Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

TECNICHE DI PROMPTING *di Large Language Model* PER IL *Word Sense Disambiguation*

Relatore:
Chiar.mo Prof.re
Fabio Tamburini

Presentata da:
Anna Tosoroni

III Sessione
Anno Accademico 2023/2024

Indice

Introduzione

Nel campo del Natural Language Processing (NLP), determinare con precisione il significato delle parole all'interno di un contesto è fondamentale. Errori nell'interpretazione dei significati possono portare a fraintendimenti e, talvolta, alla diffusione di informazioni errate. In particolare, la gestione della polisemia, ovvero la presenza di più significati per una stessa parola, rappresenta una delle sfide principali che rendono il Word Sense Disambiguation, la disambiguazione del significato delle parole, un problema di grande rilevanza.

Si considerino ad esempio le seguenti frasi:

1. *La pianta ha bisogno di più luce per crescere.*
2. *La pianta dell'edificio fornisce una visione dettagliata della distribuzione degli spazi interni ed esterni.*
3. *Un'accurata analisi della pianta del piede aiuta ad individuare problemi posturali.*

La parola *pianta* assume nelle tre frasi significati diversi, risultando così polisemica. Pur essendo scritto e pronunciato allo stesso modo, il termine cambia di senso a seconda del contesto in cui viene usato: si riferisce ad un organismo vegetale nella frase (1), alla rappresentazione grafica di un edificio in (2), alla parte inferiore del piede nell'esempio (3). Mentre per l'essere umano la distinzione tra i significati risulta immediata e spesso banale, grazie al contesto, per i sistemi NLP non vi è ancora un metodo capace di risolverlo alla perfezione. Il Word Sense Disambiguation e, dunque, la corretta interpretazione di parole polisemiche è ad oggi una sfida ancora aperta.

Nel corso del tempo sono stati sviluppati diversi metodi per affrontare il problema del WSD, distinguibili principalmente tra tecniche supervisionate ed approcci knowledge-based. I primi, rivelatisi più efficaci in termini di performance, si basano su modelli di apprendimento automatico addestrati su dataset annotati, in cui ad ogni parola polisemica è associato il significato corretto. Nei metodi knowledge-based, invece, la disambiguazione avviene grazie all'utilizzo di importanti risorse lessicali, come WordNet e BabelNet, sulla base delle definizioni e relazioni semantiche da esse riportate.

Nel primo capitolo, dopo aver introdotto il concetto di Word Sense Disambiguation, viene riportata una panoramica delle metodologie principali utilizzate per affrontare questo problema. Inoltre, si analizza nel dettaglio l'ontologia lessicale di WordNet, una delle risorse maggiormente consultate per il WSD. Vengono presentati i diversi ambiti d'applicazione del WSD, i principali limiti ancora in fase di studio ed, infine, un'analisi dei sistemi che costituiscono lo stato dell'arte attuale nel campo.

Nonostante i recenti progressi abbiano portato i sistemi supervisionati ad elevate performance nel WSD, valutati sui dataset standardizzati proposti da Raganato et al. [16], ci sono ancora delle difficoltà dovute ad una comprensione semantica insufficiente. Nei modelli supervisionati i significati delle parole sono, infatti, definiti solo in base alle occorrenze presenti nei dataset di addestramento. Il sistema, quindi, non incorpora esplicitamente il significato linguistico di un termine, ma si basa unicamente su ciò che è stato incluso nei dati di training. In questo modo, i significati poco frequenti, o mai visti, risultano difficili da trattare in maniera corretta, specialmente in presenza di dataset con risorse limitate. Per superare questi limiti sono stati sviluppati approcci più avanzati che sfruttano informazioni semantiche aggiuntive, come le definizioni dei sensi (*glosse*), direttamente integrate nei modelli neurali, o l'inserimento di ulteriori conoscenze provenienti dalle risorse lessicali di riferimento, come WordNet e BabelNet.

Riprendendo l'esempio iniziale sulla polisemia, numerosi studi hanno evidenziato come il significato di una parola dipenda fortemente dal contesto in cui è inserita. Di conseguenza, l'analisi isolata dei singoli termini risulta spesso inadeguata per una corretta disambiguazione. Per questo motivo, è fondamentale integrare informazioni relative la posizione delle parole nella frase, l'uso dei tag di Part-of-Speech (POS) e, quando possibile, l'inserimento di un contesto più ampio, al fine di migliorare la precisione nel WSD.

Con l'introduzione dei Large Language Models (LLMs) il Word Sense Disambiguation è stato, però, fortemente rivoluzionato. Questi modelli, addestrati su enormi corpora testuali, hanno dimostrato capacità estremamente avanzate nella comprensione contestuale. In particolare, il fine-tuning di LLMs per compiti specifici, come la generazione di risposte accurate o la creazione di conoscenza in domini ben precisi, ha ottenuto risultati davvero promettenti. Nel Capitolo 2 vengono esaminati i Large Language Models, come sono evoluti storicamente, come agiscono nel dettaglio e le implicazioni etiche derivanti dal loro utilizzo.

Con questo progetto si vuole sperimentare un paradigma alternativo, basato sull'uso diretto di LLMs, senza alcun fine-tuning specifico per il Word Sense Disambiguation. Attraverso il Prompt Engineering si sfrutta la conoscenza pre-addestrata degli LLMs, guidando il comportamento del modello esclusivamente tramite istruzioni precise e prompt

ottimizzati. Nel Capitolo 3 verranno approfondite le tecniche di Prompt Engineering, analizzando le strategie per incrementare l'efficacia dei prompt, migliorando così la disambiguazione semantica. A differenza dell'apprendimento supervisionato tradizionale, con l'apprendimento basato su prompt è stato possibile utilizzare LLMs pre-addestrati, nello specifico modelli open-source della famiglia LLaMA, eliminando la necessità di dati supervisionati per lo specifico compito e fasi di training computazionalmente costose.

Nel Capitolo 4 sono descritti nel dettaglio i dataset utilizzati per la valutazione, le metodologie sperimentali adottate e l'intero processo di affinamento dei prompt. Sono state condotte numerose sperimentazioni su diversi modelli LLaMA, con l'obiettivo di individuare la strategia più efficace per la disambiguazione. Una delle principali sfide affrontate è stata la gestione di termini altamente ambigui, caratterizzati cioè da una vasta gamma di significati. Per questi casi, è stato fondamentale guidare il processo di inferenza attraverso espliciti ragionamenti, sfruttando al massimo la conoscenza pre-addestrata dei modelli. L'approccio più efficace è stato raggiunto combinando prompt di Chain of Thought con prompt di verifica del senso selezionato. Nonostante richieda tempi di generazione considerevoli, dovuti alla verifica per ciascuna selezione del senso, questa tecnica ha permesso di ottenere ottime performance con LLMs di moderate dimensioni.

I risultati, presentati nel Capitolo 5, confermano il potenziale di questa metodologia, dimostrando come il Prompt Engineering possa essere una soluzione efficace per la disambiguazione dei significati delle parole senza la necessità di dataset annotati, né di costosi processi di pre-addestramento.

La repository con il codice sviluppato, i dataset utilizzati ed i risultati ottenuti è disponibile su GitHub: `WSDprompting_repository`.

Capitolo 1

Word Sense Disambiguation

Il concetto di Word Sense Disambiguation (WSD), tradotto letteralmente come disambiguazione del senso delle parole, rappresenta un problema centrale nel campo del Natural Language Processing (NLP). Tale problema consiste nell'identificare il significato specifico che una parola polisemica assume nel determinato contesto in cui appare. Il senso di una parola, dunque, dipende esclusivamente dal contesto in cui essa è utilizzata. Più formalmente, data una parola target w e un contesto c, un sistema WSD può essere descritto come una funzione f tale che

$$f(w, c) = s, \quad s \in S_w \tag{1.1}$$

dove Sw è l'insieme dei possibili significati di w.
Un modello WSD riceve in input una sequenza di contesti $c = c_0, c_1, \ldots, c_n$ e produce in output una sequenza di predizioni di sensi $s = s_i c_0, s_j c_1, \ldots, s_m c_n$.
Il modello, pertanto, associa ad ogni contesto c il relativo senso s appartenente all'insieme dei sensi candidati per quel contesto. Inoltre, viene assunto che per ogni senso s si disponga anche di una definizione (o *gloss*) $g_s = g_0, g_1, \ldots, g_n$ che definisce s. Per l'essere umano questa disambiguazione è un processo che avviene, talvolta, in maniera inconscia ed intuitiva; per i sistemi NLP, invece, rappresenta una sfida decisamente complessa, in quanto è il risultato dell'integrazione di informazioni sintattiche, semantiche e pragmatiche. Per un sistema automatico, infatti, la difficoltà del problema non è solo nello scegliere il senso più appropriato per un termine, ma risiede, soprattutto, nel modo di scegliere l'insieme di significati da considerare. Determinare il senso corretto di ogni termine, in base allo scenario in cui si presenta, permette la ricezione quanto più accurata delle informazioni e risulta ormai fondamentale in molte applicazioni.

Il termine Word Sense Disambiguation fu introdotto per la prima volta da Warren Weaver nel 1949[1]. Egli riconobbe il problema della presenza di parole polisemiche

nel contesto del Machine Translation, ovvero nella traduzione automatica di una frase da una lingua ad un'altra. Successivamente, nel 1975, Kelly e Stone[2] pubblicarono l'elenco delle regole selezionate per una corretta disambiguazione dei significati delle parole. Nonostante lo sviluppo di algoritmi con regole precise, che rendevano l'estrazione lessicale via via più automatizzata, la disambiguazione si basava ancora totalmente sulla conoscenza e sull'uso del dizionario, rendendo così limitata la possibilità di automatizzare il processo su larga scala.

Con l'avvento del Machine Learning, l'interesse per il WSD è diventato uno dei principali obiettivi delle tecniche di apprendimento supervisionato. Si è passati così a metodi che permettevano di addestrare, tramite l'utilizzo di dataset annotati, modelli in grado di mappare il contesto linguistico ai sensi delle parole. É in questo contesto che si inserisce la creazione di WordNet, la risorsa lessico-semantica più utilizzata in questo campo, descritta successivamente.

Negli ultimi anni, i progressi nel WSD sono stati guidati dall'integrazione di modelli pre-addestrati, come BERT (*Bidirectional Encoder Representations from Transformers*) e GPT (*Generative Pre-trained Transformer*), che hanno rivoluzionato il campo del NLP. Questi modelli, successivamente descritti nel Capitolo 2, hanno portato significativi miglioramenti nelle prestazioni del WSD. Tuttavia, nonostante i progressi, rimangono ancora molte le sfide aperte, come la disambiguazione in contesti multilingue e l'applicazione del WSD a lingue con risorse limitate.

1.1 I metodi

Disambiguare il senso di un termine richiede la comprensione del contesto in cui quel termine è inserito e la conoscenza dei diversi significati ad esso associati. Sono stati sviluppati numerosi approcci per affrontare il problema di WSD, classificabili in quattro principali categorie [3]:

- Apprendimento Supervisionato
- Apprendimento Non Supervisionato
- Metodi Knowledge-based
- Metodi Ibridi

Nell'apprendimento supervisionato[4] il WSD viene trattato come un problema di classificazione: un modello viene addestrato a mappare una parola target al suo significato corretto in uno specifico contesto, sulla base delle occorrenze in un corpus di

riferimento. Nel caso di parole polisemiche, il significato più appropriato per ciascun termine viene selezionato attraverso uno dei seguenti modi:

- Most Frequent Sense:
  metodo che assume che il significato più frequente di una parola nel corpus sia il più probabile anche nei contesti futuri. I significati di ogni termine, perciò, vengono ordinati in modo decrescente secondo la loro frequenza nel training set.
  Sebbene sia un approccio semplice ed efficace in contesti specifici, la sua accuratezza diminuisce quando i sensi della parola sono distribuiti in modo uniforme, o quando non esiste un senso predominante. Questa classificazione è implicitamente contenuta all'interno di WordNet.

- Decision List:
  tabelle organizzate secondo lo schema *[feature-value, sense, score]*.
  Il primo elemento, la feature, è un esempio ricavato dal training set, mentre il secondo è il senso della parola che si sta valutando, relativo alla feature in questione. Il terzo termine, invece, è il punteggio, calcolato secondo una formula probabilistica descritta da Yarowsky nel 1995[5], che rappresenta la probabilità di selezionare il senso corretto dato il valore della feature. L'algoritmo itera lungo la decision list, a partire dalle istanze con punteggio più alto, e confronta ciascuna feature con il contesto del termine da disambiguare, assegnando così il senso con la più alta probabilità.

- Naive Bayes[6]:
  metodo probabilistico che utilizza il Teorema di Bayes per calcolare la probabilità del senso Si di una parola w, data una feature fj, estratta dal contesto. L'algoritmo restituisce il senso che massimizza questa probabilità. Questo metodo prevede, però, che le features siano tra loro indipendenti, il che può limitare la sua accuratezza in alcuni casi.

Come evidenziato precedentemente, il WSD ha enormemente beneficiato dell'introduzione di reti neurali e modelli di linguaggio pre-addestrati. Questi modelli costituiscono oggi l'approccio standard per il WSD supervisionato, grazie alla loro capacità di rappresentare le parole nel contesto specifico in cui appaiono, catturando al contempo sfumature semantiche anche complesse. Un'altra tecnica supervisionata rilevante è il Gloss-based Learning, che utilizza le definizioni testuali (glosse) delle parole per ottenere informazioni sui loro significati, migliorando la disambiguazione nei casi in cui il contesto sia ambiguo. Il limite principale dei metodi supervisionati è la loro dipendenza dai corpora annotati manualmente. Questi, utilizzati come training set, permettono al modello di imparare

a prevedere il senso corretto di una parola target in contesti mai visti prima. Tuttavia, rappresentano una criticità, in quanto risultano costosi e talvolta limitati in termini di copertura linguistica.

L'apprendimento non supervisionato si basa sull'ipotesi che parole con significati simili tendano a comparire in contesti simili (Distributional Hypothesis)[7]. A differenza dell'apprendimento supervisionato, non richiede corpora annotati, ma utilizza tecniche di clustering per raggruppare le occorrenze di parole e dedurne i sensi. Un esempio significativo di apprendimento non supervisionato è rappresentato dall'uso dei Word Embeddings, che rappresentano le parole come vettori densi in uno spazio multidimensionale. Tuttavia, uno dei limiti principali di questi metodi è la tendenza a mescolare significati multipli in un'unica rappresentazione vettoriale, riducendo la loro utilità per il WSD. Inoltre, sebbene non richieda dati annotati, questo metodo tende ad essere meno accurato rispetto ai metodi supervisionati.

I metodi Knowledge-based, basati sulla conoscenza, sfruttano risorse lessicografiche e ontologie, come WordNet e BabelNet, per mappare le parole nei loro diversi significati. Il principio del loro funzionamento si basa, dunque, sull'utilizzo di queste risorse esterne, contenenti una conoscenza ben strutturata dalla quale attingere per assegnare un senso ai termini da disambiguare. I metodi knowledge-based si differenziano tra loro per procedimento e risorsa esterna utilizzata.

Il primo e più celebre metodo knowledge based è quello ideato da Lesk nel 1986[8], che necessita di un dizionario, come fonte di conoscenza, o di un insieme di definizioni che copra almeno l'intero insieme di parole da disambiguare. L'algoritmo si basa sul principio della sovrapposizione del contesto (*Contextual Overlap*): per ciascun termine viene confrontato il contesto, ovvero un intorno di parole di dimensione fissata, con tutte le definizioni dei suoi possibili significati. Il senso assegnato è quello che presenta maggiore affinità (intesa come corrispondenza di termini) con l'intorno del termine. Questo può, talvolta, costituire un limite, poiché le definizioni nei dizionari tendono ad essere sintetiche. Pertanto, sebbene il metodo basato su sovrapposizione sia semplice, non affronta quei casi in cui le parole nel contesto non hanno termini in comune con le definizioni.

Tecniche più recenti utilizzano, invece, i grafi semantici per identificare le relazioni possibili tra i sensi. Viene quindi sfruttata una rete semantica, in cui i significati delle parole sono rappresentati dai nodi e le relazioni semantiche dagli archi, per costruire un grafo del contesto. Tramite algoritmi di ricerca sul grafo, vengono poi identificati i sensi corretti analizzando le connessioni tra i nodi.

I metodi ibridi, infine, combinano caratteristiche degli approcci supervisionati, non supervisionati e knowledge-based, con l'obiettivo di sfruttare i punti di forza di ciascun sistema e ridurre al minimo i rispettivi limiti. Un esempio di approccio ibrido

per il WSD è ESCHER (Enhanced Semantics through Contextual Hybrid Extractor for Resolution)[9], un sistema che combina approcci Knowledge-based, come le informazioni semantiche estratte da WordNet, con tecniche di Machine Learning supervisionato. Il modello pre-addestrato su corpora annotati, basato su BERT, viene dunque combinato ai glossari estratti da WordNet, migliorando così l'accuratezza sia nei casi ambigui, che nei contesti non presenti nel corpus di addestramento.

1.2 Le Risorse

La disambiguazione del senso delle parole si basa su un insieme di risorse linguistiche fondamentali, denominati *Sense Inventory*. Queste risorse lessicali rappresentano il punto di riferimento essenziale per identificare i possibili significati delle parole. Fra i principali *Sense Inventory* si distinguono:

- WordNet [10]: considerato lo standard de facto per l'inglese, organizza i significati in synset, collegati fra loro da relazioni semantiche. Ogni synset rappresenta un significato concettuale distinto e offre una definizione affiancata da esempi d'uso.
- BabelNet [11]: un dizionario multilingue creato mappando insieme WordNet con Wikipedia. Offre una copertura senza precedenti per oltre 500 lingue.
- Wiktionary [12]: un progetto collaborativo che ha recentemente guadagnato popolarità. Sebbene meno strutturato rispetto a WordNet e BabelNet, Wiktionary mira a creare un dizionario separato per ogni lingua.

Fra gli elementi fondamentali che costituiscono la base per il training e la valutazione dei modelli di WSD non si possono non menzionare i corpora. Dal latino *'corpo'*, i corpora (al singolare corpus) sono collezioni di testi selezionati e organizzati per facilitare le analisi linguistiche. Vengono distinti in annotati e non annotati: nei primi, il significato delle parole del testo viene etichettato manualmente, ovvero disambiguato da parte di una persona umana, mentre i secondi sono testi "grezzi". Entrambi sono utili rispettivamente nella disambiguazione supervisionata e non supervisionata. Un corpus annotato è chiaramente più difficile da produrre, data la necessità dell'intervento umano. Tra i dataset annotati, emergono:

- SemCor [13]: il più grande corpus annotato manualmente basato su WordNet. Creato dal team di ricerca di WordNet, include 352 documenti per un totale di circa 234.000 parole annotate. SemCor è ampiamente utilizzato per l'addestramento di modelli supervisionati.

- MultiSemCor: corpus bilingue di SemCor, con sensi annotati facenti riferimento alle versioni inglese e italiana di WordNet.
- OMSTI (One Million Sense-Tagged Instances)[14]: uno dei più grandi corpus annotati con i sensi di WordNet 3.0, contiene oltre un milione di istanze annotate. É considerata una delle principali risorse nel training di sistemi supervisionati.
- XL-WSD [15]: un benchmark che include 18 lingue, che risulta particolarmente rilevante nel testing con approcci zero-shot.
- Senseval e SemEval: prodotti in occasione degli eventi internazionali dedicati alla valutazione dei sistemi WSD. Essi rappresentano un punto di riferimento nel campo NLP, offrendo benchmark standardizzati e aggiornati per misurare le prestazioni dei vari approcci.

1.3 Stato dell'Arte

La disambiguazione del senso delle parole ha registrato negli anni significativi progressi, grazie all'adozione di numerosi sistemi supervisionati e al miglioramento dei metodi knowledge-based. Per garantire un confronto equo tra i diversi sistemi esistenti di WSD, Raganato et al. [16] hanno proposto un framework di valutazione unificato (*Unified Evaluation Framework*, UEF) che include diversi dataset standardizzati, consentendo di valutare le prestazioni sia di metodi supervisionati che knowledge-based. Esso si compone di 7253 istanze da disambiguare, con parole che variano da quelle con un solo significato disponibile a termini, come "*break*", che può avere fino a 59 sensi possibili.

La metrica di valutazione per i sistemi WSD è l'F1-score, che rappresenta la media armonica tra precisione e richiamo.

Il framework UEF include due principali dataset di addestramento: SemCor [13] e OMSTI [14]. Il processo di valutazione viene eseguito sui cinque dataset che compongono UEF, provenienti da Senseval/SemEval, con le annotazioni standardizzate secondo il *sense inventory* di WordNet 3.0:

- Senseval-2 (SE2)
- Senseval-3 (SE3)
- SemEval-2007 (SE07)
- SemEval-2013 (SE13)

- SemEval-2015 (SE15)

Le tecniche di WSD, come anticipato, si suddividono generalmente tra approcci supervisionati, che utilizzano dati di addestramento annotati con il senso corretto delle parole, ed approcci knowledge-based, dove il senso corretto è determinato tramite risorse lessicali, come WordNet o BabelNet.

1.3.1 Approcci supervisionati

Gli approcci supervisionati basano l'apprendimento su corpora annotati, tra cui il più utilizzato è il dataset SemCor. I primi tentativi includevano decision trees, sistemi Naïve Bayes, reti neurali e classificatori SVM. Negli ultimi anni, invece, sono stati sviluppati modelli supervisionati più sofisticati in grado di migliorare significativamente le prestazioni. Tra i principali sistemi supervisionati che costituiscono attualmente lo stato dell'arte (SOTA), per il set di valutazione UEF, si riportano:

- IMS (*It Makes Sense*) [17]: un sistema supervisionato che utilizza SVM con features contestuali della parola target, come termini circostanti, POS tag e informazioni sulla posizione dei termini nel testo;

- IMS+emb: una versione avanzata di IMS che incorpora tra le features precedenti anche word embeddings, per migliorare ulteriormente le prestazioni. Ne esiste una terza versione, IMS-s+emb, che utilizza le stesse features con word embeddings, escludendo però le parole che circondano il termine target;

- Context2Vec [18]: modello che sfrutta architetture BiLSTM (*Bidirectional Long Short-Term Memory*) per apprendere context embeddings, ovvero rappresentazioni vettoriali del contesto, in cui sono contenute informazioni semantiche e sintattiche. Contenendo rappresentazioni più ricche e dettagliate sul significato delle parole, questi modelli rendono la disambiguazione più accurata;

- EWISER (*Enhanced WSD Integrating Synset Embeddings and Relations*) [19]: modello supervisionato che utilizza il grafo delle connessioni delle risorse lessicali, come WordNet, direttamente all'interno della rete neurale. Il sistema riesce in questo modo ad integrare tra le informazioni anche le relazioni semantiche tra i sensi delle parole (*synset embeddings*);

- ESCHER [9]: un'architettura basata su Transformers, che permette di ridurre il bias, molto comune, di scelta del significato più frequente nei dati di addestramento;

- GlossBERT [20]: un modello basato su BERT (*Bidirectional Encoder Representations from Transformers*) che costruisce coppie contesto-glossa, in cui le definizioni di ogni possibile senso vengono associate al contesto del termine target.
  GlossBERT viene addestrato nella classificazione della definizione più corretta per il contesto della parola target.

- ConSeC (*CONtinuous SEnse Comprehension*) [21]: utilizza un meccanismo iterativo (*feedback loop strategy*) attraverso il quale si analizza non solo il contesto della parola ambigua, ma anche i significati già assegnati alle parole vicine. Grazie a questa strategia, ConSec riesce a sfruttare meglio le informazioni contestuali, ottenendo risultati più accurati nel WSD.

Un'importante baseline per i metodi supervisionati è l'euristica del senso più frequente (*Most Frequent Sense*, MFS), che assegna ad ogni parola target il senso più comune nei dati di addestramento. Come si nota in Tab.1.1, i valori di F1-score ottenuti dai sistemi supervisionati sono ampiamente al di sopra dell'approccio MFS. Questo suggerisce che, in generale, questa euristica non risulta particolarmente efficace nel riconoscimento del migliore significato di parole altamente ambigue.

Inoltre, considerato che non tutte le parole presenti nei test set sono state precedentemente analizzate nel training set, viene inserito un punteggio massimo di F1-score. Questo valore, definito *Ceiling*, rappresenta quindi il punteggio massimo di accuratezza che i sistemi supervisionati possono raggiungere.

Nella tabella 1.1 vengono riportati i valori F1-score di UEF rispetto ai sistemi supervisionati SOTA, addestrati sul dataset SemCor.

Sistemi	**SemEval-2007**	**Senseval-2**	**Senseval-3**	**SemEval-13**	**SemEval-15**	**ALL**
MFS	54.5	65.6	66.0	63.8	67.1	64.8
IMS	61.3	70.9	69.3	65.3	69.5	68.4
IMS+emb	60.9	71.0	69.3	67.3	71.3	69.1
IMS-s+emb	62.6	72.2	70.4	65.9	71.5	69.6
Context2Vec	61.3	71.8	69.1	65.6	71.9	69.0
EWISER	71.0	78.9	78.4	78.9	79.3	78.3
ESCHER	76.3	81.7	77.8	82.2	83.2	80.7
GlossBERT	72.5	77.7	75.2	76.1	80.4	77.0
ConSeC	**77.4**	**82.3**	**79.9**	**83.2**	**85.2**	**82.0**
Ceiling	*93.8*	*91.0*	*94.5*	*88.6*	*90.4*	*91.5*

Tabella 1.1: Sistemi supervisionati per il WSD, addestrati con dataset SemCor

1.3.2 Approcci knowledge-based

I sistemi knowledge-based per il WSD (KBWSD) non richiedono dati annotati di addestramento, ma sfruttano conoscenze provenienti da dizionari e risorse lessicali(KB), come WordNet e BabelNet. Esistono principalmente due approcci in questa categoria:

- Metodi basati sulla similarità:
  tra i principali rientra l'algoritmo di Lesk, che confronta le definizioni dei possibili sensi con il contesto della parola target, basandosi sulla sovrapposizione di parole comuni. Una versione più efficiente di questo metodo è Lesk+emb, proposta da Basile et al. [22], che migliora il calcolo della similarità utilizzando word embeddings, andando cioè oltre la semplice corrispondenza lessicale;

- Metodi basati su grafi:
  costruiscono un grafo semantico rappresentante il contesto, seguendo le connessioni fornite dalla risorsa lessicale. Gli approcci basati su grafi generalmente ottengono risultati migliori rispetto ai metodi Lesk, grazie proprio all'uso delle relazioni semantiche all'interno della risorsa lessicale utilizzata. Al contempo, l'efficacia di questi sistemi dipende proprio dalla qualità e dalla copertura delle basi di conoscenza selezionate. Fra i principali esempi ci sono:

  - PageRank su WordNet [23]: viene creato un sottografo dei possibili sensi della parola target e, tramite l'algoritmo di PageRank, si identifica quello più rilevante;
  - UKB [24]: il contesto del termine target indirizza la distribuzione di probabilità verso i sensi più pertinenti. UKB applica *random walks* sull'intero grafo semantico di WordNet per determinare il senso corretto delle parole, sfruttando le connessioni tra i sensi. UKB-gloss è un approccio avanzato di UKB, che include tra le connessioni nel grafo le glosse disambiguate;
  - Babelfy [25]: sfrutta BabelNet come risorsa lessicale e l'algoritmo *Random Walk with Restart* per migliorare la disambiguazione.

Una baseline comune nei sistemi knowledge-based è il *WordNet 1st sense* (WN 1st), che assegna a ciascuna parola target il primo senso secondo WordNet, assumendo sia il più probabile in base alla frequenza d'uso riportata nell'ontologia.

Sistemi	SemEval-2007	Senseval-2	Senseval-3	SemEval-13	SemEval-15	ALL
Lesk	32.0	50.6	44.5	53.6	51.0	48.7
Lesk+emb	**56.7**	63.0	63.7	66.2	64.6	63.7
UKB	39.0	56.0	51.7	53.6	55.2	53.2
UKB-g	42.0	60.6	54.1	59.0	61.2	57.5
Babelfy	51.6	**67.0**	63.5	**66.4**	**70.3**	**65.5**
WN 1st sense	55.2	66.8	**66.2**	63.0	67.8	65.2

Tabella 1.2: Sistemi knowledge-based per WSD

I metodi basati sulla conoscenza, seppur interessanti per la loro indipendenza dai corpora annotati di addestramento, tendono ad ottenere prestazioni inferiori rispetto ai modelli supervisionati. Questo è dovuto principalmente alla difficoltà di costruire una base di conoscenza che risulti completa ed efficace. Inoltre, nell'approccio knowledge-based è essenziale comprendere quali siano le informazioni migliori da integrare per garantire una rappresentazione coerente e precisa di ogni significato.

Al contrario, gli approcci supervisionati apprendono direttamente dalle annotazioni presenti nei dataset di addestramento, permettendo di associare in modo più accurato determinate features al senso corretto.

1.4 Applicazioni del WSD

Il Word Sense Disambiguation trova applicazione in diversi ambiti del Natural Language Processing e dell'intelligenza artificiale, spaziando da applicazioni più classiche e consolidate, ad altre più recenti ed innovative.

1.4.1 Machine Translation

La traduzione automatica, storicamente una delle prime applicazioni del WSD, ha permesso di migliorare la precisione nella scelta delle parole nel linguaggio di destinazione. La disambiguazione risulta, infatti, necessaria ogni qualvolta un termine presenta diverse traduzioni in un'altra lingua, oppure nei casi in cui l'idioma di partenza può essere ambiguo a seconda del contesto in cui si trova. Nonostante la potenziale utilità del WSD nel Machine Translation (MT), oggi la maggior parte dei sistemi MT adotta metodi statistici, basati su un lessico predisambiguato e regole definite manualmente.

1.4.2 Information Retrieval

L'Information Retrieval (IR) consiste nel recupero di documenti pertinenti a partire da una query inserita dall'utente. Nei motori di ricerca, ad esempio, per aumentare la pertinenza dei risultati le query vengono divise in token, ricercati poi in base alla loro frequenza nella collezione di documenti. Le query di ricerca, tuttavia, possono contenere termini ambigui, per cui disambiguare in questo caso equivale ad identificare il significato corretto e, dunque, aumentare la precisione dei processi IR. Inoltre, i termini contenuti nella query possono essere strettamente legati ad altri che non vengono inseriti dall'utente, che, se riconosciuti tramite l'uso di relazioni semantiche, potrebbero aumentare il numero di risultati utili della ricerca.

1.4.3 Question answering

Il Question Answering (QA) è un sistema progettato per fornire risposte a domande espresse in linguaggio naturale. A differenza dell'IR, un sistema di QA, oltre ai singoli termini contenuti nella query, deve tener conto anche del loro ordine e della funzione grammaticale, in modo da comprenderne il contenuto e fornire una risposta diretta alla domanda posta. In questo contesto, il WSD svolge dunque un ruolo cruciale nell'interpretazione corretta della domanda.

1.4.4 Word Processing

Un altro ambito in cui facilmente si riconoscono i possibili benefici della Word Sense Disambiguation è quello dei programmi di scrittura e dei sistemi di correzione automatica. In questi casi il WSD permette di migliorare la qualità del controllo ortografico e delle correzioni automatiche, modificando il testo in base al significato delle parole nel contesto. Nelle applicazioni di scrittura predittiva, inoltre, il WSD consente di proporre suggerimenti più pertinenti, aumentando la velocità e l'accuratezza della scrittura.

1.4.5 Semantic Web

Nel Semantic Web i documenti digitali sono arricchiti da metadati che, codificando le informazioni semantiche su di essi, favoriscono l'interoperabilità tra utente e macchina. Classificando e collegando le informazioni in modo più efficace, si può così migliorare l'accesso ai dati e ai risultati di ricerca. Il WSD risulta utile, in questo caso, per l'annotazione semantica dei documenti, in quanto può associare le parole ai loro significati corretti in base a un'ontologia di riferimento.

1.5 Sfide e Limiti del WSD

Nonostante l'ampia gamma di applicazioni pratiche, il WSD presenta alcune criticità. Fra tutte emerge il problema relativo alla granularità dei significati: alcuni Sense Inventory, come WordNet, definiscono sensi con una granularità eccessiva, rendendo difficile la distinzione persino per annotatori umani. L'elevato grado di polisemia in alcuni termini, associati dunque ad un numero eccessivo di synsets, rende le differenze reciproche fra i significati così sottili da rendere complicata una disambiguazione accurata. Come sarà mostrato nel Capitolo 5, in numerosi contesti sarebbe stato sufficiente un livello di distinzione dei sensi meno dettagliato per ottenere risultati più accurati. La sfida consiste dunque nel bilanciare la granularità, evitando sia l'eccessiva semplificazione che l'eccessiva complessità.

Un'altra sfida da affrontare quando si parla di WSD è sicuramente l'evoluzione linguistica, e, con essa, la dinamicità del linguaggio. Le lingue, in continua evoluzione, vedono il costante emergere di nuovi termini e significati, spesso influenzati da fenomeni culturali e tecnologici. Questa rapida evoluzione rende difficile mantenere aggiornati i Sense Inventory, costringendoli alla continua revisione e all'ampliamento delle proprie risorse linguistiche.

Va, inoltre, considerata la problematica dovuta alla copertura linguistica, ovvero al fatto che la disponibilità di risorse annotate risulta limitata per molte lingue. La maggior parte delle risorse linguistiche utilizzate per il WSD, come i corpora annotati, è disponibile principalmente per le lingue più diffuse. Essendo, invece, limitate o talvolta inesistenti le risorse per lingue meno documentate, lo sviluppo di sistemi di WSD multilingue risulta fortemente limitato. L'espansione delle risorse multilingue, come BabelNet, ha rappresentato comunque un passo avanti, ma rimane ancora molto lavoro da fare per garantire una copertura adeguata.

Strettamente correlato ai due problemi precedenti si inserisce il così definito *Knowledge Acquisition Bottleneck*. Un problema storico che affligge lo sviluppo dell'intelligenza artificiale, esso sancisce che per quanto vasto possa essere un lessico o un Sense Inventory, non potrà coprire i significati di tutte le parole polisemiche di una lingua e non potrà stare al passo con la sua evoluzione [26]. Inoltre, essendo la creazione di corpora annotati manualmente un processo costoso e lungo, risulta difficile la costruzione di dataset sufficientemente ampi e costantemente aggiornati.

Nonostante queste sfide, la ricerca dei migliori metodi alla disambiguazione non si è fermata. I progressi nella potenza computazionale e la crescente disponibilità di risorse digitali hanno aperto nuove prospettive per superare i limiti del WSD. Le reti semantiche di grandi dimensioni, i dizionari machine-readable e l'uso di modelli di linguaggio

pre-addestrati, come i Transformer, di cui si parlerà nel Capitolo 2, hanno migliorato significativamente le performance dei sistemi di WSD.

1.6 WordNet

Il processo di disambiguazione del testo consiste essenzialmente nell'identificazione dei concetti associati ai vari lemmi, ovvero nell'assegnare, ad ogni parola, il senso più corretto in base al contesto nel quale è utilizzato. Come descritto nelle sezioni precedenti, la disambiguazione presuppone l'utilizzo di un'ontologia lessicale, dalla quale sia possibile reperire le informazioni relative ai concetti e alle relazioni che vi intercorrono. Tra le sorgenti d'informazione lessicale più ampiamente utilizzate e riconosciute vi è WordNet. WordNet è un'ontologia lessicale caratterizzata da due tipi principali di conoscenza:

- Conoscenza lessicale: derivante dalle parole intese come singoli caratteri
- Conoscenza semantica: derivante dai significati che emergono dai costrutti lessicali e dalle relazioni che intercorrono tra di essi.

Lo sviluppo di WordNet ha inizio nel 1985 presso il Cognitive Science Laboratory dell'Università di Princeton, sotto la direzione di George A. Miller [27]. Viene sviluppato uno dei database lessicali fra i più utilizzati al mondo, basato sulle teorie psicolinguistiche formulate a partire dalla memoria lessicale umana. Il team di sviluppo dello psicologo Miller, costituito da linguisti e lessicografi, decise di organizzare WordNet non come un semplice dizionario, con le sole informazioni sul significato degli elementi sintattici, bensì come un sistema costituito anche dalle relazioni tra le parole in esso contenute. In WordNet, infatti, le informazioni sono memorizzate in base al significato ed alle categorie sintattiche, e sono legate fra loro da diversi tipi di relazioni.

Ad ogni termine di WordNet vengono associati due elementi distinti:

- la Word Form: il lemma, ovvero la forma scritta associata a una stringa di caratteri;
- il Word Meaning: il significato, ovvero il concetto espresso dalla Word Form.

Quindi il punto d'inizio della classificazione delle parole secondo WordNet sono le relazioni che intercorrono fra lemma e significato. I lemmi vengono suddivisi secondo quattro categorie sintattiche: nomi, verbi, aggettivi ed avverbi. Ogni categoria sintattica è suddivisa in diversi *synsets*, insiemi di sinonimi; ad ognuno di questi insiemi è associato un unico significato, condiviso da tutti i termini ad esso associati. Il synset è l'unità base dell'ontologia WordNet, che permette di descrivere un concetto attraverso i suoi *glossa*,

definizioni utili a spiegare il significato, quando l'insieme di termini da solo non è sufficiente, e una o più brevi frasi esplicative. Un termine, ovviamente, può possedere più di un significato ed essere, quindi, presente in molti di questi insiemi, ed anche in più di una categoria sintattica. Termini che appartengono a più di un synset vengono definiti polisemici.

1.6.1 La matrice lessicale

Come anticipato nella sezione precedente, esistono delle relazioni fra la forma di una parola, ovvero il modo in cui viene scritta e letta, ed il significato ad essa associato. Queste associazioni, di tipo molti a molti, vengono espresse con il concetto di:

- Sinonimia: proprietà per cui due o più parole distinte esprimono lo stesso significato;
- Polisemia: proprietà per cui ad una stessa parola sono associati due o più significati distinti. Tali parole, ambigue dal punto di vista del significato, vengono definite polisemiche (viceversa, sono definite monosemiche quelle non ambigue, in quanto possiedono un solo significato).

La corrispondenza tra lemma e synset, ossia tra parola e significato, viene rappresentata tramite Matrice Lessicale (Fig.1.1).

Word	Word Forms				
Meanings	F_1	F_2	F_3	.....	F_n
M_1	$E_{1,1}$	$E_{1,2}$			
M_2		$E_{2,2}$			
M_3			$E_{3,3}$		
.....					
M_m					$E_{m,n}$

Figura 1.1: La matrice lessicale di WordNet

Le righe della matrice sono i significati che è possibile attribuire ad una parola (Word

Meanings). Ad ogni riga è dunque associato un synset, ovvero un gruppo di termini sinonimi. Le colonne costituiscono invece i lemmi (Word Forms); ogni colonna rappresenta un lessema, ed è quindi composta da parole polisemiche.
Se ci sono, dunque, due elementi nella stessa colonna, significa che quella Word Form ha più sensi o significati. Invece, se ci sono due elementi nella stessa riga, le due Word Form sono sinonime. Ogni elemento Eij non nullo all'interno della matrice implica che quel lemma, situato nella riga i, può essere usato per rappresentare lo specifico significato associato alla colonna j. L'insieme dei termini associati ad un synset non è comunque sufficiente a descrivere pienamente un significato. Per questo, ai synsets di WordNet viene associata anche una descrizione del significato tramite la glossa.

1.6.2 Le relazioni di WordNet

WordNet si distingue per la ricchezza di informazioni semantiche offerte, che permettono di individuare e fornire all'utente le relazioni che si instaurano tra i concetti o tra i lemmi. In particolare, le relazioni principali considerate sono:

- Relazioni lessicali tra i lemmi: coinvolgono due word form, ovvero due lemmi.
  Le principali relazioni di questo tipo sono:
  - Sinonimia:
    l'appartenenza di un termine ad un synset. Due lemmi sono considerati sinonimi se condividono lo stesso synset; La sostituzione di un lessema con l'altro in un determinato contesto non muta il valore della frase in cui compare.

    Ad esempio, i termini albero, arbusto e pianta non possono sempre essere usati in modo intercambiabile, poiché hanno significati distinti. Tuttavia, secondo la definizione di WordNet, appartengono allo stesso synset, in quanto condividono un significato comune, consentendo la distinzione da altri sensi.
  - Antonimia:
    la relazione che intercorre tra due lessemi di significato diametralmente opposto. Un termine y è un antonimo di x se vale la proposizione y = not x. Riguarda principalmente la categoria sintattica degli aggettivi.

    Ad esempio, i termini felice e triste sono antonimi, ma non essere tristi non implica necessariamente l'essere felici. Tra i due stadi esistono, infatti, molteplici stati emotivi che non appartengono pienamente a nessuno dei due estremi.
- Relazioni semantiche tra i synset: coinvolgono due concetti, ovvero due synsets.
  Le principali relazioni semantiche presenti in WordNet sono:

– Iponimia/Iperonimia:
  un synset X è iperonimo di un synset Y se X ha un significato più esteso e comprensivo, ossia può rappresentare una generalizzazione di Y. La relazione simmetrica rispetto all'iponimia è l'iperonimia. Questa relazione, anche nota come relazione *ISA* (is a kind of), è la relazione codificata con più frequenza all'interno di WordNet, ed è valida sia per i nomi che per i verbi (in questo caso si parla di troponimia). L'iponimia genera una gerarchia nella struttura semantica, secondo cui gli iponimi (concetto figlio) si trovano al di sotto dei propri iperonimi (concetto padre), formando così una gerarchia di specializzazione tra i synset.

  Ad esempio, abete è un iponimo di albero, poiché è un tipo specifico di albero. AL contrario, strumento musicale è un iperonimo di pianoforte, poiché questo rientra nella categoria, più ampia, degli strumenti musicali.

– Meonimia/Olonimia:
  esprime il concetto di *'è parte di'* all'interno della categoria sintattica dei nomi. Il synset X = x1, x2,... è un meronimo di un concetto rappresentato da Y = y1, y2,... se è accettabile la proposizione *'X è parte di Y'*. Anche la meronimia e la sua inversa olonimia costituiscono una gerarchia.

  Ad esempio, maniglia è un meronimo di porta, in quanto la maniglia è una parte della porta. La relazione di meronimia è generalmente transitiva, ma non sempre. Per esempio, se maniglia è un meronimo di porta e porta è un meronimo di casa, non è necessariamente corretto dire che maniglia è un meronimo di casa.

Capitolo 2

Large Language Models

I Large Language Models, tradotto letteralmente modelli linguistici di grandi dimensioni e più comunemente conosciuti come LLMs, sono modelli di Deep Learning progettati per comprendere, generare ed elaborare il linguaggio umano in maniera estremamente avanzata. I LLM rappresentano oggi una delle più significative innovazioni nel campo dell'Intelligenza Artificiale (AI) e del Natural Language Processing (NLP). Come il termine '*Large*' fa intuire, sono modelli di volume considerevole, sia per le dimensioni dei dataset su cui vengono addestrati, sia per il numero di parametri che li compongono. I parametri sono tutti quei valori numerici che il modello apprende durante il training e che poi determineranno il modo in cui verrà processato e generato il linguaggio. La capacità degli LLMs di gestire grandi volumi di dati e di apprendere relazioni complesse tra parole e frasi li rende strumenti potenti per una vasta gamma di applicazioni, tra cui la traduzione automatica, la generazione di testo, l'analisi semantica, la creazione di chatbot e molte altre attività legate all'elaborazione del linguaggio naturale. Grazie alla loro capacità di comprendere contesti articolati e generare risposte coerenti, i Large Language Models hanno trasformato profondamente il panorama delle tecnologie linguistiche, offrendo nuove possibilità sia nel campo della ricerca che in ambito applicativo.

2.1 Evoluzione storica

Per molti anni, a partire dal 1990, la modellazione e la generazione delle sequenze di testo venivano effettuate utilizzando semplici reti neurali ricorrenti (RNN). Le RNN erano progettate per elaborare le sequenze un token alla volta, mantenendo uno stato nascosto aggiornato ad ogni passo temporale. Tuttavia, le RNN soffrivano del *Vanishing Gradient Problem*[28], che rendeva difficile apprendere e mantenere le informazioni rilevanti su lun-

ghe sequenze. Questo fenomeno si verificava perché i gradienti utilizzati per aggiornare i parametri del modello si riducevano drasticamente man mano che la sequenza veniva elaborata, causando la perdita di informazioni cruciali. Di conseguenza, al termine di una lunga frase, il modello non riusciva a mantenere traccia delle informazioni sui token iniziali, compromettendo l'accuratezza delle sue predizioni.

Un primo passo avanti significativo si ebbe nel 1995, con l'introduzione della Long Short-Term Memory (LSTM)[29]. Questa nuova architettura, basata su una RNN innovativa, era in grado di superare il problema del gradiente, favorendo un apprendimento migliore su sequenze lunghe. Grazie all'introduzione di un sistema di celle di memoria e di meccanismi di gating, il modello LSTM riusciva a superare il problema delle dipendenze a lungo termine. Nella cella di memoria le informazioni vengono aggiunte, aggiornate o eliminate attraverso dei *gate*, ovvero delle porte che regolano il flusso delle informazioni. L'obiettivo principale è decidere quali informazioni memorizzare, quali eliminare e quali trasmettere all'output successivo, in base all'importanza che hanno per il contesto della sequenza. Ogni cella di memoria utilizza tre gate fondamentali:

- Input gate: per decidere quali nuove informazioni aggiungere alla cella di memoria;
- Forget gate: per stabilire quali informazioni della cella di memoria devono essere eliminate;
- Output gate: per selezionare le informazioni che devono essere utilizzate per l'output attuale.

I modelli LSTM sono stati per anni l'architettura standard per la modellazione delle sequenze, fino alla comparsa dei Transformers nel 2017.

Un'evoluzione parallela delle LSTM furono le Gated Recurrent Units (GRU)[30], una variante semplificata che manteneva prestazioni analoghe alle LSTM, riducendo però la complessità computazionale. Entrambe le architetture vennero utilizzate nei primi modelli Sequence-to-Sequence (Seq2Seq), in particolare nel contesto della traduzione automatica. I primi modelli Seq2Seq erano composti da un LSTM Encoder e un LSTM Decoder, che convertivano una sequenza di input in un vettore di dimensione fissa, successivamente decodificato per generare l'output. Sebbene rappresentassero un progresso significativo rispetto alle semplici RNN, questi modelli soffrivano di un collo di bottiglia causato dal vettore a dimensione fissa: se la sequenza di input era lunga, le informazioni rilevanti venivano inevitabilmente perse.

Il problema venne, in parte, risolto nel 2015 con l'introduzione del modello RNNsearch, che implementava un meccanismo di attenzione in grado di migliorare la capacità del modello di catturare le dipendenze a lungo termine. Grazie al sistema di attenzione,

il Decoder poteva "consultare" l'intera sequenza di input durante la generazione dell'output, eliminando il collo di bottiglia del vettore fisso e migliorando notevolmente le prestazioni nei compiti di traduzione automatica. Il nome RNNSearch deriva dal fatto che veniva emulata la ricerca in una frase sorgente durante la decodifica di una traduzione. Vennero, dunque, proposti due tipi di attenzione: globale (quella di RNNsearch) e locale (a finestra scorrevole), con ulteriori miglioramenti ottenuti grazie a versioni ibride. Nel contesto della traduzione automatica, l'attenzione mista rivelava una qualità superiore rispetto all'attenzione globale, mentre l'attenzione locale aveva un effetto positivo sui tempi di traduzione. Nonostante l'introduzione del meccanismo di attenzione, i modelli Seq2Seq soffrivano ancora di limitazioni legate alla loro natura sequenziale, che rendeva difficile la parallelizzazione durante l'addestramento, limitando, così, l'efficienza computazionale.

La svolta decisiva, come anticipato, avvenne nel 2017, con la pubblicazione del paper "*Attention is All You Need*"[31] da parte di un team di ricercatori di Google Brain. Questo lavoro propose una nuova architettura, il Transformer, che avrebbe cambiato per sempre il panorama dell'NLP. A differenza delle RNN e delle LSTM, i Transformer sono in grado di elaborare intere sequenze di dati in parallelo, grazie all'uso del meccanismo di *self-attention* (auto-attenzione). Uno degli autori, Jakob Uszkoreit, era convinto che l'attenzione senza la ricorrenza fosse sufficiente per la traduzione linguistica; da qui il titolo "*l'attenzione è tutto ciò di cui hai bisogno*". Il Transformer nel documento originale prevedeva un'architettura Encoder-Decoder, in cui ogni componente faceva uso di un meccanismo di *multihead attention* (attenzione multi-testa). Questo approccio consentiva al modello di "prestare attenzione" a diverse parti della sequenza simultaneamente, migliorando la capacità di apprendere relazioni a lungo raggio. Inoltre, l'assenza di ricorrenza rendeva il modello facilmente parallelizzabile, riducendo i tempi di addestramento e aumentando l'efficienza computazionale. Con il meccanismo di self-attention ogni token nella sequenza poteva calcolare la sua relazione con tutti gli altri token, migliorando significativamente la qualità delle informazioni apprese. Grazie, inoltre, alla capacità di parallelizzazione, i Transformer potevano sfruttare appieno la potenza delle GPU, riducendo drasticamente i tempi di addestramento e aprendo la strada a modelli sempre più grandi.

In seguito alla diffusione dei Transformer, la ricerca si è concentrata sull'aumento delle dimensioni dei modelli e dei dataset utilizzati per il pretraining, seguendo le *Scaling laws* proposte da [32] nel 2020. Secondo queste leggi, l'aumento del numero di parametri del modello e dei dati di addestramento portava a miglioramenti significativi delle prestazioni.

Nel 2018, Google introdusse BERT[33] (*Bidirectional Encoder Representations from*

Transformers), un modello basato su Transformer Encoder, progettato per l'apprendimento contestuale bidirezionale. L'Encoder bidirezionale di BERT permetteva di analizzare il contesto delle parole in entrambe le direzioni della sequenza, migliorando significativamente la comprensione del linguaggio rispetto ai modelli precedenti. Nell'ottobre 2019, Google iniziò ad utilizzare BERT per elaborare le query di ricerca[34].

Nel 2020, invece, Google Translate sostituì il precedente modello RNN-Encoder–RNN-Decoder con un modello Transformer-Encoder–RNN-Decoder[35].

Poco dopo, con la serie di modelli GPT (Generative Pre-trained Transformer) di OpenAI, basati su Transformer Decoder, vennero stabiliti nuovi standard nella generazione del linguaggio naturale. Con l'introduzione di GPT-3 nel 2020, un modello autoregressivo generativo con 175 miliardi di parametri, si dimostrò come l'aumento delle dimensioni potesse migliorare notevolmente le capacità generative e di comprensione del linguaggio[36].

Nel 2022, il chatbot basato su GPT-3, ChatGPT, divenne rapidamente popolare[37], innescando un boom senza precedenti attorno ai LLMs. Questo ha portato allo sviluppo di numerosi ulteriori LLMs, come:

- Gopher (2022)[38], modello con una forte enfasi sulla comprensione del linguaggio scientifico;
- Chinchilla (2022)[38], che ha ridefinito le scaling laws ottimizzando il rapporto tra il numero di parametri e il numero di token di addestramento;
- LLaMA (2023)[39], focalizzato sull'efficienza computazionale durante l'inferenza;
- Gemini, annunciato nel 2023, contendente di GPT-4 di OpenAI [40]. Progettato per essere multimodale, in quanto in grado di elaborare più tipi di dati contemporaneamente, come testo, immagini, audio, video e linguaggi di programmazione

Parallelamente, si è andato sviluppando un acceso dibattito tra modelli open-source (ad esempio, BLOOM, OPT, Falcon) e modelli chiusi come GPT-3 e Chinchilla. I modelli open-source hanno cercato di colmare il divario attraverso tecniche di *distillation* e *fine-tuning* su istruzioni sintetiche, portando alla nascita di modelli come Vicuna e Alpaca.

A partire dal 2020, i Transformer sono stati utilizzati anche al di fuori del dominio testuale, con applicazioni che spaziano dalla visione artificiale, al riconoscimento vocale, fino alla robotica e ai modelli multimodali, segnando un punto di svolta nel campo delle reti neurali. In Fig.2.1 vengono riportati gli LLMs introdotti a partire dal 2019. In giallo sono marcati i LLMs open-source.

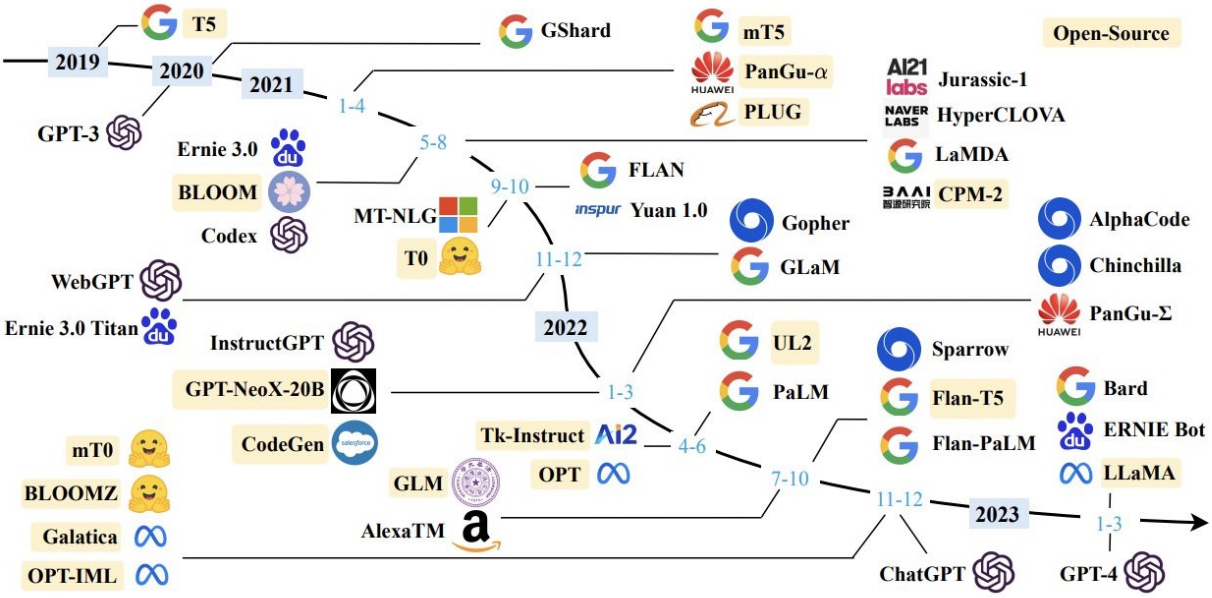

Figura 2.1: Timeline LLMs open-source e proprietari

2.2 Architettura

I Large Language Models si basano su una particolare architettura di rete neurale definita Transformer, introdotta per la prima volta da Vaswani et al. nel 2017 [31]. La loro innovazione principale risiede, come il titolo dell'articolo suggerisce, nel meccanismo di *self-attention* che li compone. Esso permette di elaborare sequenze di dati in parallelo, anche considerevoli, in maniera efficiente, catturando le relazioni tra gli elementi indipendentemente dalla loro distanza nella sequenza. Il meccanismo dell'Attenzione consente al modello di assegnare un "peso" alle parole in base alla loro importanza nel contesto. Focalizzandosi, dunque, sulle parti rilevanti, il Transformer riesce a comprendere il contesto globale di una sequenza di token.

Analizzando la struttura base di un Transformer si individuano due componenti principali: l'Encoder e il Decoder. Il ruolo dell'Encoder è quello di prendere in input una sequenza di token e trasformarla in una rappresentazione vettoriale (*embedding*), in cui è contenuto il significato del testo. Durante questo processo, il meccanismo di self-attention è bidirezionale: vengono analizzate le relazioni tra ogni token e tutte le altre parole della sequenza, sia precedenti che successive, così da ottenere una rappresentazione completa del contesto. Successivamente, il Decoder genera una sequenza di token in

output, basandosi sulla rappresentazione vettoriale prodotta dall'Encoder. A differenza dell'Encoder, il Decoder utilizza self-attention unidirezionale, poiché deve generare una sequenza token per token, tenendo conto solo delle parole precedenti già generate.

Ad esempio, in una applicazione comune di questa architettura come la traduzione automatica:

- l'Encoder elabora il testo di input (ad esempio una frase in inglese) trasformandola in una rappresentazione intermedia
- il Decoder utilizza questa rappresentazione intermedia per generare la traduzione equivalente (ad esempio la frase francese equivalente).

Nonostante il Transformer originale utilizzi sia Encoder che Decoder, molti degli LLMs moderni hanno modificato questa struttura per specializzarsi in compiti specifici. Le varianti modificano alcuni aspetti dell'architettura principale, come il modo in cui viene gestito l'input o la direzione della generazione (ad esempio, bidirezionale o autoregressiva). Tra i sistemi, precedentemente descritti nella sezione relativa all'evoluzione storica, emergono:

- BERT (*Bidirectional Encoder Representations from Transformers*)[33]: utilizza solo l'Encoder. È ottimizzato per compiti come il completamento di frasi, la sentiment analysis e la risposta a domande.
- GPT (*Generative Pre-trained Transformer*)[36]: si basa esclusivamente sul Decoder per generare testo in modalità autoregressiva, prevedendo cioè il token successivo sulla base dei token precedenti, generando una parola alla volta. Questa caratteristica rende i modelli autoregressivi particolarmente efficaci nella generazione di testi naturali, dialoghi e contenuti coerenti, ma anche creativi.

Sebbene ciascuna variante modifichi alcuni aspetti, tutti i Transformer hanno gli stessi componenti primari:

- Tokenizzatori: convertono il testo in token (parole, sottoparole o caratteri), ovvero nelle unità di base di ogni sequenza;
- Embedding Layer: convertono i token e le posizioni in rappresentazioni vettoriali (*embeddings*), contenenti informazioni semantiche e posizionali, in uno spazio multidimensionale;
- Transformer Layer: qui vengono eseguite trasformazioni ripetute sulle rappresentazioni vettoriali, estraendo via via più informazioni linguistiche. I Transformer

Layer sono di due tipologie, a seconda che si stia facendo codifica (Encoder Layer) o decodifica (Decoder Layer). Ciascun livello di questo tipo è costituito da livelli alternati di self-attention e reti feed-forward:

  - Self-Attention Layer: calcola la relazione tra ciascun token e tutti gli altri nella sequenza, assegnando un peso maggiore ai token più rilevanti;
  - Feed-Forward Network: una rete neurale a due strati che affina ulteriormente le rappresentazioni dei token.

- Classifier Layer: riconverte la rappresentazione vettoriale finale in una distribuzione di probabilità sui token, determinando il token successivo nella sequenza.

In generale, ogni layer del Transformer include una rete neurale feed-forward, responsabile della trasformazione delle rappresentazioni dei token. Queste reti sono multilayer perceptron a due strati, con una funzione di attivazione non lineare (ReLU nei primi Transformer). Il numero di neuroni nello strato intermedio è solitamente molto maggiore rispetto alla dimensione del vettore di embedding, aumentando la capacità del modello di apprendere rappresentazioni più ricche.

Il meccanismo di *multihead attention* utilizzato nell'architettura Transformer, si compone di unità di attenzione basate su prodotti scalari. Per ogni unità, il modello del trasformatore apprende tre matrici di pesi:

- la matrice di pesi della query W^Q
- la matrice di pesi chiave W^K
- la matrice di pesi del valore W^V

Il modulo accetta tre sequenze: una sequenza di query, una sequenza di chiavi e una sequenza di valori. Ciascuna sequenza, costituita da vettori, viene moltiplicata per la corrispettiva matrice. Ad esempio, ogni vettore $x_{i,query}$, appartenente alla sequenza di query, viene moltiplicato per la matrice W^Q, producendo un vettore di query $q_i{=}x_{i,query}W^Q$. La matrice con tutti i vettori di query risultanti è la matrice:

$$Q = X_{query}W^Q \tag{2.1}$$

Allo stesso modo, viene costruita la matrice chiave:

$$K = X_{query}W^K \tag{2.2}$$

e la matrice dei valori:

$$V = X_{query}W^V \tag{2.3}$$

Ottenute le matrici, il calcolo dell'attenzione per tutti i token si può esprimere come:

$$Attention(Q, K, V) = softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

dove la funzione $softmax$ viene applicata a ciascuna riga della matrice.
L'insieme delle matrici (W^Q,W^K,W^V), definita *attention head* (testa di attenzione), si concentra sui token rilevanti per ciascun token. Maggiori sono gli attention head che compongono ogni layer, maggiori saranno le relazioni catturate tra token parallelamente. I token di rilevanza possono variare dal tipo di attenzione di ciascuna testa. Ad esempio, alcuni centri di attenzione possono concentrarsi principalmente sulla parola successiva, altri si concentrano principalmente sui verbi, altri ancora sui loro oggetti diretti[41]. I calcoli per ogni attention head possono essere eseguiti in parallelo, il che consente un'elaborazione rapida. Gli output risultanti vengono, infine, concatenati e trasferiti ai livelli successivi Feed-Forward. L'intero processo può essere descritto come:

$$MultiHeadedAttention(Q, K, V) = Concat_{i\in[n_{\text{heads}}]}\left(Attention(XW_i^Q, XW_i^K, XW_i^V)\right)W^O$$

dove:

- la matrice X è la concatenazione degli embeddings
- W_i^Q,W_i^K,W_i^V sono le matrici di proiezione relative alla testa di attenzione i
- W^O è la matrice di proiezione finale relativa all'intera *multi-headed attention*.

2.3 Funzionamento

I Language Model sono distribuzioni di probabilità sulle sequenze di token [42]. Questo compito, spesso noto come Next Token Prediction (previsione del prossimo token), è stato a lungo considerato uno dei problemi centrali del Natural Language Processing[62]. I LLMs sono modelli basati su previsione autoregressiva: il token successivo di una sequenza viene previsto automaticamente sulla base dei token precedenti. Un token può essere una parola, una sottoparola o un singolo carattere. Questo processo è rappresentato matematicamente dalla seguente formula:

$$P(y \mid X) = \prod_{t=1}^{T} P(y_t \mid x_1, x_2, \ldots, x_{t-1}) \tag{2.4}$$

dove:

- X rappresenta la sequenza dei token precedenti, di lunghezza T;
- y è il token successivo che il modello sta cercando di prevedere;
- y_t è il token alla posizione t nella sequenza;
- $x_1, x_2, \ldots, x_{t-1}$ sono i token che precedono yt nella sequenza.

Come descritto dalla formula, i LLMs raccolgono molte informazioni sul contesto globale di una sequenza. Inoltre, grazie all'enorme quantità di parametri che li costituiscono, riescono a catturare relazioni semantiche, anche complesse, tra parole e frasi. Durante l'addestramento, il modello regola questi parametri per massimizzare la probabilità di prevedere correttamente il token successivo. Per rappresentare ogni token, i LLMs utilizzano *embedding* vettoriali: ogni termine viene trasformato in un vettore multidimensionale. Questa rappresentazione numerica permette al modello di cogliere le relazioni semantiche tra parole con significati simili, posizionandole vicine nello spazio vettoriale.

Le tecniche, invece, tradizionali utilizzavano rappresentazioni statiche, dove per ciascuna parola era prevista una tabella numerica. Questa forma di rappresentazione, però, non era in grado di riconoscere relazioni tra parole, come quelle con significati simili.

Grazie ai vettori multidimensionali, invece, comunemente denominati *Word Embeddings*, si ottiene una maggiore flessibilità e una migliore capacità di comprensione dell'intero contesto. Tramite l'Encoder, dunque, i Transformer riescono ad elaborare il testo rappresentandolo numericamente in *Word Embeddings*, mentre attraverso il Decoder genereranno poi l'output finale. In questo modo, è possibile comprendere e generare testo con un livello di coerenza molto elevato.

2.4 Addestramento

L'addestramento dei LLMs avviene in tre fasi principali:

- Pre-training
- Fine-tuning
- Alignment-tuning

Queste fasi permettono al modello di apprendere prima le basi del linguaggio, poi di specializzarsi in compiti specifici e, infine, di essere il più possibile affine con le preferenze umane.

2.4.1 Pre-training

Il Pre-training è la fase iniziale e più costosa dell'addestramento di un Large Language Model. Durante questa fase, il modello acquisisce la conoscenza di base di un linguaggio, imparando le strutture grammaticali, il significato delle parole e le relazioni semantiche e logiche tra concetti. Nel Pre-training il modello viene addestrato, tramite apprendimento autosupervisionato, su vastissimi corpora non annotati, contenenti centinaia di miliardi di token. In questo tipo di apprendimento, definito *Self-Supervised Learning*, il modello utilizza il contesto delle parole per auto-generare etichette, affindando così la propria comprensione linguistica. Come descritto nel Capitolo 1, il successo del pre-training di un LLM dipende dalla qualità e dalla dimensione del dataset. A seconda delle esigenze, si possono utilizzare corpus generici, specializzati o multilingua. Le tecniche comunemente utilizzate per pre-addestrare un LLM includono:

- Language Modeling: il modello impara a prevedere la parola successiva in una sequenza di testo, secondo due varianti:
  - Causal Language Modeling (CLM): data una sequenza di parole il modello cerca di prevedere la successiva. È utilizzato da modelli come GPT.
  - Masked Language Modeling (MLM): vengono mascherate alcune parole nel testo e il modello cerca di prevederle. Questa tecnica è stata resa popolare da modelli come BERT.
- Denoising Objectives: il modello impara a ricostruire il testo originale a partire da una versione disturbata o incompleta, rendendolo più robusto. Modelli come T5 (Text-to-Text Transfer Transformer)[44] utilizzano questa tecnica.

Dopo il Pre-training, il modello LLM viene definito Foundation Model, ovvero costituisce una base generica con un'enorme quantità di conoscenza. Tuttavia, esso non è ancora ottimizzato per compiti specifici, per cui a questo processo si sussegue spesso una fase di tuning.

2.4.2 Fine-tuning

Dopo aver allenato il modello di base su dati generali, si può riaddestrare su compiti specifici, come la classificazione del testo, l'analisi del sentimento di una frase o la generazione di risposte alle domande. Un esempio noto di modello pre-addestrato è GPT-3 [36], che contiene miliardi di parametri e può essere utilizzato come base per molti task successivi. Il Fine-tuning è quel processo di riaddestramento, su un dataset mirato e

spesso annotato manualmente, che permette ad un Foundation Model di adattarsi meglio a tipi di compiti specifici. Nel Fine-tuning vengono modificati i pesi del modello originale, creandone una nuova versione adattata al nuovo compito, o con una performance migliore su un dominio specifico. Nonostante sia efficace, questa tecnica può essere computazionalmente costosa, in quanto richiede l'aggiornamento di tutti i parametri del modello. Per ovviare a questo problema, si può utilizzare il Parameter-efficient tuning, che aggiorna solo una parte dei parametri, preservando gran parte della struttura del modello originale e riducendo i tempi di addestramento.

Instruction-tuning

L'Instruction-tuning è una forma avanzata di Fine-tuning in cui il modello viene addestrato a seguire istruzioni esplicite, fornite tramite esempi e descrizioni del compito. Questo processo è in grado di insegnare al modello come seguire istruzioni generiche e come generalizzare meglio su compiti mai visti prima. Si basa su multi-task prompting, esponendo il modello a diversi compiti descritti tramite prompt. Data un'instruzione e un input, che può includere testo o immagini, il modello LLM viene addestrato a generare una risposta. Come il Fine-tuning, anche l'Instruction-tuning modifica i pesi di un LLM, ma con un focus più ampio, mirato al miglioramento delle capacità generali del modello nel seguire le richieste.

Distillation

La Distillation è una tecnica che riduce la dimensione di un modello LLM mantenendo al contempo buona parte delle sue capacità predittive. Il processo prevede la creazione di un modello più piccolo (*student model*) che apprende da un modello più grande (*teacher model*). Come precedentemente riportato, la maggior parte degli LLMs fine-tuned contiene un numero elevato di parametri, richiedendo, di conseguenza, costose risorse computazionali per generare previsioni. In più, talvolta, gran parte dei parametri di questi LLMs sono irrilevanti per una specifica applicazione. Con il processo di Distillation, e, quindi, con una versione più piccola di un LLM, si possono generare predizioni più velocemente, anche se con una leggera perdita di precisione rispetto al modello originale, per via del minor numero di parametri.

La forma più comune di Distillation utilizza il processo di inferenza sul *teacher model*, per etichettare i dati di training. Questi dati vengono poi utilizzati per addestrare il nuovo modello più piccolo (*student model*). Il training set etichettato è, quindi, il modo attraverso il quale il modello più grande trasferisce la propria conoscenza al modello più piccolo.

2.4.3 Alignment-tuning

L'Alignment-tuning è la fase in cui il modello viene perfezionato per allinearsi ulteriormente alle preferenze specifiche degli utenti, migliorando la qualità delle risposte e riducendo comportamenti indesiderati, come bias o allucinazioni. Una delle tecniche più utilizzate in questa fase è il Reinforcement Learning with Human Feedback (RLHF), che utilizza, appunto, il feedback umano per migliorare le risposte del modello [45]. Il processo di RLHF si articola in tre fasi:

- Supervised Fine-Tuning, in cui il modello pre-addestrato viene perfezionato per ottenere un comportamento desiderato;
- Reward Modeling, in cui il modello impara a preferire le risposte migliori sulla base delle preferenze umane. Dato, quindi, un input e una coppia di risposte, il modello impara a dare una votazione più alta alla risposta preferita dall'utente;
- Reinforcement Learning con l'algoritmo di Proximal Policy Optimization (PPO), utilizzato per migliorare ulteriormente le prestazioni.

Un'alternativa a RLHF è il Direct Preference Optimization (DPO), che semplifica il processo evitando la necessità di un esplicito *Reward Model*, utilizzando invece una semplice classificazione binaria per imparare le preferenze umane. Le performance del modello possono essere, poi, ulteriormente migliorate grazie al Prompt Engineering, elemento centrale di questo progetto di Tesi, ampiamente descritto nel Capitolo 3.

2.5 Limiti e implicazioni etiche e sociali

Nonostante le loro capacità avanzate e la crescente diffusione in diversi settori, i Large Language Models (LLMs) presentano numerose sfide ed importanti implicazioni etiche e sociali. Uno dei principali problemi dei LLMs riguarda il fenomeno delle *hallucinations*[46], ovvero la generazione di informazioni errate o fuorvianti generate da questi modelli. Queste "allucinazioni" rappresentano un rischio significativo, in quanto possono compromettere l'affidabilità delle informazioni fornite.

A ciò si aggiunge il costo computazionale elevato; l'addestramento e l'inferenza di questi modelli, infatti, richiedono enormi risorse computazionali ed energetiche.

Un altro aspetto critico riguarda i bias e le eventuali discriminazioni generatesi con gli LLMs. Essi possono riprodurre e amplificare pregiudizi presenti nei dati di addestramento, con conseguenze che potrebbero oscillare da semplici errori di valutazione a situazioni di discriminazione vera e propria.

Un'ulteriore sfida a cui i Large Language Models devono far fronte è sicuramente legata alla privacy. Sono state spesso sollevate preoccupazioni riguardanti l'utilizzo di dati sensibili e la sicurezza delle informazioni personali nella fase di addestramento di questi modelli. É, pertanto, fondamentale adottare strategie di mitigazione del rischio per proteggere gli utenti e garantire una maggiore trasparenza nell'utilizzo dei dati.

2.6 Applicazioni nel mondo reale

Trovando applicazione in un'ampia gamma di contesti, i Large Language Models stanno ormai trasformando moltissimi settori, migliorando l'efficienza e la precisione di molte attività. Tra le principali applicazioni si evidenziano:

- Classificazione del testo: sfruttando, ad esempio, tecniche di clustering, i LLMs possono essere utilizzati per la classificazione di testi con significati o reazioni/opinioni simili. Questa applicazione trova impiego in tutti i contesti di Sentiment Analysis, nella ricerca di documenti e nella determinazione delle relazioni tra testi;

- Assistenza clienti: molti sistemi di assistenza clienti utilizzano oggi LLMs per fornire risposte rapide e coerenti alle richieste degli utenti. I chatbot basati su LLM offrono un'esperienza più naturale ed efficace rispetto ai sistemi tradizionali, garantendo un supporto rapido e personalizzato;

- Automazione delle procedure legali[47]: i LLMs trovano applicazione anche nel settore legale, dove vengono utilizzati per automatizzare il controllo e la revisione di documenti legali, riducendo i tempi di analisi e migliorando la precisione;

- Generazione di codice: modelli ormai noti, come GitHub Copilot[48], Codex di OpenAI[49], assistono gli sviluppatori nella scrittura di codice, traducendo istruzioni da linguaggio naturale in codice;

- Generazione di contenuti testuali e creativi: i Large Language Models vengono oramai utilizzati per scrivere articoli, storie, documentazione di prodotti, poesie ecc., migliorate per stile, tono e tipologia di voce.

2.7 LLaMA

LLaMA (Large Language Model Meta AI) è una collezione di Large Language Models pre-addestrati, sviluppati da Meta AI[50], per la generazione testuale e la risposta a

domande in linguaggio naturale. Lanciata per la prima volta nel 2023, la collezione LLaMA presenta modelli da 7 a 405 miliardi di parametri, mantenendo prestazioni altamente competitive rispetto a modelli proprietari, come GPT-3 e GPT-4 di OpenAI. A differenza di molti modelli di grandi dimensioni, LLaMA punta ad offrire una soluzione più efficiente ed accessibile, consentendo agli sviluppatori di eseguire il modello anche su singole GPU, favorendo così una maggiore sperimentazione a un costo ridotto.

2.7.1 Architettura e Addestramento

L'architettura di LLaMA si basa su Transformer auto-regressivi, in grado di predire il token successivo in una sequenza basandosi sui token precedenti. La pipeline di addestramento prevede due fasi principali:

- Pre-training su un enorme corpus di dati, proveniente da fonti pubblicamente disponibili.
- Fine-tuning con tecniche di Supervised Fine-Tuning (SFT) e Reinforcement Learning con Human Feedback (RLHF) per allineare il modello alle preferenze umane, migliorando l'usabilità e la sicurezza. Anche nel processo di fine-tuning sono stati inclusi dati provenienti da dataset pubblici, oltre a più di un milione di esempi annotati manualmente.

2.7.2 Versioni di LLaMA

A partire da febbraio 2023, Meta AI ha rilasciato diverse versioni di LLaMA, ciascuna con miglioramenti significativi in termini di prestazioni, scalabilità e applicazioni specifiche. L'ultima versione è LLaMA 3.3, rilasciata nel dicembre 2024 [39]. La prima versione della famiglia LLaMA include modelli di 7, 13, 30 e 65 miliardi di parametri.

Con l'introduzione di LLaMA 2 si assiste ad un significativo passo avanti, in quanto vengono rilasciate versioni più avanzate e specifiche per il dialogo, come LLaMA 2-Chat. Questo modello risulta ottimale per applicazioni di dialogo e chatbot, poiché migliora le capacità di ottenere risposte coerenti al contesto. LLaMA 2 si compone di modelli da 7, 13 e 70 miliardi di parametri. LLaMA 2 con 13 miliardi di parametri si distingue, in particolare, per l'efficienza computazionale, superando GPT-3 su molte metriche, nonostante sia ben dieci volte più piccolo.

Infine, con LLaMA 3, Meta introduce una nuova generazione di modelli nativamente multilingue e capaci di gestire compiti complessi come il coding, il ragionamento avanzato e l'integrazione con strumenti esterni. I modelli disponibili in LLaMA 3 sono di 8, 70 e

405 miliardi di parametri. Con un addestramento su 15 trilioni di token, contro i 2 trilioni di LLaMA 2, i risultati sperimentali di LLaMA 3 raggiungono prestazioni paragonabili a quelle di GPT-4 [51] in una varietà di compiti. I modelli più piccoli, al contempo, superano quelli concorrenti open-source di pari dimensioni.

2.7.3 Installazione e Configurazione

LLaMA è distribuito sotto licenza MIT, una delle più permissive nel mondo open-source. Il rilascio open degli LLMs, se effettuato in modo sicuro, rappresenta un beneficio netto per la società[52]. Questa licenza consente, infatti, agli utenti di utilizzare, modificare e distribuire il software senza restrizioni significative, a condizione di mantenere l'attribuzione agli autori originali. Tuttavia, la licenza non offre alcuna garanzia sull'uso del software e lascia agli sviluppatori la responsabilità di adottare le misure necessarie per garantire la sicurezza e l'affidabilità delle applicazioni create.

Come tutti gli LLMs, LLaMA è una tecnologia che comporta potenziali rischi d'uso [53]. Sebbene i modelli siano stati testati principalmente in lingua inglese, non è possibile prevedere ogni scenario di utilizzo. Pertanto, Meta consiglia ai ricercatori e agli sviluppatori di eseguire test di sicurezza e tuning personalizzato prima di implementare LLaMA in applicazioni reali.

LLaMA può essere eseguito sia in locale su GPU, che su ambienti cloud. L'installazione è relativamente semplice, grazie all'ampia disponibilità di pacchetti e strumenti di supporto. Per poter utilizzare LLaMA occorre:

- GPU con almeno 16 GB di VRAM per l'esecuzione di LLaMA-7B in locale. Per modelli più grandi, è necessario disporre di GPU con maggiore capacità;
- Python 3.8, o superiore;
- PyTorch per il framework di deep learning.

Una volta garantiti i requisiti di sistema, si può procedere con l'installazione tramite una delle seguenti opzioni:

- Clonare il repository ufficiale di LLaMA o utilizzare uno dei porting disponibili su Hugging Face[54];
- Scaricare i modelli ufficiali tramite l'autorizzazione concessa da Meta AI. I modelli vengono forniti come checkpoint che devono essere caricati per l'inferenza;
- Eseguire l'inferenza utilizzando gli script forniti, o integrando LLaMA in applicazioni personalizzate con PyTorch.

L'inferenza con LLaMA è simile a quella di altri modelli basati su Transformer. Gli sviluppatori possono utilizzare i modelli per generare testo, rispondere a domande, completare frasi o eseguire compiti di traduzione automatica.

Capitolo 3

Prompt Engineering

Il *Prompt Engineering* è una disciplina finalizzata allo sviluppo e all'ottimizzazione delle richieste, ovvero i *prompt*, da inviare ai Large Language Models. Questa tecnica si distingue per la capacità di sfruttare la potenza di LLMs pre-addestrati senza la necessità di un ulteriore riaddestramento. Tramite il paradigma di *Prompt Inference* si può indirizzare il comportamento di un LLM tramite richieste mirate e ben formulate, senza modificare i pesi né l'architettura del modello, ma unicamente la struttura del prompt. Vi è, dunque, una stretta dipendenza tra la risposta generata e la corretta formulazione di un prompt: modificando leggermente la richiesta, si possono ottenere output diversi e più precisi.

Come anticipato nel Capitolo 2, l'apprendimento dei LLMs ha subito un profondo cambiamento negli anni. Tra il 2017 ed il 2019, in particolare, il ruolo fino ad allora fondamentale dell'apprendimento supervisionato è stato progressivamente sostituito dal paradigma *Pre Training - Fine Tuning*[55]: un modello pre-addestrato veniva adattato a specifici compiti tramite fine-tuning, aggiornando cioè una parte o l'intero insieme di parametri. In questo paradigma, la ricerca si concentrava principalmente sull'*Objective Engineering*, ovvero sulla progettazione degli obiettivi di addestramento, sia nella fase di pre-training che in quella di fine-tuning. Esempi tipici di modelli pre-addestrati, ottimizzati in questo modo, includevano BERT[33] e RoBERTa[56]. Tale approccio, seppur semplice, presentava alcuni svantaggi, come il rischio di overfitting su dataset piccoli e il fenomeno del *Catastrophic Forgetting*, dove il modello perdeva la capacità di eseguire compiti che prima del fine-tuning era in grado di svolgere[57].

Nel 2021 si è verificato un secondo cambiamento radicale, dove il *Pre Training - Fine Tuning* è stato sostituito dal nuovo paradigma *Pre Training - Prompting - Prevision.* In questo nuovo approccio, invece di adattare i LLMs pre-addestrati ai compiti specifici tramite fine-tuning, si cerca di formulare i task nel prompt in modo che assomiglino a

quelli affrontati durante l'addestramento originario del modello. Ciò avviene attraverso l'uso di prompt testuali, che guidano il comportamento del modello senza la necessità di ulteriori addestramenti specifici [58]. Esempi tipici di prompting senza fine-tuning includono LLaMA[39] e GPT-3[36]. Il principale vantaggio di questo metodo è che permette di utilizzare lo stesso LLM per una vasta gamma di compiti, mantenendo invariati i parametri del modello. Tuttavia, questo introduce una nuova sfida: la ricerca del prompt ottimale per garantire risposte di alta qualità.

3.1 Progettazione del Prompt

L'apprendimento basato su tecniche di prompting, per essere efficace, deve tenere in considerazione numerosi elementi chiave, di seguito riportati.

3.1.1 Scelta del LLM pre-addestrato

I modelli LLMs si differenziano fra loro per dimensione, struttura architetturale (ad esempio, nella distinzione tra modelli autoregressivi, mascherati, ecc.) e capacità di adattamento ai differenti tipi di prompting. Da quest'ultima, in particolare, si possono individuare diverse tipologie di LLMs, progettate per adattarsi meglio a specifiche strategie di prompting:

- Modelli ottimizzati per Few-Shot e Zero-Shot Prompting, come GPT-4[51], LLaMA-3[39], Gemini[40], in grado di risolvere compiti del tutto nuovi, senza bisogno di fine-tuning e basandosi solo su pochi o nessun esempio.
  Alcuni studi[59] mostrano come i prompt selezionati in scenari Few-Shot generalizzino bene tra modelli di dimensioni simili, mentre la trasferibilità risulta essere scarsa tra modelli di dimensioni molto diverse.

- Modelli basati su Chain-of-Thought Prompting (CoT), come GPT-4-turbo, LLaMA-3 70B, capaci di eseguire ragionamenti più complessi quando viene esplicitato un ragionamento step-by-step nel prompt;

- Modelli per Instruction Tuning, come GPT-4-turbo, Mistral-Instruct[60], GPT-3.5-Instruct, LLaMA-3 8B-Instruct, addestrati su dataset con istruzioni specifiche e risposte ben strutturate.

3.1.2 Forma del Prompt

Esistono due principali tipologie di prompt, scelte in base al compito e al modello utilizzato per risolverlo:

- *Cloze Prompt*: il modello completa uno spazio vuoto in una stringa testuale. Vengono solitamente utilizzati con i modelli mascherati, in quanto riflettono meglio il formato utilizzato in fase di addestramento;
- *Prefix Prompt*: il modello continua una stringa prefissata. I modelli autoregressivi sono più adatti ai prefix prompt, poiché seguono una generazione sequenziale. Questa tipologia di prompt si predilige in tutti quei compiti per cui è prevista una generazione testuale.

In generale, un prompt può essere costituito da diversi elementi, variabili in base al compito specifico:

- Istruzione: il compito o la specifica istruzione specifica che il modello deve eseguire;
- Contesto: le informazioni aggiuntive che possono indirizzare il modello verso risposte migliori;
- Dati in input: l'input o la domanda per la quale il modello deve generare una risposta;
- Indicatore dell'output: l'informazione che specifica il formato o il tipo di risposta atteso.

3.1.3 Prompt Template Engineering

Il template di un prompt, definendo come il compito verrà presentato al modello, influisce notevolmente sulla qualità dell'output generato. La progettazione dei template può avvenire tramite tecniche automatizzate (*Automated Template Learning*) o gestite manualmente (*Manual Template Engineering*).

Nel *Manual Template Engineering* la creazione del prompt si basa unicamente sull'intuizione umana e su esperimenti empirici. Fra gli esempi realizzati manualmente ed ormai noti ci sono:

- il dataset LAMA (LAnguage Model Analysis)[61], che fornisce cloze prompt creati manualmente per valutare la conoscenza dei LLMs;

- Prefix prompt sviluppati manualmente per un'ampia gamma di compiti, tra cui domande e risposte, traduzione e ragionamento[62];
- Template predefiniti per il few-shot learning progettati per attività di classificazione del testo[63].

Per compiti più complessi, ottenere manualmente dei prompt davvero ottimali richiede molta sperimentazione, perciò sono state sviluppate alcune tecniche per automatizzare l'intero processo (*Automated Template Learning*).

I prompt generati automaticamente si possono distinguere in prompt discreti (*Hard Prompts*), rappresentati da stringhe testuali effettive, e prompt continui (*Soft Prompts*), rappresentati direttamente nello spazio degli embeddings del modello. Per generare prompt discreti esistono diversi metodi:

- *Prompt Mining*: si estraggono template da corpus testuali esistenti. Per creare un prompt si selezionano le stringhe contenenti un certo input x e un output y, e si identificano le parole nel mezzo (*middle words*), oppure i percorsi più frequenti che dall'input x giungono all'output y.
- *Prompt Paraphrasing*: partendo da un prompt di base, costruito manualmente o derivante dal Prompt Mining, vengono generate diverse varianti, scegliendo poi la più corretta per il compito target. Le varianti vengono costruite tramite back-translation (traducendo in un'altra lingua e poi ritornando alla lingua originale, per ottenere più parafrasi), o attraverso sostituzioni semantiche;
- *Prompt Generation*: la formulazione del prompt viene trattata come un compito di generazione testuale. Ad esempio, il modello T5 pre-addestrato[44] viene largamente utilizzato nel processo di ricerca dei template, sfruttando la sua capacità di riempire gli spazi vuoti.

I prompt continui, noti anche come *soft prompts*, non necessitano di essere leggibili dall'uomo, in quanto operano direttamente nello spazio degli embeddings del modello.
Le *word embeddings* del template non devono, dunque, essere necessariamente parole di un linguaggio naturale.

3.1.4 Prompt Answer Engineering

Oltre alla progettazione del prompt, è essenziale definire il formato della risposta attesa. A seconda del compito da eseguire, le risposte possono essere:

- Token singolo: utile nei compiti di classificazione, come nel *Sentiment Analysis*;
- Span: sequenze brevi di token, spesso utilizzate nei cloze prompt;
- Frase o paragrafo: comunemente usati con i prefix prompt, in quanto adatti a generazioni testuali o nel QA a scelta multipla[65].

Inoltre, si può decidere se generare un output in uno spazio vincolato (*constrained*) o privo di vincoli (*unconstrained*). Gli spazi vincolati vengono principalmente utilizzati per compiti con classi predefinite, come nella Sentiment Analysis, in cui si richiede una mappatura esplicita tra la risposta e la classe target. Negli spazi non vincolati, invece, la risposta può essere un qualsiasi token o frase.

3.2 Tecniche di Prompting

Per l'apprendimento basato su prompt possono essere adottate diverse strategie per interagire con i LLMs.

3.2.1 Zero-Shot Prompting

La tecnica *Zero-Shot* descrive la struttura di un prompt in cui non viene fornito in input alcun esempio esplicito del compito richiesto[66]. Un esempio tipico di prompt Zero-Shot consiste nel richiedere al modello di classificare un testo in positivo o negativo, senza fornire alcun esempio. Il LLM è in grado di comprendere il *sentiment* basandosi esclusivamente sulla sua conoscenza pregressa e sull'interpretazione del contesto. Differenti studi[67] hanno dimostrato che l'inserimento di istruzioni relative allo specifico task sono in grado di affinare l'apprendimento Zero-Shot.

3.2.2 Few-Shot Prompting

Nonostante i LLMs mostrino notevoli risultati con apprendimento su prompt Zero-Shot, vengono riscontrate alcune difficoltà nei compiti più complessi. Il Few-Shot Prompting, tecnica appartenente all'*In-Context Learning* (ICL), consente al modello di apprendere meglio il contesto, generando risposte più accurate. L'*In-Context Learning* permette, infatti, ai LLMs di apprendere un nuovo compito a partire da un piccolo insieme di esempi, inseriti a tempo di inferenza. I prompt di tipo Few-Shot forniscono al modello uno o più esempi espliciti (da cui il nome *1-shot, 3-shot, 5-shot,* ecc.)[63], che influenzeranno il modello nello specifico compito per cui è richiesta una risposta.

Nonostante i prompt di tipo Few-Shot performino molto bene in alcuni task, non sono sufficienti per problemi che richiedono ragionamenti più complessi.

3.2.3 Chain-of-Thought Prompting

Il *Chain-of-Thought* (CoT) prompting migliora significativamente la capacità di ragionamento dei LLMs, inducendoli a scomporre il problema in passaggi intermedi. Questo approccio è particolarmente efficace nei compiti che richiedono ragionamenti complessi, come problemi aritmetici, logici o simbolici Inoltre, risulta particolarmente utile nei Large Language Models molto grandi.

Un'idea emersa più di recente, proposta nel 2022 [68], si basa su *Zero-Shot CoT prompting*, che consiste nell'aggiungere una semplice istruzione come '*Let's think step by step.*' al prompt, senza alcun esempio dimostrativo. Questo metodo si è rivelato sorprendentemente efficace, nonostante l'assenza di esempi espliciti.

3.3 Linee guida

La creazione di prompt efficaci è un processo iterativo che richiede sperimentazione e ottimizzazione. Di seguito vengono riportati alcuni principi guida da seguire per la corretta formulazione di un prompt.

3.3.1 Specificità

Per ottenere risultati accurati occorre essere dettagliati e chiari nella formulazione della richiesta, evitando ambiguità e indicazioni vaghe. Più la richiesta è descrittiva, migliori saranno le risposte. Talvolta, però, si tende a voler essere troppo esaustivi nelle descrizioni, finendo coll'essere imprecisi. Nella creazione di un prompt, invece, più si è diretti, più il messaggio risulta efficace. Solitamente, si suggerisce di collocare le istruzioni all'inizio del prompt e, quando necessario, di includere esempi concreti che aiutino il modello ad una maggiore comprensione.

3.3.2 Lunghezza vs. Pertinenza

Nel costruire una richiesta efficace è necessario fornire sufficienti dettagli che contestualizzino il compito, ma senza, al contempo, sovraccaricare il prompt con informazioni irrilevanti. La lunghezza del prompt va quindi calibrata in base ai limiti del modello e all'obiettivo specifico.

3.3.3 Sperimentazione

La formulazione ottimale di un prompt richiede molta sperimentazione. Provare varie formulazioni, con parole chiave, istruzioni e contesti diversi, permette di perfezionare progressivamente il prompt e di adattarlo meglio allo specifico compito. Ogni variante possibile deve essere poi valutata nei risultati e progressivamente perfezionata, così da trovare la formulazione più efficace.

3.4 Limiti e Sfide

L'apprendimento basato su prompt ha dimostrato un potenziale significativo in diversi ambiti e differenti task, ma presenta ancora numerose sfide. Una delle principali difficoltà riguarda la scelta del modello più adatto tra i numerosi LLMs pre-addestrati disponibili. Selezionare il modello ottimale per un compito specifico è un problema complesso e computazionalmente dispendioso, che richiede un'attenta valutazione delle prestazioni e delle caratteristiche di ciascun Large Language Model.

Un altro aspetto cruciale è la corretta progettazione dei prompt. Per ottenere risultati efficaci, i prompt devono essere ottimizzati in funzione del compito e del determinato output che si vuole ottenere. Molti problemi nel Natural Language Processing coinvolgono strutture dati specifiche, come alberi, grafi, tabelle. Rappresentare in modo efficace queste strutture all'interno dei prompt richiede template e formattazioni adeguate, rappresentando, dunque, una sfida importante.

Infine, un ulteriore problema riguarda sicuramente la trasferibilità di questo approccio, ovvero capire quanto un prompt, efficace per uno specifico modello e dominio, sia trasferibile ed utilizzabile su altri modelli ed ambiti. Prompt performanti e generalizzabili tra modelli di simili dimensioni, hanno infatti, una scarsa trasferibilità con modelli di dimensioni estremamente diverse.

Capitolo 4

Materiali e Metodi

Il problema del Word Sense Disambiguation viene affrontato, in questo progetto, mediante l'uso esclusivo di LLMs con tecniche di prompt inference, senza alcun processo di fine-tuning. I LLMs, grazie alla loro capacità intrinseca di comprendere i significati delle parole, risultano essere ottimi strumenti nella disambiguazione dei significati delle parole. Sebbene il fine-tuning possa migliorare ulteriormente le prestazioni del modello, l'obiettivo di questo studio è stato dimostrare come, attraverso una formulazione ottimizzata del prompt, sia possibile ottenere risultati competitivi, senza alcuna fase di pre-addestramento. Studi recenti hanno evidenziato l'efficacia di questo approccio, dimostrando come tecniche basate su LLMs possano raggiungere, e in alcuni casi superare, le prestazioni di sistemi supervisionati per il WSD [69]. Analogamente, è stato dimostrato che l'*in-context learning* basato su LLMs risulta particolarmente efficace per compiti di classificazione nel WSD [70]. Dato, dunque, un termine specifico all'interno della sua frase di contesto, il modello valuta se un determinato significato sia compatibile con l'uso di quella parola ambigua nella frase. L'ottimizzazione del prompt è stata realizzata attraverso un processo iterativo, partendo da una formulazione iniziale e perfezionandola progressivamente mediante test successivi. Sono state sperimentate diverse strategie di prompting per migliorare la capacità del modello di identificare con precisione il significato corretto delle parole target. L'adozione di modelli differenti, i numerosi test di affinamento del prompt e l'integrazione di più tecniche di prompting hanno permesso di ottenere una disambiguazione efficace, basata esclusivamente sulle capacità di inferenza dei LLMs.

4.1 Ambiente di sviluppo

La fase di progettazione iniziale, relativa alle operazioni di pre-processing dei dati e alla corretta estrazione di tutti i significati possibili da WordNet, è stata eseguita su Google Colab Pro. Con esso, è stato possibile usufruire di potenza di calcolo adeguata grazie alla presenza di una GPU T4 con elevata capacità di memoria.

Le sperimentazioni relative, invece, alla fase di prompting sono state condotte grazie al cluster GPU Nvidia del Dipartimento di Informatica - Scienza e Ingegneria (DISI) dell'Università di Bologna. L'accesso al cluster permette di eseguire job computazionali attraverso lo schedulatore SLURM [71], consentendo una distribuzione efficiente delle richieste di calcolo sui nodi disponibili.

4.2 Dataset

Per la valutazione del modello sono stati utilizzati i cinque dataset standard per il WSD, introdotti da Raganato et al.[16] e provenienti dalle competizioni Senseval e SemEval:

- Senseval-2 (SE2)
- Senseval-3 (SE3)
- SemEval-2007 (SE07)
- SemEval-2013 (SE13)
- SemEval-2015 (SE15)

Questi dataset rappresentano un riferimento ormai consolidato per la valutazione di compiti di WSD. La loro formattazione standardizzata permette un confronto equo tra modelli e metodi diversi di NLP, in quanto condividono una struttura simile, costituita da:

- Frasi con parole target da disambiguare;
- Annotazioni manuali con i sensi provenienti da WordNet;
- Testi provenienti da fonti diverse per garantire una maggiore varietà linguistica.

Il dataset SemEval-2007 è stato utilizzato come validation set per ottimizzare il prompt, in quanto è il più piccolo tra quelli considerati (455 annotazioni di senso per sostantivi e verbi [72]. Una volta individuata la configurazione ottimale del prompt, essa è stata testata sugli altri quattro dataset.

4.2.1 Elaborazione Dataset

Nella fase iniziale del processo è stata effettuata l'elaborazione dei dataset di riferimento da cui estrarre i dati da utilizzare per il WSD. Ogni dataset dei cinque di input è affiancato dal corrispettivo *gold* dataset. Nei dataset di input sono presenti le frasi annotate, contenenti le parole target da disambiguare. Nei corrispettivi gold dataset, invece, sono riportate le coppie composte dall'ID del termine ambiguo e dal senso corretto associato. Per ogni frase annotata dei dataset di input sono stati estratti:

- `id`: ogni dataset si compone di testi. Per ogni testo ci sono più frasi. Dunque, per ogni frase una o più parole ambigue. L'`id` permette di etichettare univocamente la specifica *word*, di una specifica *sentence*, all'interno di uno specifico *text*;
- `target_word`: il termine specifico da disambiguare. Ogni parola target è stata evidenziata tra i tag `<WSD>` e `</WSD>` per facilitarne l'identificazione nel prompt.
- `words_sentence_list`: lista delle parole contenute nella frase, che vanno a formare il contesto da associare nel prompt a ciascun termine ambiguo;
- `token_length_list`: lunghezza in termini di token di ciascuna parola nella frase.

I corrispondenti gold file forniscono, per ogni `id` di parola target, il senso corretto secondo WordNet, con annotazioni manuali validate da esperti lessicografi. Queste informazioni sono state estratte ed organizzate in dizionari `gold_senses`, in cui le chiavi corrispondono agli `id` delle parole ambigue ed i valori alle liste corrispondenti con i sensi corretti. Per alcuni termini, infatti, sono presenti più di un significato possibile.

4.2.2 Estrazione dei Sensi da WordNet

L'estrazione dei possibili significati delle parole è stata effettuata interrogando WordNet 3.0 tramite la libreria `nltk` [73]. Ogni parola target è stata trasformata nella sua forma base, il lemma, ed associata al corrispondente POS tag, ovvero il ruolo della parola nel discorso (sostantivo, verbo, aggettivo o avverbio). WordNet è stato, quindi, interrogato per estrarre, dato il lemma ed il POS tag, tutti i sensi possibili di ciascuna parola. Nello specifico, da ogni lemma sono state raccolte le seguenti informazioni:

- Definizione del senso (*Gloss*);
- Frequenza d'uso del lemma;

- Esempio di utilizzo del senso (se disponibile).
  Gli esempi d'uso sono stati valutati secondo due approcci distinti. Inizialmente, sono state utilizzate frasi di esempio fornite direttamente da WordNet. Nel secondo approccio, per le parole in comune con il dataset specifico utilizzato ogni volta, sono stati estratti esempi dal dataset SemCor 3.0[13].

I dati estratti da WordNet sono stati strutturati in un dizionario in cui ogni lemma, identificato dalla propria chiave, è associato alle informazioni semantiche appena descritte.

4.3 Tecniche di Prompting

Il processo di formulazione del prompt ha richiesto la sperimentazione di numerose varianti, testate su diversi modelli e con differenti metodologie. L'obiettivo principale è stato quello di migliorare la struttura del prompt sia in termini di scelta delle parole, chiarezza e specificità delle istruzioni, sia attraverso la sperimentazione di diverse strategie di prompt inference. Per questo studio sono stati selezionati e sperimentati diversi LLMs pre-addestrati appartenenti alla famiglia LLaMA. L'accesso ai modelli LLaMA è stato reso possibile previa autorizzazione tramite la piattaforma Hugging Face [54], utilizzando un token di autenticazione rilasciato da Meta a seguito della richiesta di utilizzo. I modelli sono stati valutati in base alla loro capacità di assegnare il senso corretto alle parole ambigue presenti nei dataset di riferimento.

Inizialmente, la ricerca del prompt ottimale è stata condotta sui modelli Chat di LLaMA 2, in particolare sulla versione a 7 miliardi di parametri (LLaMA2-7b)[74]. Come punto di partenza, è stato adottato un formato di prompt ispirato al modello Alpaca, un LLM sviluppato proprio a partire dal modello LLaMA 7b fine-tuned [75]. Il template di questo prompt si compone di tre sezioni principali:

- `###Instruction:` contenente l'istruzione primaria per il modello;
- `###Input:` fornisce i dati specifici su cui il modello deve lavorare;
- `###Response:` indica il formato atteso per la risposta.

Di seguito, uno dei primi prompt usati con ZeroShot prompting:

```
### Instruction:
You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense_number for an ambiguous word based
on the context and the provided list of possible meanings.
Always respond in the following format: "Sense Number: <sense_number>"

If you are unsure, always choose the closest matching sense from the list.

### Input:
Context: "{context}".
Target word: "{target_word}".
Possible senses list for "{target_word}": {possible_meanings}

### Response:
```

Successivamente, per ottenere risposte più precise, è stato adottato lo specifico template utilizzato dal modello LLaMA 2 in fase di addestramento. Tale formato impiega dei tag specifici per distinguere le istruzioni di sistema da quelle fornite dall'utente e dalla risposta generata dal modello. Seguire un formato già noto al modello, per via del pre-training, è cruciale per garantire che il modello interpreti correttamente le istruzioni, evitando comportamenti imprevedibili. Il prompt per modelli LLaMA di tipo Chat segue la seguente struttura:

```
[INST]
<<SYS>> {{ system_prompt }} <</SYS>>
{{ user_message }}
[/INST]
```

Un esempio concreto di prompt utilizzato, conforme a questo standard, è il seguente:

```
<s>[INST]
«SYS» You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense for an ambiguous word based on the
context and the provided list of possible senses.«/SYS»

This is the context: {context}
This is the ambiguous word: {target_word}
Possible senses for {target_word} are: {possible_meanings}
Choose the most appropriate sense_number listed above.
Always respond in the following format: "Sense Number: <sense_number>"
[/INST]
```

Per ulteriori miglioramenti, sono state aggiunte descrizioni dei singoli campi per ogni significato.

```
Below are the possible senses for the target word.
Each sense includes:
- A brief description of the sense.
- The number of times this sense of the target word appears in a large corpus
of texts. Higher frequency suggests the sense is more commonly used, while
lower frequency indicates a rarer or specialized use of the word.
- An example of sentence illustrating the use of the sense.
```

Durante la progettazione e il testing dei prompt è stato possibile interagire direttamente con i modelli LLMs, configurando specifici parametri per influenzare la qualità delle risposte ottenute. L'ottimizzazione di tali parametri ha richiesto un'attenta fase di sperimentazione per individuare i valori più adatti allo specifico caso d'uso. I principali parametri adottati sono:

- *Temperature*: determina il grado di casualità nelle risposte del modello. Valori bassi favoriscono risposte più deterministiche, selezionando il token più probabile ad ogni passo. Valori più alti, al contrario, aumentano la diversità delle risposte, risultando utili per compiti creativi.

- *Top_p*: limita la scelta del modello a un sottoinsieme di token la cui probabilità cumulativa non supera una soglia p, predefinita. Invece di considerare sempre un numero fisso di token candidati, il modello sceglie in maniera dinamica tra i token più probabili, adattandosi al contesto. Valori bassi inducono il modello a scegliere solo tra token più probabili, producendo risposte più sicure e coerenti, mentre valori alti favoriscono variabilità e creatività.

- *Top_k*: limita la scelta del modello ai k token più probabili, questa volta indipendentemente dalla loro probabilità cumulativa. Valori bassi riducono la creatività del modello, rendendo le risposte più prevedibili, mentre valori alti aumentano la varietà, ma potrebbero ridurre la coerenza complessiva del testo generato.

- *Max New Tokens*: definisce il numero massimo di token generabili in una risposta. Nel progetto è stato selezionato un valore alto per permettere al modello di sviluppare un ragionamento completo nella selezione del senso corretto, soprattutto nei casi più ambigui con numerosi significati possibili;

- *Repetition Penalty*: permette di ridurre la ripetitività e favorire una maggiore diversità nei testi generati. Un token generato in precedenza, avrà minori probabilità di essere nuovamente selezionato dal modello. Un valore basso permette meno ripetizioni e una maggiore variabilità nei testi. Un valore troppo alto potrebbe provocare una possibile perdita di coerenza, portando il modello ad evitare parole chiave importanti.

Dopo aver raggiunto un limite nelle performance con modelli LLaMA Chat, che risultavano decisamente inferiori allo stato dell'arte riportato nel Capitolo 1, si è passati ai modelli LLaMA-3 Instruct [76], addestrati specificamente per seguire istruzioni in linguaggio naturale. Questi modelli, progettati per interazioni *multi-turn* tra utente e modello, supportano principalmente tre ruoli distinti:

- `System`: definisce il contesto e le linee guida per l'interazione con il modello, stabilendo regole e informazioni essenziali per una risposta adeguata;

- `User`: rappresenta l'utente umano che interagisce con il modello, fornendo input, comandi e domande;

- `Assistant`: identifica la risposta generata dal modello sulla base del contesto fornito da User e System.

Selezionato, dunque, il modello LLaMA-3.1 Instruct, con 8 miliardi di parametri(LLaMA3.1-8b), sono state testate diverse tecniche di prompting.

Zero-Shot prompting:
inizialmente, il modello riceve soltanto la parola da disambiguare nel suo contesto e l'elenco dei possibili significati estratti da WordNet;

```
system_prompt = You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense for the target word based on the
context.

user_prompt = Context: "{context}". Target word: "{target_word}".
Possible senses list for "{target_word}": {possible_meanings}

Return JSON object that contains the finalized sense_number.
Use the following format for the output.

<JSON Object with ambiguity word and the sense_number>
```

Figura 4.1: Zero-Shot prompting

Few-Shot prompting:
sono stati introdotti uno o più esempi dettagliati per guidare il modello nel processo decisionale. In particolare, si forniscono esempi di disambiguazione di parole ipotetiche, con il relativo contesto e la risposta corretta, seguiti poi dal caso reale;

```
system_prompt = You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense for the target word based on the
context.

user_prompt = Example 1
Context:  "Employees in this company enjoy several health benefits and a
generous retirement plan."
Target word:  "enjoy"
Possible senses for "enjoy":
- sense_number 1:  "derive or receive pleasure from; get enjoyment from; take
pleasure in"
Frequency:  52
Example:  "She relished her fame and basked in her glory."
- sense_number 2:  "have benefit from"
Frequency:  15
Example:  "enjoy privileges"
- sense_number 3:  "get pleasure from"
Frequency:  10
Example:  "I love cooking."
- sense_number 4:  "have for one's benefit"
Frequency:  9
Example:  "The industry enjoyed a boom."
- sense_number 5:  "take delight in"
Frequency:  6
Example:  "he delights in his granddaughter"

Response:<ambiguity word:  enjoy sense_number:2>

Following the same format above from example:
Context:  "{context}".
Target word:  "{target_word}".
Possible senses list for "{target_word}":  {possible_meanings}

Return JSON object that contains the finalized sense_number.
Use the following format for the output.
<JSON Object with ambiguity word and the sense_number>
```

Figura 4.2: Few-Shot prompting

Zero-Shot CoT (Chain-of-Thought) prompting:
prima di un vero e proprio ragionamento esplicito, è stato testato il modello con una semplice istruzione ("`Let's think step by step.`"), incentivandolo a ragionare passo dopo passo.

```
system_prompt = You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense for the target word based on the
context.

user_prompt = Context: "{context}". Target word: "{target_word}".
Possible senses list for "{target_word}": {possible_meanings}

Return JSON object that contains the finalized sense_number.
Use the following format for the output.

<JSON Object with ambiguity word and the sense_number>
Let's think step by step.
```

Figura 4.3: Zero-Shot CoT prompting

Chain-of-Thought prompting:
l'approccio più efficace si è rivelato quello che esplicita il ragionamento del modello, chiedendogli di analizzare il contesto, le parole chiave circostanti e l'intento della frase. Inoltre, come si nota dai prompt riportati, con il modello LLaMA3.1-8b Instruct è stata ottimizzata la formulazione della risposta generata dal modello. Inizialmente veniva richiesta una semplice risposta testuale, nel formato
`sense_number: <number>`.
Successivamente, invece, è stato adottato un formato standardizzato, imponendo la restituzione della risposta all'interno di un oggetto JSON. Questo ha garantito una maggiore affidabilità nell'estrazione dei dati, riducendo gli errori di interpretazione, salvo i rari casi in cui il modello generava risposte cicliche a causa di livelli di confidenza simili tra più sensi. Di seguito, è riportato il prompt di Chain-of-Thought ispirato ad uno studio di recente pubblicazione, in cui viene presentato il sistema GlossGPT[77]: un modello basato su GPT-3.5 Turbo e GPT-4 Turbo che permette di raggiungere elevate performance tramite sole tecniche di prompt inference.

```
system_prompt = You are a linguistic expert in word sense disambiguation (WSD).
Your task is to identify the correct sense for the target word based on the
context.

user_prompt = Do the following tasks.
1.  word has different meanings.  Below are possible meanings.
Comprehend the meanings.  "{possible_meanings}"
2.  You can learn more on the usage of each word and the meaning through below
Examples.  Examples are:  "{examples}".
3.  Analyze the sentence below using the following techniques and identify the
most suitable meaning of the ambiguous word.
"{context}"
- Focus on keywords in the sentence surrounding the ambiguous word.
- Think about the overall topic and intent of the sentence.
4.  Choose the sense_number that makes the most logical sense within the
context.
"{possible_meanings}"
5.  Return JSON object that contains the ambiguity word and the finalized
sense_number.  Use the following format for the output.
<JSON Object with ambiguity word and the sense_number>
```

Figura 4.4: Chain-of-Thought prompting

Infine, nuovamente sulla base dello studio sopra citato[77], è stato adottato un ulteriore metodo mirato alla gestione dei casi più complessi, ossia quelli in cui la parola ambigua poteva avere numerose interpretazioni. Questo nuovo approccio, indicato in questo progetto come *CoT-Verified Prompting*, si basa su un'ulteriore verifica da parte del modello, che deve rivalutare il significato precedentemente scelto, confermando o meno la propria decisione. Inizialmente, ogni parola ambigua viene analizzata singolarmente dal modello tramite CoT prompting, come nel caso precedente. Dopodiché le predizioni corrette vengono distinte da quelle errate, molto spesso coincidenti con le parole più ambigue, ovvero quelle con molti significati possibili. Questi casi problematici vengono ora nuovamente sottoposti al modello, con un prompt di Verifica che chiede di confermare o correggere la scelta iniziale.

```
system_prompt = Verify if the selected sense is correct for the ambiguous word
in the given sentence.  Respond with a JSON object containing the target word
and your answer ('yes' or 'no').

user_prompt = Do the following tasks.
1.Examine the sentence below.  '{context}'
2.  You are going to identify if the meaning below is suitable for
'{target_word}'.
Proposed sense:  '{predicted_sense}'
(Meaning:  {sense_definition})
3.  Is the proposed sense correct for the target word in this sentence?
Respond in JSON format with the following structure:
{{
"target_word":  "{target_word}",
"answer":  "yes" or "no"
}}
```

Figura 4.5: Prompt di Verifica del senso selezionato dal modello

Se il modello dubita della scelta iniziale, il senso selezionato viene escluso dallo spazio dei significati possibili e il prompt CoT viene riproposto, per quel termine, senza di esso.

Una volta ottimizzati template, ordine delle istruzioni, struttura del prompt e selezionata la tipologia di modello più efficace, la fase di validazione si può considerare conclusa. Individuato il migliore prompt, viene sottoposto ai quattro corpora distinti di test, effettuando una duplice valutazione: la prima basata su esempi d'uso, dei diversi significati, estratti da WordNet, la seconda su esempi d'uso provenienti da SemCor, limitatamente alle parole in comune in termini di lemma e categoria grammaticale (POS tag).

Capitolo 5

Analisi e Risultati

Questo studio si propone di valutare l'efficacia del Prompt Engineering, combinato con diverse tecniche computazionali, per il compito di Word Sense Disambiguation. Studi precedenti hanno evidenziato come l'uso di Large Language Models per il WSD sia risultato particolarmente efficace, soprattutto quando combinato con strategie di ottimizzazione dei parametri e miglioramento dei prompt [70], basato su ChatGPT. L'obiettivo principale di questo progetto è stato, quindi, sviluppare un prompt ottimizzato per migliorare l'accuratezza nella disambiguazione del senso delle parole, utilizzando modelli open-source come LLaMA-2-7b e LLaMA-3.1-8b.

I modelli utilizzati non sono stati addestrati specificamente per il compito di WSD. Si è voluto infatti dimostrare che, attraverso prompt accuratamente progettati e differenti tecniche di affinamento, si possono ottenere risultati comparabili a quelli SOTA, senza la necessità di fine-tuning o di un addestramento specifico per il task. Questo conferma che anche modelli open-source di dimensioni ridotte possono ottenere prestazioni competitive rispetto a modelli di grande scala, nonostante il minor numero di parametri.

I cinque dataset selezionati, che compongono il UEF proposto da Raganato et al.[16], rappresentano il benchmark di riferimento per il WSD. Questi dataset presentano il problema della disambiguazione come un compito di classificazione, in quanto il modello deve identificare il senso corretto di una parola target all'interno di una frase, data una lista di significati possibili. Per questo progetto il dataset SemEval-2007 è stato utilizzato come validation set, mentre gli altri quattro dataset Senseval/SemEval come test set.

Dopo l'estrazione di ogni termine da disambiguare, e i relativi significati provenienti da WordNet, sono stati sperimentati i differenti prompt riportati nel capitolo 4. La fase di sperimentazione ha permesso di identificare il prompt ottimale per estrarre la `sense key` associata alla parola target ambigua dato un contesto. L'intero processo ha coinvolto numerosi template, differenti tipologie di istruzioni con cui guidare i modelli e molteplici

tecniche di prompt engineering per sviluppare il miglior prompt possibile. Il prompt è stato perfezionato iterativamente, analizzando sistematicamente le predizioni errate per migliorarne la formulazione e ottenere risultati ottimali.

Il primo modello testato, LLaMA-2 Chat con 7 miliardi di parametri, si è rivelato efficace nel rispondere a domande a scelta multipla, mostrando, però, difficoltà nell'estrazione standardizzata delle informazioni. Le risposte fornite erano spesso variabili, rendendo complessa l'automatizzazione del processo di disambiguazione. In particolare, il modello riscontrava numerose difficoltà con termini target con molteplici significati. Sono state sperimentate diverse strategie per affinare il prompt e migliorare le performance, indicate in tabella 5.1 come:

- `Alpaca template`: formato di partenza che segue la struttura del prompt nel modello Alpaca;
- `Chat Template`: si utilizza la struttura adottata dai modelli LLaMA Chat durante le fasi di pre-training;
- `Chat template + WNsenses description`: vengono aggiunte descrizioni dettagliate dei singoli campi di ogni senso. Spiegano al modello come interpretare la definizione, la frequenza d'uso e l'esempio di ogni significato;
- `Chat template + Parameters optimization`: interagendo direttamente con i parametri specifici tramite cui il modello genera le risposte, è stata ulteriormente incrementata la capacità del LLM di comprendere il significato corretto data la parola ambigua.

I parametri di generazione, dopo molteplici test, sono stati fissati a:

- `max_new_tokens=1024`
- `temperature = 0.6`
- `top_p = 0.8`
- `top_k = 30`

I risultati ottenuti con LLaMA-2-7b Chat sono riportati nella tabella 5.1, che mostra un miglioramento progressivo delle performance. I valori indicano l'accuratezza ottenuta confrontando gli output del modello LLaMA-2-7b Chat con le *gold sense keys* corrette presenti nel dataset SemEval-2007. Per ogni metodo sono state effettuate due valutazioni, differenziate in base agli esempi d'uso utilizzati. Per ciascun significato, infatti, gli esempi

d'uso sono stati estratti da WordNet oppure, per parole con significati in comune, dal dataset Semcor 3.0. I risultati sono stati misurati utilizzando la metrica F1-score.

	Prompt template	WordNet e.g.	SemCor e.g.
Zero-Shot prompting	Alpaca template	37.6	39.6
	Chat template	41.5	**43.5**
	Chat template + WNsenses description	46.8	43.3
	Chat template + Parameters optimization	**47.3**	40.9

Tabella 5.1: Risultati per SemEval-2007 con LLaMA-2-7b Chat

In seguito a diversi test e ottimizzazioni, si è deciso di adottare modelli della serie LLaMA 3 di tipo `Instruct`. Questi modelli sono risultati ampiamente più efficaci nel comprendere le istruzioni fornite e nel completare correttamente il task di WSD assegnato. LLaMA-3.1-8b Instruct, un modello open-source con avanzate capacità di comprensione contestuale, è stato impiegato in tutte le modalità di prompt inference riportate nel capitolo 4. I risultati migliori ottenuti con SemEval-2007, per ciascuna tecnica di prompt engineering selezionata, sono riportati nella Tabella 5.2.

Tecniche di Prompting	WordNet e.g.	SemCor e.g.
Zero-Shot	64.4	65.3
Few-1-Shot	63.3	63.7
Zero-Shot CoT	66.2	64.8
Chain-of-Thought	65.5	64.2

Tabella 5.2: Risultati per SemEval-2007 con LLaMA-3.1-8b Instruct

Nonostante i progressi ottenuti, le tecniche precedenti presentano ancora numerosi limiti nella gestione delle parole ambigue meno frequenti e di quelle con un'ampia gamma di significati possibili. Senza infatti, una comprensione approfondita di ciascun senso, i modelli LLaMA, di queste dimensioni, non sempre riescono a disambiguare correttamente i termini target.

Ispirato, dunque, ad uno studio recentemente pubblicato [77], viene introdotta la tecnica del *CoT-Verified Prompting*. Dopo aver analizzato iterativamente ogni singola parola target tramite CoT prompting, si analizzano tutte le predizioni errate, casi limite di ciascun dataset. Per questi viene chiesto al modello, tramite un prompt di verifica,

se conferma o contesta il significato scelto. Grazie a questa strategia, è stato possibile ottenere un miglioramento complessivo del 13.3%, migliorando così l'accuratezza delle risposte nei casi più complessi.

I risultati successivi mostrano un'analisi dettagliata delle prestazioni del modello LLaMA-3.1-8b Instruct su tutti i dataset di Unified Evaluation Framework. Le performance sono state calcolate confrontando le predizioni corrette con il numero totale di istanze target da disambiguare. Per ciascun dataset viene riportato il confronto tra i risultati ottenuti con Chain-of-Thought prompting e con *CoT-Verified Prompting*. Questa fase finale è stata condotta utilizzando frasi d'esempio per ciascun significato interamente estratte da WordNet.

Prompting	**SemEval-2007**	**Senseval-2**	**Senseval-3**	**SemEval-13**	**SemEval-15**
CoT	65.5	71.0	68.2	74.5	77.8
CoT-Verified	78.0	81.3	79.2	84.5	87.0

Tabella 5.3: Risultati per UEF con CoT prompting e CoT-Verified prompting

	SemEval-2007	**Senseval-2**	**Senseval-3**	**SemEval-13**	**SemEval-15**
Parole target	455	2282	1850	1644	1022
Casi limite	161	591	521	354	191
Run time	2h 21	9h 59	8h 22	6h 50	4h 22

Tabella 5.4: Informazioni per UEF con CoT_Verified prompting

L'approccio *CoT-Verified Prompting* ha raggiunto:

- 78% di F1-score sul validation set SemEval-2007
- 83% di F1-score sulla media fra i restanti test set

5.1 Confronto con lo Stato dell'Arte

In Tabella 5.5 vengono ora presentate le performance di UEF rispetto ai sistemi che compongono lo stato dell'arte, con un confronto tra i risultati esistenti nel campo del Word Sense Disambiguation e quelli ottenuti in questo progetto. I risultati dei modelli mostrati utilizzano anch'essi il dataset SemEval-2007 come set di validazione, ed i restanti di UEF come test set. Questo garantisce un confronto reale ed affidabile tra i differenti sistemi SOTA esistenti.

Sistemi	SemEval-2007	Senseval-2	Senseval-3	SemEval-13	SemEval-15
Lesk	32.0	50.6	44.5	53.6	51.0
Lesk+emb	56.7	63.0	63.7	66.2	64.6
UKB	39.0	56.0	51.7	53.6	55.2
UKB-g	42.0	60.6	54.1	59.0	61.2
Babelfy	51.6	67.0	63.5	66.4	70.3
WN 1st sense	55.2	66.8	66.2	63.0	67.8
MFS	54.5	65.6	66.0	63.8	67.1
IMS	61.3	70.9	69.3	65.3	69.5
IMS+emb	60.9	71.0	69.3	67.3	71.3
IMS-s+emb	62.6	72.2	70.4	65.9	71.5
Context2Vec	61.3	71.8	69.1	65.6	71.9
GlossBERT	72.5	77.7	75.2	76.1	80.4
GlossGPT	76.2	**86.1**	**82.9**	75.4	83.0
ESCHER	76.3	81.7	77.8	82.2	83.2
ConSeC	**77.4**	82.3	79.9	**83.2**	**85.2**
This work$_{\text{best}}$	***78.0***	*81.3*	*79.2*	***84.5***	***87.0***

Tabella 5.5: Confronto dei risultati del metodo proposto con i sistemi per il WSD SOTA

I valori in grassetto indicano i migliori risultati SOTA, mentre quelli in corsivo nell'ultima riga rappresentano le performance migliori ottenute in questo progetto. Il metodo di *CoT-Verified Prompting* proposto ha ottenuto ottimi risultati nei dataset SemEval-13 e SemEval-15, superando diversi modelli supervisionati SOTA. I risultati dimostrano come l'uso di LLMs per il WSD, combinato con tecniche avanzate di Prompt Engineering, possa raggiungere prestazioni paragonabili a quelle dei modelli SOTA, senza la necessità di fine-tuning.

É, inoltre, da considerare che il metodo utilizzato è stato sviluppato e testato su modelli LLaMA con pochi miliardi di parametri. Se confrontato con sistemi più avanzati, come GlossGPT da cui è derivato l'approccio *CoT-Verified Prompting*, sperimentato su modelli GPT con oltre un centinaio di miliardi di parametri, il risultato ottenuto si può considerare promettente. Utilizzare definizioni precise per ogni possibile significato di una parola target, accompagnate dalla relativa frequenza d'uso e da frasi di esempio, hanno condotto la ricerca verso nuove possibilità per il WSD, senza dover addestrare il sistema specificamente per quel task. Inoltre, l'integrazione di un prompt di verifica, che chiede al modello di confermare o rivalutare la propria scelta, contribuisce a perfezionare ulteriormente la selezione del senso corretto.

Tuttavia, questa strategia presenta delle limitazioni: richiede un tempo di generazione delle risposte considerevole, risultando poco adatta per applicazioni in tempo reale. No-

nostante ciò, i miglioramenti osservati indicano una direzione promettente per la ricerca futura nel campo della disambiguazione del significato delle parole.

Conclusioni

In questo progetto di Tesi è stato presentato il Word Sense Disambiguation, il processo che permette di identificare il significato corretto di una parola target nel determinato contesto in cui appare. Il WSD rappresenta una sfida cruciale nel campo del Natural Language Processing. Nel corso di questa ricerca, è stato evidenziato come il problema della polisemia sia stato affrontato nel tempo con differenti approcci; metodi supervisionati, knowledge-based, ibridi, fino all'impiego di Large Language Models che hanno segnato un punto di svolta significativo nel settore.

La disambiguazione del significato delle parole non coinvolge solo la conoscenza linguistica, basata sulle relazioni sintattiche, ma richiede anche una competenza funzionale, ovvero la capacità di integrare e correlare tra loro informazioni provenienti da più risorse. Sebbene i LLMs eccellano nella competenza formale, non raggiungono ancora livelli pari a quelli umani nella competenza funzionale. L'utilizzo di LLMs nel WSD offre, dunque, un metodo ulteriore per analizzare e valutare le capacità di ragionamento di questi sistemi.

Nella prima parte della Tesi è stato introdotto nel dettaglio il Word Sense Disambiguation, delineando i concetti fondamentali e fornendo una panoramica completa delle metodologie esistenti per affrontare questo problema. Uno degli aspetti più rilevanti riguarda, infatti, la scelta dell'approccio più adatto. I metodi supervisionati, sebbene abbiano dimostrato elevate performance in termini di accuratezza, dipendono fortemente da corpora annotati manualmente, la cui creazione, spesso onerosa, rappresenta un ostacolo significativo. I metodi knowledge-based, invece, organizzano la conoscenza di importanti ontologie lessicali in reti semantiche molto estese, dove i nodi rappresentano le parole e gli archi le loro relazioni semantiche. In questa struttura a grafo la possibilità che due termini, i nodi del grafo, si trovino nello stesso contesto è rappresentata dalle relazioni tra essi, ovvero dagli archi. WordNet, da cui sono state estratte le definizioni, i significati e gli esempi d'uso di ciascuna parola disambiguata in questo progetto, rappresenta una delle risorse lessicali più note nel WSD.

Con l'introduzione dei Large Language Models, presentati nel Capitolo 2, il panorama

del WSD è stato enormemente rivoluzionato. Questi modelli, addestrati su enormi quantità di dati, hanno dimostrato notevoli capacità di comprensione semantica e contestuale, superando molte delle limitazioni dei metodi supervisionati tradizionali. Tuttavia, presentano anch'essi alcune problematiche, come la stretta dipendenza dai dati iniziali di training e la necessità di fine-tuning per poter ottenere prestazioni avanzate in domini specifici. Nonostante le elevate performance, i LLMs presentano ancora delle difficoltà nella gestione di parole con significati rari o assenti nei dataset di training.

Con questo studio si vuole, quindi, dimostrare come l'impiego di tecniche di Prompt Engineering possano guidare il comportamento dei LLMs nel Word Sense Disambiguation. Tramite l'uso esclusivo di strategie di Prompt Inference, in seguito a numerose sperimentazioni, è stato possibile ottenere risultati simili a quelli di sistemi SOTA, ma senza la necessità di costose fasi di riaddestramento dei modelli.

In particolare, l'adozione di prompt molto strutturati, come nel Chain-of-Thought Prompting, ha significativamente migliorato la qualità della disambiguazione, spingendo il modello ad esplicitare i processi di ragionamento per ciascun termine target. In questi casi, l'efficacia del WSD è risultata strettamente legata alla precisione delle istruzioni fornite nel prompt, che hanno guidato l'attenzione del modello verso le informazioni più rilevanti, come il tono della frase, l'intento del contesto e la comprensione dettagliata dei termini circostanti ogni parola target.

Tuttavia, in casi di elevata ambiguità, l'approccio basato su prompt ha mostrato alcune limitazioni, rendendo necessaria l'introduzione di un'ulteriore fase di verifica. Il metodo di *CoT-Verified Prompting* sviluppato ha permesso di ottenere risultati significativamente migliori. Con esso, i casi limite sono stati gestiti attraverso un meccanismo iterativo di verifica basato su classificazione binaria, in cui il modello doveva confermare o rivalutare le proprie scelte sul senso selezionato. Nonostante i risultati ottenuti superino alcuni dei sistemi supervisionati che compongono l'attuale stato dell'arte nel WSD, sono sicuramente necessari ulteriori studi per affrontare la gestione di parole altamente ambigue.

Un aspetto critico di questa ricerca è stato rappresentato dalle limitazioni legate alle risorse computazionali. Non è stato, infatti, possibile utilizzare Large Language Models con un elevato numero di parametri sulle macchine di cui si disponevano. È probabile che un aumento del numero di parametri nei modelli LLaMA utilizzati avrebbe migliorato ulteriormente la comprensione contestuale di ciascuna parola target ambigua. Studi precedenti, come quello da cui è ispirato il metodo di *CoT-Verified Prompting*, hanno dimostrato che modelli come GPT-3 Turbo, con 175 miliardi di parametri, ottengono risultati estremamente avanzati nelle medesime sperimentazioni.

Considerato questo limite, anziché concentrarsi esclusivamente sull'esplorazione del-

l'impatto di modelli LLMs di dimensioni maggiori, si è preferito indirizzare la ricerca sulla sperimentazione di più tecniche di prompting, svariate tipologie di istruzioni e sul test di differenti parametri di generazione delle risposte. Operando con risorse limitate, i risultati ottenuti potrebbero fornire una base di riferimento per studi futuri in cui si dispongano di risorse computazionali ridotte, consentendo di replicare ed estendere la presente ricerca. Al contempo, l'approccio proposto apre anche alla possibilità di continuare la valutazione futura utilizzando, invece, modelli LLMs a dimensioni più elevate per prestazioni migliori.

Possibili sviluppi futuri potrebbero focalizzarsi sull'uso di parametri aggiuntivi per migliorare le performance dei LLMs, come l'integrazione di sinonimi per ciascun senso o l'uso di esempi estratti da ulteriori risorse lessicali. Inoltre, studi successivi potrebbero essere condotti su più lingue, rendendo il modello più generalizzabile e applicabile in possibili contesti reali.

Un'altra questione rilevante riguarda l'applicazione concreta del WSD. Guardando agli scenari reali che il Word Sense Disambiguation può trovare, emerge sicuramente la problematica legata a quale sia l'approccio migliore per poter realmente sfruttare queste tecniche in un contesto reale. Le applicazioni che possono beneficiare di un'accurata disambiguazione potrebbero essere altamente selettive sul procedimento che viene loro applicato. Basti pensare che l'approccio proposto in questo progetto, nonostante ottenga risultati vicini a valori SOTA, trova una difficile collocazione nel costesto reale e su larga scala, in quanto richiede tempi di elaborazione e costi computazionali considerevoli.

La disambiguazione lessicale potrebbe offrire, però, vantaggi significativi per sistemi di WSD specializzati in specifici domini. In contesti reali prettamente tecnici, come ad esempio in campo medico, finanziario, i sistemi di disambiguazione potrebbero raggiungere prestazioni da subito elevate, per via del lessico estremamente specifico richiesto. Al contrario, ad esempio, di problematiche riscontrate con risorse come WordNet, dove l'elevata granularità dei sensi è risultata talvolta dannosa alla corretta disambiguazione, in domini molto specifici, in cui il grado di polisemia è ridotto, il processo di disambiguazione risulta facilitato.

In conclusione, questo progetto di Tesi ha dimostrato come l'integrazione di tecniche di Prompt Engineering con i Large Language Models rappresenti un'efficace strategia per il WSD. Nonostante le sfide ancora aperte, lo sviluppo di modelli sempre più avanzati permetteranno di migliorare ulteriormente la comprensione del linguaggio naturale, aprendo nuove opportunità per il futuro del NLP e per l'applicazione del Word Sense Disambiguation in contesti reali.

Bibliografia

[1] W. Weaver (1949)
Translation. In Machine Translation of Languages: Fourteen Essays
Cambridge, Mass.: Technology Press of the Massachusetts Institute of Technology, pp.15-23

[2] E.F. Kelly & P.J. Stone (1975)
Computer Recognition of English Word Senses
Elsevier, Amsterdam

[3] E. Agirre & P. Edmonds (2007)
Word Sense Disambiguation: Algorithms and Applications
Springer Science and Business Media

[4] M. Bevilacqua, T. Pasini, A. Raganato & R. Navigli (2021)
Recent trends in word sense disambiguation: A survey.
International Joint Conference on Artificial Intelligence, online, pp.4330-4338

[5] D. Yarowsky (1995)
Unsupervised Word Sense Disambiguation Rivaling Supervised Methods.
In 33rd annual meeting of the association for computational linguistics, Cambridge, Massachusetts, USA, pp.189-196

[6] I. Rish (2001)
An empirical study of the naive Bayes classifier.
In IJCAI 2001 workshop on empirical methods in artificial intelligence, New York, USA, Vol. 3, No. 22, pp.41-46

[7] S. Brody & M. Lapata (2008)
Good neighbors make good senses: Exploiting distributional similarity for unsupervised WSD.

In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), Manchester, UK, pp.65-72

[8] M.E. Lesk (1986)
Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone.
In Proceedings of the 5th annual international conference on Systems documentation, New York, USA, pp. 24-26

[9] E. Barba, T. Pasini & R. Navigli (2021)
ESC: Redesigning WSD with Extractive Sense Comprehension.
In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,
pp. 4661-4672

[10] G.A. Miller, R. Beckwith, C. Fellbaum & D. Gross (1990)
Introduction to WordNet: An on-line lexical database.
International journal of lexicography, 3(4), pp.235-244

[11] R. Navigli & S. Ponzetto (2012)
BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network.
Artificial intelligence, 193, pp.217-250

[12] Wiktionary:Statistics
`https://en.wiktionary.org/wiki/Wiktionary:Statistics`

[13] G.A. Miller, C. Leacock, R. Tengi & R.T.Bunker (1993)
A semantic concordance.
In Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, pp.21-24

[14] OMSTI Training Data
`http://lcl.uniroma1.it/wsdeval/training-data`

[15] T. Pasini, A. Raganato & R. Navigli (2021)
XL-WSD: An extra-large and cross-lingual evaluation framework for word sense disambiguation.
In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 15), online, pp.13648-13656

[16] A. Raganato, J. Camacho-Collados & R. Navigli (2017)
Word sense disambiguation: A unified evaluation framework and empirical comparison.
In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1 Long Papers, Valencia, Spain, pp.99-110

[17] Z. Zhong & H.T. Ng (2010)
It makes sense: A wide-coverage word sense disambiguation system for free text.
In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, Uppsala, Sweden, pp.78–83

[18] O. Melamud, J. Goldberger & I. Dagan (2016)
context2vec: Learning generic context embedding with bidirectional lstm.
In Proceedings of the 20th SIGNLL conference on computational natural language learning, Berlin, Germany, pp.51-61

[19] M. Bevilacqua & R. Navigli (2020)
Breaking Through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information.
In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, online, pp.2854–2864

[20] L. Huang, C. Sun, X. Qiu & X. Huang (2019)
GlossBERT: BERT for word sense disambiguation with gloss knowledge.
arXiv preprint arXiv:1908.07245

[21] E. Barba, L. Procopio, & R. Navigli (2021)
ConSeC: Word sense disambiguation as continuous sense comprehension.
In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, pp.1492-1503

[22] P. Basile, A. Caputo & G. Semeraro (2014)
An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model.
In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin, Ireland, pp.1591–1600

[23] R. Mihalcea (2005)
Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling.

In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada. Association for Computational Linguistics., pp.411–418

[24] E. Agirre & A. Soroa (2009)
Personalizing PageRank for Word Sense Disambiguation.
In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), Athens, Greece, pp. 33-41

[25] A. Moro, A. Raganato & R. Navigli (2014)
Entity linking meets word sense disambiguation: a unified approach.
Transactions of the Association for Computational Linguistics, Cambridge, MA, pp.231-244

[26] J. Gonzalo & F. Verdejo (2007)
Automatic Acquisition of Lexical Information and Examples.
Word Sense Disambiguation: Algorithms And Applications, Springer, pp.253-274

[27] G.A. Miller (1995)
WordNet: a lexical database for English
Communications of the ACM, 38(11), pp.39-41

[28] S. Hochreiter (1998)
The vanishing gradient problem during learning recurrent neural nets and problem solutions.
International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6(02), pp.107-116

[29] S. Hochreiter (1997)
Long Short-term Memory
Neural Computation MIT-Press, 9(8), pp.1735-1780

[30] N. Gruber & A. Jockisch (2020)
Are GRU cells more specific and LSTM cells more sensitive in motive classification of text?
Frontiers in artificial intelligence, 3, 40

[31] A. Vaswani (2017)
Attention is all you need.
Advances in neural information processing systems, 30, I

[32] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu & D. Amodei (2020)
Scaling laws for neural language models.
arXiv preprint arXiv:2001.08361.

[33] J. Devlin, M. W. Chang, K. Lee & K. Toutanova (2019)
Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding.
In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp.4171-4186

[34] B. Schwartz (2020)
Google: BERT now used on almost every English query.
`https://searchengineland.com/google-bert-used-on-almost-every-english-query-34219`

[35] I. Caswell & B. Liang (2020)
Recent advances in google translate.
Google AI Blog, 8.

[36] openAI-GPT-3
`https://openai.com/index/gpt-3-apps/`

[37] W.D. Heaven (2023)
The inside story of how ChatGPT was built from the people who made it.
MIT Technical Review.

[38] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals & L. Sifre (2022)
Training compute-optimal large language models.
arXiv preprint arXiv:2203.15556

[39] K. Wiggers (2024)
Meta unveils a new, more efficient Llama model
TechCrunch | Startup and Technology News
`https://techcrunch.com/2024/12/06/meta-unveils-a-new-more-efficient-llama-model/`

[40] D. Milmo (2023)
Google says new AI model Gemini outperforms ChatGPT in most tests.
The Guardian ISSN 0261-3077

[41] K. Clark, U. Khandelwal, O. Levy, & C. Manning (2019)
What Does BERT Look at? An Analysis of BERT's Attention.
arXiv preprint arXiv:1906.04341

[42] C.E. Shannon (1948)
A mathematical theory of communication.
The Bell system technical journal, 27(3), pp.379-423

[43] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer & P. S. Roossin (1990)
A statistical approach to machine translation.
Computational linguistics, 16(2), Venue CL, pp.79-85

[44] Google-research (2024)
T5: Text-To-Text Transfer Transformer
`https://github.com/google-research/text-to-text-transfer-transformer`

[45] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang & X. Xie (2024)
A survey on evaluation of large language models.
ACM Transactions on Intelligent Systems and Technology, 15(3), pp.1-45

[46] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto & P. Fung (2023)
Survey of hallucination in natural language generation.
ACM computing surveys, 55(12), pp.1-38

[47] N. Guha, J. Nyarko, D. Ho, C. Ré, A. Chilton, A. K, A. Chohlas-Wood, A. Peters, B. Waldon, D. Rockmore, D. Zambrano, D. Talisman, E. Hoque, F. Surani, F. Fagan, G. Sarfaty, G. Dickinson, H. Porat, J. Hegland, J. Wu, J. Nudell, J. Niklaus, J. Nay, J. Choi, K. Tobia, M. Hagan, M. Ma, M. Livermore, N. Rasumov-Rahe, N. Holzenberger, N. Kolt, P. Henderson, S. Rehaag, S. Goel, S. Gao, S. Williams, S. Gandhi, T. Zur, V. Iyer & Z. Li. (2023)
Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models.
Advances in Neural Information Processing Systems, 36, pp.44123-44279

[48] GitHub Copilot
`https://github.com/features/copilot`

[49] CodexAI
https://openai.com/index/openai-codex/

[50] Meta AI LLaMA
https://www.llama.com/

[51] OpenAI GPT-4
https://openai.com/index/gpt-4/

[52] E. David (2023)
Meta's AI research head wants open source licensing to change
The Verge
https://www.theverge.com/2023/10/30/23935587/
meta-generative-ai-models-open-source

[53] E.M. Bender, T. Gebru, A. McMillan-Major & S. Shmitchell (2021)
On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?
In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21). Association for Computing Machinery, New York, NY, USA, pp.610–623
https://doi.org/10.1145/3442188.3445922

[54] Hugging Face Models
https://huggingface.co/models

[55] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou & H.W. Hon (2019)
Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems, 32*

[56] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer & V. Stoyanov (2019)
Roberta: A robustly optimized bert pretraining approach.
arXiv abs/1907.11692

[57] M. McCloske & N.J. Cohen (1989)
Catastrophic interference in connectionist networks: The sequential learning problem. *In Psychology of learning and motivation (Vol. 24). Academic Press*, pp.109-165

[58] A. Radford & J. Wu (2019)
Language models are unsupervised multitask learners.
OpenAI blog, 1(8), 9

[59] E. Perez, D. Kiela & K. Cho (2021)
True few-shot learning with language models.
In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS'21), pp.11054-11070

[60] Mistral 7b-Instruct
`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`

[61] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A.H. Miller & S. Riedel (2019)
Language models as knowledge bases?
In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, pp.2463–2473

[62] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever & D. Amodei (2020)
Language models are few-shot learners.
Advances in neural information processing systems, 33, pp.1877-1901

[63] T. Schick & H. Schütze (2020)
Exploiting cloze questions for few shot text classification and natural language inference.
arXiv preprint arXiv:2001.07676

[64] Z. Jiang, F.F. Xu, J. Araki & G. Neubig (2020)
How can we know what language models know?
Transactions of the Association for Computational Linguistics, 8, pp.423-438

[65] D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark & H. Hajishirzi (2020)
UNIFIEDQA: Crossing format boundaries with a single QA system.

In Findings of the Association for Computational Linguistics: EMNLP 2020, pp.1896–1907

[66] B. Romera-Paredes & P. Torr (2015)
An embarrassingly simple approach to zero-shot learning.
In Proceedings of the International Conference on Machine Learning. PMLR, Lille, France, pp.2152–2161

[67] J. Wei, M. Bosma, V.Y. Zhao, K. Guu, A.W. Yu, B. Lester, N. Du, A.M. Dai & Q.V. Le (2021)
Finetuned language models are zero-shot learners.
arXiv preprint arXiv:2109.01652

[68] T. Kojima, S.S. Gu, M. Reid, Y. Matsuo & Y. Iwasawa (2022)
Large language models are zero-shot reasoners.
Advances in neural information processing systems, 35, pp.22199-22213

[69] M. Ortega-Martín, O. Sierra, A. Ardoiz, J. Álvarez, J.C. Armenteros & A. Alonso (2023)
Linguistic ambiguity analysis in ChatGPT.
arXiv preprint arXiv:2302.06426.

[70] T. Sumanathilaka, N. Micallef & J. Hough (2024)
Can LLMs assist with Ambiguity? A Quantitative Evaluation of various Large Language Models on Word Sense Disambiguation.
arXiv preprint arXiv:2411.18337.

[71] SLURM Documentation
`https://slurm.schedmd.com/overview.html`

[72] S. Pradhan, E. Loper, D. Dligach & M. Palmer (2007)
Semeval-2007 task-17: English lexical sample, srl and all words.
In Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007), Prague, Czech Republic, pp.87-92

[73] NLTK Documentation
`https://www.nltk.org/`

[74] LLaMA2-7b Chat Model
`https://huggingface.co/meta-llama/Llama-2-7b-chat-hf`

[75] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang & T. B. Hashimoto (2023)
Alpaca: A strong, replicable instruction-following model.
Stanford Center for Research on Foundation Models, 3(6), 7
`https://crfm.stanford.edu/2023/03/13/alpaca.html`

[76] LLaMA3.1-8b Instruct Model
`https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`

[77] D. Sumanathilaka, N. Micallef & J. Hough (2025)
GlossGPT: GPT for Word Sense Disambiguation using Few-shot Chain-of-Thought Prompting.
The 8th International Conference on Emerging Data and Industry (EDI40), Patras, Greece