"Natural Language Processing with Python"

Steven Bird, Ewan Klein, Edward Loper

È facile acquisire milioni di parole di testo. Cosa possiamo farne, presupponendo di saper scrivere alcuni semplici programmi? In questo capitolo ci concentreremo sulle seguenti domande:

1. Cosa possiamo ottenere combinando semplici tecniche di programmazione con ampie quantità di testo?
2. Come possiamo estrarre automaticamente parole-chiave e frasi che riassumano lo stile ed il contenuto di un testo?
3. Quali strumenti e tecniche ci fornisce per tale lavoro il linguaggio di programmazione Python?

Quantificare con il Linguaggio: Testi e Parole

Conosciamo tutti il testo, in quanto lo leggiamo e scriviamo ogni giorno. In questo paragrafo considereremo il testo come *dati grezzi* per i programmi che scriviamo, programmi che lo manipolano ed analizzano in una varietà di modi interessanti. Ma prima di poterlo fare dobbiamo iniziare ad usare l'interprete Python.Una delle cose più apprezzate di Python è che permette di digitare direttamente sul suo **interprete** interattivo ovvero il programma che gestisce i tuoi programmi Python. Si può accedere all'interprete Python utilizzando una semplice interfaccia grafica chiamata Sviluppo Ambientale Interattivo (in inglese Interactive DeveLopment Environment da cui l'acronimo IDLE).

Il prompt >>> indica che l'interprete Python è ora in attesa dell'introduzione dei dati. Quando copiate esempi da questo libro, non digitate il ">>>". Ora, cominciamo, utilizzando Python come una calcolatrice:

```
>>> 1 + 5 * 2 - 3
8
>>>
```

Una vola che l'interprete ha terminato di calcolare la soluzione e la visualizza, il prompt riappare. Questo significa che l'interprete Python è in attesa di un'altra istruzione.

Iniziare ad usare NLTK

Prima di proseguire dovreste installare NLTK, scaricabile gratuitamente da `http://www.nltk.org/`. Seguite le istruzioni del sito per scaricare la versione richiesta dalla vostra piattaforma. Una volta installato NLTK, avviate l'interprete Python come prima, ed installate i dati necessari per il libro digitando i seguenti due comandi sul prompt Python, dopodiché selezionate la collezione. Una volta che I dati sono stati scaricati sul vostro apparecchio, potete caricarne un po' usando l'interprete Python. Il primo passo è digitare un comando speciale sul prompt Python che dice all'interprete di caricare una parte dei testi così da esplorarli: `from nltk.book import *`. Questo dice "dal modulo NLTK `book` , carica tutti gli oggetti." Il modulo contiene tutti i dati di cui avrai bisogno mentre leggete questo capitolo. Dopo aver inviato un messaggio di benvenuto, caricate il testo di diversi libri (ci impiega pochi secondi). Ecco di nuovo il comando, insieme al risultato che vedrete. Fate attenzione che lo spelling ed la punteggiatura siano corretti, e ricordate che non dovete digitare`>>>` .

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

Ogni volta che vogliamo esplorare tra questi testi, dobbiamo solo inserire i loro nomi nel prompt Python:

```
>>> text1
<Text: Moby Dick by Herman Melville 1851>
>>> text2
<Text: Sense and Sensibility by Jane Austen 1811>
>>>
```

Ora che possiamo usare l'interprete Python, e abbiamo qualche dato con cui lavorare, siamo pronti per iniziare.

Cercare il testo

Ci sono molti modi per esaminare il contesto di un testo oltre che semplicemente leggendolo. Un esame delle concordanze ci mostra ogni occorrenza di una data parola, assieme ad un po' di contesto. Andiamo a cercare la parola mostruoso in *Moby Dick* inserendo `text1` seguito da un punto, poi il termine `concordance`, e poi inserendo `"monstrous"` tra parentesi.

```
>>> text1.concordance("monstrous")
Building index...
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came
towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or
ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some
were thick
d as you gazed , and wondered what monstrous cannibal and savage
could ever hav
that has survived the flood ; most monstrous and most mountainous !
That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse
and more de
th of Radney .'" CHAPTER 55 Of the monstrous Pictures of Whales . I
shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I
am strongly
ere to enter upon those still more monstrous stories of them which
are to be fo
ght have been rummaged out of this monstrous cabinet there is no
telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes
cast up dead u
>>>
```

Comprendere il linguaggio naturale automatico

Siamo esplorando il linguaggio dalla base, con l'aiuto di testi e del linguaggio di programmazione Python. Ad ogni modo, siamo anche interessati a sfruttare la nostra conoscenza del linguaggio e del calcolo per costruire tecnologie di linguaggio utili. Ora coglieremo l'opportunità di allontanarci dal nocciolo del codice per dipingere un'immagina più grande del processo di linguaggio naturale.

Ad un livello puramente pratico, abbiamo tutti bisogno di navigare l'universo dell'informazione rinchiuso nel testo sulla rete. I motori di ricerca sono cruciali per la crescita e la popolarità del web, ma hanno alcuni punti deboli. Ci vogliono abilità, conoscenza ed un po' di fortuna per ricavare risposte da domande come: *Quali siti turistici posso visitare tra Philadelphia e Pittsburgh con un budget limitato? Cosa pensano gli esperti delle fotocamere digitali SLR? Quali pronostici a proposito del mercato dell'acciaio furono fatti da commentatori credibili nella settimana passata?* Avere un computer per rispondere automaticamente coinvolge

una serie di mansioni di programmazione del linguaggio, incluse l'estrazione delle informazioni, la deduzione ed il riassunto, ed avrebbe bisogno di essere portata su una bilancia ad un livello di robustezza che è oltre le nostre attuali capacità.

Ad un livello più filosofico, una sfida a lungo termine con l'intelligenza artificiale è stata costruire macchine intelligenti, ed una parte cruciale del comportamento intelligente è capirne il linguaggio. Per molti anni l'obiettivo è stato visto molto difficilmente. Ad ogni modo, come le tecnologie NLP sono maturate, e metodi robusti per analizzare testi illimitati sono diventati più diffusi, la prospettiva della comprensione del linguaggio naturale è riemersa come obiettivo plausibile

In questa sezione descriviamo alcune tecnologie di comprensione del linguaggio, per darvi un'idea delle sfida interessanti che vi attendono.

Traduzione meccanica

Per un lungo periodo, la traduzione meccanica (MT) è stata il Sacro Graal della comprensione linguistica, cercando in definitiva di fornire una traduzione idiomatica di alta qualità tra ogni coppia di lingue. Le sue radici affondano ai primi giorni della Guerra Fredda quando la promessa di una traduzione automatica condusse al sostanziale finanziamento del governo, e con esso, alla genesi del NLP stesso.

Oggi, sistemi di traduzione pratica esistono per particolari coppie di lingue, ed alcune sono integrate in motori di ricerca del web. Ad ogni modo, questi sistemi hanno alcuni seri punti deboli. Possiamo esaminarli con l'aiuto del "babelizer" di NLTK (che viene caricato automaticamente quando si importa il materiale di questo capitolo usando `fromnltk.book import *`). Questo programma converte la frase da tradurre in un linguaggio specifico, poi riconverte in inglese la frase risultante da tradurre. Si ferma dopo 12 ripetizioni, o se riceve una traduzione che sia stata già prodotta (indicando un ciclo):

```
>>> babelize_shell()
NLTK Babelizer: type 'help' for a list of commands.
Babel> how long before the next flight to Alice Springs?
Babel> german
Babel> run
0> how long before the next flight to Alice Springs?
1> wie lang vor dem folgenden Flug zu Alice Springs?
2> how long before the following flight to Alice jump?
3> wie lang vor dem folgenden Flug zu Alice springen Sie?
4> how long before the following flight to Alice do you jump?
5> wie lang, bevor der folgende Flug zu Alice tun, Sie springen?
6> how long, before the following flight to Alice does, do you jump?
```

```
7> wie lang bevor der folgende Flug zu Alice tut, tun Sie springen?
8> how long before the following flight to Alice does, do you jump?
9> wie lang, bevor der folgende Flug zu Alice tut, tun Sie springen?
10> how long, before the following flight does to Alice, do do you
jump?
11> wie lang bevor der folgende Flug zu Alice tut, Sie tun Sprung?
12> how long before the following flight does leap to Alice, does
you?
```

Osservate che il sistema traduce correttamente *Alice Springs* dall'inglese al tedesco (nella linea che inizia per `1>`), ma ritraducendo in inglese finisce per diventare *Alice jump* (linea 2). La preposizione *before* è inizialmente tradotta nella corrispondente preposizione tedesca vor, ma dopo nella congiunzione *bevor* (linea 5). Dopo la linea 5 le frasi sono diventate senza senso (ma notate la varia formulazione indicata dalle virgole, e il cambio da *jump* a *leap*). Il sistema di traduzione non ha riconosciuto se una parola facesse parte di un nome proprio, e ha interpretato male la struttura grammaticale. I problemi grammaticali sono più ovvii nell'esempio seguente. John ha trovato il maiale, o il maiale ha trovato John?

```
>>> babelize_shell()
Babel> The pig that John found looked happy
Babel> german
Babel> run
0> The pig that John found looked happy
1> Das Schwein, das John fand, schaute gl?cklich
2> The pig, which found John, looked happy
```

La traduzione meccanica è difficile perché la parola data potrebbe avere svariate possibili traduzioni (a seconda del suo significato), e perché l'ordine della parole deve essere cambiato per mantenere la struttura grammaticale della lingua d'arrivo. Oggi queste difficoltà si stanno affrontando raccogliendo massicce quantità di testi paralleli da giornali e siti web governativi che pubblicano documenti in due o più lingue. Avendo un documento in tedesco ed inglese, e possibilmente un dizionario bilingue, possiamo automaticamente accoppiare le frasi, un processo chiamato **allineamento di testo**. Una volta che abbiamo un milione o più coppie di frasi, possiamo analizzare parole ed espressioni corrispondenti, e costruire un modello che possa essere usato per tradurre un nuovo testo.

Limitazioni di NLP

I sistemi del linguaggio naturale che sono stati impiegati per applicazioni reali non possono ancora svolgere ragionamenti di senso comune o aspirare alla comprensione del mondo in maniera generale e robusta. Possiamo aspettare che questi difficili problemi di intelligenza artificiale vengano risolti, ma non frattempo è necessario vivere con alcune gravi limitazioni sulla capacità di ragionamento e comprensione dei sistemi di linguaggio naturale. Di conseguenza, sin dall'inizio un importante scopo della ricerca NLP è stato fare progressi sul difficile compito di costruire tecnologie che "comprendano il linguaggio", usando tecniche superficiali ma potenti invece di capacità di comprensione e ragionamento illimitate.

Una sfida per il futuro.

Una sfida a lungo termine con l'intelligenza artificiale è stata costruire macchine intelligenti, ed una parte cruciale del comportamento intelligente è capirne il linguaggio. Per molti anni l'obiettivo è stato visto molto difficilmente. Ad ogni modo, come le tecnologie NLP sono maturate, e sono diventati più diffusi metodi robusti per analizzare testi illimitati, la prospettiva della comprensione del linguaggio naturale è riemersa come obiettivo plausibile.