per gli utenti. A volte però capita che nei listini dei commercianti siano presenti delle informazioni non interessanti per 7Pixel e che quindi si sceglie di ignorare. In questi casi si sceglie di annotare tali colonne con un campo speciale che è *niente*.

Per non appesantire troppo la descrizione si è deciso di non mostrare i grafici di tutti i tagli di *Recall* ma solo il taglio ai primi 5 risultati. Come si può notare vengono confermati i risultati presentati nel paragrafo precedente. Le colonne sono ordinate in base ai risultati ottenuti dall'algoritmo di STAN in ordine decrescente e, infatti, le colonne in cui STAN ottiene i migliori risultati sono le tre colonne numeriche: prezzo, disponibilità e spese di spedizione. Si può notare come per queste colonne STAN ottenga dei risultati migliori a Karma, ma al contrario Karma ottiene risultati migliori sulla maggior parte delle colonne testuali come: categoria, prodotto, marca, descrizione breve, immagine e link. In particolare la differenza è evidente su immagine e link dove l'algoritmo di STAN ha MRR inferiore al 10% mentre quello di Karma si aggira intorno al 90%.



**Figura 6.1:** MRR calcolato sulle singole colonne per STAN e Karma.



**Figura 6.2:** Recall at 5 calcolata sulle singole colonne per STAN e Karma.

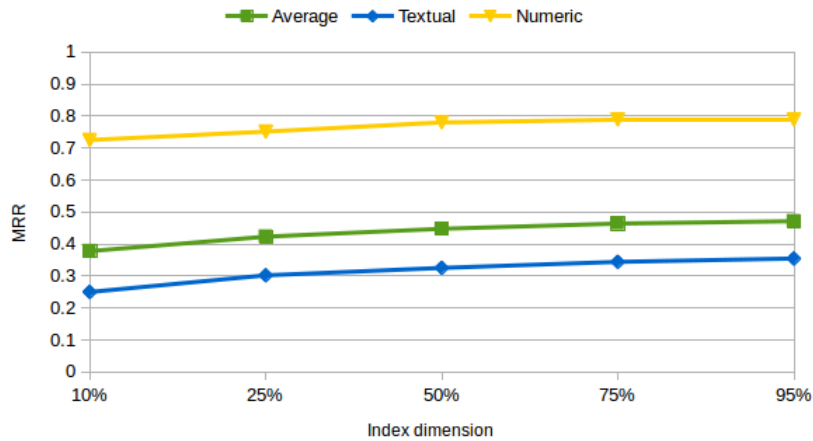
### 6.1.4 Qualit  in relazione alla dimensione dell'indice

La sperimentazione   stata condotta con la tecnica statistica della *k-fold cross-validation*. La tecnica prevede di suddividere casualmente l'insieme dei listini che compongono il dataset in  $k$  parti di uguale numerosit . Ad ogni passo  $k-1$  parti vengono utilizzate per popolare l'indice mentre sulla porzione rimanente dei listini si effettua l'annotazione. Questo viene ripetuto per ognuna delle  $k$  parti. In questo modo dunque   stata valutata la qualit  dei suggerimenti al variare della dimensione dell'indice. Nel dataset di 7Pixel, infatti, il volume di dati su cui si basa l'annotazione   molto grande, dunque condurre questo tipo di analisi   importante per valutare il comportamento degli algoritmi al crescere del volume di informazione. Per questo nei grafici presentati di seguito si metter  in evidenza l'andamento di MRR e *Recall* al crescere del volume dell'indice. In particolare la dimensione dell'indice   misurata in percentuale rispetto alla dimensione totale del dataset. Sebbene possa sembrare scontato che al crescere dell'informazione corrisponda un

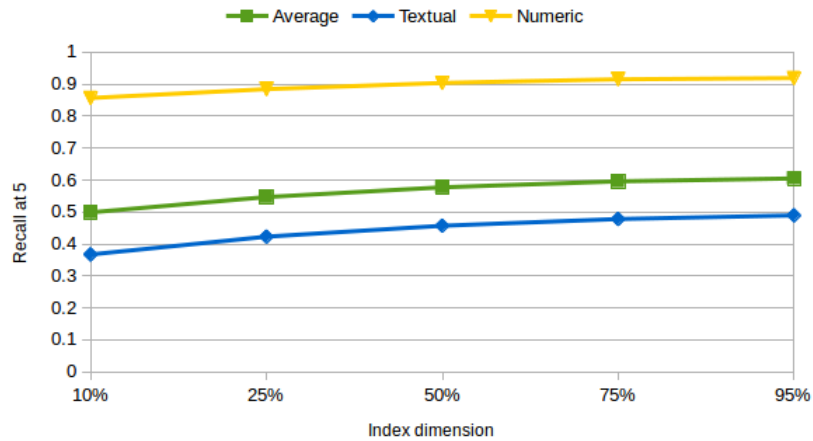
### 6.1. QUALITÀ DEI SUGGERIMENTI NEL CONTESTO DELL'ECOMMERCE

miglioramento della qualità si dimostrerà che nel caso di Karma questo non è vero.

Analizzando i grafici 6.3 e 6.4, sebbene l'incremento sia modesto, si nota che la qualità dei suggerimenti fornita da STAN migliora al crescere dell'informazione utilizzata per popolare l'indice. Infatti sia nel caso del MRR che della *Recall* si ha un incremento praticamente lineare in rapporto al volume dell'indice. In particolare questo andamento è evidente sia studiando le colonne di tipo testuale che numerico.



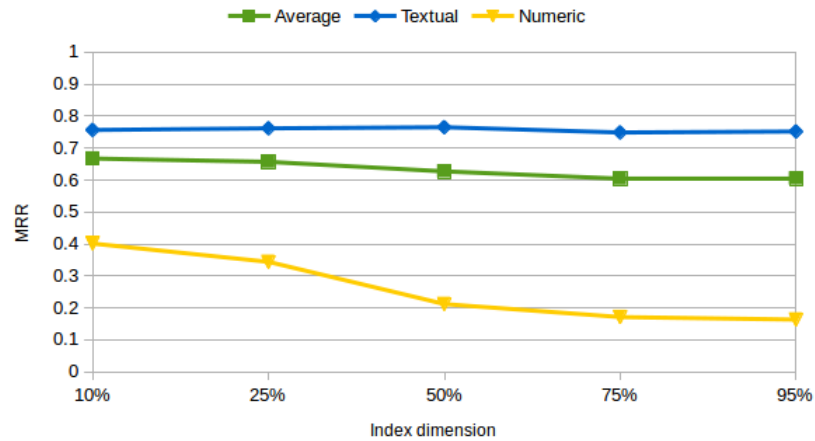
**Figura 6.3:** L'andamento del MRR al crescere della dimensione dell'indice per STAN.



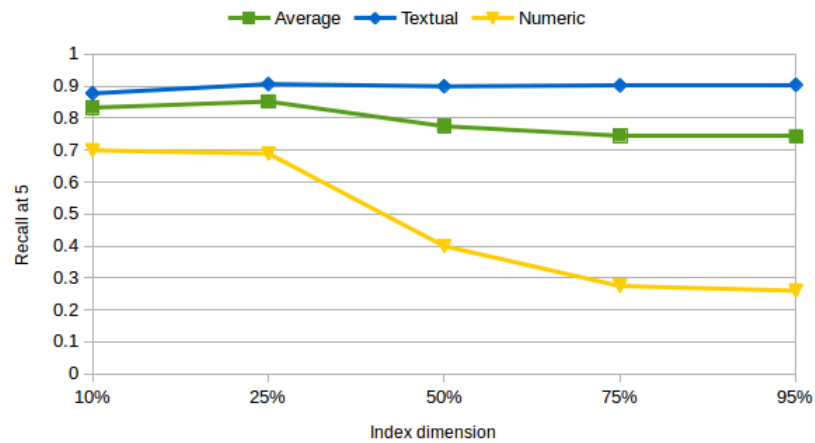
**Figura 6.4:** L'andamento della Recall at 5 al crescere della dimensione dell'indice per STAN.

## CAPITOLO 6. SPERIMENTAZIONE

Al contrario, osservando i grafici 6.5 e 6.6 risulta evidente che l'algoritmo di Karma ha un calo delle performance al crescere dell'informazione presente nell'indice. Quello che si evince, in particolare, è un netto peggioramento nella qualità dei suggerimenti relativi alle colonne numeriche. Questo probabilmente significa che il modello statistico su cui si basa Karma per l'annotazione dei dati numerici non è adatto per gestire grandi volumi di dati. Al contrario invece l'annotazione effettuata sulle colonne testuali non sembra essere particolarmente influenzata dalla dimensione dell'indice.



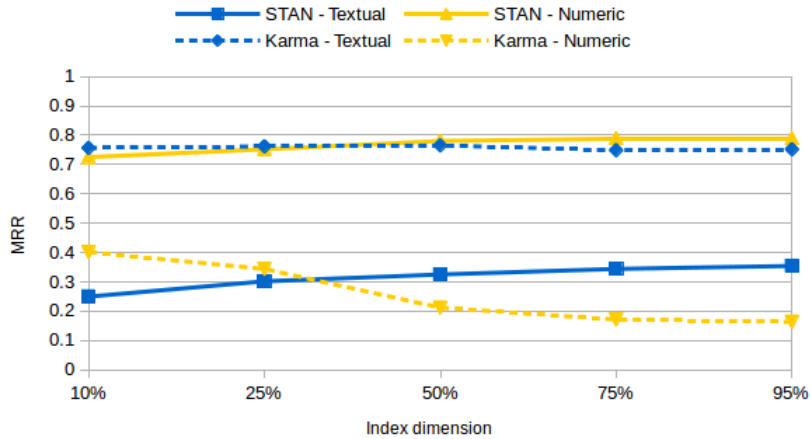
**Figura 6.5:** L'andamento del MRR al crescere della dimensione dell'indice per Karma.



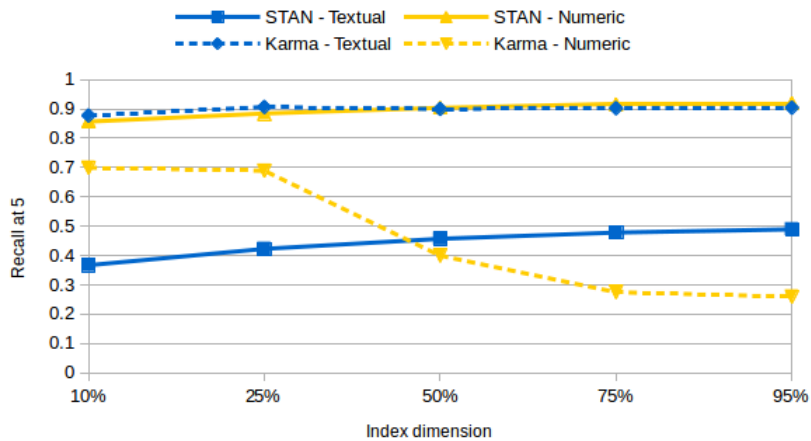
**Figura 6.6:** L'andamento della Recall at 5 al crescere della dimensione dell'indice per Karma.

### 6.1. QUALITÀ DEI SUGGERIMENTI NEL CONTESTO DELL'ECOMMERCE

Infine nelle figure 6.7 e 6.8 vengono analizzati MRR e Recall ponendo l'accento sul confronto diretto tra i risultati dei due algoritmi. Come si può notare si evince quello già espresso in precedenza, cioè che i due algoritmi hanno dei risultati praticamente complementari per quanto riguarda l'annotazione testuale e numerica. Infatti per entrambe le metriche si può notare come i due algoritmi ottengano risultati in un range simile di qualità ma per tipologia di annotazioni invertite.



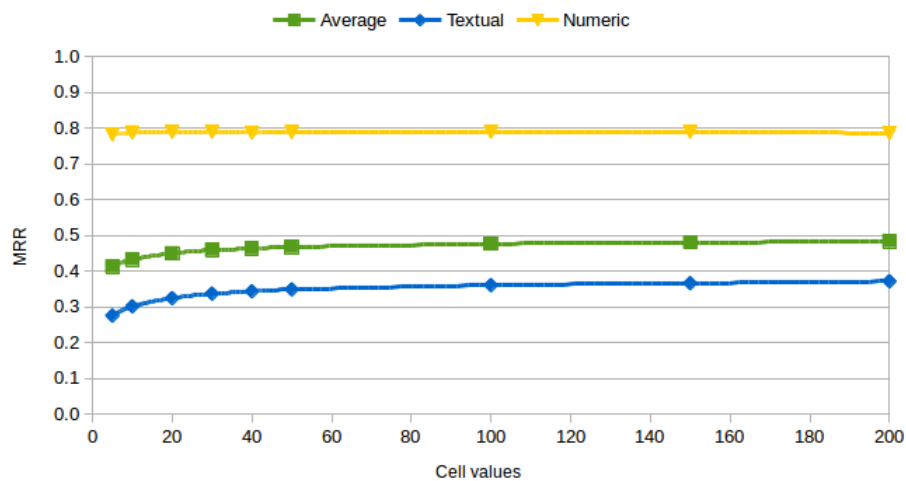
**Figura 6.7:** Il confronto tra STAN e Karma relativo al MRR al crescere della dimensione dell'indice.



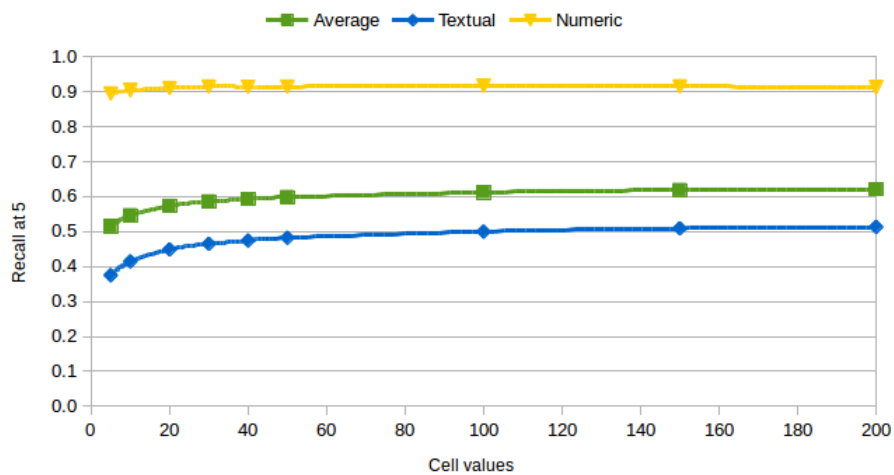
**Figura 6.8:** Il confronto tra STAN e Karma relativo alla Recall al crescere della dimensione dell'indice.

### 6.1.5 Qualità in relazione al numero di valori

Una ulteriore variabile su cui si è deciso di condurre la sperimentazione è il numero di celle utilizzate come campione della colonna per eseguire l'annotazione. In questo caso la sperimentazione è stata effettuata solo per STAN dato che questi risultati saranno analizzati in seguito in relazione ai tempi per decidere la dimensione migliore del campione da utilizzare. Inoltre l'algoritmo di Karma non prevede il campionamento dei valori delle colonne.



**Figura 6.9:** L'andamento del MRR al crescere del numero di celle per STAN.



**Figura 6.10:** L'andamento della Recall at 5 al crescere del numero di celle per STAN.



Come si evince dai grafici 6.9 e 6.10 l'andamento di MRR e *Recall* presenta una crescita logaritmica al crescere del numero di celle utilizzate come campione. Infatti si nota un aumento della qualità, seppur minimo, passando da 5 a 40 celle mentre aumentando oltre il numero di celle l'incremento nella qualità è quasi nullo. Un'altra cosa che emerge da questi grafici è il manifestarsi di questo comportamento solo nel caso delle colonne testuali. Per le colonne numeriche, infatti, il numero di celle utilizzate non sembra influire particolarmente sulla qualità dei suggerimenti.

## **6.2 Esperimenti di carico**

Per rendere STAN un provider di annotazioni sia all'interno di 7Pixel che fuori è necessario che i suoi tempi di risposta siano accettabili. Per questo motivo è stata condotta una sperimentazione sui tempi medi impiegati dall'algoritmo di STAN per effettuare l'annotazione di una colonna. La sperimentazione è stata condotta su entrambi i dataset oggetto della tesi: quello dell'eCommerce proprio di 7Pixel e quello delle School Closing di Chicago. L'obiettivo di questa sperimentazione è misurare le performance dell'algoritmo al variare di due fattori: il numero di celle utilizzato come campione della colonna e la dimensione dell'indice. Analizzando in questo modo il comportamento dell'algoritmo, e incrociando i risultati con l'output della sperimentazione sulla qualità dei suggerimenti si cercherà quindi di individuare la combinazione migliore di queste due variabili bilanciando performance e qualità.

### **6.2.1 Tempi nel contesto dell'eCommerce**

L'analisi dei tempi di annotazione medi per l'algoritmo di STAN nel contesto eCommerce mostra che il tempo di annotazione di una colonna è direttamente proporzionale sia al numero di celle che alla dimensione dell'indice. Per primi sono stati testati i tempi in relazione al numero di celle usate per l'annotazione. In questo caso si è deciso di fissare la dimensione dell'indice al 75% del totale del dataset. In questo modo si ha a disposizione un buon volume di informazione che consente di ottenere una buona qualità nei suggerimenti come si evince dalla relativa sperimentazione. Analizzando quindi i risultati in tabella 6.3 emerge che i tempi crescono in proporzione al numero di celle e in particolare il tempo medio per annotare una colonna va da poco più di 1 secondo a quasi 3.

Numero di celle	Tempo
5	1,225 s
10	1,407 s
20	1,677 s
30	2,248 s
40	2,352 s
50	2,487 s
100	2,519 s
150	2,608 s
200	2,704 s

**Tabella 6.3:** I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto eCommerce in rapporto al numero di celle.

Allo stesso modo fissando un numero di celle ragionevole sono stati calcolati i tempi medi in rapporto alla dimensione dell'indice. In particolare si è scelto di condurre questa sperimentazione fissando il numero di celle utilizzate a 50. Come si può notare nella tabella 6.4 i tempi di annotazione medi di una colonna sono direttamente proporzionali alla dimensione dell'indice e i tempi variano da circa 350 millisecondi a circa 3,5 secondi.

Dimensione dell'indice	Tempo
10%	347 ms
25%	1,283 s
50%	1,810 s
75%	2,612 s
95%	3,534 s

**Tabella 6.4:** I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto eCommerce in rapporto alla dimensione dell'indice.

Dopo aver studiato questi risultati in rapporto alla qualità dei suggerimenti forniti dall'algoritmo si è deciso di utilizzare all'incirca 40 celle per l'annotazione. Oltre questa soglia, infatti, non si notano particolari miglioramenti nella qualità dei suggerimenti mentre i tempi variano di poco nell'intorno di questo numero di celle. Per quanto riguarda la dimensione dell'indice è ovviamente sempre meglio avere un dataset di allenamento il più ricco possibile in modo da avere maggiori possibilità di annotare un nuovo listino. Nel caso di problemi di performance tuttavia si può pensare di ridurre la dimensione dell'indice al 75% selezionando tra tutti i listini quelli che in qualche modo sono più informativi e che facilitano l'annotazione di quelli nuovi.

### 6.2.2 Tempi nel contesto School Closing

Nel condurre questa sperimentazione l'unica variabile che è stata analizzata è il numero di celle. In questo caso infatti l'indice è composto da tutte le istanze della knowledge base di DBPedia e non avrebbe senso frazionarlo. L'annotazione è invece testata sulle tabelle appartenenti al dataset dello scenario delle School Closing di Chicago.

Come emerge dall'osservazione della tabella 6.5 anche in questo caso il tempo medio di annotazione di una colonna è direttamente proporzionale al numero di celle utilizzate come campione. In particolare però si può notare che in questo caso l'annotazione impiega decisamente più tempo. Infatti il range dei tempi va da circa 10 secondi fino a circa 40 secondi annotando con 50 valori. La scelta del numero di celle ottimo in questo caso è stata condotta solo sulla base dei tempi. La sperimentazione relativa alla qualità dei suggerimenti per questo dataset, infatti, è stata effettuata nel corso di un altro lavoro. Considerando che i tempi medi di annotazione sono più elevati rispetto al caso dell'eCommerce si è deciso di selezionare un numero di celle inferiore. In particolare si è deciso di utilizzarne 20 in modo da avere un tempo medio di annotazione intorno ai 15 secondi.

Numero di celle	Tempo
5	9,776 s
10	13,352 s
20	14,898 s
30	20,036 s
40	24,381 s
50	39,313 s

**Tabella 6.5:** I tempi di annotazione di una colonna per l'algoritmo di STAN nel contesto School Closing in rapporto al numero di celle.

## 6.3 Considerazioni

Dai risultati della sperimentazione è emerso che STAN e Karma si comportano in modo molto diverso a seconda che si trattino colonne di tipo testuale o numerico. Tra i due approcci Karma ottiene risultati migliori sulle colonne testuali, al contrario fa STAN per le colonne numeriche. Sebbene le colonne numeriche siano in numero inferiore rispetto a quelle testuali spesso nel dominio di 7Pixel si rivelano essere tra le più ambigue. Infatti, in molti casi risulta difficile distinguere colonne come prezzo, disponibilità e spese di spedizione quando il loro ordine di grandezza è molto simile. Per questo tipo di colonne dunque l'algoritmo si rivela più che sufficiente per l'annotazione. Tuttavia per quanto riguarda le colonne testuali la qualità dei suggerimenti può essere migliorata in modo da fornire a 7Pixel un

servizio migliore. L'algoritmo infatti è stato applicato senza alcun adattamento al dominio specifico, e questo dunque lascia buoni margini di miglioramento. In particolare gli scarsi risultati ottenuti da STAN nell'annotazione delle colonne di tipo testuale possono essere migliorati introducendo alcune euristiche specifiche di dominio. Ad esempio riconoscere alcuni pattern ricorrenti come "http://" aiuterebbe facilmente la corretta individuazione di colonne come *Link* e *Immagine* per le quali l'algoritmo attualmente ottiene risultati pessimi.

Dall'analisi dei tempi medi di annotazione, inoltre, risulta evidente che l'algoritmo è troppo lento per gli standard di una applicazione web. Sul web, infatti, gli utenti sono poco tolleranti riguardo alla velocità di risposta di una applicazione. Secondo alcuni studi di Human Computer Interaction, infatti, per trasmettere agli utenti un'opinione positiva è necessario che non vi siano ritardi superiori agli 8 secondi [20]. Oltre gli 8 secondi infatti gli utenti perdono attenzione nell'interazione ed aumenta la probabilità che abbandonino il sito. Attualmente i tempi medi di annotazione registrati per il caso di studio delle School Closing si aggirano intorno ai 15 secondi, per cui ben oltre gli 8. Sarà dunque necessario investire del tempo per capire come rendere più efficiente l'annotazione. L'ottimizzazione dei tempi dell'algoritmo è inoltre necessaria al fine di rendere più efficiente il servizio di API di STAN.

### 6.4 Vantaggi dell'approccio incrementale

Come già descritto in precedenza STAN implementa un approccio incrementale nella definizione di una ontologia locale che consente agli utenti di specificare un proprio vocabolario man mano che si annota la tabella. In questo modo non si è costretti a definire in precedenza l'ontologia per importarla nel tool come in Karma e Open Refine. In questa sezione verrà analizzato l'utilizzo di STAN in confronto proprio a Karma ed Open Refine nel caso dell'annotazione delle tabelle del caso di studio delle School Closing. L'obiettivo di questa sperimentazione è proprio quello di dimostrare quanto sia utile poter sfruttare un approccio incrementale piuttosto che un approccio di tipo *top-down* in cui l'ontologia viene definita prima di annotare una tabella.

Provando ad annotare le tabelle del caso di studio delle School Closing di Chicago è risultato evidente che non sempre le ontologie condivise esistenti sono sufficienti per assegnare ad una tabella la semantica che si vorrebbe. In particolare è emerso che nel 32% dei casi (tabella 6.6) è risultato necessario ricorrere ad una annotazione non propria di un'ontologia esistente. Principalmente i casi in cui si deve ricorrere ad un'ontologia locale sono legati a quelle colonne la cui semantica è strettamente legata al dominio della tabella oppure che contengono dati di tipo statistico. Nel primo caso rientrano colonne contenenti identificativi di dominio come *ISBE ID* per le scuole di Chicago e *FBI Code* per i crimini, oppure anche

#### 6.4. VANTAGGI DELL'APPROCCIO INCREMENTALE

colonne di dettaglio come le colonne *Arrest* e *Domestic* che descrivono un crimine. Le colonne del secondo caso, invece, sono tutte quelle colonne che descrivono dati statistici con una terminologia propria del dominio. La tabella legata ai dati sulla povertà di Chicago, ad esempio, contiene per lo più dati statistici come la percentuale di case sovraffollate o la percentuale di cittadini senza lavoro la cui semantica non può essere rappresentata se non con una proprietà definita appositamente per il dominio.

Tabella	Colonne annotate con ontologia locale
Scuole	27%
Censimento	0%
Povertà	77%
Crimini	23%
<b>Media</b>	32%

**Tabella 6.6:** La percentuale di colonne annotate con una ontologia locale per ciascuna tabella del caso di studio delle School Closing.

Come è stato descritto spesso dunque risulta necessario definire una proprietà o un concetto di un vocabolario locale specifico per la tabella che si sta annotando. Per gestire questa situazione sia in Karma che in Open Refine è necessario interrompere l'annotazione e creare un'ontologia in formato OWL da importare. Al contrario con STAN l'effort necessario è minimo dato che è sufficiente definire la nuova proprietà o concetto direttamente tramite l'interfaccia dell'applicazione.



## Capitolo 7

# Sviluppi futuri

STAN è stato progettato con un duplice fine: l'utilizzo come provider di mapping per il caso specifico dell'eCommerce in 7Pixel e come strumento di annotazione di dati eterogenei nel mondo degli Open Data. In questo capitolo dunque vengono analizzate le prospettive future del tool che molto probabilmente prevedono la separazione di queste due anime e la nascita di due strumenti diversi ma con un focus comune. Innanzitutto, dunque, vengono presentati i due scenari in cui in futuro ci si aspetta di vedere l'applicazione di STAN, in seguito vengono presentate alcune funzionalità aggiuntive di carattere più trasversale che si vorrebbe implementare nel tool.

### 7.1 Utilizzo in 7Pixel

Al momento STAN è deployato nei sistemi di 7Pixel, tuttavia non è ancora utilizzato. Quello che si vorrebbe fare nei prossimi mesi è integrare le funzionalità di STAN nel ciclo di mappatura dei listini dei commercianti. Come già accennato in precedenza nei sistemi di 7Pixel esiste già un software in grado di supportare la mappatura dei listini che viene utilizzato dai responsabili dei contenuti dei siti di TrovaPrezzi.it e Kirivo.it. In particolare questo software viene utilizzato per classificare ciascuna offerta di un listino all'interno della categoria corretta. Tuttavia per poter classificare le offerte è necessario che prima vengano riconosciuti i campi che descrivono il listino definendo dei mapping tra i campi utilizzati dal commerciante e l'ontologia di 7Pixel. Al momento gli esperti di dominio effettuano questa operazione a mano ma con l'utilizzo di STAN il loro compito potrebbe essere ridotto al solo supervisionamento. Per cui nei prossimi mesi si implementerà nel software di 7Pixel un'estensione per consentire l'interrogazione delle API di STAN e utilizzare le annotazioni suggerite come valori per i campi del listino.

In aggiunta, 7Pixel sta pensando di iniziare lo sviluppo di uno strumento di annotazione da fornire ai commercianti in modo da alleggerirsi di una parte del lavoro di mappatura. I commercianti dunque sarebbero chiamati ad utilizzare l'ontologia di 7Pixel per annotare il proprio listino sebbene essi non conoscano la semantica

utilizzata da 7Pixel per mappare i campi. A maggior ragione quindi in un caso come questo STAN si rivelerebbe utile come provider e suggeritore di annotazioni e per questo è già stata inoltrata una proposta a 7Pixel per l'integrazione di STAN anche in questa applicazione.

### 7.2 Progetto Open Source

Al di fuori di 7Pixel si cercherà di promuovere STAN come strumento Open Source per l'annotazione di dati tabellari in formato Open Data in modo da supportare la comunità scientifica nella definizione di dati con una qualità a cinque stelle [25]. STAN, infatti, potrebbe essere usato nella comunità degli Open Data o dalle pubbliche amministrazioni come strumento per tradurre facilmente dati in formato tabellare in dati di qualità in formato RDF.

Quello che si vorrebbe fare è condividere pubblicamente il codice sorgente di STAN per favorire la collaborazione tra le persone della comunità Open Source per lo sviluppo congiunto di nuove funzionalità. Il codice potrebbe essere condiviso, ad esempio, sulla piattaforma GitHub<sup>1</sup> in seguito alla rimozione dalla codebase di tutti i riferimenti ai dati commerciali di 7Pixel.

### 7.3 Endpoint SPARQL

Al momento STAN non memorizza le triple ma le genera solo nel momento in cui viene chiesto di produrre l'export da scaricare. Tuttavia se STAN memorizzasse le triple generate per ciascuna tabella annotata dall'utente si potrebbe esporre un endpoint SPARQL tramite il quale consentire agli utenti di effettuare interrogazioni sulle tabelle integrate. In questo modo STAN diventerebbe un tool di integrazione completo e gli utenti non dovrebbero più utilizzare un proprio triple store per usufruire dei dati integrati.

Per sviluppare tale funzionalità si potrebbe integrare in STAN un server Virtuoso<sup>2</sup>, un software Open Source dedicato alla creazione di endpoint SPARQL. Virtuoso infatti consente di costruire un server per ospitare dati in formato RDF e al di sopra di essi si occupa automaticamente di esporre un endpoint SPARQL per effettuare interrogazioni sui dati sia tramite interfaccia grafica che tramite API. Una volta integrato il server Virtuoso dunque gli utenti avranno un'ulteriore opzione per usufruire dei dati annotati. Oltre alla possibilità di scaricare i mapping e le triple RDF, infatti, agli utenti sarà data la possibilità di aggiornare il proprio "triple store online" con i dati contenuti nella tabella appena annotata integrandoli con i dati annotati in precedenza.

---

<sup>1</sup><https://github.com/>

<sup>2</sup><http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main>



## **7.4 Approfondimento su ABSTAT**

Per supportare ulteriormente gli utenti nella scelta dell'annotazione migliore si potrebbe sfruttare ABSTAT come fonte per approfondire la semantica e l'utilizzo di una proprietà. Infatti al momento l'utente seleziona una proprietà con l'ausilio di un sistema di autocompletamento e dei suggerimenti forniti da STAN tuttavia nessuno di questi due elementi descrive come quella proprietà viene utilizzata all'interno di un dataset. Un utente potrebbe infatti ignorare che l'interpretazione che ha dato del significato di una proprietà è sbagliata e potrebbe utilizzarla in modo non corretto producendo dati di cattiva qualità.

Una soluzione che non sia troppo restrittiva per l'utente ma che allo stesso tempo gli consenta di utilizzare correttamente le proprietà di una knowledge base potrebbe essere l'introduzione in STAN di un link di approfondimento su ABSTAT. Come già detto ABSTAT fornisce ai suoi utilizzatori una migliore comprensione di dataset complessi e voluminosi e, in particolare, è in grado di fornire informazioni riguardo a come vengono utilizzate le proprietà all'interno del dataset descrivendo quali tipi di dati mettono in relazione e con che frequenza. Per implementare la modifica si potrebbe generare in STAN un link di approfondimento per ogni proprietà suggerita. ABSTAT già consente la visualizzazione delle informazioni che servono agli utenti di STAN tuttavia è necessario sviluppare un supporto per la gestione dei link di approfondimento provenienti da STAN.

## **7.5 Supporto di altre knowledge base**

Al momento STAN supporta l'annotazione con le proprietà e i tipi dell'ontologia di DBPedia ma in futuro si vorrebbe dare agli utenti la possibilità di scegliere tra diverse knowledge base e ontologie. Ad esempio per cominciare si potrebbe fornire il supporto per ontologie note come FOAF, Schema.org, Dublin Core, Yago, GeoNames, ... Questo significherebbe dare agli utenti la possibilità di configurare in STAN le ontologie da utilizzare per l'annotazione e, in base alla configurazione, fare in modo che suggerimenti e autocompletamento siano coerenti.

## **7.6 Supporto di altri formati**

Così come si prevede di supportare nuove knowledge base e ontologie per l'annotazione, in futuro si prevede di estendere STAN in modo che sia in grado di supportare altri formati per l'import di dati tabellari. Al momento infatti STAN supporta soltanto l'import di dati in formato CSV ma in futuro il supporto potrebbe essere esteso ai formati XML e JSON considerando che il linguaggio di mapping utilizzato (RML) è già in grado di supportare tali formati.

Inoltre dato che STAN è pensato per annotare tabelle estratte da pagine web un ulteriore formato di dati interessante da gestire potrebbe essere l'HTML. Nel

linguaggio di markup HTML infatti esistono dei tag specifici che definiscono la struttura di una tabella in termini di righe e colonne. Dato un indirizzo web STAN potrebbe essere in grado di analizzare la pagina alla ricerca dei tag HTML che descrivono una tabella per poi importarla automaticamente.

### 7.7 Filtraggio object type

Come già introdotto nella sezione 5.3.5 al momento STAN consente di specificare l'object type di una proprietà in modo molto permissivo. Questo permette in alcuni casi di produrre delle annotazioni di cattiva qualità. Per questo tra gli sviluppi futuri si prevede anche di limitare l'occorrere di tale situazione. L'obiettivo non è quello di limitare l'utente nelle sue azioni ma di facilitarlo nella definizione di un object type coerente alla proprietà scelta e conseguentemente facilitarlo a produrre dati di qualità. Un'idea potrebbe essere quella di evidenziare in qualche modo gli object type corretti così da invitare l'utente a selezionarli. Un'alternativa invece potrebbe essere notificare l'utente qualora scelga un object type non corretto.

### 7.8 Workspace ampliata

Al momento la workspace personale messa a disposizione da STAN prevede la memorizzazione solo dell'ultima tabella su cui si è lavorato. In futuro si potrebbe pensare di estendere il numero di tabelle che il sistema è in grado di memorizzare per ciascun utente. In questo modo si darebbe la possibilità agli utenti di tenere traccia di tutti o parte dei propri progetti di annotazione per eventualmente portarne avanti anche più di uno alla volta. Questo implica inoltre la creazione di un pannello di gestione della propria workspace per fornire all'utente alcune funzionalità legate ai progetti come la creazione di un nuovo progetto, la selezione di un progetto o la cancellazione di esso.

### 7.9 Selezione del campione più rappresentativo

Al momento il processo di annotazione di una colonna prevede la selezione casuale di un campione di celle utilizzando come unico criterio l'univocità di ciascun valore selezionato. La scelta di selezionare casualmente le celle è stata dettata dalle esigenze di tempo della tesi tuttavia presenta almeno due problemi. Per prima cosa l'annotazione per una stessa colonna può variare a seconda degli elementi selezionati. Inoltre la selezione casuale potrebbe escludere alcuni valori rilevanti della colonna che invece ridurrebbero l'ambiguità in fase di annotazione.

Per questo motivo in futuro si vuole integrare nell'algoritmo di annotazione un criterio di selezione che sia più ponderato rispetto alla semplice selezione casuale. In particolare si vuole cercare di selezionare il campione di celle più rappresentativo della colonna, cioè quell'insieme di valori che più probabilmente porta ad annotare

## **7.9. SELEZIONE DEL CAMPIONE PIÙ RAPPRESENTATIVO**

---

correttamente la colonna. Non è ancora ben chiaro come fare a discriminare tra i valori per selezionare i più rappresentativi, tuttavia un punto di partenza potrebbe essere la selezione di valori frequenti e eterogenei.



## Capitolo 8

# Conclusioni

In questa tesi è stato presentato STAN, un tool disponibile come applicazione web per l'annotazione semantica di dati in formato tabellare. La progettazione e lo sviluppo dell'applicazione sono stati guidati da una profonda analisi dello stato dell'arte e di alcuni casi d'uso. Grazie a questi casi concreti è stato inoltre possibile studiare i pattern più ricorrenti di annotazione in modo da definire formalmente una serie di modelli che sono stati poi utilizzati con un duplice obiettivo: come linee guida per lo sviluppo delle funzionalità di STAN e come metro di paragone tra STAN e gli altri tool esistenti da stato dell'arte.

Grazie a STAN è possibile ricostruire la semantica di una tabella sfruttando una o più ontologie. Annotare semanticamente più tabelle sfruttando la stessa ontologia di fatto riproduce un tipo di architettura classica per i sistemi di Data Integration: la virtualizzazione. Infatti dove nella Data Integration classica si producono mapping verso uno schema globale costruito sopra i database da integrare in questo caso si producono mapping verso una ontologia condivisa a partire da un insieme di tabelle da integrare. STAN, infatti, date in input una tabella ed una ontologia è in grado di produrre i mapping per definire la tabella nei termini dell'ontologia. La definizione dei mapping viene eseguita dagli utenti mediante una interfaccia grafica e non richiede quindi il possesso di particolari competenze tecniche. Utilizzando la stessa ontologia per diverse tabelle dunque STAN è in grado di abilitare l'integrazione dei dati contenuti nelle diverse tabelle.

STAN ovviamente non è l'unico strumento esistente da stato dell'arte che consente di annotare semanticamente dati in formato tabellare tuttavia si ritiene che possieda alcune caratteristiche che lo contraddistinguano dagli altri tool. Innanzitutto STAN consente di definire una propria ontologia in maniera incrementale durante il processo di annotazione e questo permette agli utenti di utilizzare una semantica propria per la strutturazione dei dati. Un'altra differenza tra STAN e gli altri tool è la struttura puramente pensata per il web. STAN infatti è una applicazione web utilizzabile online che permette a ciascun utente di avere il suo

spazio di lavoro personale. Gli altri tool invece sono pensati per essere scaricati ed utilizzati da un singolo utente sulla propria macchina. STAN inoltre a differenza degli altri tool incorpora al suo interno un algoritmo di annotazione automatica *instance based* basato sul confronto di un campione dei valori della colonna con i valori di migliaia di proprietà di una knowledge base. Inoltre STAN è l'unico tool che, sfruttando il proprio algoritmo di annotazione, espone un servizio di API pubbliche permettendo ad applicazioni di terze parti di effettuare l'annotazione automatica di gruppi di valori. Infine STAN a differenza degli altri tool è uno strumento modulare che può essere esteso con diversi algoritmi di annotazione. L'unico requisito è che i nuovi algoritmi rispettino il formato utilizzato nelle API.

Un contributo della tesi è anche la sperimentazione condotta proprio sull'algoritmo di annotazione per valutarne la qualità e le performance in relazione allo stato dell'arte. La sperimentazione è infatti stata condotta per studiare l'algoritmo sotto due diversi punti di vista: per valutarne la qualità dei suggerimenti nel caso specifico dell'eCommerce e per valutarne le performance in termini di tempo. Sebbene l'algoritmo non sia un contributo della tesi, uno degli obiettivi è quello di integrarsi con i sistemi di 7Pixel e quindi la sperimentazione è necessaria per garantire all'azienda lo sviluppo di un tool di qualità. La sperimentazione dunque è stata condotta esclusivamente su un dataset composto da listini commerciali di proprietà dell'azienda utilizzando come verità le annotazioni effettuate in passato. Nel paragone con lo stato dell'arte si è scelto di confrontarsi con l'algoritmo utilizzato in Karma. Karma, infatti è uno dei pochi tool completi di Table Annotation con cui si confronta STAN, inoltre l'algoritmo implementato in Karma è l'unico il cui codice sia accessibile pubblicamente.

Dai risultati della sperimentazione è emerso che STAN e Karma si comportano in modo molto diverso a seconda che si trattino colonne di tipo testuale o numerico. Tra i due approcci Karma ottiene risultati migliori sulle colonne testuali, al contrario fa STAN per le colonne numeriche. Sebbene le colonne numeriche siano in numero inferiore rispetto a quelle testuali spesso nel dominio di 7Pixel si rivelano essere tra le più ambigue. Infatti, in molti casi risulta difficile distinguere colonne come prezzo, disponibilità e spese di spedizione quando il loro ordine di grandezza è molto simile. La sperimentazione è stata condotta senza adattare gli algoritmi al dominio specifico e questo dunque lascia ampi margini di miglioramento. In particolare gli scarsi risultati ottenuti da STAN nell'annotazione delle colonne di tipo testuale possono essere migliorati introducendo alcune euristiche specifiche di dominio. Ad esempio riconoscere alcuni pattern ricorrenti come "http://" aiuterebbe facilmente la corretta individuazione di colonne come *Link* e *Immagine* per le quali l'algoritmo attualmente ottiene risultati pessimi.

Un'ulteriore osservazione grazie alla sperimentazione, inoltre, è che non sempre le ontologie condivise esistenti sono sufficienti per assegnare ad una tabella la

---

semantica che si vorrebbe. Spesso infatti quando si trattano dati di un dominio molto specifico l'utilizzo di ontologie condivise risulta essere troppo generico e non sufficiente a designare la corretta semantica di una tabella. In questi casi dunque è necessario definire ed utilizzare un vocabolario locale al dominio della tabella. Questa situazione è gestita efficacemente da STAN mentre lo è meno in altri tool esistenti come Karma e Open Refine. Infatti, Karma e Open Refine adottano un approccio *top down* in cui è necessario aver definito l'ontologia completa da utilizzare prima di iniziare l'annotazione di una tabella. Per definire dunque una nuova proprietà sarebbe necessario interrompere l'annotazione, creare un'ontologia in formato OWL e infine importarla. Al contrario con STAN l'effort necessario è minimo dato che è sufficiente definire la nuova proprietà o concetto direttamente tramite l'interfaccia dell'applicazione.

Lo sviluppo di STAN è stato pensato principalmente con due finalità: l'integrazione nei sistemi di 7Pixel e la diffusione come tool Open Source di annotazione. Al momento queste due anime dell'applicazione convivono e la logica che le differenzia viene specificata solamente in fase di deployment. Tuttavia, probabilmente in futuro queste due anime potrebbero divergere sempre più con l'introduzione di nuove funzionalità e quindi gestire la logica che le differenzia solamente in fase di deployment potrebbe diventare a tendere troppo oneroso e complesso. Per questo in futuro si prevede di scindere STAN in due progetti esaltando individualmente le sue due anime. Nel caso di 7Pixel ci si concentrerà sul rendere l'algoritmo di annotazione più performante nel dominio dell'eCommerce introducendo alcune euristiche specifiche che consentano di ottenere suggerimenti di annotazione di maggior qualità. Al di fuori di 7Pixel invece si cercherà di promuovere STAN come strumento Open Source per l'annotazione di dati tabellari in formato Open Data e per questo ci si concentrerà sull'interfaccia grafica e sull'aggiunta di nuove funzionalità.





# Bibliografia

- [1] Ben Adida, Mark Birbeck, Shane McCarron, and Ivan Herman. Rdfa core 1.1: Syntax and processing rules for embedding rdf through attributes. August 2013.
- [2] Carlo Batini, Marco Cremaschi, Angela Locoro, Andrea Maurino, Matteo Palmonari, Anisa Rula, Blerina Spahiu, and Claudio Venturini. Methodologies for deployment and usage of the comsode publication platform (odn), tools and data. Technical report, DISCO, Dipartimento di informatica, sistemistica e comunicazione, September 2014.
- [3] Beck. *Test Driven Development: By Example*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [4] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.
- [5] Herbert D. Benington. Production of large computer programs. *Annals of the History of Computing*, 5(4):350–361, Oct 1983.
- [6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
- [7] P. Buche, J. Dibia-Barthelemy, L. Ibanescu, and L. Soler. Fuzzy web data tables integration guided by an ontological and terminological resource. *Knowledge and Data Engineering, IEEE Transactions on*, 25(4):805–819, April 2013.
- [8] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, August 2008.
- [9] Isabel F. Cruz, Venkat R. Ganesh, Claudio Caletti, and Pavan Reddy. GIVA: a semantic framework for geospatial and temporal data integration, visualization, and analytics. In *21st SIGSPATIAL International Conference on*

## BIBLIOGRAFIA

---

- Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, pages 534–537, 2013.
- [10] Isabel F. Cruz, Venkat R. Ganesh, and Seyed Iman Mirrezaei. Semantic extraction of geographic data from web tables for big data integration. In *Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR 2013, 5th November, 2013, Orlando, Florida, USA*, pages 19–26, 2013.
  - [11] Isabel F. Cruz and Huiyong Xiao. The Role of Ontologies in Data Integration. *Journal of Engineering Intelligent Systems*, 13(4), 2005.
  - [12] Souripriya Das, Seema Sundara, and Richard Cyganiak. R2rml: Rdb to rdf mapping language (w3c working draft), 2011.
  - [13] Marisa de la Torre, Molly F. Gordon, Paul Moore, and Jennifer Cowhy. School closings in chicago, understanding families’ choices and constraints for new school enrollment. *UChicago Consortium*, 2015.
  - [14] A. Dimou, M.V. Sande, J. Slepicka, P. Szekely, E. Mannens, C. Knoblock, and R. Van de Walle. Mapping hierarchical sources into rdf using the rml mapping language. In *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pages 151–158, June 2014.
  - [15] Ying Ding, Dieter Fensel, Michel C. A. Klein, Borys Omelayenko, and Ellen Schulten. The role of ontologies in ecommerce. In *Handbook on Ontologies*, pages 593–616. 2004.
  - [16] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.
  - [17] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 976–987, March 2014.
  - [18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. 1999.
  - [19] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
  - [20] Dennis F. Galletta, Raymond Henry, Scott McCoy, and Peter Polak. Web site delays: How tolerant are users? *JOURNAL OF THE ASSOCIATION FOR INFORMATION SYSTEMS*, 5:1–28, 2003.
  - [21] Jonathan Gray, Lucy Chambers, and Liliana Bounegru, editors. *The Data Journalism Handbook*. Oreilly & Associates Inc, 2012. CC BY-SA 3.0.

- 
- [22] Alon Y. Halevy. Structured data on the web. In Yishai A. Feldman, Donald Kraft, and Tsvi Kuflik, editors, *Next Generation Information Technologies and Systems*, volume 5831 of *Lecture Notes in Computer Science*, pages 2–2. Springer Berlin Heidelberg, 2009.
- [23] Z. R. Hesabi, Z. Tari, A. Goscinski, A. Fahad, I. Khalil, and C. Queiroz. *Data Summarization Techniques for Big Data—A Survey*. Springer, 2015.
- [24] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at <http://www.w3.org/TR/owl2-primer/>.
- [25] Krzysztof Janowicz, Pascal Hitzler, Benjamin Adams, Dave Kolas, and Charles Vardeman. Five stars of linked data vocabulary use. *Semantic Web*, 5(3):173–176, 2014.
- [26] Craig A. Knoblock, Pedro Szekely, Jose Luis Ambite, Shubham Gupta, Aman Goel, Maria Muslea, Kristina Lerman, Mohsen Taheriyan, and Parag Mallick. Semi-automatically mapping structured sources into the semantic web. In *Proceedings of the Extended Semantic Web Conference*, Crete, Greece, 2012.
- [27] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.
- [28] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [29] E. L. Lehmann and Joseph P. Romano. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, New York, third edition, 2005.
- [30] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233–246, New York, NY, USA, 2002. ACM.
- [31] Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Halevy. Harnessing the deep web: Present and future.
- [32] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

## BIBLIOGRAFIA

---

- [33] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic Message Passing for Generating Linked Data from Tables. In *Proceedings of the 12th International Semantic Web Conference*. Springer, October 2013.
- [34] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. Publisher: John Benjamins Publishing Company.
- [35] Matteo Palmonari, Anisa Rula, Riccardo Porrini, Andrea Maurino, Blerina Spahiu, and Vincenzo Ferme. Abstat: Linked data summaries with abstraction and statistics. In Fabien Gandon, Christophe Guéret, Serena Villata, John Breslin, Catherine Faron-Zucker, and Antoine Zimmermann, editors, *The Semantic Web: ESWC 2015 Satellite Events*, volume 9341 of *Lecture Notes in Computer Science*, pages 128–132. Springer International Publishing, 2015.
- [36] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.
- [37] S.K. Ramnandan, Amol Mittal, Craig A. Knoblock, and Pedro Szekely. Assigning semantic labels to data sources. In Fabien Gandon, Marta Sabou, Harald Sack, Claudia d’Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann, editors, *The Semantic Web. Latest Advances and New Domains*, volume 9088 of *Lecture Notes in Computer Science*, pages 403–417. Springer International Publishing, 2015.
- [38] Sunita Sarawagi and Soumen Chakrabarti. Open-domain quantity queries on web tables: Annotation, response, and consensus models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, pages 711–720, New York, NY, USA, 2014. ACM.
- [39] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006.
- [40] Oreste Signore. Rdf per la rappresentazione della conoscenza. KM2002 Conference Proceedings, Milan, March 27-28 March, 2002, 2002.
- [41] James Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.
- [42] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.
- [43] Ziqi Zhang. Towards efficient and effective semantic table interpretation. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pages 487–502, 2014.