Assistenti Virtuali per la Pubblica Amministrazione

Mattia Atzeni, Maurizio Atzori
Università degli Studi di Cagliari
m.atzeni38@studenti.unica.it, atzori@unica.it

Abstract

Questo articolo discute un approccio basato su ontologie per la creazione di assistenti virtuali intelligenti. Il sistema descritto permette di estrapolare una rappresentazione formale del significato di interrogazioni espresse in linguaggio naturale. In particolare, gli esperimenti effettuati permettono di tradurre in codice Java i comandi in linguaggio naturale ricevuti in input dal sistema. L'approccio descritto permette di creare assistenti facilmente estendibili e si presta ad essere utilizzato in svariati contesti applicativi. L'articolo si focalizza soprattutto sulle applicazioni nel campo della pubblica amministrazione.

1 Introduzione

Negli ultimi anni, uno degli ambiti in cui l'uso di tecniche di Intelligenza Artificiale (IA) ha riscosso maggior successo è stato sicuramente lo sviluppo di chatbot e assistenti personali intelligenti. Questo tipo di strumenti permette all'utente di esprimere richieste tramite il linguaggio naturale e si presta ad essere utilizzato in diversi contesti applicativi, con notevoli potenzialità di impatto anche nel campo della pubblica amministrazione. Ad alto livello, i possibili obiettivi dell'applicazione di tali tecnologie in questo ambito includono:

- ridurre i costi dell'assistenza;
- rendere più facilmente accessibili le informazioni;
- ridurre l'afflusso dell'utenza agli uffici;
- semplificare il lavoro dei funzionari della pubblica amministrazione.

Gli approcci più diffusi in letteratura per la realizzazione di sistemi in grado di interpretare espressioni in linguaggio naturale sono spesso basati su tecniche di apprendimento supervisionato [Quirk *et al.*, 2015; Lin *et al.*, 2017; Iyer *et al.*, 2018]. Solitamente, gli assistenti virtuali intelligenti e gli agenti conversazionali si basano in primo luogo sull'identificazione ad alto livello dell'azione (*intent*) richiesta dall'utente [Wang e Manning, 2012; Kim, 2014]. Le alternative più diffuse e sviluppate a livello commerciale includono *Google Assistant*[1], *Amazon Alexa*[2], *Siri*[3] e *Cortana*[4]. Il funzionamento di questi sistemi tuttavia spesso non è ampiamente documentato e risulta difficile investigarne il funzionamento ed estenderne le capacità.

Questo articolo, invece, introduce un approccio non supervisionato e basato su ontologie, per l'interpretazione di interrogazioni e comandi espressi in linguaggio naturale. Inoltre, vengono discusse le potenziali applicazioni di questo sistema nel campo delle pubbliche amministrazioni.

2 CodeOntology e AskCO

AskCO [Atzeni e Atzori, 2018] è un progetto che permette di analizzare interrogazioni complesse espresse in linguaggio naturale, al fine di convertirle in una rappresentazione formale del loro significato. In particolare, il caso d'uso analizzato in [Atzeni e Atzori, 2018] mostra come sia possibile utilizzare AskCO per tradurre in codice Java delle domande ricevute in input dal sistema. A titolo d'esempio, la semplice domanda:

What is the cube root of the max between 20 and 27?

viene tradotta autonomamente nel seguente codice sorgente:

```
Math.cbrt(Double.max(20, 27))
```

la cui esecuzione restituisce il risulato atteso: `3.0`. Ciò è reso possibile dalla realizzazione di un'ontologia, chiamata CodeOntology [Atzeni e Atzori, 2017], che contiene informazioni estratte da progetti Java, in particolare OpenJDK 8[5]. In questo modo, AskCO riesce ad esaminare diversi metodi e può scegliere i candidati che meglio corrispondono al comando ricevuto in input, sulla base di features sintattiche e semantiche estratte dal nome del metodo, dal nome della classe dichiarante e dai commenti inseriti all'interno del codice. Tra le features utilizzate da AskCO, troviamo misure di similarità calcolate tramite *Word Embeddings* e tecniche di *Named Entity Disambiguation* (NED). Le prestazioni del sistema sono state valutate tramite un confronto con *WolframAlpha*[6] su un dataset contenente diverse richieste espresse in linguaggio naturale, riguardanti la risoluzione di espressioni matematiche

[1] https://assistant.google.com
[2] https://developer.amazon.com/alexa
[3] https://www.apple.com/siri/
[4] https://www.microsoft.com/cortana
[5] https://openjdk.java.net/
[6] https://www.wolframalpha.com/

o l'esecuzione di compiti che richiedono la manipolazione di stringhe.[7] La Tabella 1 riassume i risultati ottenuti.

Tabella 1: Confronto tra AskCO e WolframAlpha

	AskCO	**WolframAlpha**
Numero Domande	120	120
Domande Elaborate	116	108
Risposte Corrette	109	98
Precisione (globale)	0.91	0.82
Precisione (domande elaborate)	0.94	0.91

3 Applicazioni nel campo della Pubblica Amministrazione

Sistemi che permettano di elaborare il linguaggio naturale allo scopo di indentificare le intenzioni dell'utente si prestano ad essere applicati in svariati contesti. In questo articolo, mostriamo, tramite dei semplici casi d'uso, come AskCO possa essere adoperato nel campo delle pubbliche amministrazioni. A tal fine, sono stati implementati in Java dei semplici *intent* mock-up di esempio, disponibili su GitHub[8]. Le possibili azioni previste includono:

- rinnovo di documenti;
- prenotazioni di visite mediche;
- richieste di informazioni;
- pagamenti di multe o tasse.

L'approccio *coarse-grained* descritto in [Atzeni e Atzori, 2018] può essere utilizzato per associare, a frasi espresse in linguaggio naturale, una lista di metodi, ordinati in maniera tale che gli intent più adatti a rispondere alle esigenze dell'utente ricoprano le posizioni iniziali della lista. La Figura 1 mostra alcuni esempi di frasi ricevute in input dal sistema e delle porzioni dei relativi ranking prodotti in output.

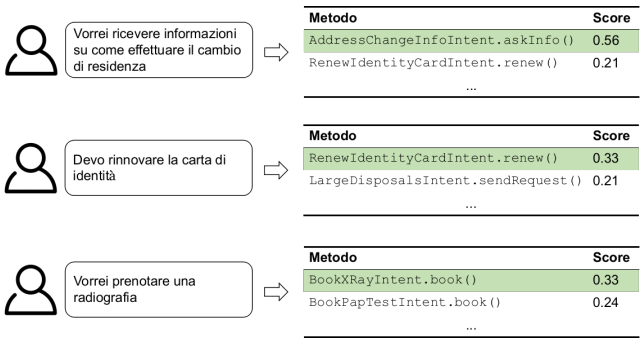

Figura 1: Esempio di applicazione di AskCO per le azioni definite nel campo delle pubbliche amministrazioni

Come descritto nella Sezione 2, AskCO riesce ad identificare il metodo corretto sulla base di diverse features, alcune delle quali sono estratte dai nomi del metodo e della classe dichiarante. Per supportare molteplici linguaggi, è prevista la possibilità di utilizzare appositi tag nei commenti, in maniera tale da fornire una traduzione dei nomi di metodi e classi. A questo punto, il sistema identifica il linguaggio in cui è espressa l'interrogazione ricevuta in input [Nakatani, 2010] ed utilizza di conseguenza le features e i modelli appropriati.

Uno dei principali vantaggi dell'approccio descritto, rispetto all'uso di altri metodi per la realizzazione di assistenti virtuali, risiede sicuramente nella possibilità di estendere rapidamente e in maniera semplice le funzionalità del sistema. Nell'esempio analizzato in questa sezione, infatti, per aggiungere delle nuove azioni è sufficiente creare nuove classi che implementino l'interfaccia `Intent`.

4 Conclusioni

È stato illustrato un approccio basato su ontologie per la creazione di assistenti virtuali che possano essere applicati in diversi contesti. In particolare, le possibili applicazioni nel campo della pubblica amministrazione risultano promettenti per ridurre i costi dell'assistenza e rendere più semplice e immediata l'esecuzione di alcuni compiti ricorrenti.

Riferimenti bibliografici

[Atzeni e Atzori, 2017] Mattia Atzeni e Maurizio Atzori. CodeOntology: RDF-ization of Source Code. In *The Semantic Web – ISWC 2017*, pages 20–28, Cham, 2017. Springer International Publishing.

[Atzeni e Atzori, 2018] Mattia Atzeni e Maurizio Atzori. What Is the Cube Root of 27? Question Answering Over CodeOntology. In *The Semantic Web – ISWC 2018*, pages 285–300, Cham, 2018. Springer International Publishing.

[Iyer *et al.*, 2018] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, e Luke Zettlemoyer. Mapping language to code in programmatic context. In *Proceedings of EMNLP 2018*, pages 1643–1652, 2018.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[Lin *et al.*, 2017] Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, Luke Zettlemoyer, e Michael D. Ernst. Program synthesis from natural language using recurrent neural networks. Technical Report UW-CSE-17-03-01, University of Washington, 2017.

[Nakatani, 2010] Shuyo Nakatani. Language detection library for java, 2010. https://github.com/shuyo/language-detection.

[Quirk *et al.*, 2015] Chris Quirk, Raymond Mooney, e Michel Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 878–888, 2015.

[Wang e Manning, 2012] Sida Wang e Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94. Association for Computational Linguistics, 2012.

[7] https://doi.org/10.6084/m9.figshare.6071729.v1

[8] https://github.com/codeontology/AskCOPAIntents