DOI: 10.26314/GARR-Conf18-proceedings-03

Un framework semantico per la ricerca e l'esecuzione di codice sorgente

Mattia Atzeni, Maurizio Atzori

Università degli Studi di Cagliari

Abstract. Questo articolo introduce un framework semantico per la ricerca automatica e la conseguente esecuzione di codice sorgente open source. A tal fine, abbiamo recentemente introdotto CodeOntology, un approccio ideato per applicare lo stack tecnologico fornito dal Semantic Web, nel dominio dell'ingegneria del software. In questo modo, l'informazione estratta dal codice sorgente può essere facilmente collegata con risorse esterne, in maniera da permettere interessanti analisi e ricerche semantiche che sarebbero altrimenti impossibili. Inoltre, proponiamo un approccio algoritmico che si basa su CodeOntology, al fine di selezionare un insieme di metodi rilevanti, che vengono successivamente combinati per tradurre una specifica, espressa in linguaggio naturale, in codice sorgente basato sul paradigma orientato agli oggetti

Keywords. Comprensione del Linguaggio Naturale, Semantic Web, Question Answering

Introduzione

Nonostante il bisogno di crescente automazione da parte di una vasta gamma di utenti differenti (Atzeni & Atzori, 2018c), la libera disponibilità online di un'enorme quantità di codice sorgente esibisce ancora un considerevole, ma al tempo stesso inespresso, potenziale per lo sviluppo di tecniche di intelligenza artificiale, finalizzate a sfruttare questa notevole quantità di informazione. Pertanto, il lavoro descritto in questo articolo ambisce a risolvere il problema di fornire un framework per la ricerca e l'esecuzione di componenti software, tramite l'introduzione di approcci semantici che permettano di tradurre complesse interrogazioni e comandi espressi in linguaggio naturale in un appropriato codice sorgente.

Questo problema di ricerca può essere scomposto in due sotto-problemi principali: fornire una rappresentazione semantica dei sistemi software e dei linguaggi di programmazione orientati agli oggetti ed utilizzare la base di conoscenza risultante, per rispondere in maniera automatica ad interrogazioni espresse in linguaggio naturale. Il primo sotto-problema può essere risolto tramite l'uso di CodeOntology (Atzeni & Atzori, 2017a), una risorsa che consiste di tre contributi principali:

- un'ontologia OWL 2 che rappresenta il dominio dei linguaggi di programmazione orientati agli oggetti;
- un parser che permette di serializzare codice sorgente in triple RDF;
- un dataset contenente circa 2.5 milioni di triple RDF, generate tramite l'applicazione del parser sul codice sorgente del progetto OpenJDK 8.

Ciò permette di eseguire interrogazioni espresse nel linguaggio SPARQL per scopi diffe-

Cagliari, 3-5 ottobre 2018 - ISBN: 978-88-905077-8-6

renti, quali l'analisi statica del codice sorgente, il rilevamento di design pattern ed anti-pattern, la ricerca semantica di codice (Atzeni & Atzori, 2017b).

Seguendo questa linea di ricerca, il presente articolo definisce un approccio semantico (Atzeni & Atzori, 2018a) che sfrutta la conoscenza estratta da CodeOntology per eseguire comandi in linguaggio naturale e fornire una risposta a domande complesse che richiedono la generazione di codice ad hoc.

1. Descrizione del sistema

Questa sezione descrive un approccio semantico per la traduzione in codice di comandi espressi in linguaggio naturale (Atzeni & Atzori, 2018b). Il sistema si basa su CodeOntology per la ricerca di metodi definiti in OpenJDK, tali che la loro firma sia compatibile con i valori specificati dall'utente. I risultati restituiti da questa query sono successivamente ordinati per selezionare il candidato che meglio corrisponde alla specifica in linguaggio naturale. L'ordinamento dei candidati selezionati si basa sia su misure semantiche che su indicatori sintattici, come spiegato meglio nella Sezione 2. Data una descrizione in linguaggio naturale, il sistema esegue inizialmente il parsing delle dipendenze. (Manning et al. 2014), come mostrato nella Figura 1.

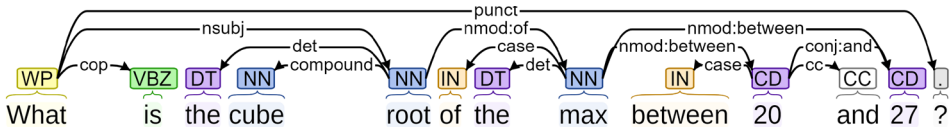


Fig. 1
Risultato del parsing delle dipendenze per una domanda ricevuta in input

Il grafo risultante viene trasformato in un albero, tale che l'insieme dei nodi N possa essere partizionato in due sottoinsiemi $\mathcal{L}$ e $\mathcal{M}$, dove:

- $\mathcal{L}$ è il sottoinsieme corrispondente agli argomenti letterali;
- $\mathcal{M}$ contiene i nodi che corrispondono ad espressioni che denotano l'invocazione di un metodo;
- ogni nodo $i \in \mathcal{L}$ è una foglia;
- $N = \mathcal{L} \cup \mathcal{M}$ e $\mathcal{L} \cap \mathcal{M} = \emptyset$.

Ogni nodo $i \in \mathcal{M}$ è etichettato con un ranking di metodi $\mathcal{R}_i$, cioè una sequenza

$$(m_1, s_1),\ldots,(m_n,s_n)$$

tale che:

- m_i : Method per ogni $i = 1\ldots n$
- $s_i \in [0,1]$ per ogni $i = 1\ldots n$
- $i < j \Rightarrow s_i \geq s_j$

I metodi candidati sono selezionati tramite l'esecuzione di interrogazioni su CodeOntology, basate sul tipo dei parametri e sul tipo di ritorno atteso. Successivamente, ad ogni metodo viene associato uno score reale, usando le features descritte nella Sezione 2.

A questo punto, si rende necessario risolvere un problema di programmazione lineare intera, che prevede di massimizzare il seguente obiettivo, dove $x_{ij}=1$ se il j-esimo metodo

del ranking $\mathcal{R}_i$ è selezionato, ed $x_{ij} = 0$ altrimenti:

$$\sum_{i \in \mathcal{M}} \sum_{(m_{ij}, s_{ij}) \in \mathcal{R}_i} x_{ij} \cdot s_{ij}$$

con i seguenti vincoli:

- $x_{ij} \in \{0,1\}$;
- $\sum_j x_{ij} = 1$, per ogni $i \in \mathcal{M}, 1 \leq j \leq |\mathcal{R}_i|$;
- la combinazione dei metodi selezionati può essere compilata.

Per migliorare ulteriormente l'approccio algoritmico descritto, si applica un metodo euristico che agisce sulla struttura dell'albero e adotta una ricerca greedy per massimizzare la seguente funzione:

$$z(\mathcal{T}_k) = \frac{1}{|\mathcal{M}_k|} \cdot \sum_{i \in \mathcal{M}_k} \sum_{(m_{ij}, s_{ij}) \in \mathcal{R}_i^k} x_{ij} \cdot s_{ij} - \lambda \cdot NTED(\mathcal{T}_k, \mathcal{T}_o),$$

dove $\mathcal{T}_k$ è l'albero all'iterazione k, $\mathcal{M}_k$ è l'insieme dei nodi non letterali in $\mathcal{T}_k$ e $\mathcal{R}^k{}_i$ è il ranking associato al nodo i in $\mathcal{T}_k$. Inoltre, λ è una costante ed $NTED(\mathcal{T}_k, \mathcal{T}_0)$ è la tree edit distance normalizzata tra $\mathcal{T}_k$ e $\mathcal{T}_0$.

2. Esperimenti

L'ordinamento dei metodi Java è stato valutato tramite un dataset contenente domande estratte da StackOverflow. Sono state sperimentate diverse combinazioni delle seguenti misure sintattiche e semantiche:

- similarità di Levenshtein normalizzata, calcolata tra il nome del metodo e la specifica in linguaggio naturale;
- n-gram overlap con il nome della classe dichiarante;
- n-gram overlap con il commento relativo al metodo considerato;
- similarità del coseno tra il vettore medio (Mikolov et al. 2013) associato al commento del metodo ed il vettore medio associato alla descrizione in linguaggio naturale;
- frazione dei link a DBpedia condivisi tra il commento del metodo e il comando in linguaggio naturale, annotati tramite l'uso di TagMe (Ferragina & Scaiella, 2012).

La combinazione di queste features permette di raggiungere una Mean Average Precision pari a 0.92 sul dataset menzionato in precedenza. L'approccio è stato ulteriormente valutato su un benchmark contenente 120 domande relative all'esecuzione di espressioni matematiche e manipolazione di stringhe. É possibile applicare una soglia $t \in [0,1]$ sul valore obiettivo, in modo da rilevare le domande che non possono essere elaborate correttamente dal sistema. La Tabella 1 confronta il nostro approccio con i risultati ottenuti da WolframAlpha.

3. Conclusioni

Questo articolo ha introdotto un approccio che fa uso di CodeOntology, al fine di suppor-

Tab. 1
Risultati sperimentali

	CodeOntology	WolframAlpha
Numero di domande	120	120
Domande elaborate	116	108
Risposte corrette	109	98
Precisione (globale)	0.91	0.82
Precisione (domande elaborate)	0.94	0.91

tare la ricerca e l'esecuzione di metodi, tramite semplici interrogazioni espresse in linguaggio naturale. Gli esperimenti sono stati condotti su OpenJDK, ma l'approccio è generale e può essere applicato per qualunque linguaggio di programmazione.

Riferimenti bibliografici

M. Atzeni, M. Atzori (2018a), What is the Cube Root of 27? Question Answering over CodeOntology, The Semantic Web – ISWC 2018: 17th International Semantic Web Conference, Monterey, California, USA.

M. Atzeni, M. Atzori (2018b), Translating Natural Language to Code: an Unsupervised Ontology-Based Approach, 2018 IEEE International Conference on Artificial Intelligence and Knowledge Engineering, Laguna Hills, California, USA.

M. Atzeni, M. Atzori (2018c), Towards Semantic Approaches for General-Purpose End-User Development, 2nd IEEE International Conference on Robotic Computing, IRC 2018, Laguna Hills, California, USA.

M. Atzeni, M. Atzori (2017a), CodeOntology: RDF-ization of Source Code, The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, ISWC 2017, Vienna, Austria, pp. 20–28.

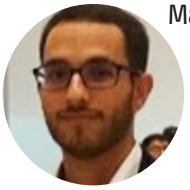
M. Atzeni, M. Atzori (2017b), CodeOntology: Querying Source Code in a Semantic Framework, 16th International Semantic Web Conference (Posters & Demo), Vienna, Austria.

P. Ferragina, U. Scaiella (2012), Fast and accurate annotation of short texts with Wikipedia pages., IEEE Software, 29(1), pp. 70–75.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky (2014), The Stanford CoreNLP natural language processing toolkit, Association for Computational Linguistics (ACL), System Demonstrations, pp. 55–60.

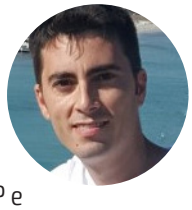
T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean (2013), Distributed representations of words and phrases and their compositionality, Advances in Neural Information Processing Systems, Curran Associates, pp. 3111–3119

Autori



Mattia Atzeni - m.atzeni38@studenti.unica.it

Mattia Atzeni è studente di dottorato presso l'Università degli Studi di Cagliari. Ha conseguito la Laurea Magistrale in Informatica presso la stessa università, ad Aprile 2018. I suoi principali interessi di ricerca includono la comprensione del linguaggio naturale, l'intelligenza artificiale e il machine learning.



Maurizio Atzori - atzori@unica.it

Maurizio Atzori è Professore Associato presso l'Università degli Studi di Cagliari. Ha conseguito il dottorato presso l'Università di Pisa nel 2006 ed è stato membro del KDD Lab presso l'Istituto di Scienza e Tecnologia dell'Informazione del CNR. È stato visiting researcher presso la Purdue University (Indiana, USA), la Sabanci University (Istanbul, Turchia) e presso la UCLA (Los Angeles, USA). I suoi principali interessi di ricerca includono AI, NLP e grafi di conoscenza.