Natural Language Processing

Dispense del corso A.A. 2024/2025

Gabriel Rovesti
Dipartimento di Matematica, Università di Padova

19 aprile 2025

Sommario

Queste dispense raccolgono il materiale del corso di Natural Language Processing, focalizzandosi sugli aspetti teorici e pratici dell'elaborazione del linguaggio naturale. Il documento copre: introduzione alla linguistica computazionale, fondamenti di machine learning, preprocessing del testo, rappresentazioni vettoriali, modelli del linguaggio, reti neurali per NLP, e applicazioni come sentiment analysis, question answering, e machine translation.

Indice

Capitolo 1

Introduzione al Natural Language Processing

1.1 Definizione e Obiettivi

Il Natural Language Processing (NLP) è l'insieme dei metodi che rendono il linguaggio umano accessibile ai computer. L'obiettivo principale è permettere alle macchine di comprendere e generare testo in linguaggio naturale, simulando la capacità umana di processare il linguaggio.

Il testo rimane una delle principali modalità con cui le informazioni vengono condivise, rendendo l'NLP un campo di fondamentale importanza nell'informatica moderna.

1.2 Applicazioni Significative

L'NLP ha numerose applicazioni in diversi campi:

- **Analisi** - Parafrasi, implicazione testuale, sentiment analysis, attribuzione di autore, rilevamento di fake reviews e account, question answering, categorizzazione di testo, rilevamento di fake news, topic modeling.
- **Generazione** - Traduzione automatica, giornalismo automatico, question answering, chatbot.
- **Intelligenza Artificiale Generativa** - Sistemi come Copilot (testo a codice), DALL-E/Midjourney (testo a immagine), Pika/Sora (testo a video).

1.3 NLP e Discipline Correlate

L'NLP si interseca con diverse discipline:

- **Linguistica**: lo studio scientifico del linguaggio
- **Etica**: questioni di accessibilità, bias, privacy e libertà di internet
- **Scienze Sociali Computazionali**: NLP come strumento di analisi
- **Information Retrieval**: recupero di informazioni pertinenti

1.4 Approcci all'NLP

Esistono due principali paradigmi nell'NLP:

- **NLP Tradizionale (basato sulla linguistica)**: utilizzo di preprocessing complesso del testo (tokenizzazione, rimozione di stopwords, analisi sintattica, ecc.) prima dell'applicazione di algoritmi di machine learning. L'esperienza del ricercatore è fondamentale nella scelta del preprocessing adeguato.
- **Deep Learning End-to-End**: approccio che potenzialmente non richiede preprocessing. L'input è semplicemente una frase (codificata numericamente) e l'output è direttamente il risultato desiderato (ad esempio, la polarità del sentimento). Questi modelli richiedono grandi quantità di dati per l'addestramento.

1.5 Sfide del Machine Learning per NLP

L'NLP presenta diverse sfide specifiche per il machine learning:

- Dominio discreto (a differenza, ad esempio, della computer vision)
- Difficoltà nell'applicare algoritmi di ottimizzazione basati sul gradiente
- Frasi non sempre ben formate (ad esempio, da parte di non madrelingua)
- Importanza di valutare il tipo di linguaggio specifico di un task
- Compositionalità del linguaggio (il significato è una funzione complessa delle parti e della sintassi)
- Presenza di molte parole poco frequenti
- Creazione continua di nuove parole (slang)

1.6 Modelli Pre-addestrati

I modelli di deep learning end-to-end necessitano di molti dati per addestrare i numerosi parametri. Per colmare questo divario, sono stati sviluppati modelli pre-addestrati:

- BERT e derivati, XLNet, GPT e versioni successive
- Reti neurali molto grandi già addestrate su task specifici per ottenere una buona rappresentazione del significato di parole/frasi
- Possono essere utilizzati come punto di partenza per risolvere un task specifico (fine-tuning)
- Prompt Engineering: perfezionare i modelli pre-addestrati in modo che possano ricevere la descrizione di un task invece di essere fine-tuned per esso

Capitolo 2

Elementi di Linguistica

2.1 Fondamenti di Linguistica

La linguistica è lo studio scientifico del linguaggio, in particolare la relazione tra la forma del linguaggio e il suo significato.

Un esempio chiaro è la varietà di parole usate in lingue diverse per lo stesso concetto: Computer, ordinateur, ordenador, calcolatore. Il significato attribuito a una forma è solo una convenzione, pertanto può cambiare per una persona o un gruppo, a seconda del contesto sociale.

2.2 Livelli di Descrizione Linguistica

La descrizione linguistica può essere organizzata in diversi livelli, dalla forma al significato:

1. **Fonologia**: studio dei sistemi sonori delle lingue umane, cioè come i suoni sono organizzati e utilizzati.
2. **Morfologia**: studio della formazione e della struttura interna delle parole.
3. **Sintassi**: studio delle regole e dei vincoli che governano l'organizzazione delle parole in frasi.
4. **Semantica**: studio del significato delle espressioni linguistiche come parole, frasi e proposizioni.
5. **Pragmatica**: studio del modo in cui le espressioni linguistiche con i loro significati semantici vengono utilizzate per specifici obiettivi comunicativi.

2.3 Morfologia

La morfologia è lo studio della formazione e della struttura interna delle parole.

- **Morfemi**: le più piccole unità di linguaggio dotate di significato
- **Parola**: combinazione di uno (o più) morfemi radice e affissi (prefissi/suffissi/infissi/circonfissi)
- Esempi: un+speak+able, carry+ing

2.3.1 Tipi di Morfologia

- **Morfologia derivazionale**: crea nuove parole con nuovi significati (e spesso con nuove parti del discorso)
  - Esempio: parse -> parser
- **Morfologia flessiva**: aggiunge informazioni a una parola coerenti con il suo contesto all'interno di una frase (non cambia la semantica del morfema radice)
  - Esempio di tempo: Walk -> walked

2.3.2 Classificazione delle Lingue

Le lingue possono essere classificate in base alla loro ricchezza morfologica:

- **Lingue isolanti** (inglese, cinese): morfologicamente semplici
- **Lingue sintetiche** (finlandese, turco, ebraico): possono avere molti affissi attaccati a un morfema radice

2.4 Sintassi

La sintassi studia le regole e i vincoli che governano l'organizzazione delle parole in frasi.

2.4.1 Part of Speech (POS)

Una categoria di parole che svolgono ruoli simili all'interno della struttura sintattica di una frase:

- **Definizione distribuzionale**: tutte le parole che manterrebbero la grammaticalità della frase
- **Definizione funzionale**: il ruolo della parola (es. avverbi = modificano verbi; verbi = predicati; sostantivi = argomenti)

Esistono diversi set di tag, ad esempio:

- Penn Treebank
- Universal Part-of-Speech Tags

2.4.2 Frasi e Costituenti

Le parole formano raggruppamenti chiamati frasi o costituenti:

- Espressioni che svolgono lo stesso ruolo (es. "un libro" e "un libro molto interessante sulla grammatica")
- Ogni frase ha una testa sintattica, la parola "principale" (es. "libro")
- La testa sintattica determina la distribuzione esterna

2.4.3 Alberi di Dipendenza

Gli alberi di dipendenza evidenziano le relazioni importanti (semantiche) tra le parole:

- Mostrano chi ha compiuto l'azione specificata dal verbo e verso chi
- Utili per gli utenti, ma presentano problemi di ambiguità sintattica durante la costruzione

2.5 Semantica

La semantica è lo studio del significato delle espressioni linguistiche come parole, frasi e proposizioni. Si concentra su ciò che le espressioni significano convenzionalmente, piuttosto che su ciò che potrebbero significare in un particolare contesto (quest'ultimo è l'oggetto della pragmatica).

2.5.1 Significato delle Parole

- **Lessema**: un insieme di forme di parole che condividono lo stesso significato fondamentale. Es. run, runs, ran, running
- **Lemma**: una delle forme del lessema che rappresenta la sua forma canonica (forma di dizionario). Es. run
- **Ambiguità lessicale**: quando lo stesso lemma ha più significati. Es. bank

2.5.2 Relazioni tra Lessemi con lo stesso Lemma

- **Omonimia**: due lessemi identici con significati non correlati. Es. bank (banca), bank (riva)
- **Polisemia**: due lessemi identici il cui significato è semanticamente correlato per estensione. Es. bank (istituzione finanziaria), bank (repository biologico)

2.5.3 Relazioni tra Lessemi con Lemmi diversi

- **Sinonimia**: due lessemi diversi con lo stesso significato o molto simile. Es. divano, sofà; automobile, macchina
- **Antonimia**: due lessemi diversi con significato quasi opposto. Es. freddo, caldo; leader, seguace
- **Iponimia**: due lessemi diversi di cui il significato di uno è più specifico dell'altro. Es. auto, veicolo; laptop, computer
- **Iperonimia**: due lessemi diversi di cui il significato di uno è meno specifico dell'altro. Es. mobili, sedia; computer, laptop

2.5.4 Significato della Frase

- **Principio di Composizionalità**: il significato di un'espressione complessa è determinato (solo) dalla sua struttura e dal significato dei suoi costituenti
- Eccezioni: espressioni idiomatiche
- Dipendenza dal contesto
- Gli ascoltatori sfruttano molti tipi di segnali per ottenere il significato comunicativo previsto

2.6 Pragmatica

La pragmatica studia il modo in cui le espressioni linguistiche con i loro significati semantici vengono utilizzate per specifici obiettivi comunicativi. Si occupa del significato di un testo in un contesto specifico.

2.6.1 Atti Linguistici

Un'azione eseguita attraverso il linguaggio:

- "Puoi passarmi il sale?" - richiesta (non domanda)
- Utile nei sistemi di dialogo

2.6.2 Discorso

Il discorso si riferisce a un testo con più sottotemi e relazioni di coerenza tra loro, come:

- Spiegazione
- Elaborazione
- Contrasto

Il dialogo è un tipo cooperativo di discorso, in cui sono coinvolti due o più partecipanti.

Capitolo 3

Fondamenti di Machine Learning per NLP

3.1 Introduzione al Machine Learning

Uno degli obiettivi del Machine Learning è simulare il ragionamento induttivo: dato un insieme di esempi di un fenomeno, inferire la regola $y = h(x)$ che lo governa.

x	y
1	2
2	4
3	6
4	?

Figura 3.1: Esempio di apprendimento induttivo: determinare $h(x)$

In questo esempio, si può inferire che $h(x) = 2x$.

3.2 Fasi del Machine Learning

In un algoritmo di Supervised Learning distinguiamo due fasi:

1. **Training** (stima di h dagli esempi): utilizziamo un insieme di coppie $(x^{(i)}, y^{(i)})$ per trovare una funzione h che mappa gli input $x^{(i)}$ agli output $y^{(i)}$

2. **Prediction**: una volta addestrato il modello, possiamo usare $h()$ per prevedere nuovi valori. Ad esempio, $h(5) = 10$, $h(6) = 12$, ecc.

3.3 Apprendimento Supervisionato

- **Obiettivo**: dare la "risposta giusta" per ogni esempio nei dati
- **Definizione formale**: dato un dataset $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$, dove le m coppie sono estratte in modo i.i.d. (indipendente e identicamente distribuito) da una distribuzione $P(x, y)$

- **Obiettivo**: trovare la "migliore" funzione $h : X \to Y$ sfruttando D

Il dominio X può essere qualsiasi cosa, normalmente è un vettore d-dimensionale $\mathbb{R}^d$. Il codominio Y determina il tipo di problema:

- $Y = \{0, 1\}$ o $Y = \{-1, 1\}$: classificazione binaria (classi mutuamente esclusive)
- $Y = \{0, 1, \ldots, k\}$: problema di classificazione multiclasse (classi mutuamente esclusive)
- $Y = \{0, 1, \ldots, k\}^d$: problema di classificazione multi-etichetta
- $Y = \mathbb{R}$: regressione

3.4 Teorema No Free Lunch

Il training set da solo non è sufficiente per selezionare h, poiché ci sono infinite funzioni coerenti con il training set. Il teorema No Free Lunch afferma che:

- "Ogni algoritmo di ML di successo deve fare delle assunzioni"
- "Non esiste un singolo algoritmo di ML che funzioni per ogni situazione"

3.5 Stima del Rischio

Come selezionare la migliore h? Il rischio empirico $R(h, D)$ è uno stimatore non distorto del rischio vero $R(h, P)$.

Si può utilizzare un insieme di test per stimare il rischio, a condizione che:

- L'insieme di test (ma anche il training set) sia rappresentativo di P
- Non sia utilizzato durante la selezione di h

3.6 Problemi Comuni: Bias e Varianza

Il rischio empirico $R(h, D)$, dove D è il training set, può presentare alcuni problemi:

- **Alta varianza**: sensibilità ai cambiamenti nel training set
- **Alto bias**: incapacità del modello di rappresentare tutti i possibili training set

3.6.1 Overfitting e Underfitting

- **Underfitting**: incapacità di adattarsi ("alto" errore) al training set e, di conseguenza, anche al test set
- **Overfitting**: il modello si adatta troppo bene al training set, ma non è in grado di generalizzare bene al test set

Il trade-off tra bias e varianza è centrale nel machine learning:

- Funzione troppo semplice (alto bias): rischio di underfitting
- Complessità ottimale: buon equilibrio tra bias e varianza
- H troppo "potente" (alta varianza): rischio di modellare il rumore, overfitting

3.7 Regolarizzazione

La regolarizzazione è uno strumento per "controllare" la complessità di un algoritmo di apprendimento. Di solito prevede un iperparametro i cui valori estremi producono:

- Un modello molto complesso da un lato (che potrebbe portare all'overfitting)
- Un modello troppo semplice dall'altro (che potrebbe portare all'underfitting)

3.8 Selezione del Modello

La tecnica Hold-out: manteniamo un sottoinsieme di v campioni dal training set (il validation set) per valutare i nostri iperparametri.

- Un classificatore/regressore viene addestrato su $m - v$ campioni
- I parametri (θ) e gli iperparametri (λ) vengono ottimizzati rispettivamente sul training e sul validation set

3.9 Algoritmi di Machine Learning

3.9.1 Regressione Lineare

La regressione lineare è uno degli algoritmi più semplici del machine learning.

- **Modello**: $h_\theta(x) = \theta_0 + \theta_1 x$
- **Obiettivo**: scegliere θ_0, θ_1 in modo che $h_\theta(x)$ sia vicino a y per i nostri esempi di training $\{(x^{(i)}, y^{(i)})\}$
- **Funzione di costo**: somma dei quadrati delle distanze (residui)

Per ottimizzare i parametri, si utilizza l'algoritmo di Gradient Descent:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla J(\boldsymbol{\theta}^k) \tag{3.1}$$

dove $\boldsymbol{\theta} = \{\theta_0, \ldots, \theta_n\}$ e $\eta > 0$ è il learning rate.

3.9.2 Naive Bayes

Il classificatore Naive Bayes è un algoritmo generativo che costruisce un modello di come una classe potrebbe generare alcuni dati di input.

Si basa sul teorema di Bayes:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{3.2}$$

L'assunzione "naive" (ingenua) è che le caratteristiche (features) f_i siano indipendenti tra loro data la classe:

$$P(f_1, f_2, \ldots, f_n|c) = \prod_{i=1}^{n} P(f_i|c) \tag{3.3}$$

Questa assunzione semplifica notevolmente il calcolo, anche se nella realtà le features sono raramente indipendenti.

3.9.3 Percettrone e Regressione Logistica

Percettrone

Il Percettrone è un algoritmo di apprendimento per problemi di classificazione binaria. Dato $x \in \mathbb{R}^d$, il suo spazio di ipotesi è composto da tutte le funzioni lineari nello spazio di input: $\theta^T x$.

Regressione Logistica

La Regressione Logistica è un modello discriminativo che utilizza la funzione logistica (sigmoide) per trasformare l'output di una funzione lineare in una probabilità:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{3.4}$$

$h_\theta(x)$ può essere interpretato come la probabilità che x appartenga alla classe 1.

3.9.4 Reti Neurali

Le reti neurali artificiali sono composte da neuroni (unità computazionali) collegati tra loro. Un neurone artificiale prende un input vettoriale x, lo combina linearmente con dei pesi θ e applica una funzione di attivazione non lineare:

$$h_\theta(x) = f(\theta^T x) \tag{3.5}$$

Le reti neurali possono avere più strati (hidden layers) che permettono di apprendere rappresentazioni sempre più astratte dei dati.

I parametri della rete vengono appresi attraverso l'algoritmo di backpropagation, basato sul gradient descent. La superficie di errore è complessa e può presentare molti minimi locali.

3.10 Funzioni di Valutazione

3.10.1 Metriche per la Classificazione

Per valutare le prestazioni di un classificatore, si utilizzano diverse metriche:

- **Accuracy**: percentuale di predizioni corrette
- **Matrice di confusione**: confronta le etichette predette con quelle reali, evidenziando i tipi di errori
- **Precision**: $TP/(TP + FP)$ - quante delle istanze predette positive sono realmente positive
- **Recall**: $TP/(TP+FN)$ - quante delle istanze realmente positive sono state predette positive
- **F1 Score**: media armonica di precision e recall

3.10.2 Data Sbilanciati

In caso di classi sbilanciate (una classe molto più frequente dell'altra), l'accuratezza può essere fuorviante. In questi casi, è preferibile usare metriche come precision, recall, F1 score, o l'area sotto la curva ROC.

Capitolo 4

Preprocessing del Testo e Creazione di Dataset

4.1 Definizione di un Task di Apprendimento

Un task di apprendimento supervisionato è definito come la ricerca di una funzione $f : X \to Y$, conoscendo una lista di coppie (x, y) estratte in modo i.i.d dalla distribuzione "vera" sottostante $P(x, y)$.

- X dominio di input (es. testo in linguaggio naturale)
- Y potrebbe essere:
  - $\{0, 1\}$: classificazione binaria
  - $\{0, \ldots, N\}$: classificazione multiclasse
  - un numero reale: regressione (similarità tra frasi)
  - $\{0, 1\}^m$: classificazione multi-etichetta (tagging di immagini)

4.2 Annotazione dei Dati

Le etichette y possono essere ottenute tramite:

- Annotazione manuale
- Sfruttamento di metadati disponibili
- Combinazione di entrambe le strategie

Una campagna di annotazione deve essere:

- Espressiva (creando istanze effettive di $P(x, y)$)
- Replicabile
- Scalabile

4.3 Raccolta di Dati

La raccolta dei dati deve garantire che non venga introdotto alcun bias, o almeno che ne siamo consapevoli:

- Raccolta di tweet per prevedere il sentiment dell'intera popolazione (bias di densità)
- Raccolta di post Facebook dal 2009 (bias temporale)
- Raccolta di dati da un dominio specifico, ad esempio articoli scientifici (bias di dominio)

È importante avvertire che il modello ML potrebbe apprendere il bias stesso (sezione Limitazioni del paper).

4.3.1 Bias nella Raccolta dei Dati

- **Auto a guida autonoma**: più probabilità di riconoscere pedoni bianchi rispetto a pedoni neri, con conseguente diminuzione della sicurezza per gli individui dalla pelle più scura.
- **Screening CV**: l'algoritmo ML era distorto contro le donne perché basato sui CV ricevuti dall'azienda negli ultimi 10 anni, prevalentemente da uomini.
- **Gestione dell'assistenza sanitaria ad alto rischio**: un algoritmo usato per identificare i pazienti che trarrebbero beneficio dalla "gestione dell'assistenza sanitaria ad alto rischio" utilizzava la spesa sanitaria come proxy per il bisogno di cure, favorendo involontariamente le persone più ricche che hanno accesso a cure più costose.

4.3.2 Bias Implicito

- Alcuni algoritmi di riconoscimento delle immagini, invece di imparare a riconoscere il tipo di animali, imparano a distinguere lo sfondo.
- Se nel dataset non ci sono immagini dell'aquila con uno sfondo non blu, il lavoro di annotazione potrebbe essere inutile.

4.4 Procedura di Annotazione

4.4.1 Preparazione

- **Determinare cosa annotare**:
  - Assicurarsi di avere una conoscenza teorica completa delle teorie rilevanti per il problema
  - Es. Tecniche di inganno nei discorsi politici
- **Espressività vs scalabilità**:
  - Qual è il task a cui siamo realmente interessati?
  - Tale task è il più utile?

4.4.2 Selezione delle Risorse

- **Selezionare gli annotatori**:
  - Meglio avere accesso a esperti nel dominio -> solitamente costosi
  - Alcune aziende (MTurk) offrono opportunità di crowdsourcing per le annotazioni
  - Più economico ma forse meno impegnato, necessità di garantire la qualità
- **Selezionare un software/piattaforma di annotazione** (Inception, Anafora, MTurk)

4.4.3 Formalizzazione e Implementazione

- **Formalizzare le istruzioni per il task di annotazione** (replicabilità):
  - Chiare
  - Non troppo complicate
  - Non ambigue
  - Non basate sul fatto che l'utente capisca cosa intendiamo "tra le righe"
- **Configurare lo schema di annotazione e eseguire un'annotazione pilota** (interna ed esterna):
  - Permette di testare, rivedere e modificare le istruzioni
  - Crea etichette gold che possono essere utilizzate per verificare quantitativamente le prestazioni degli annotatori
- **Avviare la campagna di annotazioni**:
  - Avere più annotazioni per esempio quando possibile
  - Etichette gold assegnate per voto di maggioranza

4.5 Accordo tra Annotatori

Il razionale dell'accordo tra annotatori è che se più annotatori indipendenti pensano che y^* sia l'etichetta corretta per l'input x, allora probabilmente lo è.

4.5.1 Metriche di Accordo

- **Raw Agreement**: proporzione di etichette in accordo
  - Assume che tutte le annotazioni siano ugualmente difficili/importanti
  - Non tiene conto della probabilità di concordare per caso
- **Kappa di Cohen** per etichette discrete tiene conto del caso:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{4.1}$$

  dove:

  - P_o = proporzione di annotazioni per le quali A,B sono d'accordo
  - P_e = accordo atteso per caso

4.5.2 Interpretazione del Kappa di Cohen

- $-P_e/(1-P_e) \leq \kappa \leq 1$
- $\kappa < 0$ (disaccordo)
- $\kappa = 0$ (accordo per caso)
- $\kappa = 1$ (accordo perfetto)
- L'interpretazione di κ dipende dal dominio: in generale 0.8 o superiore è considerato un accordo molto buono

4.6 Preprocessing del Testo

4.6.1 Tokenizzazione

La tokenizzazione consiste nel convertire il testo da un insieme di simboli a una sequenza di parole/token:

- Dividere il testo in base a un insieme di caratteri (spazio bianco, punteggiatura,...)
- Sfide:
  - Parole composte: Prize-winning, isn't, pd-meno-elle
  - Abbreviazioni: U.S., Ph.D, New York
- La tokenizzazione è specifica per lingua:
  - Il tedesco scrive sostantivi composti senza spazi. Esempio: Computerlinguistik, 'linguistica computazionale'
  - L'italiano e lo spagnolo incorporano verbi e clitici, che sono un tipo speciale di pronomi. Esempio: comprarlo > comprare + lo, 'comprarlo'

4.6.2 Tokenizzazione a Sottorparole

Se il corpus di addestramento contiene le parole "foot" e "ball", ma non la parola "foot-ball" → quest'ultima è una parola sconosciuta. La tokenizzazione a sottoparole riduce la dimensione del vocabolario ed è diventata il metodo di tokenizzazione più comune per i grandi modelli linguistici.

Tokenizzatore BPE (Byte Pair Encoding)

L'algoritmo BPE viene solitamente eseguito all'interno delle parole, senza unire attraverso i confini delle parole. A tal fine, si utilizza un marcatore speciale di fine parola. L'algoritmo itera attraverso i seguenti passaggi:

1. Iniziare con un vocabolario composto da tutti i caratteri individuali
2. Scegliere i due simboli A, B che sono più frequentemente adiacenti
3. Aggiungere un nuovo simbolo unito AB al vocabolario
4. Sostituire ogni coppia adiacente A, B nel corpus con AB
5. Fermarsi quando il vocabolario raggiunge la dimensione k, un iperparametro

4.6.3 Normalizzazione

Dopo la tokenizzazione del testo, quali token dovrebbero essere uniti/separati?

- Per il nostro scopo, dobbiamo distinguere tra "great" e "Great"?
- E tra "apple" e "Apple"?

La normalizzazione del testo consiste in trasformazioni di stringhe che rimuovono distinzioni irrilevanti per le applicazioni a valle:

- Abbreviazioni e contrazioni: isn't -> is not
- Standardizzazione dei numeri: 1,000 -> 1000 -> !NUM
- Standardizzazione delle date: August 11, 2015 -> 11/08/2015 -> !DATE
- Allungamento espressivo, ad esempio, cooooool -> cool
- Variazioni ortografiche, ad esempio in testi storici "Nel mezzo del cammin", "amor 'ca nullo amato"

4.6.4 Pro e Contro della Normalizzazione

- La normalizzazione riduce la dimensione dello spazio delle caratteristiche, che può aiutare nella generalizzazione
- Rischio di unire distinzioni linguisticamente significative

Oltre alla normalizzazione, per ridurre lo spazio delle caratteristiche possiamo:

- Eliminare le parole meno frequenti
- Eliminare le stopwords (non sempre una buona idea, es. attribuzione di autore)

4.6.5 Stemming

Lo stemming si riferisce al processo di taglio di una parola con l'intenzione di rimuovere gli affissi.

Lo stemming è problematico dal punto di vista linguistico, poiché a volte produce parole che non sono nella lingua, o parole che hanno un significato diverso.

Esempi:

- arguing > argu, flies > fli
- playing > play, caring > car
- news > new

Molto più comunemente usato in IR che in NLP. Gli stemmer Porter e Snowball sono molto popolari (basati su regole). Per le lingue con poche risorse sono disponibili anche stemmer statistici.

4.6.6 Lemmatizzazione

- Lo stemming opera rimuovendo i suffissi di una parola
- Non sempre restituisce una parola propria
- Un lemma è la forma canonica di una parola, cioè la sua forma di dizionario
- Es. There are Geese -> goose

A differenza dello stemming, la lemmatizzazione dipende dalla corretta identificazione di:

- La parte del discorso intesa
- Il significato di una parola in una frase (basato sul contesto)

Lemmatizzazione e stemming si escludono a vicenda, e la prima richiede molte più risorse della seconda.

4.6.7 Altri Task di Preprocessing del Testo

- **Identificazione della lingua**: task di rilevamento della lingua di origine per il testo di input.
  - Preliminare al controllo ortografico, tokenizzazione, espansione di acronimi, ecc.
  - Diverse tecniche statistiche per questo task: frequenza di parole funzionali, modelli di linguaggio N-gram, misura di distanza basata sull'informazione mutua, ecc.
- **Controllo ortografico**: correzione di errori grammaticali nel testo.
  - Particolarmente utile per testi scritti rapidamente, come Twitter e recensioni di Amazon.

  - Uso di algoritmi di corrispondenza di stringhe approssimativa come la distanza di Levenshtein per trovare ortografie corrette.
  - Problema: una parola mal ortografata potrebbe comunque essere nella lingua.

- **Punteggiatura**: i segni di punteggiatura nel testo devono essere isolati e trattati come se fossero parole separate.
  - Critico per trovare i confini della frase e per identificare alcuni aspetti del significato (punti interrogativi, punti esclamativi, virgolette).
  - Molti simboli di punteggiatura sono utilizzati anche nelle abbreviazioni (punto), nei nomi di aziende (Yahoo!), nelle parole composte (-), ecc.

- **Caratteri speciali**: attenzione particolare per
  - Emoticon: :) ;) ecc., usare espressioni regolari
  - Emoji: ecc., usare librerie specializzate

4.7 Rappresentazione del Testo per il Machine Learning

Una volta completato il preprocessing, il task richiede l'uso del Machine Learning per essere risolto. La maggior parte (non tutti) degli algoritmi di Machine Learning presuppone che i dati di input siano in forma vettoriale.

- **Tipi di parole**: le parole distinte che appaiono in un documento
- **Token di parole**: le singole occorrenze di parole in un documento

Possiamo costruire un vettore dove ogni posizione corrisponde a un tipo di parola (scegliamo le parole che ci interessano, normalmente tutte quelle che appaiono nel nostro corpus di training).

4.7.1 Bag of Words (BoW)

Una rappresentazione tipica dei dati è l'utilizzo di un dizionario di parole e delle loro frequenze.

Es. "The quick brown fox is on the table"

The brown quick on fox dog is table the

I testi vengono confrontati contando il numero di corrispondenze tra le parole (prodotti scalari dei vettori sopra).

4.7.2 N-Grammi

Un n-gramma è una sequenza di n parole adiacenti in un testo. Gli n-grammi sono un'alternativa praticabile alla bag of words.

Es. "The quick brown fox is on the table"

The quick, quick brown, brown fox, ..., the table

I testi vengono confrontati contando il numero di n-grammi corrispondenti (prodotti scalari dei vettori).

Capitolo 5

Word Embeddings e Modelli di Linguaggio Neurali

5.1 Rappresentazioni Vettoriali del Testo

5.1.1 One-Hot Encoding

Nell'one-hot encoding:

- La rappresentazione per ogni parola è ortogonale alle altre
- Tutte le parole sono ugualmente dissimili tra loro
- Ma "dog" dovrebbe essere più vicino a "cat" che a "table"

5.1.2 Embeddings

Idealmente, vorremmo ottenere una rappresentazione vettoriale che:

- Permetta di determinare il grado di similarità tra le parole
- Questi vettori densi risultanti sono chiamati embeddings

5.2 Semantica Distribuzionale

La semantica distribuzionale si basa sull'ipotesi che parole che appaiono in contesti simili abbiano significati simili.

5.2.1 Ipotesi Distribuzionale

- Quanti più contesti due parole condividono, tanto più sono simili
- Possiamo costruire vettori basati sui contesti in cui le parole appaiono
- Possiamo quindi calcolare la similarità tra parole usando il prodotto scalare o la similarità del coseno

La similarità del coseno tra due vettori è calcolata come il coseno dell'angolo tra i vettori:

$$\cos(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} \tag{5.1}$$

- $\cos(\mathbf{v}, \mathbf{v}) = 1$
- $\cos(\mathbf{v}, \mathbf{w}) = 0$ se $\mathbf{v}$ e $\mathbf{w}$ sono ortogonali
- $\cos(\mathbf{v}, \mathbf{w}) = -1$ se $\mathbf{v}$ e $\mathbf{w}$ sono in direzione opposta

5.3 Modelli Basati sul Conteggio

5.3.1 Matrice Termine-Documento

- Parole nelle righe e documenti nelle colonne
- Ogni elemento (i, j) corrisponde alla frequenza della parola i nel documento j

5.3.2 Matrice Termine-Termine

- Ogni colonna si riferisce a una parola
- Per ogni parola nella riga i, contiamo, per ogni colonna j, il numero di volte in cui la parola della colonna appare all'interno di una finestra di dimensione L, a sinistra o a destra, in qualsiasi documento

5.3.3 TF-IDF

- **TF (term frequency)**: $\text{tf}_{t,d} = \text{count}(t, d)$
- **IDF (inverse document frequency)**: $\text{idf}_t = \log \frac{N}{\text{df}_t}$, dove N è il numero totale di documenti e df_t è il numero di documenti in cui appare t
- **TF-IDF**: $\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$

Usare conteggi grezzi è problematico per le stopwords (hanno troppo impatto).

5.3.4 PPMI per Matrici Termine-Termine

- **PMI (Pointwise Mutual Information)**: misura quanto spesso due eventi x e y si verificano insieme, rispetto a quanto ci aspetteremmo se fossero indipendenti

$$\text{PMI}(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \tag{5.2}$$

- **PPMI (Positive PMI)**: restringe i valori PMI solo a quelli positivi

5.3.5 Analisi Semantica Latente (LSA)

Rappresentare una parola con una matrice termine-termine genera vettori lunghi (dimensione del vocabolario). Consideriamo una matrice C dove $C_{i,j}$ rappresenta l'occorrenza del termine i nel contesto j.

- Può essere scritta come un prodotto di 3 matrici: $C = USV^T$
- $C = m \times n$, $U = m \times l$, $S = l \times l$, $V = l \times n$
- S è una matrice diagonale, i valori nella matrice diagonale sono i valori singolari

Obiettivo: approssimare C con una matrice più piccola dove la norma di Frobenius sia minimizzata.

La migliore approssimazione di C, cioè con il minimo errore rispetto alla norma di Frobenius, si ottiene prendendo la sottomatrice corrispondente ai primi l valori singolari della Singular Value Decomposition (SVD).

5.4 Word2vec

5.4.1 Modello Skip-Gram

- **Idea**: invece di contare quante volte ogni parola w si verifica vicino a c, addestrare un classificatore su un task di predizione binaria: "È probabile che w compaia vicino a c?"
- Il classificatore dovrà imparare una buona rappresentazione compressa per i contesti di una parola
- Data una parola t, prevedere se una parola c è nel contesto (problema di classificazione)

Esempio: "I am happy because I am learning" genera coppie termine-contesto (t, c):

- Esempi positivi: $(+|happy, am)$, $(+|happy, because)$, $(+|happy, I)$
- Esempi negativi (campionati casualmente): $(-|happy, window)$, $(-|happy, in)$, $(-|happy, eat)$

Il modello stima la probabilità che la parola sia nel contesto:

$$P(context|word) = \frac{1}{1 + e^{-s(w,c)}} \tag{5.3}$$

dove $s(w, c)$ è la similarità tra le due parole.

5.4.2 Negative Sampling

Per addestrare il modello, si massimizza la probabilità delle parole di contesto e si minimizza la probabilità delle parole non di contesto.

Sketch dell'algoritmo:

1. Trattare la parola target e una parola di contesto vicina come esempi positivi

2. Campionare casualmente altre parole nel lessico per ottenere esempi negativi
3. Usare la regressione logistica per addestrare un classificatore a distinguere questi due casi
4. Usare i pesi appresi come embeddings

5.4.3 Dettagli del Modello Skip-Gram

- Ogni parola è rappresentata da un vettore d-dimensionale e ha 2 rappresentazioni: come w e come c (abbiamo $d \times 2|V|$ parametri nel nostro modello)
- Dopo l'apprendimento possiamo sommare i vettori w_i e c_i per la parola i o semplicemente usare w_i
- Le parole di contesto sono campionate casualmente favorendo le parole più vicine (la dimensione della finestra L è un altro iperparametro del modello)

5.4.4 Collegamento tra Skip-gram e Fattorizzazione della Matrice

Il modello Skip-Gram con negative sampling è collegato alla fattorizzazione della matrice di C:

$$C_{i,j} = \text{PMI}(i, j) - \log k \tag{5.4}$$

dove:

- $\text{PMI}(i, j)$ è il PMI della parola i nel contesto j
- k è il numero di esempi negativi campionati

Possiamo usare PPMI per evitare valori $-\infty$ quando $\text{count}(i, j) = 0$.

5.5 Proprietà degli Word Embeddings

5.5.1 Proprietà Semantiche

- Gli embeddings di parole possono catturare significati relazionali e risolvere problemi di analogia:
  - "Apple is to tree as grape is to ..."
  - Si può utilizzare il modello del parallelogramma: $v_{grape} - v_{apple} + v_{tree} \approx v_{vine}$
- Gli embeddings possono essere visualizzati:
  - Elencando le parole in V con la più alta similarità del coseno con la parola
  - Proiettando le d dimensioni di un word embedding in 2 dimensioni (t-distributed stochastic neighbor embedding, t-SNE)

5.5.2 Word Embeddings Diacronici

È possibile confrontare embeddings addestrati su dati diversi (ma correlati), ad esempio embeddings addestrati su dati dello stesso dominio in anni diversi.

- W_t e W_{t+1} sono embeddings addestrati su periodi di tempo diversi
- Non possiamo confrontare direttamente vettori da W_t e W_{t+1} poiché i dati su cui sono stati addestrati non sono gli stessi
- Dobbiamo allineare i due embeddings prima di confrontarli
- Q è una matrice di trasformazione (rotazione e allineamento)

Questo è chiamato problema di Procrustes ortogonale e la sua soluzione è UV^T dove U, V sono il risultato della decomposizione SVD di $W_{t+1}(W_t)^T$.

5.5.3 Embeddings per Parole Polisemiche

Cosa succede agli embeddings di parole con significati multipli?

- "Tie": 1) un articolo di abbigliamento; 2) un pareggio; 3) un atto fisico
- Arora et al. mostrano che:

$$v(tie) \approx a_1 v(tie_1) + a_2 v(tie_2) + a_3 v(tie_3) \tag{5.5}$$

  dove a_i è proporzionale alla frequenza relativa del significato i

- $v(tie_1)$ è un atomo di discorso (topic) che può essere appreso (qui rappresentato da un insieme di parole rappresentative)

5.5.4 FastText

- Le parole, specialmente nelle lingue morfologicamente ricche, hanno una struttura interna
- FastText calcola rappresentazioni per n-grammi di caratteri (usa <> per denotare inizio/fine della parola)
- Esempio: "where" = <wh, whe, her, ere, re>
- Rappresenta una parola come la somma degli embeddings degli n-grammi di caratteri che la compongono
- Il miglior n dipende dalla lingua e dal task
- Buoni risultati su task di analogia sintattica e su parole rare

5.5.5 GloVe

- GloVe combina modelli come skip-gram che usano informazioni locali sulla finestra di contesto con informazioni globali
- Con un certo grado di semplificazione, gli embeddings sono inizializzati in base alle statistiche di co-occorrenza globale tra parole (calcolate dalla matrice termine-termine)

5.6 Modello Continuous Bag Of Words (CBOW)

- L'apprendimento della rappresentazione delle parole può essere inquadrato come un task di apprendimento
- Prendere un enorme corpus di testo
- Scorrerlo con una finestra scorrevole, spostandosi di una parola alla volta
- Ad ogni passo, c'è una parola centrale e parole di contesto (altre parole in questa finestra)
- Prevedere le probabilità della parola centrale dati i contesti
- Aggiustare i vettori per aumentare queste probabilità
- La rappresentazione della parola sarà un sottoprodotto del task di apprendimento
- Le parole di contesto sono codifiche one-hot mediate
- N è la dimensione degli embeddings
- Si noti che l'ordine delle parole non è modellato (da qui il nome)

5.6.1 Dettagli del Modello CBOW

- **Funzione di Perdita**: Cross Entropy Loss
- Gli embeddings delle parole possono essere estratti da:
  1. Ogni colonna di W_1 è l'embedding della parola corrispondente
  2. Ogni riga di W_2 è l'embedding della parola corrispondente
  3. La media di W_1 e W_2 trasposta

5.7 Modelli di Linguaggio Neurali

5.7.1 Introduzione ai Modelli di Linguaggio

I modelli di linguaggio neurali (NLM) hanno ampiamente sostituito i modelli di linguaggio N-gram.

Vantaggi principali dei NLM:

- Possono incorporare informazioni contestuali arbitrariamente distanti, pur rimanendo trattabili dal punto di vista computazionale e statistico
- Possono generalizzare meglio su contesti di parole simili e sono più accurati nella previsione delle parole

Svantaggi:

- Rispetto ai modelli di linguaggio N-gram, i NLM sono molto più complessi, più lenti, richiedono più tempo per l'addestramento e sono meno interpretabili
- Per molti compiti (specialmente quelli più piccoli) i modelli di linguaggio N-gram sono ancora lo strumento giusto

5.7.2 Architettura Generale per NLM

1. Ottenere una rappresentazione vettoriale per il contesto precedente
2. Generare una distribuzione di probabilità per il token successivo

Il primo punto dipende dall'architettura NN, il secondo è indipendente dal modello. Le scelte più naturali per l'architettura NN sono le reti neurali ricorrenti (RNN), ma sono state sfruttate anche le reti neurali feedforward (FNN) e le reti neurali convoluzionali (CNN).

5.7.3 NLM Feedforward

- Come il modello di linguaggio N-gram, il NLM feedforward usa la seguente approssimazione:
$$P(w_t|w_{1:t-1}) \approx P(w_t|w_{t-N+1:t-1}) \tag{5.6}$$
e una finestra mobile che può vedere $N-1$ parole nel passato
- Rappresentiamo ogni parola di input w_t come un vettore one-hot x_t di dimensione $|V|$

Fasi del modello:

1. Convertire i vettori one-hot per le parole nella finestra di $N-1$ in word embeddings di dimensione d
2. Concatenare gli $N-1$ embeddings
3. Passare attraverso un hidden layer
4. Calcolare gli score (logits) per ogni parola nel vocabolario
5. Applicare softmax per ottenere probabilità

5.7.4 Addestramento del NLM Feedforward

I parametri del modello sono $\theta = \{E, W, U\}$.

La distribuzione vera y_t per la parola in posizione t è un vettore one-hot di dimensione $|V|$ con:

- $y_t[ind(w_t)] = 1$
- $y_t[k] = 0$ altrove

Usiamo la loss di cross-entropy per addestrare il modello:

$$L_{CE}(\hat{y}_t, y_t) = -\sum_{k=1}^{|V|} y_t[k] \log \hat{y}_t[k] = -\log \hat{y}_t[ind(w_t)] \tag{5.7}$$

5.7.5 Reti Neurali Ricorrenti (RNN)

- Srotolare la rete raggiunge una profondità orizzontale
- Ogni step temporale usa lo stesso peso condiviso
- L'output può essere influenzato dall'intera sequenza vista finora
- Lo stato nascosto può essere un riassunto della sequenza di input

NLM Ricorrente

- I modelli di linguaggio RNN elaborano l'input una parola alla volta, prevedendo la parola successiva da:
  - La parola corrente
  - Lo stato nascosto precedente
- Le RNN possono modellare la distribuzione di probabilità $P(w_t|w_{1:t-1})$ senza l'approssimazione della finestra $N-1$ del NLM feedforward
- Lo stato nascosto del modello RNN può in principio rappresentare informazioni su tutte le parole precedenti

Equazioni del modello:

$$e_t = Ex_t \tag{5.8}$$
$$h_t = g(Uh_{t-1} + We_t) \tag{5.9}$$
$$\hat{y}_t = \text{softmax}(Vh_t) \tag{5.10}$$

dove:

- $x_t : |V| \times 1$ è una rappresentazione one-hot della parola w_t
- $E : d \times |V|$ è una matrice apprendibile con i word embeddings
- $U, W : d \times d$ sono matrici apprendibili
- $h_t : d \times 1$ è il vettore nascosto al passo t
- $V : |V| \times d$ è una matrice apprendibile
- $\hat{y}_t : |V| \times 1$ è una distribuzione di probabilità

Addestramento del NLM Ricorrente

- Il numero di parametri del modello è $O(|V|)$, poiché d è una costante
- Ad ogni passo t durante l'addestramento:
  - La previsione si basa sulla sequenza corretta di token $w_{1:t}$
  - Ignoriamo ciò che il modello ha previsto ai passi precedenti
- L'idea che diamo sempre al modello la sequenza di storia corretta per prevedere la parola successiva è chiamata teacher forcing
- Il teacher forcing ha alcuni svantaggi: il modello non è mai esposto agli errori di previsione; quindi al momento dell'inferenza il modello non è in grado di riprendersi dagli errori

Long Short-Term Memory (LSTM)

- Le RNN attuali tendono a utilizzare informazioni di contesto locali e hanno difficoltà a sfruttare informazioni distanti
  - Es. "The flights the airline was canceling were _____" [full]
- La rete Long Short-Term Memory (LSTM) ha l'obiettivo di decidere quali informazioni mantenere per il futuro
- LSTM introduce gate per questo scopo: NN feedforward seguita da un'attivazione sigmoide, seguita da una moltiplicazione elemento per elemento
  - Gate di dimenticanza: elimina informazioni dal contesto che non sono più necessarie
  - Gate di aggiunta: sceglie le informazioni da aggiungere al contesto
  - Gate di output: decide quali informazioni sono necessarie per lo stato nascosto corrente

5.7.6 Generazione di Testo

Il testo generato dal campionamento del nostro NLM dovrebbe essere:

- Coerente: il testo deve avere senso
- Diversificato: il modello deve essere in grado di produrre campioni molto diversi

C'è un trade-off tra i due. Strategie possibili:

- Scegliere sempre la più probabile (greedy)
- Campionamento casuale: la probabilità di ogni parola di essere selezionata è la sua probabilità calcolata dal modello di linguaggio
- Le parole rare sono così tante che, sommando la loro probabilità, otteniamo una probabilità non insignificante

Campionamento Top-k

- Selezionare le k parole più probabili, rinormalizzare la probabilità e poi applicare il campionamento casuale
- Funziona se le top k parole includono la maggior parte della massa di probabilità
- Se la distribuzione è quasi piatta, le top-k rappresentano solo una piccola parte della massa di probabilità

Campionamento Top-p

- Mantenere il top p percento della massa di probabilità
- Selezionare il più piccolo insieme di parole $V^{(p)}$ tale che:

$$\sum_{w \in V^{(p)}} P(w|w_{<t}) \geq p \tag{5.11}$$

- L'obiettivo è avere una misura in grado di catturare le parole più probabili tale che la loro massa di probabilità sia massimizzata (indipendentemente dalla distribuzione P)

Temperatura

- La temperatura rimodella la distribuzione di probabilità:

$$y = \text{softmax}(V/\tau) = \frac{\exp\left(\frac{V_i h_t}{\tau}\right)}{\sum_j \exp\left(\frac{V_j h_t}{\tau}\right)} \tag{5.12}$$

  dove V_i denota la i-esima riga di V

- $\tau = 1$ lascia y invariato, $\tau > 1$ la appiattisce, $\tau < 1$ la rende più asimmetrica

Capitolo 6

Architetture Neurali per NLP

6.1 LSTM

Le reti neurali ricorrenti (RNN) tendono a utilizzare informazioni di contesto locali e hanno difficoltà a sfruttare informazioni distanti. Long Short-Term Memory (LSTM) è un tipo di RNN progettato per affrontare questo problema.

6.1.1 Struttura e Funzionamento

LSTM introduce gate per determinare quali informazioni mantenere:

- **Gate di dimenticanza**: elimina informazioni dal contesto che non sono più necessarie
- **Gate di aggiunta**: sceglie le informazioni da aggiungere al contesto
- **Gate di output**: decide quali informazioni sono necessarie per lo stato nascosto corrente

LSTM calcola per ogni input (time step t):

- Uno stato nascosto h_t, che potrebbe essere l'input per il layer successivo o utilizzato per calcolare direttamente un output
- Un vettore di contesto c_t che riassume le informazioni passate rilevanti

6.1.2 Equazioni dei Gate

Sia $a = [a_1, a_2, \ldots, a_n] \in \mathbb{R}^n$, $b = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^n$, allora $a \odot b = [a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_n \cdot b_n] \in \mathbb{R}^n$ (moltiplicazione elemento per elemento).

Gate di dimenticanza:

$$f_t = \sigma(U_f h_{t-1} + W_f x_t) \tag{6.1}$$

$$k_t = c_{t-1} \odot f_t \tag{6.2}$$

k_t è il contesto precedente con alcune informazioni rimosse.

Gate di aggiunta:

$$g_t = \tanh(U_g h_{t-1} + W_g x_t) \tag{6.3}$$
$$i_t = \sigma(U_i h_{t-1} + W_i x_t) \tag{6.4}$$
$$j_t = g_t \odot i_t \tag{6.5}$$

j_t è l'informazione dallo stato nascosto precedente che verrà aggiunta al contesto. Infine, il nuovo contesto è $c_t = j_t + k_t$.
Gate di output:

$$o_t = \sigma(U_o h_{t-1} + W_o x_t) \tag{6.6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6.7}$$

6.2 Modelli Encoder-Decoder

I modelli encoder-decoder sono utilizzati per compiti di sequenza a sequenza, dove l'input è una sequenza e l'output è un'altra sequenza.

6.2.1 Tipi di Task

- **Classificazione di testo**: dato un testo (input di lunghezza variabile), determinare un singolo output
  - Esempio: Sentiment di una recensione
- **Etichettatura di sequenza**: dato un testo di n token, determinare un output per ciascun token (output della stessa lunghezza dell'input)
  - Esempio: Part of Speech Tagging
- **Sequenza a sequenza**: per un input di n token, l'output è di m elementi (di solito con $m \neq n$)
  - Esempio: Machine Translation, Summarization, ecc.

6.2.2 Struttura Encoder-Decoder

- **Encoder**: $x_1, \ldots, x_n \to h_1, \ldots, h_n$
- **Contesto**: $h_1, \ldots, h_n \to c$ una rappresentazione compressa di $h_1, \ldots, h_n$ in un vettore di dimensione fissa
- **Decoder**: $c \to h_1, \ldots, h_m$ (m potrebbe essere diverso da n)

LSTM, reti convoluzionali e transformer possono essere tutti impiegati come encoder/decoder.

6.2.3 Traduzione con Encoder-Decoder

Seguendo la struttura encoder-decoder, prima calcoliamo c basato sulla frase di input, poi iniziamo a generare y_t basato su $y_{<t}$ e c.

- Per evitare di "inquinare" c ad ogni time step, non lo aggiorniamo durante la generazione (ogni y_t è condizionato anche sullo stesso vettore c)

$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c) \tag{6.8}$$

- Formalmente:

$$c = h_n^e \tag{6.9}$$
$$h_0^d = c \tag{6.10}$$
$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, c) \tag{6.11}$$
$$\hat{y}_t = \text{softmax}(h_t^d) \tag{6.12}$$

6.2.4 Addestramento Encoder-Decoder

- Un esempio di input è una coppia di frasi separate da un token specifico (notare che non abbiamo un singolo contesto qui)
- L'addestramento viene eseguito come per qualsiasi NN (usiamo il teacher forcing)
- Durante l'inferenza, dopo un errore il decoder potrebbe deviare sempre più dalla traduzione "corretta"

6.3 Attention

- L'attention è un meccanismo sviluppato per superare i problemi delle RNN
- È un blocco di una NN
- È il componente principale di una nuova classe di architetture di reti neurali, i Transformer
- L'attention imita il recupero di un valore v_i per una query q basata su una chiave (indirizzo) k_i in un database (memoria)

$$\text{attention}(q, k, v) = \sum_i \text{similarity}(q, k_i) v_i \tag{6.13}$$

6.3.1 Motivazione per l'Attention

- L'architettura dell'encoder RNN potrebbe non preservare informazioni su tutte le parole, ma le ultime parole tendono ad avere un impatto maggiore sul vettore di riepilogo
- Potrebbe trascurare parole importanti per il nostro task
- Più in generale, potrebbe perdere informazioni importanti

6.3.2 Come Funziona l'Attention

1. Calcolare la similarità tra q e ogni parola (o stato nascosto h_i)
2. Calcolare il softmax dei valori di similarità e ottenere score α
3. Gli stati nascosti vengono composti insieme con i loro score
4. Abbiamo ottenuto un vettore alternativo v che dipende dalla query q (e dalla funzione di similarità)
5. La buona notizia è che il vettore q può essere appreso come qualsiasi altro parametro della rete

6.3.3 Attention in Encoder-Decoder

- Invece di avere un singolo vettore di contesto c che riassume le informazioni dell'encoder come prima, c dipende anche dallo stato precedente del decoder (h^d_{i-1})

$$c_i = f(h^e_1, \ldots, h^e_n, h^d_{i-1}) \tag{6.14}$$

- Nella fase di decodifica

$$h^d_i = g(\hat{y}_{i-1}, h^d_{i-1}, c_i) \tag{6.15}$$

- Lo stato nascosto della fase di decodifica dipende da un contesto diverso per ogni token
- Permette a ogni $\hat{y}_i$ di concentrarsi su ogni stato dell'encoder (diverse parti della frase di input) tramite il c_i personalizzato
- L'attention presuppone che tutti gli stati nascosti siano esplicitamente disponibili
- Relazione many-to-many tra output e stati nascosti di input
- È possibile sfruttare relazioni a lungo termine

6.3.4 Funzioni di Attention

$$\text{attention}(q, k, v) = \sum_i \text{similarity}(q, k_i) v_i \tag{6.16}$$

- $\text{similarity}(q, k_i) = q^T k_i$
- $\text{similarity}(q, k_i) = q^T W k_i$ (pesi W da apprendere, utili quando q, k non sono nello stesso spazio, ad es. domanda e risposta)
- $\text{similarity}(q, k_i) = w_q^T \tanh(W[q; k_i])$
- In alternativa, si potrebbe utilizzare una funzione kernel

6.3.5 Varianti di Attention

- L'attention potrebbe essere definita gerarchicamente, ad esempio per codificare un documento date le sue frasi
- Hard Attention (si concentra su un elemento, possibilmente favorendo l'interpretabilità)
- In alcuni casi è disponibile l'informazione gold (in MT se sappiamo che in una lingua c'è allineamento per certe parole, allora quando traduciamo nella direzione opposta ci aspettiamo lo stesso allineamento)
- Forzare l'attention ad apprendere alcune proprietà motivate linguisticamente

Capitolo 7

Applicazioni dell'NLP

7.1 Sentiment Analysis

7.1.1 Definizione e Obiettivi

L'identificazione del sentiment di un testo può essere utile per:

- Determinare l'atteggiamento delle persone verso un marchio, una legge, un prodotto, una persona...
- Come condurre "sondaggi" - questo è molto popolare nella computer social science
- Analizzare l'umore di una persona: cercare i primi segni di depressione dai social media, ecc...
- Il sentiment può essere un componente per applicazioni a valle
  - Es. riassumere riunioni o trovare parti di una conversazione in cui le persone sono particolarmente eccitate o coinvolte
  - Rilevare la frustrazione in un'interazione tra un essere umano e un agente conversazionale

7.1.2 Teorie delle Emozioni

- Teoria delle 6 emozioni (Ekman), presumibilmente presenti in tutte le culture:
  - Sorpresa, felicità, rabbia, paura, disgusto, tristezza
- Teoria di Plutchik ha 8 emozioni in 4 coppie opposte:
  - Gioia vs Tristezza, Fiducia vs Disgusto, Paura vs Rabbia, Sorpresa vs Anticipazione
- Altri lavori identificano 3 assi:
  - Valenza: la piacevolezza dello stimolo
  - Arousal: l'intensità dell'emozione provocata dallo stimolo
  - Dominanza: il grado di controllo esercitato dallo stimolo

7.1.3 Lessici di Sentiment

- **Lessici binari** che rappresentano il sentiment (valenza) delle parole:
  - General Inquirer (1967): 1915 parole positive, 2291 parole negative
  - MPQA Subjectivity lexicon (2005): 2718 parole positive, 4912 parole negative, e una lista di parole e frasi soggettive
- **Lessici che considerano ulteriori aspetti**:
  - The NRC Word-Emotion Association Lexicon, EmoLex (2013), dà etichette binarie a 14k parole secondo lo schema di Plutchik
  - LIWC, Linguistic Inquiry and Word Count, è un set ampiamente utilizzato di 73 lessici contenenti oltre 2300 parole (2007)
- **Lessici non binari**:
  - The NRC Valence, Arousal, and Dominance - VAD - (2018) assegna punteggi di valenza, arousal e dominanza a 20.000 parole

7.1.4 Sentiment Analysis Basata su Lessico

- Quando si tratta di documenti (lunghi), combinare una rappresentazione bag-of-words con il sentiment delle parole potrebbe essere sufficiente
  - Ci aspetteremmo che le parole associate al suo vero sentiment sovrastino le altre
- Per frasi o documenti brevi questo è meno efficace:
  - Negazione → "That's not bad for the first day"
  - Linguaggio figurato, satira → "This film should be brilliant. The actors are first grade. However, the film is a failure"
- Soluzione parziale:
  - Usare n-grammi ("not bad") e calcolare il sentiment per ogni n-gramma
  - Usare l'albero sintattico per stabilire relazioni tra la negazione e l'oggetto negato

7.1.5 Apprendimento di Lessici

- I lessici potrebbero dover essere adattati al dominio, alla cultura o al contesto corrente
  - "Small" era (è) positivo per gli smartphone, negativo per gli appartamenti
- Da qui la necessità di imparare/fine-tunarli dai dati
- **Asse Semantico**:
  - Definire vettori positivi e negativi in uno spazio di embedding di parole

  - Definire l'asse tra tali vettori (il vettore che li collega)
  - Proiettare le parole nell'asse e determinare se il positivo o il negativo è più vicino

7.1.6 Label Propagation

- Definire un grafo: un nodo per parola, un arco per ciascuna delle k parole più simili (pesare gli archi in base alla stessa funzione di similarità)
- Definire parole seed
- Propagare le polarità sul grafo dalle parole seed
- Eseguire random walk partendo da ogni nodo e muovendosi verso un nodo vicino con probabilità proporzionale al peso dell'arco
- Valutare i nodi nel grafo: contare la percentuale di volte in cui un nodo è stato raggiunto da qualsiasi camminata che parte da un nodo etichettato positivamente rispetto al numero totale di volte in cui è stato visitato
- I valori di confidenza per i punteggi possono essere ottenuti ripetendo le camminate da diversi set seed

7.1.7 Sentiment Treebank

- Dato un albero di analisi sintattica di una frase, ogni nodo è annotato per il Sentiment (inclusa l'intera frase)
- Il sentiment di un nodo può essere studiato in termini del sentiment dei nodi figli
- Gli autori propongono una rete neurale ricorsiva, ma sono possibili altri approcci

7.2 Question Answering

7.2.1 Definizione e Tipi

- Il Question Answering è il task di fornire automaticamente una risposta per una domanda posta in linguaggio naturale
- Varie istanziazioni, ad esempio riguardo al tipo di domanda:
  - Factoid: nomi di entità e frasi brevi
  - Domande "perché" (Why did Ahab continue to pursue the white whale?)
  - Domande "come" (How did Queequeg die?)
  - Richieste di riassunti (What was Ishmael's attitude towards organized religion?)
  - Domande su dominio chiuso/aperto (prenotazione aerea/Wikipedia)

7.2.2 Sottotipi di QA

- **Fonte/Contesto**
  - Insiemi di documenti (corpus)
  - Un singolo documento
  - Knowledge Base
  - Tipi di dati non linguistici (GPS, immagini, sensori, ...)
- **Tipo di Risposta**
  - Un singolo fatto
  - Una spiegazione
  - Un documento
  - Una frase o paragrafo estratto dal testo
  - Un'immagine o altro tipo di oggetto
  - Un'altra domanda

7.2.3 Reading Comprehension

- QA su un passaggio di testo
- "Una macchina comprende un passaggio di testo se, per qualsiasi domanda riguardante quel testo a cui la maggioranza dei madrelingua può rispondere correttamente, quella macchina può fornire una stringa che quei madrelingua concorderebbero sia risponde a quella domanda, sia non contiene informazioni irrilevanti per quella domanda".

7.2.4 Corpora per Reading Comprehension

- **CNN/DailyMail**
  - Oltre 1 milione di coppie QA di tipo cloze con articoli di CNN e Mail online come contesto
- **CBT**
  - 700k coppie QA con libri per bambini come contesto. Scegliere uno tra 10 candidati
- **SQuAD**
  - 100k coppie QA manuali con 500 articoli di Wikipedia come contesto. La risposta è uno span

7.2.5 Approcci per il Reading Comprehension

- **Sliding Window**: prevede una risposta basata su semplici informazioni lessicali in una finestra scorrevole
  - Massimizzare la similarità bag-of-word tra la risposta e la finestra scorrevole nel passaggio dato
- **Regressione Logistica** (baseline SQuAD)
  - Caratteristiche dai candidati tra cui lunghezze, frequenze di bigrammi, frequenze di parole, tag POS dello span, caratteristiche lessicali, caratteristiche del percorso dell'albero di dipendenza, ecc.
  - Prevede se uno span di testo è la risposta finale in base a tutte quelle informazioni

7.2.6 Approcci Neurali

- **FastQA**: caratteristiche di base indicano se un token in un passaggio appare nella domanda
  - Le caratteristiche sono pesate rispetto alla similarità del contesto e della query
- **QANet**
  - Non un'architettura RNN, basata su self-attention
  - Viene utilizzata un'ulteriore attenzione domanda-contesto per calcolare la rilevanza dei due
- **Attention over Attention**
  - Quali parole del documento corrispondono meglio alle parole più importanti nella query?

7.2.7 Ragionamento Multi-step

- Potrebbe diventare chiaro che sono necessarie più informazioni post-facto
  - John è andato nel corridoio
  - John ha messo giù il pallone
  - Q: Dov'è il pallone?
- Passo 1: Attenzione a "pallone"
- Passo 2: Attenzione a "John"
- Dataset: HotpotQA (2018)
- Una formulazione generale di modelli che accedono alla memoria esterna attraverso l'attention e un'istanziazione specifica per QA a livello di documento
- Idea: calcolare l'attention anche sulla frase selezionata nel passaggio precedente

- Quando fermare il ragionamento?
  - Dopo un numero predefinito di passaggi
  - Quando prestiamo attenzione a un simbolo "stop reasoning"
  - Avere un predittore esplicito "stop reasoning"

7.2.8 Language Models come Knowledge Bases

I modelli di linguaggio possono essere utilizzati come basi di conoscenza per rispondere a domande fattuali:

- Ad esempio, "Dove è nato Dante?"
- Possiamo far leggere al modello di linguaggio il testo di input
- Formulare un template di risposta

7.2.9 Valutazione QA

Per valutare i sistemi di Question Answering, si utilizzano metriche come:

- Precisione: percentuale di risposte corrette
- F1: media armonica tra precisione e recall, particolarmente utile quando la risposta può essere parzialmente corretta
- BLEU/ROUGE: per risposte più lunghe o generate

7.3 Sequence Labelling

7.3.1 Definizione e Tipi

Il sequence labelling consiste nell'assegnare etichette discrete a elementi discreti in una sequenza:

- Part-of-Speech Tagging
- Named Entity Recognition
- Aspect-based Sentiment Analysis

Due tipi di approcci:

- **Ricerca locale**
  - Sequenza di problemi di classificazione: prevedere l'etichetta per ogni posizione nella sequenza di input
- **Ricerca globale**
  - Problema di ottimizzazione combinatoria sull'intero insieme di sequenze candidate

7.3.2 Part-of-Speech Tagging

- Il part of speech è una categoria di parole che svolgono ruoli simili all'interno della struttura sintattica di una frase
- Tutti i parlanti condividono le parole nelle classi POS chiuse, potrebbero avere parole diverse per quelle aperte

Classi Aperte

- **Sostantivo**: potrebbe verificarsi con determinanti (una capra), prendere possessivi (le entrate annuali di IBM)
  - Nomi propri, nomi comuni (nomi numerabili e di massa)
- **Verbi**: si riferiscono ad azioni e processi
- **Aggettivi**: termini che descrivono proprietà e qualità
- **Avverbi**: modificano verbi o altri avverbi

Classi Chiuse

- **Preposizioni** (prima di sintagmi nominali): on, under, over, near, by, at, from
- **Particelle** (insieme a un verbo): up, down, on, off, in, out, at, by
- **Determinanti** (spesso all'inizio di una frase): a, an, the
- **Congiunzioni**: and, but, or, as, if, when
- **Pronomi**: she, who, I, others
- **Verbi ausiliari**: can, may, should, are
- **Numerali**: one, two, three, first, second, third

7.3.3 Part-of-Speech Tagging come Task

- Il processo di assegnare un marcatore di parte del discorso o di classe lessicale a ciascuna parola in un corpus
- Task di etichettatura di sequenza: assegniamo, a ciascuna parola x_i in una sequenza di parole di input, un'etichetta y_i
- Task di disambiguazione: i termini potrebbero avere più di un POS. Ad esempio, la parola "back":
  - earnings growth took a back/JJ seat (aggettivo)
  - a small building in the back/NN (sostantivo)
  - Dave began to back/VB toward the door (verbo base)
  - I was twenty-one back/RB then (avverbio)

Quanto è comune l'ambiguità dei tag?

- La maggior parte dei tipi di parole (85-86%) non sono ambigui
- Le parole ambigue (14-15%) sono comuni: il 55-67% dei token delle parole in un testo corrente sono ambigui

7.3.4 Approcci al POS Tagging

Approccio basato su regole

- Iniziare con un dizionario
- Assegnare tutti i possibili tag alle parole dal dizionario
- Scrivere regole a mano per rimuovere selettivamente i tag
- Lasciare il tag corretto per ogni parola

Classificatore a finestra scorrevole

- Ogni token viene classificato in modo indipendente (ad esempio da un percettrone), ma le caratteristiche sono derivate dai token circostanti
- Caratteristiche tipiche:
  - Forma della parola (lowercase) del token corrente
  - Forme delle parole dei token precedenti, token successivi
  - Capitalizzazione del token corrente (maiuscolo, minuscolo, N/A)
  - Tipo del token corrente (cifre, lettere, simboli)
  - Vari prefissi e suffissi del token corrente
  - Se il token corrente è con trattino
  - Se il token è il primo o l'ultimo nella frase

7.3.5 Algoritmo di Viterbi

- Usando il classificatore precedente non possiamo produrre un'etichettatura "globale"
- Potremmo formulare il problema come:

$$\hat{y} = \arg\max_{y \in Y} \text{score}(x, y) \tag{7.1}$$

- Tuttavia, se consideriamo una frase lunga e il numero di tag POS, la dimensione dello spazio Y cresce esponenzialmente
- Se la funzione di punteggio è decomponibile

$$\text{score}(x, y) = \sum_{i=1}^{n} \text{score}(y_i, y_{i-1}, x) \tag{7.2}$$

- Possiamo trovare algoritmi più veloci
- Dato una funzione di punteggio decomponibile, Viterbi restituisce il tag con il punteggio più alto per il testo
- Es. Frase di input: they can fish
- Tag POS: N, V
- Numero di possibili etichettature (numero di percorsi sul grafico): 2×2×2=8
- Assumiamo una funzione di punteggio definita sul tag precedente e corrente e sulle caratteristiche basate sulla parola della frase

7.3.6 POS con il Percettrone

- Come apprendere i parametri dell'algoritmo di apprendimento? Collins combina Viterbi e il Percettrone (approccio esteso in [2])
  1. Prevedere la migliore sequenza di tag z usando Viterbi con le impostazioni correnti
  2. Per ogni vettore di caratteristiche collegato a un tag erroneamente previsto (w.r.t. y)
     (a) Aumentare il punteggio delle caratteristiche corrette, diminuire il punteggio delle caratteristiche errate
- Caratteristiche: (tag token precedente/corrente, parola/tag corrente)
- Es. z = the/D man/N saw/N the/D dog/N; y = the/D man/N saw/V the/D dog/N
- $w_2 = w_1 + a_{N,V} + a_{saw,V} - a_{N,N} - a_{saw,N}$

7.4 Named Entity Recognition

7.4.1 Definizione e Categorie

- Il Named Entity Recognition (NER) comporta l'identificazione di nomi propri nei testi e la classificazione in un insieme predefinito di categorie di interesse
- Tre categorie universalmente accettate:
  - persona, luogo e organizzazione
- Altri task comuni:
  - Riconoscimento di data/ora
  - Espressioni, misure (percentuale, denaro, peso, ecc.),
  - Indirizzi email, ecc.
  - Altre entità specifiche del dominio: nomi di farmaci, condizioni mediche, nomi di navi

7.4.2 Aree Grigie nelle Etichette NER

Le definizioni delle categorie sono intuitivamente chiare, ma ci sono molte aree grigie:

- Organizzazione vs. Luogo: "England won the World Cup" vs. "The World Cup took place in England"
- Azienda vs. Artefatto: "shares in MTV" vs. "watching MTV"
- Luogo vs. Organizzazione: "she met him at Heathrow" vs. "the Heathrow authorities"

7.4.3 NER come Task di Etichettatura di Sequenza

Il formato BIO:

- Begin, Inside, Outside tag per ogni token
- B: inizio di un'entità
- I: all'interno di un token
- O: il token non ha Named Entity
- Es. The (O) United (B_Country) States (I_Country) of (I_Country) America (I_Country) played (O) is (O) an (O) English (O) speaking (O) country (O)
- La notazione BIO può essere estesa con ulteriori simboli (End of entity)

7.4.4 Approcci al NER

Può essere diviso in due sottoattività:

1. Trovare i confini di un'entità
2. Determinare il tipo di entità

Approcci:

- Gli stessi del POS tagging (es. window slide)
- Reti Neurali Ricorrenti

Reti Neurali per Sequenze

- 3 layer di unità:
  - Layer di input
  - Layer nascosto: mantiene informazioni temporali
  - Layer di output: prevede il prossimo time step
- $x = [x_1, x_2, \ldots, x_T]$
- $h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$
- $y_t = \sigma(W_{yh}h_t + b_y)$

Caratteristiche per il NER

- Caratteristiche locali come per POS
- Gazetteer (dizionari con identità note)
- Primo carattere del token maiuscolo; tutto il token in lettere maiuscole, caso misto
- Tag POS
- Stem o lemma della parola
- Punteggiatura (ha un periodo interno, I.B.M.), apostrofo (O'Brien), &
- Frequenza della parola

7.4.5 Altri Task di Etichettatura di Sequenza

- **Tokenizzazione**
  - Prevedere tag Start/NonStart per ogni carattere (necessario per il cinese)
- **Code Switching**: uso di più lingue nello stesso testo
  - Riconoscere i punti di commutazione
- **Dialogue Acts** (riconoscere se un enunciato è una proposizione, pone una domanda, ecc.)

Capitolo 8

Constituency Trees e Dependency Parsing

8.1 Costituenti

- Un costituente è un gruppo di parole all'interno degli enunciati che può essere dimostrato agire come un'unità singola
- Un sintagma nominale è una sequenza di parole che "agiscono" come un sostantivo:
  - They love
  - The Broadway coppers love
  - A high-class spot such as Mindy's attracts
- Mentre il sintagma nominale può essere riposizionato nella frase, le parole all'interno di un sintagma nominale non possono

8.2 Alberi di Parsing

- Un albero di parsing è un albero che evidenzia la struttura sintattica di una frase secondo una grammatica formale
  - Mostra le relazioni tra parole o sotto-frasi
- L'albero di parsing dei costituenti si basa sul formalismo delle grammatiche context-free. In questo tipo di albero, la frase è divisa in costituenti, cioè sotto-frasi che appartengono a una categoria specifica nella grammatica.
  - VP $\rightarrow$ V NP
- Un albero di dipendenza è un grafo dove le parole sono nodi, e gli archi sono etichettati con la relazione tra i nodi.

8.3 Albero dei Costituenti

- Le parole sono nodi foglia, i loro genitori sono tag POS, gli antenati dei tag POS sono costituenti della frase: VP, NP, PP...

- Ambiguità:
  - Chi sta indossando il pigiama?
  - Molti parsing grammaticalmente corretti ma semanticamente irragionevoli
  - L'uso del ML supervisionato permette di apprendere interpretazioni di buon senso

8.4 Parsing di Dipendenza

- Albero di Parse di Dipendenza (grafo aciclico con un nodo radice):
  - Le parole appaiono nei nodi interni
  - I discendenti modificano (arricchiscono) il nodo testa
- Un albero di dipendenza evidenzia importanti informazioni (semantiche) (chi ha fatto l'azione specificata dal verbo a chi)
- Gli archi in un parser di dipendenza potrebbero essere etichettati con il tipo di relazione

8.5 Tecniche di Parsing di Dipendenza

Due paradigmi:

- **Parsing di dipendenza basato su grafi**
  - Formulare il parsing come un problema di ottimizzazione combinatoria su un insieme (possibilmente ristretto) di alberi di dipendenza
  - Alberi completi come input
- **Parsing di dipendenza basato su transizioni**
  - Formulare il parsing come una sequenza di problemi di classificazione locale: ad ogni punto nel tempo, prevedere una delle diverse azioni del parser
  - Costruisce l'albero passo dopo passo

8.5.1 Parsing di Dipendenza Basato su Grafi

- Data una frase x e un insieme $Y(x)$ di alberi di dipendenza candidati per x, vogliamo trovare un albero con il punteggio più alto $\in Y(x)$:

$$\hat{y} = \arg \max_{y \in Y(x)} \text{score}(x, y) \tag{8.1}$$

- La complessità computazionale dipende dalla scelta di $Y(x)$ e dalla funzione di punteggio

Il Modello Arc-factored

- Il punteggio di un albero di dipendenza è espresso come la somma dei punteggi dei suoi archi:

$$\text{score}(x, y) = \sum_{(h,d) \in y} s(x, h, d) \tag{8.2}$$

- Il punteggio di un singolo arco può essere calcolato da qualsiasi algoritmo di apprendimento che riceve la testa e il dipendente come input
- Nel modello arc-factored, l'albero di dipendenza con il punteggio più alto può essere trovato in tempo $O(n^3)$ (n = lunghezza della frase)
- Maximum spanning tree su un grafo diretto con tutte le relazioni di dipendenza
- Estendere il modello arc-factored (ad esempio facendo dipendere il punteggio di un arco dagli archi con la stessa testa o ascendenti) rende il problema intrattabile

8.5.2 Approcci Basati su Transizioni

- Gli approcci basati su grafi potrebbero essere troppo esigenti dal punto di vista computazionale per grandi dataset
- Gli approcci basati su transizioni utilizzano un classificatore locale per costruire un singolo albero di dipendenza passo dopo passo
- Solitamente lineare rispetto alla lunghezza della frase di input
- Data una frase di input, iniziare con un albero di dipendenza vuoto
- Poi chiamare un classificatore, che prevede la transizione che il parser dovrebbe fare per passare a una prossima configurazione
  - Estendere l'albero di dipendenza parziale
- Questo processo viene ripetuto finché il parser raggiunge una configurazione terminale
  - Albero di dipendenza completo

Algoritmo Arc-Standard

- Un parsing di dipendenza basato su transizioni
- L'algoritmo arc-standard può prevedere solo alberi di dipendenza proiettivi
  - Ogni sottoalbero corrisponde a una sequenza contigua di parole (nessuno dei due archi si intersecano)
  - Albero non proiettivo: yesterday è tra "pizza which was vegetarian"

Arc-Standard consiste in:

- Un buffer, che contiene quelle parole nella frase che devono ancora essere elaborate

  - Inizialmente, il buffer contiene tutte le parole

- Uno stack, che contiene quelle parole nella frase che sono attualmente in fase di elaborazione
  - Inizialmente, lo stack è vuoto
- Un albero di dipendenza parziale
  - Inizialmente, questo albero contiene tutte le parole della frase, ma nessun arco di dipendenza
- **Transizione Shift (SH)**: rimuove la parola più anteriore dal buffer e la spinge in cima allo stack
- **Transizione Left-arc (LA)**: crea una dipendenza dalla parola più in alto nello stack alla seconda parola più in alto, ed elimina la seconda parola più in alto
- **Transizione Right-arc (RA)**: crea una dipendenza dalla seconda parola più in alto nello stack alla parola più in alto, ed elimina la parola più in alto
- Ad ogni fase, un classificatore decide la migliore azione tra:
  - Transizione Shift (SH), Transizione Left-arc (LA), Transizione Right-arc (RA)

Transizioni Valide e Proprietà dell'Algoritmo

- **Transizioni valide**
  - SH è valido se il buffer contiene almeno una parola
  - LA e RA sono validi se lo stack contiene almeno due parole
- Il numero di transizioni che l'algoritmo arc-standard impiega per costruire un albero per una frase con n parole è 2n - 1
- **Soundness**
  - Ogni sequenza di transizione valida che inizia nella configurazione iniziale e termina in qualche configurazione terminale costruisce un albero di dipendenza proiettivo
- **Completeness**
  - Ogni albero di dipendenza proiettivo può essere costruito da qualche sequenza di transizione valida che inizia nella configurazione iniziale e termina in qualche configurazione terminale

Classificatore per Arc-Standard

- Il classificatore può utilizzare le seguenti caratteristiche:
  - Le parole nel buffer
  - Le parole sullo stack
  - L'albero di dipendenza parziale
- Il classificatore è stato implementato come una rete neurale feed forward in [1]
  - Prestazioni state-of-the-art nel 2014

Etichette Gold per Algoritmi Basati su Transizioni

- Dobbiamo tradurre l'albero di dipendenza gold in azioni gold per il classificatore (approccio oracle statico):
  - Scegliere LA se questo creerebbe un arco dall'albero gold-standard, e se tutti gli archi dalla seconda parola più in alto nello stack sono già stati assegnati dal parser (perché quella parola verrebbe rimossa dallo stack e non potrebbe formare alcun arco)
  - Scegliere RA se questo creerebbe un arco dall'albero gold-standard, e se tutti gli archi dalla parola più in alto nello stack sono già stati assegnati dal parser
  - Altrimenti, scegliere SH

8.5.3 Approcci Neurali

- Chen et al. [1] incorporano le prime 3 parole sullo stack e nel buffer, così come certi discendenti delle parole principali sullo stack
  - Dimensione dell'embedding della parola = 50
  - Oltre agli embedding delle parole, utilizzano anche embedding per i tag part-of-speech e le etichette di dipendenza
  - Dimensione dell'embedding del tag = dimensione dell'embedding dell'etichetta = 50
  - La dimensione dell'input risultante della FNN è 2400
  - Minimizzare la loss di cross entropy rispetto alle etichette dell'oracle statico
- Kiperwasser et al. propongono di utilizzare un set minimo di caratteristiche principali basate su embedding contestualizzati ottenuti da una Bi-LSTM
  - LSTM è una rete neurale ricorrente il cui scopo è preservare relazioni a lungo termine
  - Bi-LSTM: il contesto è calcolato da sinistra a destra e da destra a sinistra
  - Calcola la rappresentazione delle prime 2 parole nello stack e 1 nel buffer
- L'algoritmo di Kiperwasser et al. è stato esteso per essere uno scorer per approcci basati su grafi

- Data una coppia di parole, assegnare un punteggio come coppia (testa, dipendente)
- La loss massimizza il margine (y* etichetta gold)

8.5.4 Valutazione del Parsing di Dipendenza

- Unlabelled Attachment Score (UAS): percentuale di parole che sono collegate al genitore corretto
- Label Accuracy (LA): percentuale di parole che hanno l'etichetta di relazione corretta
- Labelled Attachment Score (LAS): percentuale di parole che sono collegate al genitore corretto con l'etichetta di relazione corretta

Capitolo 9

Machine Translation

9.1 Introduzione alla Machine Translation

- La Machine Translation (MT) è uno dei problemi più antichi e più studiati nell'NLP
- Dati di addestramento - Corpora paralleli:
  - Discorsi politici (Europarl, Hansards)
  - Sottotitoli di film

9.2 Valutazione della MT

- **Adeguatezza**: il contenuto linguistico della lingua di destinazione riflette quello della lingua di input
- **Fluidità**: la traduzione dovrebbe leggersi come un testo fluente nella lingua di destinazione
- Idea: confrontare la traduzione con la traduzione di riferimento
- BLEU: quale frazione di n-grammi della traduzione appare nel riferimento?
- I punteggi degli n-grammi vengono poi sommati insieme

9.2.1 BLEU

- BLEU, come definito sopra, può essere manipolato
  - P_n potrebbe essere 0, i conteggi devono essere smussati
  - "on on on on on" avrebbe un punteggio p_1 alto con "the cat is on the table" - Ogni parola nel riferimento può essere usata solo una volta
  - BLEU tende a favorire traduzioni brevi (il denominatore è più basso) - viene aggiunta una penalità per le traduzioni brevi

9.2.2 ROUGE

- **ROUGE-N**: Sovrapposizione di N-grammi tra i riassunti del sistema e di riferimento (la frase di riferimento è ora al denominatore)
  - ROUGE 1 (sovrapposizione di unigrammi)
  - ROUGE 2 (sovrapposizione di bigrammi)
  - ROUGE-L: basato sulla sottosequenza comune più lunga (LCS)

9.3 MT su Corpora Non Paralleli

- Se non vengono fornite frasi parallele, dobbiamo allineare le frasi nelle due lingue
- Abbiamo bisogno di una funzione di costo che prende un intervallo di frasi di origine e un intervallo di frasi di destinazione e restituisce un punteggio che misura quanto è probabile che questi intervalli siano traduzioni
  - Similarità del coseno tra embedding multilingue delle due frasi
- Un algoritmo di allineamento che prende questi punteggi per trovare un buon allineamento tra i documenti
  - Algoritmi di programmazione dinamica per trovare il miglior allineamento dati i punteggi a coppie tra le frasi

9.4 Neural Machine Translation

- La Machine Translation può essere implementata come un modello Encoder-Decoder
- Quasi tutti fanno la traduzione frase per frase
- Il decoder è simile a un modello di linguaggio in cui condizionamo anche sulla frase di input (z)
- Come per altre applicazioni, le NN ricorrenti (LSTM) sono state utilizzate per implementare l'architettura Encoder-Decoder

9.4.1 Modelli Seq2seq

- Problemi con i modelli Seq2seq
  - Una frase è rappresentata come un vettore fisso
  - Tutte le parole non sono ugualmente importanti per prevedere una parola target
  - È difficile mantenere relazioni a lungo termine
  - È stato osservato che le prestazioni dei modelli seq2seq si degradano con la lunghezza della frase da tradurre

9.4.2 Modelli di Attention

- Per ogni parola target, concentrarsi solo su specifiche parole di origine
- Data la parola target precedente, il meccanismo di attention "valuta" ogni parola di origine
- Dopo l'addestramento, nella fase di previsione, potrebbero esserci parole nella frase di input che non sono state viste durante l'addestramento
  - Nomi propri
  - Forme morfologiche non viste
- L'attention può aiutare:
  - Calcoliamo l'allineamento tra le parole nella frase
  - Cerchiamo di abbinare con un dizionario o scomponendo la parola morfologicamente e ottenendo la forma radice

Capitolo 10

Disinformation Detection: Fact-checking e Fake News

10.1 Introduzione alla Disinformazione

La disinformazione è un problema crescente nell'era digitale, con fake news che si diffondono rapidamente attraverso i social media e altre piattaforme online.

10.2 Linguaggio delle Fake News

Le fake news tendono ad avere caratteristiche linguistiche distintive:

- Titoli più sensazionalistici
- Contenuto testuale più semplice e ripetitivo
- Maggiore somiglianza con la satira rispetto alle notizie reali

10.3 Fact-checking

10.3.1 Pipeline di Fact-checking

1. Determinare se l'affermazione è meritevole di verifica
2. Verificare se l'affermazione è già stata verificata
3. Recuperare prove
4. Verificare l'affermazione

10.3.2 Affermazioni Meritevoli di Verifica

- ClaimBuster: primo sistema per il rilevamento della check-worthiness
- Assunzione: tutte le affermazioni verificate dalle organizzazioni di fact-checking sono quelle meritevoli di fact-checking

Caratteristiche per la Check-Worthiness

- **Caratteristiche a livello di frase**
  - TF.IDF bag of words
  - Tag POS
  - Numero di parole
  - Sentiment
  - Tempo del verbo
  - Caratteristiche linguistiche
  - Named Entities
  - Numero di simboli
- **Caratteristiche a livello di contesto**
  - Posizione
  - Dimensione del segmento
  - Metadati: reazione del pubblico
  - Metadati: menzione dell'avversario
- **Caratteristiche miste**
  - Caratteristica di topic
  - Word2Vec
  - Discorso
  - Contraddizione ("Non ho detto questo")
  - Similarità con affermazioni precedentemente fact-checked

10.3.3 Fact-checking Automatico

- Problema: le persone non si fidano dei metodi automatici
  - Tuttavia, si fidano dei fact-checker umani
- Proposta: Cercare affermazioni precedentemente fact-checked
  - I politici tendono a ripetere le stesse affermazioni!

Definizione del Task

Data un'affermazione di input meritevole di verifica e un insieme di affermazioni verificate, classificare quelle affermazioni verificate, in modo che le affermazioni che possono aiutare a verificare l'affermazione di input, o una sotto-affermazione in essa, siano classificate sopra qualsiasi affermazione che non sia utile per verificare l'affermazione di input.

10.3.4 Fact-checking Basato su Knowledge Graph

- Affermazione: "Sadiq Khan è un cittadino del Regno Unito"
- Regole: isCitizenOf(X, Y) ← mayorOf(X, Z), hasCapital(Y, Z)
- Le regole possono essere generate automaticamente dall'affermazione e dalle relazioni nel KG
- Le prove vengono recuperate dal KG per soddisfare le regole e produrre spiegazioni

10.3.5 Fact-checking Utilizzando Solo Testo

- Dataset "Liar, Liar Pants on Fire"
- Etichette:
  - Pants-fire
  - False
  - Barely true
  - Half-true
  - Mostly-true
  - True
- Metadati disponibili per gli oratori:
  - Affiliazioni di partito
  - Lavoro attuale
  - Stato di origine
  - Storia creditizia (conteggi di dichiarazioni imprecise per ogni oratore)

10.3.6 Perplexity della Disinformazione

La disinformazione tende ad avere un'alta perplexity (una misura di quanto il testo è prevedibile per un modello di linguaggio).

10.3.7 Modelli di Linguaggio come Fact-checkers

I modelli di linguaggio pre-addestrati possono essere utilizzati come basi di conoscenza per verificare le affermazioni fattuali.

10.4 Generazione di Fake News

- Grover: Generatore di notizie
  - Modello di linguaggio come generatore
  - Un'architettura leggermente modificata può essere utilizzata per la classificazione
  - Architettura avversaria: l'avversario genera fake news, il verificatore classifica una notizia come vera o falsa

10.5 Rilevamento di Inconsistenze Multimodali

Un approccio promettente per rilevare le fake news è identificare inconsistenze tra diverse modalità (ad esempio, testo e immagine).

Capitolo 11

Conclusioni e Direzioni Future

11.1 Riassunto del Corso

Questo corso ha coperto i fondamenti teorici e pratici del Natural Language Processing, dalle basi linguistiche ai modelli neurali avanzati. Abbiamo esplorato:

- Elementi di linguistica rilevanti per l'NLP
- Fondamenti di machine learning applicati all'NLP
- Preprocessing del testo e creazione di dataset
- Rappresentazioni vettoriali del testo (embeddings)
- Modelli di linguaggio neurali
- Architetture neurali specifiche per l'NLP (LSTM, Attention, ecc.)
- Applicazioni pratiche come sentiment analysis, question answering, machine translation e fact-checking

11.2 Tendenze Attuali nell'NLP

Le tendenze attuali nell'NLP includono:

- Modelli di linguaggio di grandi dimensioni pre-addestrati (come BERT, GPT, ecc.)
- Prompt engineering come alternativa al fine-tuning
- Modelli multimodali che integrano testo con altre modalità (immagini, audio, ecc.)
- Approcci di apprendimento con pochi esempi o zero-shot
- Focus crescente sull'interpretabilità e l'etica nell'NLP

11.3 Sfide e Direzioni Future

Nonostante i progressi significativi, rimangono diverse sfide nell'NLP:

- Riduzione del bias nei modelli di linguaggio
- Miglioramento della comprensione del linguaggio naturale per lingue con poche risorse
- Sviluppo di modelli che possono effettivamente "comprendere" il linguaggio piuttosto che semplicemente modellarlo statisticamente
- Creazione di sistemi NLP che possono ragionare, pianificare e interagire in modo più simile all'uomo
- Integrazione della conoscenza del mondo reale nei sistemi NLP

Bibliografia

[1] Dan Jurafsky and James H. Martin, *Speech and Language Processing (3nd edition)*, Prentice Hall, 2008.

[2] Jacob Eisenstein, *Natural Language Processing*, MIT Press, 2019.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is All You Need*, NIPS, 2017.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, NAACL, 2019.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient Estimation of Word Representations in Vector Space*, ICLR, 2013.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, *GloVe: Global Vectors for Word Representation*, EMNLP, 2014.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, *Enriching Word Vectors with Subword Information*, TACL, 2017.

[8] Danqi Chen and Christopher Manning, *A Fast and Accurate Dependency Parser using Neural Networks*, EMNLP, 2014.

[9] Eliyahu Kiperwasser and Yoav Goldberg, *Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations*, TACL, 2016.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, ICLR, 2015.