Query in linguaggio naturale per il dominio della dieta mediterranea

Luca Anselma, Dario Ferrero, Alessandro Mazzei
Dipartimento di Informatica, Università di Torino, Italy
{anselma,mazzei}@di.unito.it, dario.ferrero@edu.unito.it

Abstract

English. This paper presents an ongoing work for allowing users to ask questions in natural language to a database management system on the domain of recipes. The system translates questions from Italian language to SQL exploiting an interlingua represented by a logical formalism such as relational calculus. There are two specific features of this project: first, the use of relational calculus as semantic representation for interlingua translation; second, the role played by pragmatic information, that is the Mediterranean diet domain implicitly encoded in the database schema.

Italiano. *Questo articolo presenta un progetto in corso sulla per permettere a degli utenti di porre domande a un database management system nell'ambito del dominio delle ricette. Il sistema traduce le domande da italiano a SQL sfruttando una rappresentazione logica interlingua che consiste nel calcolo relazionale. Ci sono due specifiche caratteristiche di questo progetto: l'uso del calcolo relazionale come rappresentazione semantica della traduzione interlingua e il ruolo giocato dall'informazione pragmatica, che consiste nell'uso del dominio della dieta mediterranea che è stato codificata nello schema del database.*

1 Introduzione

L'accesso a grandi moli di dati da parte di persone comuni è una delle molteplici sfide affrontata da anni per permettere una più fluida interazione uomo-macchina. La natura della comunicazione umana presenta molti aspetti difficili da formalizzare tramite regole precise: ambiguità nell'uso di parole ed espressioni, differenze culturali ed idiomatiche, l'affidamento ad informazioni implicite date dal contesto. Questo è vero anche in specifici domini, come quello che lega le persone al cibo (Jurafsky, 2014). A questo scopo da diversi anni sono state argomento di ricerca le *Interfacce al Linguaggio Naturale (NLIs)*, sistemi in grado di interpretare, modellare ed eseguire un'interrogazione formulata in Linguaggio Naturale su un qualche tipo di base di conoscenza (basi di dati, ontologie, ecc.). Le applicazioni sono dunque tra le più svariate, dalle semplici barre di ricerca di un sito web, ai più recenti Voice Assistant ed altri servizi anche web-based (Balloccu et al., 2021). L'uso del linguaggio naturale ha vantaggi e svantaggi se paragonato a linguaggi formali come le query scritte in uno dei linguaggi di interrogazione dei database. Uno dei vantaggi è la familiarità dell'utente con il linguaggio naturale. Uno svantaggio consiste nell'estrema espressività del linguaggio umano, che rende impossibile una copertura totale del lessico e quindi delle possibili interpretazioni del significato.

In questo lavoro presentiamo una NLI per interrogare una base di dati nel dominio della dieta mediterranea. Una domanda su tale dominio espressa in lingua italiana verrà analizzata mediante l'uso di una grammatica *feature-based context-free*, rappresentata mediante il *calcolo relazionale su tuple*, e infine trasformata in *SQL* mediante un processo ricorsivo basato sulla pragmatica del dominio applicativo. Un elemento distintivo di questo progetto, rispetto ai classici esempi di conversione basati sulla logica del primo ordine (Warren and Pereira, 1982), è l'uso esplicito del *calcolo relazionale* per rappresentare il significato della frase. Inoltre, un ruolo importante è giocato dalla pragmatica del

dominio, che lega il linguaggio usato nelle domande alla formalizzazione delle ricette, e che risulta codificata nello schema del database.

2 Alcuni lavori recenti LN→SQL basati sulla sintassi

In questa sezione consideriamo alcuni lavori sul tema della traduzione da linguaggio naturale a SQL che fanno uso della sintassi, seguendo lo schema proposto in (Affolter et al., 2019). Una rassegna più estesa si può trovare in (Amer-Yahia et al., 2021).

Negli *approcci basati sul Parsing* si derivano gran parte delle informazioni ricavabili a partire dalla struttura della frase in input. Tramite questo approccio è possibile riconoscere dipendenze complesse tra i singoli elementi sintattici, spesso introdotte da particolari espressioni o da preposizioni: si attribuisce quindi un valore semantico alla struttura logica della domanda analizzata. *Querix*, ad esempio, fornisce un'interfaccia per l'interrogazione di ontologie tramite la generazione di una query SPARQL (Kaufmann et al., 2006). Questo risultato è ottenuto tramite tre componenti: *query analyzer*, *matching center* e *query generator*. Il query analyzer parte dalla domanda in Linguaggio Naturale e, tramite componenti esterne quali lo Stanford Parser e il database lessicale WordNet, restituisce uno scheletro della query dato dalla sequenza delle categorie sintattiche proprie delle parole in input (Nome, Verbo, Preposizione, Congiunzione, ecc.). Tramite il matching center si cercano inizialmente dei pattern Soggetto-Proprietà-Oggetto all'interno delle categorie estratte dalla frase (la sua struttura assume quindi più importanza rispetto ai sistemi basati su keyword). Si cercano poi possibili match tra le parole in input e le risorse dell'ontologia (entrambi accresciuti dei rispettivi sinonimi di WordNet), ed infine si cerca di ottenere corrispondenze tra i risultati di queste due fasi intermedie. I matching delle triple così ottenute permettono al query generator di costruire una o più query SPARQL, ad ognuna delle quali è assegnato un ranking. L'ambiguità data dai molteplici risultati è risolta mostrando all'utente una finestra di dialogo che permette di selezionare la query più pertinente. L'efficacia del sistema dipende in gran parte dalla qualità del vocabolario dato dall'ontologia su cui è usato, limitando potenzialmente il range di domande esprimibili. Questa assenza di adattabilità è sia la debolezza che il maggior punto di forza di Querix, in quanto garantisce la più completa portabilità sui domini di qualsiasi ontologia utilizzata.

Gli *approcci basati su Grammatiche* si appoggiano largamente sulle regole di produzione: a differenza dei sistemi descritti in precedenza, in questi approcci è possibile stabilire a priori quali tipi di domande sia interpretabile e quali no. Da questa capacità segue quindi la possibilità di implementare un interfacciamento più interattivo, dove la validità di una domanda in input può essere determinata durante la stessa fase di inserimento e guidata da parte di feedback visivi. In *TR Discover*, ad esempio, la prima fase di traduzione della query consiste nel suo parsing su una *Feature-Based Context-Free Grammar* (FCFG) dove regole lessicali, generate dai nomi degli attributi della base di dati, e regole grammaticali, le quali definiscono la composizione delle informazioni estratte, permettono di ottenere composizionalmente come rappresentazione intermedia una formula nella Logica del Primo Ordine (FOL) (Song et al., 2015). Generare questo primo tipo di formulazione porta una notevole flessibilità permettendo una traduzione finale sia in SQL che in SPARQL. Nella seconda fase è effettuata un'ulteriore fase di analisi, questa volta sulla formula logica, tramite un FOL parser e un'apposita grammatica. Un importante punto di forza di TR Discover consiste nel modulo di *auto-suggestion*: il fatto che la grammatica definisca precisamente il tipo di domande valide può essere sfruttato dal sistema per suggerire all'utente possibili domande. A partire quindi dalle parole di un input parziale, i suggerimenti sono generati dalle strutture grammaticali derivabili e ordinati a seconda della "popolarità" delle entità riconosciute. Il sistema, tuttavia, presenta chiare limitazioni di espressività, non essendo in grado di analizzare query richiedenti quantificazioni e permettendo le negazioni solo traducendo in SPARQL.

3 Dominio applicativo: il progetto MAdiMan

Nel dominio della dieta alimentare il progetto *MADiMAN (Multimedia Application for Diet MANagement)* si pone come obiettivo quello di fornire un intermediario semi-automatizzato (un "dietista virtuale") tra l'utente e la gestione della sua alimentazione: a partire da informazioni quali dati biologici individuali, una definizione della dieta

da seguire (basata su quantità di macronutrienti) e un aggiornamento costante sullo storico dei pasti, il sistema cloud-based è in grado di proporre diversi servizi volti alla pianificazione dei pasti futuri, alla persuasione dell'utente verso una sana alimentazione e all'interazione efficace con queste informazioni (Anselma et al., 2018; Anselma and Mazzei, 2020; Mazzei et al., 2015; Anselma and Mazzei, 2015).

3.1 La base di dati: Gedeone

Nel considerare il dominio di interesse è necessario pensare ai dati stessi su cui andiamo a lavorare. *Gedeone*[1] è un portale di Coop Italia dedicato a promuovere un sano stile di vita e dieta proponendo consigli, articoli e soprattutto ricette. Al fine del progetto MADiMAN, le informazioni presenti su questo sito sono state manipolate e trasposte in una base di dati con DBMS PostgreSQL, rendendo disponibili un gran quantitativo di ricette (circa 500), passi preparativi, ingredienti, valori nutrizionali e vari metadati.

Un'ulteriore componente dello schema analizzato include dati e tabelle su utenti e la loro dieta, lo storico dei pasti e delle pesate.

Per gli obiettivi di questo lavoro è stato deciso di considerare la parte del database relativa alle ricette trascurando la parte relativa alla pianificazione dei pasti considerando il punto di vista di un utente non esperto del funzionamento della memorizzazione dei dati, ancor meno di come questi siano strutturati internamente.

4 Architettura del prototipo LN→SQL

Primariamente è stato costruito un corpus di 12 domande in Linguaggio Naturale[2] (Tabella 1). Gli scopi di questo piccolo corpus sono principalmente due. Primo, nel progettare un sistema che si interfacci con una persona tramite un unico passo in un caso d'uso ipotetico (inserimento testuale), è stato necessario pensare a delle domande plausibili che un utente medio potrebbe porre tramite applicativo. È stato ipotizzato a questo fine che le informazioni più desiderabili derivino dalla possibilità di consultare il ricettario, ponendo domande implicitamente relative alla propria dieta personale (per esempio, sapere l'apporto calorico di un certo piatto) così come richiedendo istruzioni ed ingredienti. Secondo, per permettere il testing di un prototipo, ognuna di queste domande è stata scritta affinché verificasse l'abilità del sistema nel riconoscere e derivare all'interno della struttura sintattica particolari costrutti, strutture ed operazioni appartenenti alla query finale. Prendendo spunto da (Affolter et al., 2019), ogni domanda è stata categorizzata attraverso varie label, principalmente corrispondenti agli operatori SQL *Join*, *Filtraggio* su attributo (nella clausola WHERE), *Negazione*, *Ordinamento* esplicito, *Raggruppamento*, *Aggregazione*, *Sottoquery*.

#	Frase
1	Quali sono le ricette senza fritture?
2	Quali sono le ricette con meno di 500 kcal?
3	Quali sono i piatti con più di 40g di proteine?
4	Quali sono i piatti con meno carboidrati?
5	Quali sono le preparazioni delle ricette primaverili?
6	Quanta acqua ho consumato il giorno @data?
7	Quante calorie ho assunto ogni giorno dal @date1 al @date2?
8	Quali sono le ricette di pesce in ordine di difficoltà?
9	Quali sono le ricette facili col minor tempo di preparazione?
10	Quali sono le ricette con cottura in forno o a vapore?
11	Quali sono gli ingredienti per @people persone della ricetta @recipe?
12	Quali sono i piatti con pochi carboidrati?

Tabella 1: Le 12 domande nel dominio delle ricette di Gedeone ideate per la sperimentazione. Il simbolo @ indica delle variabili nella domanda.

A ogni domanda definita nel corpus è stata associata una corrispondente query SQL, la cui correttezza è stata verificata sul database. Mentre si è cercato il più possibile di semplificare il formato della traduzione finale di ogni domanda in input, la necessità a fini sperimentali di definire domande sempre più complesse ha portato ad ottenere alcune query con strutture articolate, il che ha talvolta reso arduo il processo di traduzione.

Il processo di traduzione di ogni domanda del corpus è sequenziale e si divide in una fase preliminare di preanalisi, e in due fasi principali concatenate fra di loro.

Nella preanalisi la domanda in italiano è suddivisa in token. Oltre a ciò sono riconosciuti ed estratti possibili valori di attributi relativi alla base di dati considerata ed è attuata dove necessario una sostituzione con sinonimi. Par-

[1] http://www.gedeone-e-coop.it

[2] Il corpus completo è disponibile online: http://www.di.unito.it/~mazzei/papers/clic2021/QnAcorpus.pdf

```
PP[PT=?pt, RL=?rl, LF=?fl, ORD=?ord]
  ->
P[PT=?pt] NP[RL=?rl, LF=?fl, ORD=?ord]

NN[ORD=<\x.asc(x, id)>] -> 'ordine'
```

Figura 1: Un frammento della FCFG usata per rappresentare la semantica di una domanda in Calcolo Relazionale.

tendo dalla terza frase della Tabella 1, ovvero *Quali sono i piatti con più di 40g di proteine?*, la preanalisi, dopo aver eseguito tokenizzazione e sostituzione dei sinonimi, restituisce: `[quali, sono, i, ricetta, con, più, di, 40, grammi, di, proteineg]`.

La prima fase prende in input i token ottenuti in preanalisi e restituisce un albero sintattico annotato, ottenuto tramite il parsing dei token su una FCFG. È ottenuta, come rappresentazione intermedia, un'espressione nel Calcolo Relazionale su Tuple.

Infine, nella seconda fase, le informazioni presenti nell'albero sintattico sono estratte tramite diverse visite, durante ognuna delle quali è costruita una clausola della corrispondente query in SQL. Nel seguito dettagliamo il funzionamento delle due fasi principali e poi analizziamo le prestazioni del sistema.

4.1 Prima fase: dal Linguaggio Naturale al Calcolo Relazionale

In questo lavoro abbiamo usato delle grammatiche libere da contesto con feature[3] (FCFG) utilizzando la libreria Python NLTK (Bird, 2006). Similmente a (Warren and Pereira, 1982; Song et al., 2015), abbiamo inizialmente usato una rappresentazione intermedia di tipo logico per rappresentare il significato della domanda. L'obbiettivo era appunto quello di realizzarne una traduzione il più possibile fedele a una formula della FoL, la quale esprimesse precisamente il significato della query finale nel dominio (tabelle coinvolte, espressioni per la clausola WHERE, valori di attributi, ecc.).

Dopo le prime sperimentazioni si è riscontrato che questa forma intermedia, per quanto intuitiva e formale, non permetteva una facile traduzione finale nella seconda fase. Inoltre, fare affidamento esclusivamente su una singola formula composta limitava i potenziali vantaggi dati dall'uso delle FCFG. In questo cambio di direzione è stato riscontrato che una rappresentazione intermedia più vicina all'obbiettivo della traduzione in SQL poteva essere data da una query in calcolo relazionale. Il *Calcolo Relazionale su Tuple con dichiarazione di Range* è un linguaggio formale che permette di esprimere in modo dichiarativo una interrogazione su una base di dati relazionale. In questo formalismo una query comprende tre componenti: Target List (TL), Range List (RL) e Logic Formula (LF) (Codd, 1972) (cf. Fig. 1). Per ognuna di queste riportiamo una breve descrizione, seguita dal metodo scelto per poterla ottenere e rappresentare a partire dalla domanda in italiano e tramite la grammatica basata su feature.

Target List: specifica quali attributi compaiono nel risultato. Poiché il riconoscimento degli attributi necessari dipende sia da quali vengano esplicitati nella domanda, sia dalle tabelle coinvolte, questa componente è quasi completamente determinata nella seconda fase (a posteriori della parsing sintattico). Tramite le regole lessicali della grammatica sono riconosciuti gli eventuali attributi di tabelle, i cui simboli terminali sono annotati dalla loro rappresentazione nella feature LF, specificandone il tipo tramite la feature booleana +ATTR. Gli elementi restanti della Target List saranno in seguito derivati tramite una mappatura tabella-attributi.

Range List: specifica le variabili libere nella Formula Logica, cioè le tabelle su cui variano le variabili della LF per generare il risultato. Similarmente agli attributi, anche le tabelle possono essere riconosciute esplicitamente (ed annotate con una feature +TABLE), oppure derivate da attributi riscontrati, ma non appartenenti a nessuna tabella trovata fino a quel momento. In entrambi i casi la loro semantica è raccolta nella feature RL di una produzione terminale e, come per gli attributi, i diversi elementi saranno raccolti nella fase successiva attraversando l'albero sintattico annotato.

Logic Formula: specifica una formula che il risultato deve soddisfare. A differenza delle precedenti, la LF (feature LF) è composta ricorsivamente e i suoi predicati possono essere introdotti non solo nelle regole lessicali, ma anche nelle produzioni non terminali tramite il riconoscimento di particolari strutture sintattiche. La composizione della formula a partire da feature di nodi diversi è effettuata tramite le astrazioni del *lambda-calcolo*:

[3] La grammatica realizzata è disponibile online: `http://www.di.unito.it/~mazzei/papers/clic2021/calcolo_relazionale.fcfg`

definendo predicati parziali in un nodo è possibile applicarli a formule ed argomenti provenienti dagli altri nodi della stessa produzione in una semplificazione chiamata *Beta-riduzione*. Questa avviene automaticamente durante il parsing sintattico tramite NLTK, ed il risultato, se presente, è visibile nella radice dell'albero sintattico.

A differenza di SQL, il Calcolo Relazionale, per propria natura, non esprime esplicitamente un ordinamento sull'insieme di tuple risultante (corrispondente al costrutto *ORDER BY* di SQL). Per compensare questo aspetto si è deciso di utilizzare una feature aggiuntiva ORD, definita in modo simile alla feature LF ma utilizzata esclusivamente per mantenere tramite predicati le informazioni di un eventuale ordinamento esplicito (riconosciuto a livello lessicale da espressioni come "in ordine di", "con meno/più X", ecc.) In generale il processo di traduzione della prima fase è completamente automatizzato: il parser utilizzato è generato tramite una funzione della libreria NLTK e necessita solamente della definizione di una grammatica. Il parsing dei token in input restituisce dunque un albero sintattico annotato con feature.

Partendo dalla preanalisi della frase *Quali sono i piatti con più di 40g di proteine?*, si ottiene l'analisi sintattica in Fig. 2(a), e la corrispondente rappresentazione in calcolo relazionale in Fig. 2(b).

4.2 Seconda fase: dal Calcolo Relazionale a SQL

La costruzione dell'interrogazione SQL utilizza l'albero sintattico e la corrispondente formula del calcolo relazionale ottenute entrambe nella fase precedente, insieme a ulteriori dati provenienti dal dominio di interesse, e che sono implicitamente codificate nel database di riferimento. Mentre con la prima fase si è potuto modellare la conoscenza derivata dalla domanda iniziale, molto di ciò che ci serve sapere non è direttamente derivabile da una domanda in italiano. Inoltre, poiché l'obbiettivo di questo progetto rimane la creazione di un sistema che operi su un dominio definito, è necessario che le informazioni più importanti riguardanti questo siano definite ed accessibili nel sistema: queste informazioni sono fondamentali per stabilire i legami tra ciò che è stato chiesto e i nomi effettivi di tabelle e attributi.

Il processo di traduzione finale segue un algoritmo generale comune: per ogni clausola principale di SQL viene visitato l'albero sintattico e, grazie anche alle informazioni dal dominio, viene generata la clausola componendo ivalori nelle feature. In particolare:

SELECT: La lista di attributi da restituire corrisponde alla Target List del Calcolo Relazionale. È possibile riconoscere due tipi di attributi appartenenti a questa. (i) Ogni tabella introdotta, esplicitamente o meno (riconosciuta a partire dalla feature +TABLE oppure derivata da un suo attributo), comprende uno o più attributi di default, oltre a quelli di chiave primaria. Questo tipo di mappatura è un esempio di informazione pragmatica proveniente dal dominio e codificata nella tabella. (ii) Per quanto riguarda invece gli attributi non presenti in alcuna tabella, questi possono semplicemente essere riconosciuti da una regola lessicale (tramite la feature +ATTR) ed aggiunti in coda all'elenco.

FROM: Un'assunzione importante è stata fatta nella gestione dei join, ovvero che le tabelle avessero sempre delle chiavi esterne definite esplicitamente. Ciò semplifica notevolmente la traduzione, in quanto è sufficiente conoscere le tabelle coinvolte per poter generare l'intera clausola comprensiva delle condizioni di join. L'algoritmo di generazione riconosce i vincoli di chiave esterna tramite due cicli, che compongono la clausola risultante con gli attributi necessari. Da notare che sia i vincoli relazionali, sia gli alias delle tabelle sono definiti tramite dizionari.

WHERE: Questa clausola è l'unica costruita a partire dalle informazioni provenienti dalla LF. I predicati di questa sono indicati in un formalismo più simile a quello della FOL rispetto al Calcolo Relazionale, al fine di aderire alla sintassi delle formule di NLTK. Per ognuno dei predicati si effettua una traduzione corrispondente rappresentazione in SQL.

ORDER BY: Come la clausola WHERE, la costruzione di questa clausola è determinata dal valore della feature ORD nella radice dell'albero sintattico. Per i fini della nostra implementazione è stato deciso di adottare, nel caso in cui la feature ORD risulti vuota, un ordinamento di default sull'attributo di chiave primaria della tabella principale coinvolta.

Considerando ancora la terza frase della Tabella 1, il risultato della seconda fase di traduzione è in Fig. 2-(c).

```
(S[FL=<maggiore(ric(ricetta),proteineg,40)>, ORD=?ord]
  (VP[] (PI[NUM='pl', QRT='qual'] Quali) (IV[NUM='pl'] sono))
  (NP[FL=?fl]
    (DET[NUM='pl'] i)
    (NN[RL=<ric(ricetta)>, +TABLE] ricetta))
  (PP[FL=<maggiore(ric(ricetta),proteineg,40)>, ORD=?ord, PT='with', RL=?rl]
    (P[PT='with'] con)
    (NP[FL=<maggiore(ric(ricetta),proteineg,40)>]
      (ADV[FL=<\z y x.maggiore(x,y,z)>] più)
      (PP[FL=<40>, ORD=?ord, PT='of', RL=?rl]
        (P[PT='of'] di)
        (NP[FL=<40>] (CARD[FL=<40>] 40) (NN[] grammi)))
      (PP[FL=<proteineg>, PT='of', RL=<ric(ricetta)>]
        (P[PT='of'] di)
        (NN[+ATTR, FL=<proteineg>, RL=<ric(ricetta)>] proteineg)))))
```

(a)

```
{TL=ric.id, ric.descrizione, ric.nome , ric.proteineg | RL=ric(ricetta) |
LF=maggiore(ric(ricetta),proteineg,40)}
```

(b)

```
SELECT ric.id, ric.nome, ric.descrizione, ric.proteineg
FROM Ricetta ric WHERE ric.proteineg>40 ORDER BY ric.id ASC;
```

(c)

Figura 2: Il risultato della prima fase di analisi (a-b) e della seconda fase di analisi (c) sulla frase *Quali sono i piatti con più di 40g di proteine?*.

5 Valutazione preliminare

Nella sperimentazione preliminare è stato possibile implementare la traduzione solamente di 7 domande rispetto alle 12 inizialmente definite nel corpus[4]. Seguono i limiti incontrati per alcune delle interrogazioni non realizzate.

La domanda 6, *Quanta acqua ho consumato il giorno @data*, è stata tralasciata poiché richiederebbe una gestione delle informazioni riguardanti un utente che abbia effettuato l'accesso al sistema.

Non è stato possibile realizzare la domanda 10, *"Quali sono le ricette con cottura in forno o a vapore"* per la natura dell'operazione richiesta (un OR esclusivo), la quale richiedeva una sotto-query. Al momento non è implementato un meccanismo che rilevi la presenza di sotto-interrogazioni a partire dalla struttura della frase.

Nella domanda 11, *"Quali sono gli ingredienti per @people persone della ricetta @recipe?"*, la difficoltà incontrata è stata la gestione dei valori NULL ottenibili in certi attributi.

La domanda 12, *"Quali sono i piatti con pochi carboidrati?"*, è stata realizzata nella versione semplificata *"Quali sono i piatti con meno carboidrati?"*: la difficoltà nell'includere una sotto-query (con annessa Window Function di SQL) ha portato a interpretare "pochi carboidrati" non come valori appartenenti al primo quartile sulla distribuzione totale, ma come quantità al di sotto di una soglia di default.

6 Conclusioni

In questo lavoro sono stati presentati i primi risultati di un progetto ancora in corso per interpretare una domanda in linguaggio naturale come un'interrogazione SQL nel dominio della dieta mediterranea. Le due caratteristiche principali di questo progetto sono state l'uso del calcolo relazionale su tuple per rappresentare il significato della frase (Sezione 4.1) e l'uso delle specificità del dominio per trasformare poi tale rappresentazione in SQL (Sezione 4.2).

In futuro intendiamo completare le interrogazioni contenute nel corpus proposto per poi integrare il prototipo in un agente conversazionale sul dominio della gestione di una dieta salutare. Considerando la presenza di indeterminatezza nel linguaggio naturale, potrebbe essere interessante dare supporto per estensioni del modello relazionale che trattano indeterminatezza (Anselma et al., 2016).

[4]Un elenco completo dell'output è disponibile a `http://www.di.unito.it/~mazzei/papers/clic2021/OutputPrototipo.pdf`

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819.

Sihem Amer-Yahia, Georgia Koutrika, Frederic Bastian, Theofilos Belmpas, Martin Braschler, Ursin Brunner, Diego Calvanese, Maximilian Fabricius, Orest Gkini, Catherine Kosten, Davide Lanti, Antonis Litke, Hendrik Lücke-Tieke, Francesco Alessandro Massucci, Tarcisio Mendes de Farias, Alessandro Mosca, Francesco Multari, Nikolaos Papadakis, Dimitris Papadopoulos, Yogendra Patil, Aurélien Personnaz, Guillem Rull, Ana Claudia Sima, Ellery Smith, Dimitrios Skoutas, Srividya Subramanian, Guohui Xiao, and Kurt Stockinger. 2021. INODE: building an end-to-end data exploration system in practice [extended vision]. *CoRR*, abs/2104.04194.

Luca Anselma and Alessandro Mazzei. 2015. Towards diet management with automatic reasoning and persuasive natural language generation. In *Portuguese Conference on Artificial Intelligence*, pages 79–90. Springer.

Luca Anselma and Alessandro Mazzei. 2020. Building a persuasive virtual dietitian. *Informatics*, 7(3):27.

Luca Anselma, Luca Piovesan, and Paolo Terenziani. 2016. A 1nf temporal relational model and algebra coping with valid-time temporal indeterminacy. *Journal of Intelligent Information Systems*, 47(3):345–374.

Luca Anselma, Alessandro Mazzei, and Andrea Pirone. 2018. Automatic reasoning evaluation in diet management based on an italian cookbook. In *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management*, pages 59–62.

Simone Balloccu, Ehud Reiter, Matteo G. Collu, Federico Sanna, Manuela Sanguinetti, and Maurizio Atzori. 2021. Unaddressed challenges in persuasive dieting chatbots. In Judith Masthoff, Eelco Herder, Nava Tintarev, and Marko Tkalcic, editors, *Adjunct Publication of the 29th ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2021, Utrecht, The Netherlands, June 21-25, 2021*, pages 392–395. ACM.

Steven Bird. 2006. NLTK: the natural language toolkit. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.

E. F. Codd. 1972. Relational completeness of data base sublanguages. *Research Report / RJ / IBM / San Jose, California*, RJ987.

Dan Jurafsky. 2014. *The language of food: A linguist reads the menu*. WW Norton & Company.

Esther Kaufmann, Abraham Bernstein, and Renato Zumstein. 2006. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th international semantic web conference (ISWC 2006)*, pages 980–981.

Alessandro Mazzei, Luca Anselma, Franco De Michieli, Andrea Bolioli, Matteo Casu, Jelle Gerbrandy, and Ivan Lunardi. 2015. Mobile computing and artificial intelligence for diet management. In *International Conference on Image Analysis and Processing*, pages 342–349. Springer.

Dezhao Song, Frank Schilder, Charese Smiley, Chris Brew, Tom Zielund, Hiroko Bretz, Robert Martin, Chris Dale, John Duprey, Tim Miller, et al. 2015. Tr discover: A natural language interface for querying and analyzing interlinked datasets. In *International Semantic Web Conference*, pages 21–37. Springer.

David H.D. Warren and Fernando C.N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122.