I Computer e il Linguaggio Naturale

You shall know a word by the company it keeps.
R. Firth

Valerio Basile *Dipartimento di Informatica*
Università degli Studi di Torino, Italia

Il linguaggio naturale, con le sue strutture e irregolarità, è il principale oggetto di studio della `Linguistica`, ma anche di `Informatica` e `Intelligenza Artificiale`; interessate analizzarlo, comprenderlo, ed elaborarlo con strumenti computazionali. Le discipline della `Linguistica Computazionale` ed `Elaborazione del Linguaggio Naturale` nascono con lo scopo di coniugare conoscenze da vari campi di studio per svelare tramite computer ed algoritmi i segreti della lingua. In questo articolo, viene presentata una panoramica delle tante sfaccettature di queste aree di ricerca, i problemi che esse affrontano, e le soluzioni proposte negli ultimi decenni di letteratura scientifica.

Introduzione

Il linguaggio naturale è tra le principali espressioni dell'intelligenza umana, una modalità di comunicazione che si è evoluta in forme estremamente diverse e complesse, e che ha suscitato la curiosità degli studiosi fin dagli albori della filosofia. Lo chiamiamo `naturale` per distinguerlo dai linguaggi `artificiali`, ad esempio i linguaggi formali come linguaggi di programmazione, la notazione matematica, o musicale. Allo stesso tempo, il linguaggio naturale è qualcosa che tutti conoscono e usano in modo, appunto, naturale, spesso senza rendersi consciamente conto di seguire delle precise regole linguistiche.

In quanto fenomeno strettamente correlato con l'intelligenza, il linguaggio naturale è ampiamente studiato, oltre che da filosofi, linguisti, scienziati cognitivi, ecc., anche da informatici e ingegneri che si occupano di Intelligenza Artificiale. La **Linguistica Computazionale** (Computational Linguistics in inglese) è quella area di studio e di ricerca che coniuga linguistica e informatica, allo scopo di analizzare il linguaggio naturale con metodi informatici, con modelli statistici, regole formali, e programmi per computer.

Linguistica Computazionale è un termine usato a volte come sinonimo di **Elaborazione del Linguaggio Naturale** (Natural Language Processing, spesso abbreviato in NLP), anche se ci sono ampie discussioni sulle differenze tra queste terminologie. Senza voler prendere una posizione netta, una visione ragionevolmente condivisa sostiene che per Linguistica Computazionale si intende lo studio della lingua supportato da metodologie computazionali, mentre con NLP ci si riferisce alla estesa famiglia di tecniche computazionali che trattano la lingua come dato principale.

In questo articolo, cercherò di fornire delle basi per un'infarinatura generale che copra il più possibile della Linguistica Computazionale, o quantomeno dei problemi più studiati tra quelli di cui questa disciplina si occupa. Realisticamente, l'obiettivo di questo articolo è quello di suscitare nei lettori la scintilla di curiosità che li spinga a investigare oltre, a partire dai numerosi link e puntatori sparsi per l'articolo.

Si rende infine necessario tracciare dei confini riguardo il contenuto del presente articolo, data l'enorme mole di letteratura intorno alla Linguistica Computazionale. Due ampie aree di ricerca sono state volutamente escluse dalla trattazione principale. Innanzitutto, il processamento automatico del linguaggio naturale può naturalmente distinguersi nei due filoni della `comprensione` e della `generazione` del linguaggio naturale (Natural Language Understanding e Generation, rispettivamente, in inglese). In questo articolo, si tratterà principalmente del primo, al cui studio è dedicata una comunità internazionale di gran lunga più numerosa. L'altra distinzione riguarda la lingua `parlata` e la lingua `scritta`. Questo articolo tratterà quasi esclusivamente di tecniche sviluppate per l'elaborazione della lingua scritta.

Linguistica e Computer

La comprensione, ma anche la sola modellazione o processamento automatico, di un'espressione in linguaggio naturale da parte di una macchina è un problema molto complicato. Per questo motivo, l'evoluzione della disciplina ha portato alla sua divisione in molteplici sotto-problemi, la cui soluzione è spesso più gestibile con metodi informatici.

Restringendo il campo alla lingua scritta, dal punto di vista dell'elaboratore eletronico una espressione in linguaggio naturale è, in prima battuta, una sequenza (o `stringa`) di caratteri. Già a questo livello, la sua rappresentazione digitale non è scontata, e infatti sono nati e proliferati una serie di standard come `ASCII` e più recentemente `UNICODE`, allo scopo di convertire i caratteri in codici numerici processabili da un computer. In questo articolo non ci soffermiamo su tali standard di rappresentazione. Piuttosto, nel resto di questa sezione, vedremo come il linguaggio naturale viene processato su livelli via via più astratti a partire dalle sequenze di caratteri fino al significato di espressioni complesse.

Caratteri, parole, punteggiatura e frasi

Per un essere umano in grado di leggere, è scontato guardare ad un testo (come quello che compone il presente articolo) come un insieme di parole e frasi. Tuttavia, questa astrazione non è affatto scontata per un computer, che, come abbiamo scritto, vede il suo input come una stringa di caratteri. Il processo di suddividere un testo in frasi, e le frasi in parole, prende il nome di **tokenizzazione**, da *token*, unità elementare di un testo.

Il problema è all'apparenza banale: basta individuare i segni di punteggiatura che dividono le frasi, e successivamente individuare le parole, separate da spazi o da altri segni di punteggiatura. È così semplice? La risposta è si nella maggior parte dei casi, ma questo semplice insieme di regole lascia fuori una quantità non indifferente di eccezioni. Vediamo un esempio:

> "Giovanni è tornato dagli U.S.A. C'era già stato l'anno scorso."

In questo testo di esempio, ci sono due frasi, composte da 5 e 7 parole rispettivamente (più il punto finale). Si vede come per un sistema basato su semplici regole possa essere problematico separare le coppie di parole "c'era" e "l'anno". ancora più problematiche sono le decisione di separare la prima frase dalla seconda, e mantenere unita l'abbreviazione "U.S.A.": come decidere se i punti fanno parte di un'abbreviazione o sono usati come comuni segni di punteggiatura?

I moderni sistemi di tokenizzazione utilizzano per la maggior parte complessi sistemi di regole, differenti per ogni lingua. Sono stati però proposti anche tokenizzatori più sofisticati, basati su tecniche di apprendimento automatico (più avanti si parlerà più in dettaglio di tali tecniche). Un sistema di tokenizzazione di questo tipo è `Elephant` [2], che impara le regole automaticamente da un *corpus* corretto a mano, indipendentemente dalla lingua. Il nome del sistema fa riferimento alla metafora dell' "elefante nel salotto", alludendo a come la tokenizzazione sia un problema ancora aperto, anche se dato per scontato e risolto in molti contesti.

Lemmi, forme, inflessioni

Quando apriamo un dizionario per cercare un termine, troviamo le `forme base` delle parole di una certa lingua, o cosiddetti `lemmi`. In italiano, ad esempio, la forma base di un verbo è l'infinito, la forma base di un sostantivo è il maschile singolare, e così via. In linguistica, questa distinzione è nota come la differenza tra lemmi e `forme` delle parole. Ad ogni lemma, ad esempio un verbo come "andare" sono associate in genere molte forme differenti, che rispecchiano *feature* come tempi e modi verbali, numero (singolare o plurale), genere (maschile o femminile), ed altre. A seconda della lingua, le `inflessioni morfologiche` dei lemmi sono più o meno numerose e più o meno regolari. Alcune lingue hanno una morfologia più semplice e regolare, come l'inglese[1], altre, come le lingue romanze, più complesse. Tra le lingue considerate più ricche morfologicamente si annoverano usualmente turco e finlandese.

La **morfologia lessicale** (lo studio della forma delle parole) ha sempre interessato i linguisti computazionali, per via della sua natura fatta di regole ed eccezioni. Il *task* principale rispetto alla morfologia in una tipica *pipeline* NLP è quello della **lemmatizzazione**: data in input una parola (quindi una forma in generale flessa), ricavare il corrispondente lemma. Questo task è importante non solo ai fini dell'analisi morfologica in sé, ma anche perché è spesso strumentale ad analisi successive, come la ricerca di parole nel testo da analizzare all'interno di risorse lessicali.

Un approccio semplice ma efficace alla lemmatizzazione consiste nell'implementazione di **formari** computazionali, tabelle che associano ad ogni lemma di una lingua tutte le sue possibili forme flesse. Questo approccio ha il vantaggio di non richiedere computazioni complesse, al di là del mantenimento in memoria della tabella delle forme. Inoltre, in questo modo è possibile gestire le flessioni morfologiche irregolari (es. "andare" → "vado") direttamente, senza bisogno di aggiungere regole particolari. D'altro canto, questo semplice approccio ha un problema di `scalabilità`, poiché diviene difficile considerare termini altamente specifici (es. in ambito bio-

[1] Nella mia tesi di dottorato, ho mostrato come i quattro suffissi *-s, -ed, -ing, -en* coprano la maggior parte della morfologia lessicale della lingua inglese [3]

Forma	Lemma	Feature
gattini	gattino	NOUN-M:p
andarono	andare	VER:ind+past+3+p
fastidiosetto	fastidioso	ADJ:dim+m+s

Tabella 1: *Esempio del contenuto del formario Morph-it! [4].*

medico) o neologismi. Per l'italiano, un formario liberamente utilizzabile è `Morph-it!`[2], contenente 504906 forme flesse per 34968 lemmi [4]. Un esempio del contenuto di `Morph-it!` è riportato in Tabella 1, dove nella terza colonna sono visibili le *feature* morfologiche nominate in precedenza, come parte del discorso, genere, persona, e così via.

Tra i metodi computazionali più studiati in ricerca per l'analisi morfologica troviamo quelli basati su formalismi chiamati **automi a stati finiti**, e più precisamente su **trasduttori**. Questo tipo di formalismi descrive una serie di `stati` e regole di `transizione` tra essi. L'automa prende in *input* sequenze di valori da un vocabolario predefinito (ad esempio lettere, parole, o morfemi) e, a seconda del loro contenuto, vengono attivate diverse regole, formando un percorso all'interno del grafo diretto che costituisce l'automa. Gli automi a stati finiti sono fondamentali, tra le altre cose, per il riconoscimento della correttezza sintattica dei linguaggi formali, come i linguaggi di programmazione. I trasduttori, in particolare, hanno la caratteristica di emettere simboli all'attivazione di ogni regola di transizione.

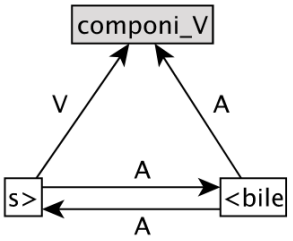

Figura 1: *Esempio di analisi morfologica usando trasduttori a stati finiti. Immagine presa dall'articolo di Tamburini e Melandri [5].*

Un sistema che implementa un modello basato su trasduttori a stati finiti per creare un analizzatore morfologico per l'italiano è `AnIta` [5]. La Figura 1 mostra un grafo che descrive il fram-

[2] `https://docs.sslmit.unibo.it/doku.php?id=resources:morph-it`

mento del sistema capace di riconoscere le parole "comporre", "scomporre", "componibile", e "scomponibile", partendo dalla forma base verbale evidenziata. Le etichette sugli archi in figura indicano il riconoscimento di un verbo (V, come nel caso di "'scomporre') o di un aggettivo (A).

Infine, sono stati proposti approcci alla lemmatizzazione basati su metodi di apprendimento automatico supervisionato. Un esempio tra questi è `morfette` [6], un sistema basato su un classificatore *maximum entropy*, che apprende da una collezione di parole morfologicamente analizzate regole di analisimi morfologica codificate come una sequenza di `correzioni`, come aggiunte, cancellazioni, e sostituzioni di lettere.

Parti del discorso e sintassi

Per molti, il primo approccio con la linguistica, alle scuole elementari, prende la forma dell'analisi grammaticale. Una parte di questo processo è costituito da una versione semplificata dell'analisi morfologica vista nella sezione precedente. L'altro *task* complementare è l'individuazione delle `parti del discorso` (*part of speech*, in inglese). Il *part-of-speech tagging* è il *task* che ha come *input* una sequenza di parole, debitamente separate tra loro (**tokenizzazione**) ed eventualmente lemmatizzate e corredate da informazioni morfologiche, e produce una etichetta (**tag**) per ogni parola che ne identifica la parte del discorso, come ad esempio, verbo, nome, avverbio, articolo determinativo, ecc. Anche questo *task* è importante di per sé, ma anche propedeutico per elaborazioni successive.

Innanzitutto, per etichettare le parole con parti del discorso, bisogna stabilire quali sono le possibilità. Diversi *tagset* sono stati proposti in letteratura, più o meno dipendenti da una certa lingua, ad esempio dalle `Università della Pennsylvania` e di `Standford` per l'inglese. Molte estensioni sono nate per catturare fenomeni specifici di alcune lingue o famiglie di lingue. Recentemente, l'iniziativa `Universal Dependencies`[3] ha lavorato per integrare ed armonizzare i diversi standard, e proporre uno standard, appunto, universale, compreso un tagset di parti del discorso. L'ultima versione (2.5) delle specifiche del il consorzio UD prevedono un *tagset* formato da 17 categorie, di cui 8 sono `classi chiuse`, ovvero contenenti un numero finito di parole per ogni lingua, 6 sono `classi aperte`, e 3 coprono le restanti categorie di *token*. Un esempio di classe chiusa di parole è quella delle preposizioni, un insieme fisso in ogni determinata lingua. Il *tagset* `UD` è raffigurato in Tabella 2. In aggiunta ai 17 *tag*, lo standard `UD` prevede una serie di *feature* per codificare le caratteristiche grammaticali e lessicali aggiuntive, come quelle viste nella sezione sulla morfologia computazionale.

Dato un *tagset* e definito il *task*, come riusciamo tramite metodi computazionali ad assegnare le corrette parti del discorso ad ogni parole di una frase o di un testo? Innanzitutto, notiamo una differenza con il *task* di analisi morfologica che abbiamo visto nella sezione precedente. Mentre per quello può essere sufficiente analizzare una parola alla volta, in questo caso il `contesto` gioca un ruolo fondamentale. L'ambiguità delle parti del discorso è infatti un fattore notevole da prendere in considerazione. Vediamo una frase di esempio:

> "Domani volo a Milano, ma il volo è in ritardo."

In questo esempio, "volo" è un verbo alla prima persona singolare alla sua prima occorrenza, e un sostantivo la seconda. Tuttavia, leggendo la frase non abbiamo dubbi riguardo la sua interpretazione, poiché il contesto rende evidente la giusta attribuzione delle parti del discorso. Alcuni casi, rari, sono effettivamente ambigui anche in presenza dell'intera frase, come la seguente frase conosciuta in linguistica:

> "La vecchia porta la sbarra."

Lascio al lettore il divertente compito di individuare tutti possibili significati di questa frase[4]. Anche senza considerare casi estremi come l'esempio citato, rimane il problema di costruire sistemi automatici per il `POS`-*tagging*. I primi approcci computazionali a questo *task* sono stati basati sulla creazione di regole che associano sequenze di parole con determinate caratteristiche grammaticali ai rispettivi *tag* di parte del

[3] `https://universaldependencies.org/`

[4] Una frase in inglese dalle caratteristiche analoghe è *time flies like an arrow*.

Classi aperte		Classi chiuse		Altri	
ADJ	Aggettivo	ADP	Preposizione	PUNCT	Punteggiatura
ADV	Avverbio	AUX	Ausiliare	SYM	Simbolo
INTJ	Interiezione	CCONJ	Congiunzione coordinata	X	altro
NOUN	Sostantivo	DET	Articolo		
PROPN	Nome proprio	NUM	Numero		
VERB	Verbo	PART	Particella		
		PRON	Pronome		
		SCONJ	Congiunzione subordinata		

Tabella 2: *Il tagset di parti del discorso dello standard Universal Dependencies.*

discorso. Tali regole vanno a formalizzare vere e proprie grammatiche di una lingua, e il loro numero può crescere fino a diventare difficilmente gestibile.

Negli anni '70, grazie al lavoro condotto presso la `Brown University`, è stato reso disponibile il primo *corpus* di testi in lingua inglese annotato con parti del discorso. Il processo di annotazione, condotto a mano da volontari, si è protratto per diversi anni, ma è stato di importanza fondamentale non solo per la creazione di programmi `POS`-*tagger* automatici, ma per lo sviluppo dell'intera disciplina della Linguistica Computazionale. A partire dal `Brown Corpus`, infatti, sono stati sviluppati modelli statistici per il *tagging* di sequenze, basati principalmente su `modelli nascosti di Markov` e `Maximum Entropy`, e, in tempi più recenti, reti neurali. Tali metodi imparano le sequenze di `POS`-*tag* più probabili data una sequenza di input di parole con le loro caratteristiche grammaticali, lessicali e morfologiche, *output* dei *task* visti in precedenza.

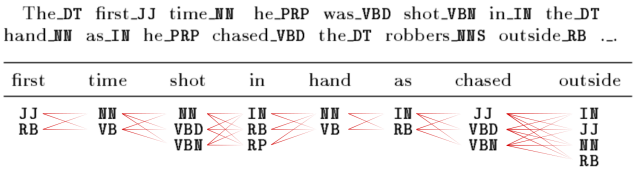

Figura 2: *Ambiguità di parti del discorso e tutte le possibili sequenze di POS-tag. Immagine tratta dall'articolo di Màrquez et al. [9].*

La predizione della corretta sequenza di etichette è interessante dal punto di vista computazionale. Si prenda ad esempio la frase in Figura 2, dove per ciascuna parola sono indicate le possibili parti del discorso. Un sistema come un modello nascosto di Markov mira ad individuare la sequenza più probabile di parti del discorso, quindi, in termini computazionali, il percorso nel grafo sovraimposto alla frase nella figura, che massimizza la probabilità totale. Su ogni arco del grafo si possono infatti immaginare dei numeri che indicano la probabilità di passare da uno stato al successivo, in modo simile al trasduttore a stati finiti visto in precedenza per il *task* di analisi morfologica. Queste probabilità possono essere stimate a partire da un corpus annotato con parti del discorso, come il `Brown Corpus`. I possibili percorsi, tuttavia, rimangono un numero elevato, che cresce `esponenzialmente` al crescere della dimensione dell'*input* (la lunghezza della frase). Già in questa breve frase di esempio le possibili combinazioni di `POS`-*tag* sono $2 \cdot 2 \cdot 3 \cdot 3 \cdot 2 \cdot 2 \cdot 3 \cdot 4 = 1,728$. Sono pertanto stati proposti algoritmi per risolvere questo problema in modo ottimizzato, tra cui probabilmente il più conosciuto è l'algoritmo di `Viterbi` [8], basato sulla tecnica della `programmazione dinamica`.

Nel corso di un paio di decadi, modelli sempre più sofisticati hanno raggiunto valori di accuratezza dal 90% dei primi modelli fino ai valori vicini al 98% dei sistemi più recenti. È comunque interessante notare come una accuratezza del 98% si traduca in circa un errore di assegnamento di parte del discorso ogni 20 parole, una *performance* ancora lontana da quella umana. Allo stato attuale, pertanto, il `POS`-*tagging* è considerato un problema aperto su cui la ricerca rimane attiva e prolifica.

Sintassi del linguaggio naturale

Le parole che compongono un'espressione in linguaggio naturale non appaiono in ordine casuale in una frase. Al contrario, la loro precisa sequenza, e come ogni parola si relaziona alle altre sono fattori fondamentali per comprenderne il significato. La sintassi è l'area della linguistica che studia i rapporti e le relazioni tra le parole, e il modo

in cui questi, insieme, compongono espressioni di senso compiuto.

La letteratura linguistica ha prodotto innumerevoli grammatiche e formalismi per descrivere le regole che permettono alle parole di legarsi tra loro e formare espressioni più lunghe, e non basterebbe un articolo di queste dimensioni neanche a farne una rapida carrellata. Ci limiteremo quindi ad una distinzione a grana grossa tra le due maggiori famiglie di formalismi sintattici del linguaggio naturale.

In generale, l'analisi sintattica di un frammento di linguaggio naturale ha come risultato un `albero`, un caso specifico di grafo in cui ogni nodo ha uno ed un solo nodo `padre`, ad eccezione di un nodo speciale chiamato `radice`, che non ha padre. In nodi situati più "in basso" nell'albero, cioé quei nodi che sono collegati solo ciascuno al proprio nodo padre, sono chiamati `foglie`. Un albero, in informatica, è la maniera naturale di rappresentare informazioni gerarchiche. L'albero che rappresenta la struttura sintattica di una frase è diverso a seconda del tipo di grammatica che si sta utilizzando. Nelle grammatiche a `costituenti` le foglie sono le parole della frase, la radice è la frase intera, e i nodi intermedi sono raggruppamenti di parole, chiamti appunto `costituenti`, che si relazionano tra loro in base alle regole della grammatica usata. Per fare un esempio, guardiamo l'albero a costituenti in Figura 3, dove la radice è il livello più in basso e le foglie sono in alto nei *box* colorati[5]. Qui il formalismo grammaticale utilizzato è la `Combinatory Categorial Grammar`, una grammatica relativamente semplice in termini di regole ma nonostante ciò molto espressiva. In `CCG`, ogni parola ha una `categoria` sintattica che può essere semplice (`N`, `NP`, oppure la categoria speciale `S` che indica l'intera frase), oppure complessa. Le categorie complesse si costruiscono con le categorie semplici più i due operatori \ e / che indicano rispettivamente "questo elemento si compone con un altro elemento a destra" e "a destra". La `CCG` quindi fornisce delle regole su come due costituenti, ognuno con la propria categoria, si combinano per formarne uno nuovo. Ad esempio, NP/N è una categoria complessa che può essere combinata con un N alla propria destra per formare un NP. Seguendo queste regole di composizione, il *parser* (il programma che effettua il *parsing*, ovvero l'analisi sintattica) è capace di ricostruire un albero sintattico completo a partire dalle categorie dei figli.

L'altra famiglia di formalismi sintattici è quella delle grammatiche `a dipendenze`. In questo tipo di grammatiche, Tutti i nodi, compresi radice e foglie, sono parole della frase, e gli archi che le collegano sono etichettati con le loro `dipendenze` ovvero il tipo di relazione che sussiste tra coppie di parole. Un esempio di albero di dipendenze si trova in Figura 4, prodotto dalla versione online del software `UDpipe`, una *pipeline* `NLP` sviluppata all'interno della già citata iniziativa `Universal Dependencies`[6]. Si puo vedere come ad esempio tutta la frase "Tom sta masticando uno stuzzicadenti" `dipenda` dal verbo principale "masticare", il cui soggetto (`nsubj`) è Tom e il cui oggetto (`obj`) è lo stuzzicadenti.

Sia per le grammatiche a costituenti sia per le grammatiche a dipendenze, in letteratura sono stati (e continuano ad essere) proposti un gran numero di algoritmi di `parsing`. Anche per questo *task*, mentre ai principi dello sviluppo della linguistica computazionale sono stati proposti sistemi basati su regole, a partire dagli anni '80 la creazione di corpora annotati con analisi sintattiche (i cosiddetti *treebank*, da *tree*, albero) ha permesso lo sviluppo di metodi basati sulla statistica e l'apprendimento automatico. Senza entrare in eccessivo dettaglio, la maggior parte degli algoritmi più comuni al giorno d'oggi sono di apprendimento automatico supervisionato, basati o sulle `transizioni`, ovvero che cercano di predire la dipendenza tra coppie di parole, o basati su algoritmi che operano sull'intero grafo delle possibili dipendenze tra le parole di una frase.

Le parole e il loro significato

Vedremo ora quei task che riguardano la `semantica`, ovvero lo studio del significato del linguaggio naturale. Il primo livello su cui si può concentrare l'attenzione alla ricerca del si-

[5] Questo esempio è stato estratto dal Parallel Meaning Bank [7], un corpus annotato multilingue liberamente accessibile online presso `https://pmb.let.rug.nl/explorer`.

[6] UDpipe è utilizzabile liberamente, e fornisce analisi linguistica su un gran numero di lingue: http://lindat.mff.cuni.cz/services/udpipe/

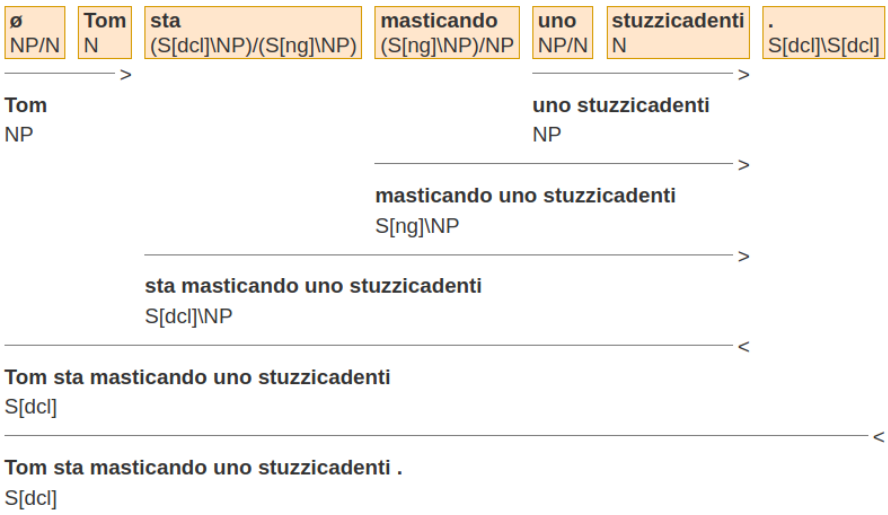

Figura 3: *Esempio di albero sintattico a costituenti che rappresenta la sintassi della frase "Tom sta masticando uno stuzzicadenti", secondo la Combinatory Categorial Grammar.*

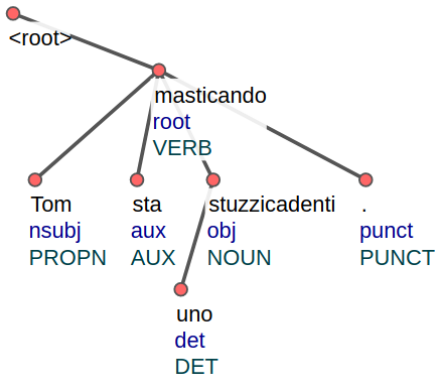

Figura 4: *Albero di dipendenze per la frase "Tom sta masticando uno stuzzicadenti", prodotto da UDpipe.*

gnificato di un'espressione in linguaggio naturale è quello delle parole e del loro significato individuale, ovvero la `semantica lessicale`.

Aprendo un dizionario, possiamo leggere parole e i loro significati, e alcune parole ne hanno più di uno. Questo fenomeno è noto come **polisemia**. Si può parlare di diversi gradi di polisemia, da parole che hanno un solo significato univoco (spesso più una parola è specifica, più tende ad avere un solo senso), fino a parole che possono indicare concetti completamente sconnessi tra loro (esempio: "calcio" come sport e come elemento chimico). La Tabella 3 mostra, a titolo esemplificativo, i sensi della parola "piano" che si trovano in `Wikizionario`[7]

In linguistica computazionale, per elaborare questo tipo di conoscenza sono state create risorse accessibili con strumenti informatici, ad esempio da un programma *software*. La più conosciuta di queste è *WordNet*, un dizionario elettronico inglese creato all'`Università di Princeton` [10]. In `WordNet`, parole e sensi formano un `grafo bipartito`[8]: ogni parola (lemma, più precisamente) è connesso ad uno o più sensi, ed ogni senso è espresso da una o più parole. Sono così rappresentate in maniera intuitiva relazioni come la **sinonimia** (diverse parole connesse allo stesso senso), o **polisemia** (diversi sensi connessi alla stessa parola). I sensi in `WordNet` sono rappresentati in effetti proprio come `synset`, insiemi di sinonimi, oltre ad avere una definizione e degli esempi. La Figura 5 mostra uno *screenshot* dell'interfaccia `Web` di `WordNet` 3.1[9] con i sensi del sostantivo inglese "rock".

La `WordNet` di `Princeton` non è l'unica rete semantica lessicale pubblicamente accessibile. Negli anni, sono state pubblicate molte `WordNet` in diverse lingue, con una struttura simile all'originale. Grazie all'iniziativa `Open Multilingual WordNet` [11], buona parte di queste risorse sono collegate tra loro e sono liberamente scaricabili o interrogabili tramite librerie *software* come il `Natural Language Toolkit` per `Python`[10].

[7] `https://it.wiktionary.org/wiki/piano`

[8] In Informatica, un grafo bipartito è un grafo in cui è possibile dividere i nodi in due gruppi con la proprietà che nessuna coppia di nodi all'interno dello stesso gruppo è connessa da un arco.

[9] `http://wordnetweb.princeton.edu/perl/webwn`

[10] `http://compling.hss.ntu.edu.sg/omw/`

Dominio	Definizione	Esempio
economia, tecnologia, ingegneria	insieme di regole prestabilite per condurre a termine un compito	
matematica, geometria, fisica	insieme di punti individuati dalla combinazione lineare di due vettori linearmente indipendenti applicati nel medesimo punto, le cui curvature fondamentali sono entrambe nulle	
architettura	livello di una struttura o di un edificio	sali al primo piano, dov'è la direzione
musica	sinonimo di pianoforte	mi diletto con il piano quando ho tempo
araldica	pianura	
geografia	estensione rettilinea di un terreno	

Tabella 3: *I significati del sostantivo "piano" secondo il Wikizionario.*

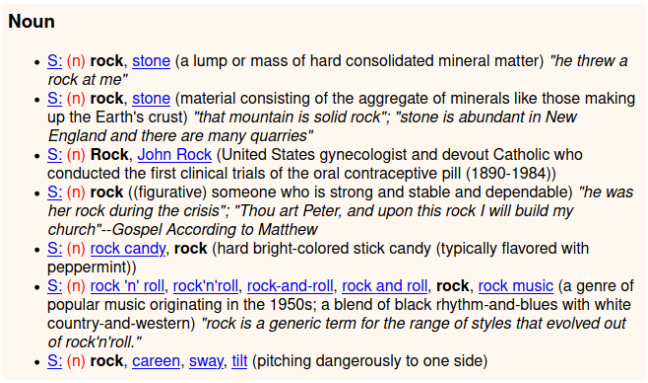

Figura 5: *Screenshot dell'Interfaccia Web di WordNet che riporta i sensi della parola inglese "rock" (sostantivo).*

Il task di attribuire il senso corretto alle parole di un'espressione in linguaggio naturale prende il nome di **disambiguazione**. È un task molto studiato in linguistica computazionale, e considerato difficile. Un algoritmo classico per la disambiguazione è l'algoritmo di `Lesk` [12], basato sulla disponibilità di un dizionario, come ad esempio `WordNet`. L'algoritmo di `Lesk` funziona nel seguente modo: data una parola e le definizioni dei suoi sensi, per ognuno di questi di conta `il numero di parole in comune` tra la definizione e il resto della frase all'interno della quale occorre la parola da disambiguare. Per esempio, considerando la parola "piano" nella frase:

> "Per trasloco dovettero spostare la centralina di tutto l'edificio ad un altro piano".

si nota come la parola "edificio" compaia sia nella frase, sia nella definizione di uno dei sensi della parola, visti nella Tabella 3, mentre non sono presenti sovrapposizioni con le altre definizioni. Si può quindi derurre che "piano" in questo contesto indica "livello di una struttura o di un edificio". L'algoritmo di Lesk è molto semplice, ma cattura l'aspetto fondamentale cruciale per la disambiguazione dei significati delle parole, ovvero che il significato dipende dal contesto. Robert Firth si riferiva proprio a questo fenomeno nello scrivere la frase citata sotto al titolo di questo articolo.

Un approccio computazionale completamente diverso al problema della semantica lessicale è quello della **semantica distribuzionale**. Questo approccio muove dalla osservazione esplicitata poco prima sul ruolo del contesto di una frase nell'attribuzione del significato alle sue parole, ed in particolare si basa fortemente sul concetto di **co-occorrenza** delle parole. Diciamo che due parole co-occorrono quando si trovano entrambe entro una distanza minima (da decidere a seconda delle applicazioni) l'una dall'altra in un testo. Una maniera per descrivere numericamente un *corpus* di testi può essere pertanto una matrice di co-occorrenze, una grande tabella in cui ogni riga e ogni colonna rappresentano una parola, e nella celle ai loro incroci viene riportato il numero di volte che si è osservata una co-occorrenza tra i due rispettivi termini. Una maniera alternativa di descrivere in termini simili un *corpus* è anche una matrice parole-documenti, in cui le righe sono sempre le parole, mentre le colonne rappresentano i documenti che compongono il *corpus*, e ad ogni incrocio si contano le occor-

Parola	D_1	D_2	D_3	D_4	D_5
Cane	1	2	1	0	0
Gatto	0	1	4	0	0
Aereo	0	1	0	1	3

Tabella 4: *Esempio di matrice di co-occorrenze parole-documenti con tre parole, cinque documenti (indicati da D_1, ..., D_5 e in ogni cella il numero di occorrenze di ogni parola in ogni documento.*

renze di ciascuna parola in tali documenti. In entrambi i casi, in corrispondenza di ogni parola, si ottiene un `vettore` numerico, una sequenza di numeri di dimensione fissa, ovvero il numero di parole nel vocabolario nel caso di matrice parole-parole o il numero di documenti nel caso di matrice parole-documenti. La Tabella 4 mostra un esempio di matrice di co-occorrenza parole-documenti. In ogni riga, si trova una parola seguita da un vettore di dimensione cinque (il numero dei documenti). Per esempio, secondo questa matrice, la parola "cane" occorre una volta nel primo documento e due volte nel secondo, mentre la parola "aereo" non occorre nel primo documenti, e così via.

Queste rappresentqazioni vettoriali delle parole sono molto utili in linguistica computazionale, poiché codificano numericamente il contesto in cui le parole occorrono, e rendono quindi verificabile computazionalmente la `ipotesi distribuzionale` [13]:

> "Parole che sono usate e occorrono in contesti simili tendono a convogliare significati simili."

In termini matematici, ci sono diversi modi di calcolare la distanza tra due vettori. Una misura molto usata per questo scopo è la **similarità coseno**:

$$K(\vec{X}, \vec{Y}) = \frac{\vec{X} \cdot \vec{Y}}{||\vec{X}||||\vec{Y}||}$$

ossia l'angolo compreso tra i due vettori, indipendentemente dal numero di dimensioni dello spazio vettoriale, normalizzato in modo da risultare un numero compreso tra -1 (vettori opposti) e 1 (vettori coincidenti). Due vettori con una alta similarità coseno sono vicini tra loro nello spazio vettoriale, in virtù del fatto di rappresentare parole che condividono un maggior numero di contesti. Secondo l'ipotesi distribuzionale, quindi, vettori con maggiore similarità coseno rappresentano parole il cui significato è vicino. Ad esempio, nel caso delle tre parole dell'esempio in Tabella 4, la similarità coseno delle rispettive coppie di parole è la seguente:

$$K(cane, gatto) \approx 0.59$$

$$K(cane, aereo) \approx 0.25$$

$$K(gatto, aereo) \approx 0.27$$

Questo strumento matematico è eccezionalmente versatile e potente, fornendo una maniera automatica e non supervisionata di calcolare rappresentazioni semantico-lessicali a partire da corpora di testo. Numerosi approcci hanno esteso questa tecnica, come ad esempio la `Latent Semantic Analysis` [14], che fa uso di `decomposizione ai valori singolari` per ridurre il numero delle dimensioni e quindi ovviare al problema della sparsità delle co-occorrenze (due parole dal significato simile potrebbero non trovarsi negli stessi contesti semplicemente perché più rare). In tempi più recenti si è iniziato a parlare di *word embedding* per riferirsi alle rappresentazioni vettoriali di parole, e sono stati proposti nuovi metodi basati sulle reti neurali per calcolarle, come nell'algoritmo `word2vec` pubblicato da `Google`.

Semantica della frase, Pragmatica e oltre

In questa sezione, abbiamo visto una carrellata di alcuni dei *task* più studiati in linguistica computazionale, dallo studio della morfologia fino alla semantica lessicale. I campi di applicazione della lingusitica compoutazionale non si esauriscono qui naturalmente, e una loro panoramica comprensiva necessiterebbe un volume di grandi dimensioni. Vorrei quindi presentarne solo alcuni, lasciando alla curiosità del lettore il compito di trovarne di interessanti.

Per cominciare, la semantica non è limitata alla sua accezione lessicale vista nella sezione precedente. Come per la sintassi, esistono diverse teorie e metodologie che mirano a fornire una rappresentazione formale (e quindi computazionale) del significato del linguaggio naturale. Tra queste, ricordiamo la `Discourse`

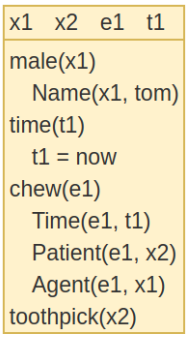

Figura 6: *Esempio di Discourse Representation Structure che rappresenta la semantica della frase "Tom sta masticando unostuzzicadenti".*

`Representation Theory` [1], che rappresenta una frase o un testo per mezzo di formule della logica del prim'ordine, o la più recente `Abstract Meaning Representation`. Il corpus multilingue `Parallel Meaning Bank`, ad esempio, è annotato con le rappresentazioni `DRT` del significato delle sue frasi (Figura 6). Un altro esempio di teoria formale della semantica è la `semantica dei frame`, dove il significato di un'espressione è caratterizzato in termini di *frame* (situazioni) e degli elementi partecipanti, ognuno col proprio ruolo.

Un gran numero di *task* sono relativi all'area chiamata `pragmatica`, ovvero lo studio della lingua dal punto di vista dell'intenzione comunicativa, e in generale di ciò che va al di là del significato in senso stretto di un'espressione in linguaggio naturale. Un *task* molto popolare sia in ricerca accademica sia in ambito industriale è l'`analisi del sentimento` (*sentiment analysis*, o *opinion mining*), che si occupa di classificare ed analizzare le opinioni soggettive e le emozioni espresse in un frammento di linguaggio naturale. Questo task è considerato cruciale anche in ambiti commerciali (es. per misurare l'interesse di clienti verso un prodotto o un servizio) e giornalistico-comunicativi (es. per conoscere l'opinione di utenti di social media), fino ad arrivare ad applicazioni ad alto impatto sociale come l'analisi automatica di cyberbullismo e hate speech.

Ci sono infine innumerevoli applicazioni che abbiamo lasciato fuori dall'esposizione per motivi di spazio, tra cui riportiamo la traduzione automatica (es. `Google Translate o DeepL`), il `question answering` e gli assistenti virtuali (es. `Alexa` o `Siri`), riassunto automatico, generazione di didascalie per immagini, e tante altre. Tutte queste applicazioni hanno come denominatore comune l'uso di una o più delle tecniche descritte in questo articolo, declinate nelle maniere più disparate per diversi obiettivi.

Apprendimento Automatico e NLP

In diversi punti in questo articolo abbiamo visto come molti task di `NLP` vengano abitualmente approcciati con metodi di apprendimento automatico, più comunemente conosciuto con il termine inglese *machine learning*. L'apprendimento automatico è una disciplina estesa e complessa di per sé, per la quale un intero articolo delle dimensioni di quello presente basterebbe solo a riassumerne le caratteristiche. Possiamo però dire che, tra le diverse famiglie di tecniche di apprendimento automatico in uso nel campo del `NLP`, la maggior parte si rifà all'apprendimento `supervisionato` (gli altri tipi principali di apprendimento automatico sono quello `non supervisionato` e il *reinforcement learning*).

Nell'apprendimento supervisionato, un modello matematico-statistico, come ad esempio una rete neurale, viene `addestrato` utilizzando una notevole quantità di `istanze`. Queste possono essere, ad esempio, frasi in linguaggio naturale o parole con associate delle etichette. Le istanze devono essere pertanto preventivamente trasformate in un formato comprensibile al calcolatore, ovvero in serie di numeri, che prendono il nome di *feature*. Nel caso del linguaggio naturale le feature possono essere molteplici: conteggi di parole in una frase, feature morfologiche, `POS`-*tag*, *word embedding*, e così via.

Il programma di apprendimento automatico ha lo scopo di creare un modello numerico che associ un certo insieme di valori delle *feature* alla corrispondente etichetta. In pratica, per addestrare un modello supervisionato si codificano le *feature* delle istanze creando un *training set* (un insieme di istanze di addestramento) e si utilizza poi un ulteriore insieme di dati (il *test set*) con le loro *feature* per misurare la qualità delle predizioni del modello. Il *training set* è tipicamen-

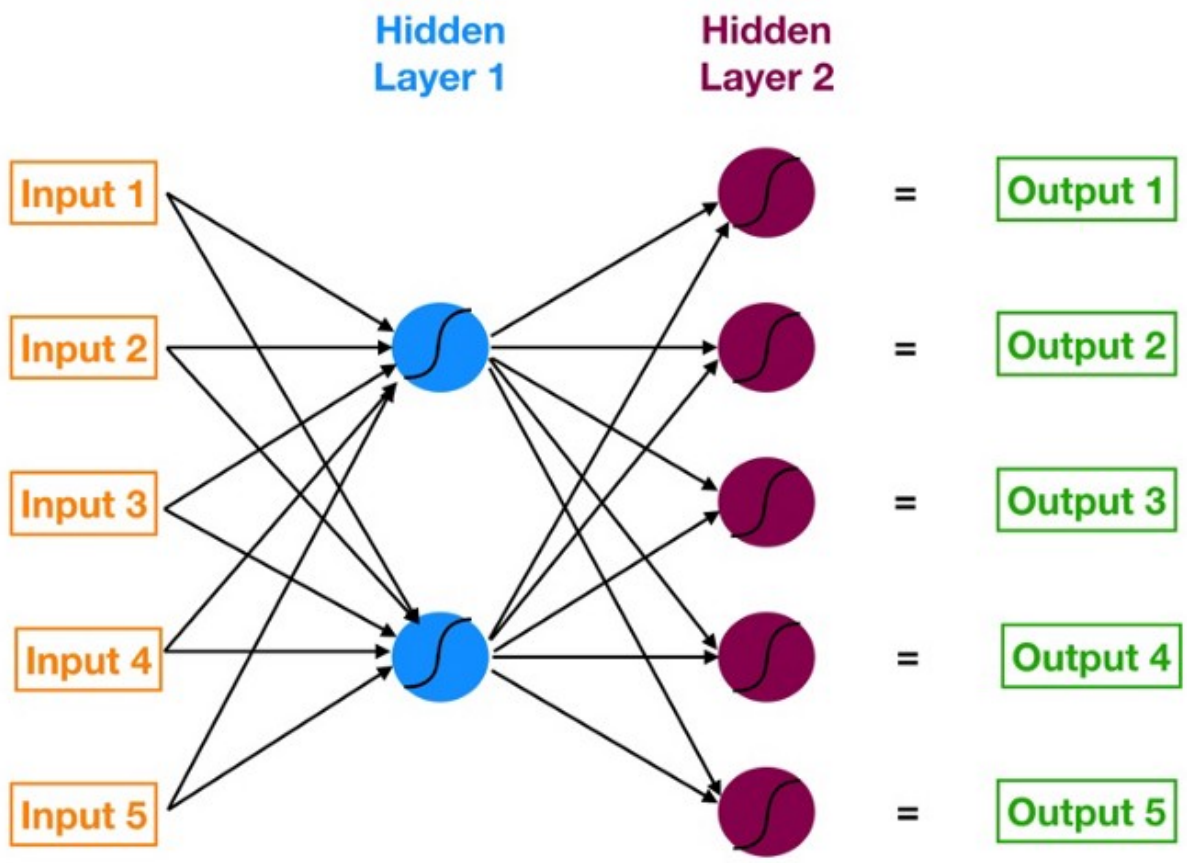

Figura 7: *Esempio di architettura di una rete neurale. Da Towards Data Science:* `https://towardsdatascience.com/understanding-neural-networks-19020b758230`

te molto più grande del *test set*, solitamente in proporzione 80%/20% oppure 90%/10%.

I modelli esistenti per apprendimento automatico sono innumerevoli, e nuovi modelli vengono presentati in continuazione da una comunità internazionale numerosa ed estremamente attiva. Tra i modelli più semplici troviamo il classificatore *naïve Bayes*, che sfrutta il noto teorema di Bayes della statistica per costruire un modello che predica la probabilità di un certo risultato a partire da una serie di osservazioni (feature). L'aggettivo *naïve* in questo caso si riferisce al fatto che questo algoritmo si basa sull'assunzione di indipendenza tra le *feature*, una assunzione relativamente `ingenua` appunto, dato che nella maggior parte dei casi le `feature` che codificano un frammento di testo hanno invece un certo grado di dipendenza reciproca.

Un altro algoritmo di apprendimento automatico supervisionato che ha conosciuto ampio successo in applicazioni `NLP` è il `Support Vector Machine`, in particolare in *task* di classificazione. Senza scendere in troppo dettaglio matematico, questo approccio vede le istanze del *training set* come dei punti in uno spazio ad alta dimensionalità, ovvero il numero di dimensioni è uguale al numero delle `feature`. Ogni istanza è quindi un punto le cui coordinate sono date dalle sue feature. L'algoritmo `SVM` poi procede iterativamente ad individuare il piano (`iperpiano` è la terminologia corretta) che separa in maniera ottimale le classi in cui sono divise le istanze.

Infine, la famiglia di metodi computazionali più usata al giorno d'oggi per task di apprendimento automatico supervisionato è quella delle `reti neurali`. Questi algoritmi, vagamente ispirati dal funzionamento del cervello umano, sono basati sulla costruzione di una rete di unità elementari di calcolo, strutturate a strati — un esempio di rete neurale è riportato in Figura 7. Le *feature* numeriche sono introdotte nel primo strato (*layer* di *input*) in maniera sequenziale, e

la computazione procede modificando ad ogni nuovo *input* i `pesi`, dei numeri che influiscono sulla computazione dei numeri contenuti nei nodi degli strati successivi. Meccanismi di apprendimento come l'algoritmo di *back propagation* fanno sì che i pesi vengano modificati in modo da ridurre l'errore rilevato tra la previsione della rete data dall'ultimo strato (*layer* di *output*) e il valore atteso che si trova nel *training set*. Le reti neurali rappresentano un ampio campo di studio, e ne esistono di moltissime forme e con diverse funzioni. Il sito `Neural Network` Zoo[11] dell'`Istituto Asimov` fornisce una interessante panoramica di alcune delle architettura più popolari.

La Valutazione dei Sistemi di NLP

Nello sviluppo di un qualunque sistema *software*, è di fondamentale importanza mettere in piedi una rigorosa e sistematica procedura di `valutazione`. Cosa vuol dire e cosa implica questa affermazione?

Valutare un sistema di `NLP` significa, in essenza, misurare la qualità della sua analisi rispetto a quello che ci si può aspettare da un essere umano medio, capace di comprendere e produrre linguaggio naturale in maniera normale. L'*output* di un sistema `NLP` va quindi confrontato con un insieme codificato di giudizi umani, un'adeguata metrica va applicata per fornire un valore numerico che indichi la somiglianza tra i due dati. Tale insieme di giudizi umano prende generalmente il nome di **gold standard** (mutuando terminologia dalla finanza).

Abbiamo già accennato ad un esempio di valutazione quantitativa nella sezione sul `POS`-*tagging*, notando come un'`accuratezza` del 99% (un valore numerico) possa considerarsi buona per una macchina ma scadente per un essere umano. Qui con il termine **accuratezza**, si intende una precisa funzione matematica: dato un insieme di possibili etichette $L = \{NN, NNP, VB, ADV, ADJ, ...\}$, una lista di n predizioni $P = \{p_1, ..., p_n\}, p_i \in L$, e un `gold standard` $G = \{g_1, ..., g_n\}, g_i \in L$,

$$\texttt{accuratezza} = \frac{\sum_{i=1}^{n} \texttt{hit}(p_i, g_i)}{n}$$

dove

$$\texttt{hit(x, y)} = \begin{cases} 1 & sex = y \\ 0 & sex \neq y \end{cases}$$

ovvero, l'`accuratezza` è il rapporto tra il numero di predizioni corrette e il numero totale di istanze da predire. Esistono molte altre metriche, per misurare diversi aspetti e applicabili a diversi *task*.

Non abbiamo ancora detto come ottenere il `gold standard`, i dati di riferimento necessari per valutare la *performance* di un sistema. A seconda del problema, ci sono diverse tecniche possibili, ma si può dire che nella maggior parte dei casi un `gold standard` si costruisce con un processo di `annotazione manuale`. Così come nel caso del `Brown corpus` a cui si è accennato in precedenza, la creazione di *corpus* annotato richiede un grande sforzo di tempo da parte di più persone. La presenza di molteplici annotazioni per ogni istanza da annotare è cruciale per garantirne la qualità, poiché persone diverse possono fornire giudizi differenti sullo stesso fenomeno, per errore, per una diversa interpretazione delle istruzioni, o per una genuina divergenza di opinioni. Una volta ottenute le annotazioni, si procede a costruire il vero e proprio `gold standard`, applicando un principio di maggioranza e in alcuni casi discutendo uno ad uno i casi in cui gli annotatori si sono trovati in disaccordo — questo processo è noto come **armonizzazione**.

Parallelamente all'armonizzazione, è importante tenere traccia della misura in cui gli annotatori si sono trovati in accordo sulle loro decisioni. A questo scopo si utilizzano misure di `inter-annotator agreement` (anche chiamato `inter-rater reliability`), una famiglia di funzioni matematiche che forniscono un valore numerico che indica il grado di accordo tra i giudizi dati su un insieme di dati. La letteratura in materia di `inter-annotator agreement` è ampia, ma nel campo del `NLP` tra le misure più usate troviamo la `Kappa di Cohen`, una misura di agreement tra due annotatori, e la `Kappa di Fleiss`, una generalizzazione che copra anche il caso in cui gli annotatori siano in numero supe-

[11] https://www.asimovinstitute.org/neural-network-zoo/

riore a due. La particolarità delle misure Kappa è che nella loro definizione tengono conto della possibilità che gli annotatori abbiano espresso lo stesso giudizio casualmente. Le misure di agreement sono importanti, e solitamente riportate nella descrizione di un *corpus* annotato, anche perché in un certo senso forniscono un limite superiore alla *performance* che ci si può aspettare da un sistema automatico. Una misura di agreement bassa è infatti un segnale che il *task* a cui sono sottoposti gli annotatori è difficile, e pertanto non è ragionevole richiedere ad un algoritmo una *performance* elevata sul medesimo *task*.

Come alternativa al processo lungo e costoso di annotazione manuale, in anni recenti sono state proposte piattaforme *on-line* tramite le quali si può sottomettere un insieme di dati ed ottenere un gran numero di annotazioni da parte di partecipanti in tutto il mondo, con una spesa relativamente contenuta. Tali piattaforme, di cui la più conosciuta è Mechanical Turk di Amazon[12] agiscono da intermediario tra il ricercatore e i partecipanti che vengono pagati tipicamente pochi centesimi per ogni istanza annotata. Questa alternativa, chiamata **crowdsourcing** allevia il problema dei lunghi tempi di annotazione da parte di un piccolo gruppo di esperti, ed è più economica dell'assunzione diretta di personale dedicato. D'altro canto, il ricercatore ha un minor controllo sulle persone che esprimono i giudizi che vanno a formare il gold standard, e pertanto si rende necessario mettere a punto procedure per garantire la qualità e la buona fede delle annotazioni.

L'annotazione manuale, sia in forma di crowdsourcing sia come annotazione da parte di esperti, è la metodologia più comune per creare gold standard per *task* di linguistica computazionale, ed è stata studiata sotto molti punti di vista. Recentemente, alcuni studiosi si stanno interrogando sulla validità della procedura usuale di annotazione e armonizzazione, ad esempio notando come il processo di armonizzazione a maggioranza tenda a rimuovere opinioni di minoranza che possono però essere rilevanti per il fenomeno studiato [15].

Per completare il quadro, quando si valuta un sistema NLP è buona norma presentare anche i risultati di un algoritmo il più possibile semplice, che fornisca una *baseline*, un termine di paragone verso il basso per misurare l'effettiva bontà del sistema proposto. Alcuni *task* hanno *baseline* molto elevate – è noto il caso della disambiguazione, per la quale scegliere il senso più comune di ogni parola porta ad una *performance* assoluta elevata. Una buona *baseline* ed una accurata misura di agreement, insieme, forniscono l'intervallo all'interno del quale può variare il risultato della valutazione di un sistema secondo una determinata metrica, facilitandone quindi l'interpretazione.

La Linguistica Computazionale in Italia

In tutto il mondo, la ricerca in Linguistica Computazionale e in Elaborazione Linguaggio Naturale si tiene principalmente all'interno di dipartimenti universitari di Informatica, Linguistica, ma anche Ingegneria o Scienze Cognitive e altri, oltre che istituti di ricerca pubblici e privati, e aziende di ogni dimensione. L'Italia non è da meno, potendo vantare un buon numero di gruppi di ricerca su tutto il territorio, in molte Università e presso centri come il Consiglio Nazionale delle Ricerche (sede dell'Istituto di Linguistica Computazionale «A. Zampolli»[13]) e la Fondazione Bruno Kessler.

Nel 2015, si è costituita l'Associazione Italiana di Linguistica Computazionale (AILC)[14], allo scopo di promuovere e supportare le attività di ricerca e divulgazione in quest'area sul nostro territorio. Tra le sue attività, AILC organizza annualmente un convegno (Italian Conference on Computational Linguistics, CLiC-it), quest'anno alla sua settima edizione[15]. Inoltre, AILC supporta l'organizzazione della campagna di valutazione delle tecnologie del linguaggio EVALITA, che dal 2007 propone ad ogni sua edizione una serie di task invitando ricercatori, studenti e aziende a sviluppare sistemi all'avanguardia per stimolare e valutare lo stato dell'arte.

[12] https://www.mturk.com/

[13] http://www.ilc.cnr.it/it/content/istituto

[14] https://www.ai-lc.it

[15] http://clic2020.ilc.cnr.it/it/home/

Le attività della AILC si svolgono in parallelo e con frequenti punti di contatto con la Associazione Italiana per l'Intelligenza Artificiale (AIxIA)[16], stabilita nel 1988, che coordina e supporta l'attività di ricerca e divulgazione sui temi della IA in Italia. Il workshop Natural Language for Artificial Intelligence (NL4AI), quest'anno alla quarta edizione, si tiene annualmente all'interno del convegno nazionale della AIxIA, e rappresenta un punto di contatto tra le due comunità[17].

Gli atti delle conferenze sopracitate, contenenti tutti gli articoli ivi presentati, sono generalmente disponibili gratuitamente online.

Conclusione

In questo articolo si è data una panoramica dei molteplici problemi e soluzioni che si incontrano quando si applicano tecniche computazionali allo studio dei fenomeni del linguaggio naturale. Abbiamo passato in rassegna numerosi *task* relativi ai diversi livelli di analisi linguistica, con cenni alle tecniche classiche e quelle più all'avanguardia, e rilevato l'importante di valutare i sistemi NLP con criteri scientifici rigorosi.

Come detto nell'introduzione, interi campi di studio sono stati esclusi dalla trattazione. Tra questi, la generazione del linguaggio naturale, che comporta tutta una serie di problematiche peculiari, a cominciare dalla sua stessa definizione. Se per l'analisi del linguaggio naturale, tema principale di questo articolo, l'*input* del problema è sempre qualche tipo di espressione linguistica e l'*output* è qualche struttura formale, come etichette di parti del discorso, o alberi sintattici, a seconda del *task*, nel caso della generazione non è affatto stabilito da dove debba partire un programma che produca linguaggio naturale. Abbiamo anche escluso l'intero campo dello studio computazionale del linguaggio parlato, che include oltre a molti degli aspetti visti in questo articolo anche sfide tecnologiche e teoriche legate al suono e alla sua forma d'onda, all'intonazione, e così via.

La comunità italiana e internazionale che studia questi problemi è ampia e vivace, e sempre aperta a nuovi studiosi interessati ai fenomeni della lingua. Se anche uno dei lettori di Ithaca si avvicinerà così all'affascinante mondo della Linguistica Computazionale, questo articolo avrà raggiunto il suo scopo.

[16] https://aixia.it/

[17] http://sag.art.uniroma2.it/NL4AI/

[1] H.Kamp, J. van Genabith, U. Reyle: *Discourse representation theory. Handbook of philosophical logic*. Springer, Dordrecht (2011).

[2] K. Evang et al.: *Elephant: Sequence Labeling for Word and Sentence Segmentation*, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA (2013).

[3] V. Basile: *From Logic to Language: Natural Language Generation from Logical Forms*, University of Groningen, Groningen, Netherlands (2015).

[4] E. Zanchetta, M. Baroni: *Morph-it! A free corpus-based morphological resource for the Italian language*, Proceedings from the Corpus Linguistics Conference Series, Vol. 1, University of Birmingham, Birmingham, United Kingdom (2005)

[5] F. Tamburini, M. Melandri: *AnIta: a powerful morphological analyser for Italian*, Proceedings of the Eighth International Conference on Language Resources and Evaluation, European Language Resources Association, Istanbul, Turkey (2012).

[6] G. Chrupala, G. Dinu, J. van Genabith: *Learning Morphology with Morfette*, Proceedings of the sixth International Conference on Language Resources and Evaluation, European Language Resources Association, Marrakech, Morocco(2008).

[7] L. Abzianidze, J. Bjerva, K. Evang, H. Haagsma, R. van Noord, P. Ludmann, D. Nguyen, J. Bos: *The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations*, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Valencia, Spain.

[8] G. Forney: *The Viterbi Algorithm: A Personal History*, University of Southern California (2005).

[9] L. Màrquez, L. Padro, H. Rodríguez: *A machine learning approach to POS tagging*, Machine Learning, 39.1, Springer (2000).

[10] G. A. Miller: *WordNet: A Lexical Database for English*, Communications of the Association for Computing Machinery, 38-11, Association for Computing Machinery, New York, NY, USA (1995).

[11] F. Bond, R. Foster: *Linking and extending an open multilingual wordnet*, Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Sofia, Bulgaria (2013).

[12] M. Lesk: *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*, Proceedings of the 5th annual international conference on Systems documentation, Association for Computing Machinery, New York, NY, United States (1986).

[13] Z. S. Harris: *Distributional Structure*, Word, 10(2-3) (1954).

[14] T. K. Landauer, S. T. Dumais: *A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*, Psychological review, 104(2), American Psychological Association (1997).

[15] M. Klenner, A. Göhring, M. Amsler: *Harmonization Sometimes Harms*, Proceedings of the 5th Swiss Text Analytics Conference and the 16th Conference on Natural Language Processing, CEUR Workshop Proceedings, Zurich, Switzerland (2020).

Valerio Basile: è Ricercatore di Informatica presso il gruppo Content-centered Computing del Dipartimento di Informatica dell'Università degli Studi di Torino, e dell'Hate Speech Monitoring Lab. Ha studiato Informatica a Bologna, laureandosi con il Prof. Fabio Tamburini con una tesi sulla semantica distribuzionale, e ha conseguito il dottorato di ricerca presso l'Università di Groningen, sotto la supervisione del Prof. Johan Bos con una tesi sulla generazione del linguaggio naturale. I suoi interessi di ricerca sono nel campo dell'Intelligenza Artificiale e della Elaborazione del Linguaggio Naturale. In particolare, i suoi lavori sono soprattutto negli ambiti della semantica computazionale, analisi del sentimento e del linguaggio abusivo, generazione del linguaggio naturale, analisi dei social media, e creazione di risorse linguistiche come il Groningen Meaning Bank e TWITA. È membro dell'Associazione Italiana di Linguistica Computazionale, della Associazione Italiana per l'Intelligenza Artificiale, comitati scientifici di numerose conferenze internazionali, e revisore per diverse riviste scientifiche. Durante il picco della pandemia di COVID-19 del 2020, ha pubblicato un bollettino giornaliero su linguistica computazionale e oltre, *The Copula and the Bit*.[18]

[18] http://valeriobasile.github.io/tcatb/